UNIVERSIDAD DEL VALLE

ESCUELA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

AN AUTOMATIC CROP ROWS GENERATOR USING AERIAL HIGH-RESOLUTION IMAGES FOR PRECISION AGRICULTURE

By

FABIO ANDRÉS HERRERA ROZO

B.Eng. Topographic Engineer, PgD. in Geomatics

MASTER IN ENGINEERING WITH A MAJOR IN SYSTEMS ENGINEERING

Santiago de Cali, Colombia

January 23, 2019

AN AUTOMATIC CROP ROWS GENERATOR USING AERIAL HIGH-RESOLUTION IMAGES FOR PRECISION AGRICULTURE

Presented by:

Fabio Andrés Herrera Rozo

A document submitted in partial fulfillment of the requirements for the degree of:

Master in Engineering with a Major in Systems Engineering

Supervised by:

María Patricia Trujillo, Ph.D.

Iván Mauricio Cabezas Troyano, Ph.D.

Research Line:

Computer Vision Techniques in Precision Agriculture

Research Group:

Multimedia and Computer Vision Research Group (**MMV-LAB**)

UNIVERSIDAD DEL VALLE

Santiago de Cali, Colombia

January 23, 2019

Disclaimer

The author declares that the research presented in this document is an original work and has not been submitted before to any institution for assessment purposes.

Further, all sources used has been acknowledged and cited in the reference section.

Fabio Andrés Herrera Rozo
Santiago de Cali - Colombia
January 23, 2019


Please consider the environment before printing this document.

We hereby approve this document

_____

_____

_____

_____

_____

María Patricia Trujillo, Ph.D.

Principal Advisor

_____

Iván Mauricio Cabezas Troyano, Ph.D.

Advisor

_____

Committee Member (Chair)

_____

Committee Member

_____

(Date)

Santiago de Cali - Colombia

Dedicatory

This work is dedicated to my lovely family.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisors María Patricia Trujillo Ph.D. and Iván Mauricio Cabezas Ph.D. whose provided detailed guidance and encouragement throughout the course of preparing for and conducting the research.

Special thanks to my colleagues from Colombian Sugarcane Research Center (CENICAÑA) and sugarcane mills agronomy and precision agriculture departments who provided insight and expertise that greatly assisted the research.

To my students of the Fundamentals of GIS course (2018-II) who assisted me on validation processes through manual crop rows digitizing.

Finally, express my gratitude to everyone who supported me throughout the course of this Master research project. I am thankful for their guidance, valuable constructive criticism, and friendly advice during the development process of the project.

Abstract

High spatial resolution images obtained by cameras onboard on unmanned aerial vehicles (i.e drones) have significant potential in modern agriculture, as they allow the accuracy and efficiency of some field operations to be improved from their analysis and interpretation. Georeferenced crop rows are used as an input for guiding precision agricultural machinery. Agricultural machinery is supported by highly accurate, real-time kinematic global satellite navigation systems. Georeferenced crop rows generation is an expensive and time-consuming task. This project addressed a crop rows generation problem in sugarcane crops through image processing and computer vision techniques. Using high-resolution aerial imagery from unmanned aerial vehicles an automated method was designed and built to generate georeferenced crop rows quickly and accurately. The computational tool ***"Crop Rows Generator CRG v1.0"*** was developed. CRG involves computer vision techniques and a high-performance computing approach which are capable of processing high-resolution large images and on these images detect, generate and mapping crop rows in sugarcane fields, with a few clicks. CRG is open source and consists of a client interface (*CRG-QGIS Plugin*) integrated as a plugin into the geographic information system software (QGIS Desktop), a processing core accessed by a command line interface (*CRG-CLI*) and resources exposed through on HTTP methods by REST API (*CRG-API*). Generate crop rows using the developed tool improves processing above to 200% compared to manual method and beyond 650% compared to the mechanized method. Crop Rows generation performance is above to 83% and overall most of the horizontal errors are in range between ±(2.5 to 10) cm.

Keywords: *Computer Vision, Crop Rows, GIS, HPC, Precision Agriculture, QGIS.*

Resumen

Las imágenes de alta resolución espacial obtenidas por cámaras instaladas en vehículos aéreos no tripulados (drones) tienen un potencial significativo en la agricultura moderna, puesto que permiten mejorar la precisión y la eficiencia de algunas de las operaciones en campo a partir de su análisis e interpretación. Las líneas de surcado georreferenciadas son utilizadas como insumo para el guiado de maquinaria agrícola de precisión. La maquinaria en campo se apoya en sistemas globales de navegación por satélite cinemáticos y en tiempo real, los cuales son altamente precisos. La generación de líneas de surcado georreferenciadas es una tarea costosa y que consume mucho tiempo. En este proyecto se abordó el problema de la generación de líneas surcado en cultivos de caña de azúcar a través del procesamiento de imágenes y técnicas de visión por computador. Fue diseñado y construido un método automatizado para generar de forma rápida y precisa líneas de surcado georreferenciadas utilizando imágenes aéreas de alta resolución obtenidas mediante vehículos aéreos no tripulados. Fue desarrollada una herramienta computacional ***"Crop Rows Generator (CRG) v1.0"*** la cual se apoya en técnicas de visión por computador y bajo un enfoque de computación de alto rendimiento es capaz de procesar imágenes de gran tamaño y alta resolución y sobre estas imágenes detectar, generar y cartografiar las líneas de surco en campos de cultivos de caña de azúcar de una manera fácil y precisa,con unos pocos clics. CRG es de código abierto y consta de una interface cliente (*CRG-QGIS Plugin*) integrada a manera de plugin dentro del software de SIG (QGIS Desktop), un núcleo de procesamiento que se encarga de las tareas más complejas y que se accede mediante una interfaz de línea de comandos (*CRG–CLI*) y a través de recursos expuestos sobre métodos HTTP en un API tipo REST (*CRG-API*). Generar líneas de surcado mediante la herramienta desarrollada optimiza el tiempo de proceso en más de un 200% frente el metodo manual y en más de un 650% frente a la generación realizada por un tractor. El desempeño en la generación de lineas de surco está por encima del 83% y en general la mayoría de los errores horizontales se encuentran en el rango de $\pm$(2.5 a 10)cm.

Palabras clave: *Visión por Computador, Surcos, GIS, HPC, Agricultura de Precisión, QGIS.*

Table of Contents

List of Tables

List of Figures

List of Code blocks

Acronyms

The following table describes the meaning of various abbreviations and acronyms used throughout the document. The page where it is defined or first used is also given.

Abbreviation	Meaning	Page
ANN	Artificial Neural Network.	30
API	Application Programming Interface.	19
BBOX	Bounding Box.	63
CPU	Central Processing Unit.	21
CV	Computer Vision.	1
CMYK	Cyan, Magenta, Yellow, and black color space.	28
FT	Fourier transform.	38
FURPS	Functionality, Usability, Reliability, Performance and Supportability.	74
GCS	Geographic Coordinate System.	18
GeoTIFF	Georeferenced Tagged Image File Format.	46
GeoJSON	GEO JavaScript Object Notation.	63
GHG	Global Greenhouse Gas.	11
GIS	Geographic Information System.	1
GNSS	Global Navigation Satellite System.	12
GPS	Global Positioning System.	1
GSD	Ground Sampling Distance.	89
GT	Growing Trait.	89
GUI	Graphical User Interface.	19
HSV	Hue,Saturation,Value color space.	28
HT	Hough transform.	38
HPC	High Performance Computing.	20

Abbreviation	Meaning	Page
KML	Keyhole Markup Language.	18
LED	Light Emitting Diode.	12
LSD	Line Segment Detector.	38
$LARS$	Low Altitude Remote Sensing.	13
$MIMD$	Multiple Instruction Multiple Data.	21
MBR	Minimum Bounding Rectangle.	63
MG	Machine Guidance.	12
$OPENCV$	Open Computer Vision.	79
$OMBB$	Oriented Minimum Bounding Box.	66
PA	Precision Agriculture or precision farming.	1
PC	Personal Computer.	129
PCS	Projected Coordinate System.	17
PT	Plant Cane trait.	89
$OBIA$	Object-Based Image Analysis.	42
$QGIS$	Open-source desktop geographic information system application.	19
RGB	Red-Green-Blue color space.	1
$RPAS$	Remotely Piloted Aircraft System similar term as UAV.	1
RTK	Real Time Kinematic.	1
RC	Ratoon Crop trait.	89
$SIMD$	Single Instruction Multiple Data.	21
TP	True Positive.	90
TN	True Negative.	90
FP	False Positive.	90
FN	True Negative.	90
UAV	Unmanned Aerial Vehicle commonly known as a **drone**.	13
$WGS84$	World Geodetic System 84.	18
XML	eXtendible Markup Language.	18

1. Introduction

The availability of georeferenced spatial data has opened new opportunities for crop management in modern agriculture [62]. Innovative agricultural techniques, known as site-specific farming, prescription farming, precision farming, or precision agriculture (PA), are developing technologies which lead to incorporate the advanced techniques to enhance farm outcomes and also enrich the farm feedstock's in profitable and environmentally sensible manner [78].

PA usage can assist farmers in decision making about seed selection, crop production, disease monitoring, weed control, pesticides and fertilizers usage, among others. PA is a combination of technologies such as global positioning systems (GPS), real-time kinematic global navigation satellite systems (RTK-GNSS), remote sensing (RS), electronic sensors and devices, geographic information systems (GIS), and computer vision (CV) in order to provide the necessary information to make the local management of the agricultural activities at within-field detail feasible. An accuracy of centimeter level is needed to achieve the required operation objectives.

Unmanned aerial vehicles also known as drones or RPAS represent a new way to collect field-level data. The most compelling reason for using drones is that the results are on-demand; whenever and wherever needed, a drone can be easily and quickly deployed. Drones can provide farmers with an eye on crop from the air and airborne cameras can take multispectral images, capturing data using visual spectrum (RGB) as well as infrared. Some soil and plant properties can be estimate by image processing methods such as historical yield maps, core and leaf samples, aircraft and satellite imagery. However, image resolution is the most important problem in these methods [3]. Also, these methods often can be time consuming, and when data is collected it can take a long time to process and analyze [99].

PA demands very high spatial and temporal resolution, measuring even characteristics of individual plants in some applications [20]. The trends towards precision farming techniques are reliant on specific location data including taking data of multiple image sources. Image processing techniques is one way that these data sets could be used to assist by providing certain analysis on high-resolution Images to be used for decision-making [6].

Farm machinery guidance uses RTK-GNSS positioning technologies to assist drivers in following an optimal path, thus minimizing risks of overlaps. Machine vision systems applied to agricultural tasks have great potential [14, 24]. The use of technology, including vision systems, in agricultural applications could reduce certain manual tasks and a crop production cost, besides, they can contribute to the productivity and competitiveness of farmers to ensure agricultural supplies. Automatic steering completely takes over steering of the farm equipment from a driver allowing an operator to reduce labor or fatigue and engage in core agricultural tasks. Demand for high-precision guidance and mapping in agriculture has resulted in tractor systems pioneering the way in vehicle guidance systems have achieved steering accuracy of less than 2.5 cm [90].

PA is supported by highly accurate data, for instance a centimeter resolution imagery captured by a drone is used in crop rows mapping. Image pre-processing can be demanded when the illumination levels and other aspects of the field cannot be controlled. Thereby, algorithms should be developed to deal with these problems. In addition, the processing pipeline between orthorectified images and an output map of crop rows should be fully automated, fast and efficient in order to be efficient. Moreover, and according to our understanding, there is no an open source software available that provides these characteristics.

Current research in computer science is moving forward the utilization of advanced computer vision techniques and high-performance computing in PA problems. Thus, the

objective of this research is to develop an automated open source application integrated into a GIS software capable of generating accurate geospatial information for PA applications. In this manner, time and resources could be optimized for certain field operations in PA. From the application domain, sugarcane is an essential crop for the Colombian Cauca River Valley economy. In 2017 there are more than 240.000 hectares growing in five administrative departments(Caldas, Risaralda, Quindio, Valle del Cauca, and Cauca)[18].

1.1. Research Problem

High precision georeferenced crop rows are required as an input for machinery guidance in precision agriculture. Generate crop rows is time-consuming and an expensive task. If crop rows data is not available, it is necessary to move a tractor across the entire field, which is not environmental friendly due to fuel consumption. Also, perform a manual generation of crop rows using a GIS software may be feasible nonetheless manual methods are difficult, expensive with the lowest efficiency due to human limitations. Approximately 6.6 km of crop rows may exist in a field of one hectare planted at a 1.5 m. Sugarcane fields are on average of 6 ha in the Cauca Valey. Thus, collecting georeferenced crop rows by traditional methods is a demanding and exhaustive task. However, nowadays, a vision of the entire field can be obtained through aerial images captured from a drone, where clearly a crop row is perfectly identified. Nevertheless, image quality in outdoor agricultural environments is affected by uncontrolled conditions, such as variable lighting (shadows, deficient or high illumination), among others, and different field conditions can make difficult an automatic crop rows detection and mapping. Moreover, an automated or semi-automated application for quickly and accurately georeferenced crop rows generation in sugarcane is not available.

1.2. Research Question

This research aims to answer a question:

- How can it be obtained high-precision georeferenced crop rows, in sugar cane fields, using computer vision techniques with images captured by a drone?

1.3. Objectives

The general objective and the specific objectives are described below.

1.3.1. General Objective

- Design and build a method for generating georeferenced sugarcane crop rows using high-resolution aerial images.

1.3.2. Specific Objectives

- Select a high-performance computing approach for processing aerial high-resolution images.
- Select computer vision techniques to identify and generate crop rows.
- Develop an open source application for crop rows generation by integrating selected techniques and the HPC approach.
- Test the functionality and evaluate the performance of the developed open source application.
- Compare obtained results against the acquired by traditional methods.

1.4. Document Structure

The remainder of this document is organized as follows: background information in the areas of precision agriculture, geographic information systems, computer vision, and image processing as well as high-performance computing is given in Chapter Two. Similar researches related to crop rows generation approaches and its applications in different crops is reviewed in Chapter Three. All the conducted work of this research is presented in Chapter Four. Project findings and suggests areas of future research are summarized in Chapter Five. Seven appendices are included with additional information.

1.5. Objectives Fulfillment

Specific objectives fulfillment is reflected in Section 4. A high-performance computing approach for processing aerial high-resolution images was selected in Section 4.2. Computer vision techniques to identify and generate crop rows are conducted in Section 4.3.1. In Section 4.4 an open source application for crop rows generation by integrating selected computer vision techniques and the HPC approach was develop. Evaluate the performance and test the functionality of the developed open source application is reflected in Section 4.4.6. Obtained results against the acquired by traditional methods are compared in Section 4.6.

2. Background

This section provides a brief background information from the disciplines which contribute to this project: agriculture and computing science. Therefore, the need to establish a conceptual baseline is essential. Topics related to geo-informatics, precision agriculture, high-performance computing, and image processing, among others, are presented.

2.1. Sugarcane Crop Rows

Furrows are small and parallel channels, made to conduct water in order to irrigate and drain the crop. A crop is usually grown on the ridges between the furrows [31]. Furrows are very important during the different tillage activities in a sugarcane crop. Crop rows direction is determined by the field design that, at the same time, it depends on the topography of a site.

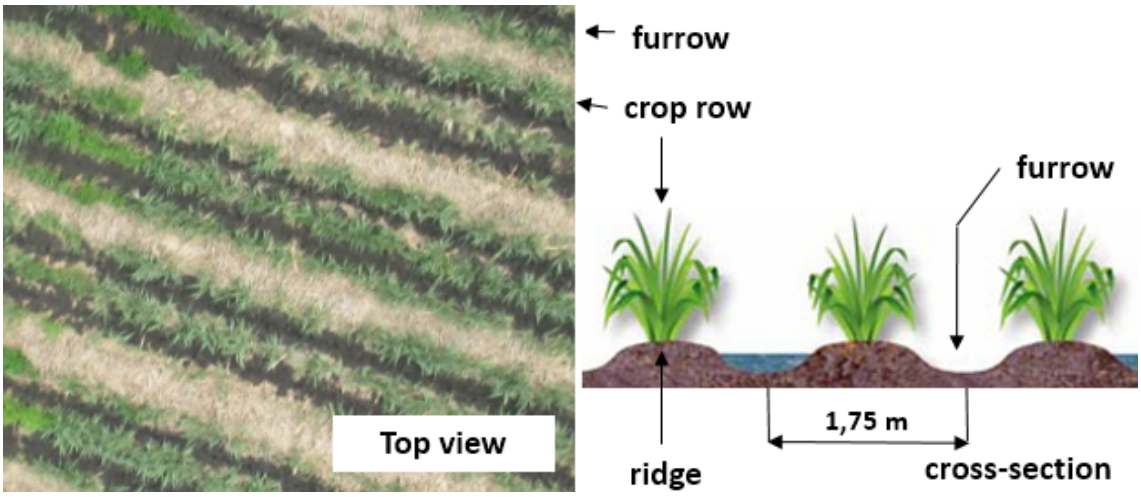

Figure 2.1.: An aerial view and cross-section of furrows and ridges in a sugarcane crop.

In the Colombian sugar industry of the valley geographical area of the Cauca river, the most common row spacing distance is 1.75 m and length between 130 to 150 m, due to

mechanization and irrigation practices. Nevertheless, when the paired furrow system is used, similar to the one used in the pineapple planting consists of planting in two narrow furrows spaced 0.75 m apart and leaving a wide street of 1.5 m [100]. The depth of the furrow depends on the quality of the soil preparation and can vary between 25 and 35 cm. In some cases, the distance between the wheels of the tractor is used to define planting distance (row spacing) [100]. Figure 2.1 shows an aerial view of part of a cultivated sugarcane field and there are two notable green colors inside it, the crop aligned in crop rows itself and some weed which usually covers parts of the field. Residues of the last harvest are located closest to the crop as a yellow cover, as well, furrows which usually are in the bare soil are the brown-dark. A schematic cross-section with main characteristics is shown on the right side of the image.

2.1.1. The utility of georeferenced Crop Rows

A list of the most common agricultural operations in sugarcane fields where georeferenced crop rows are important.

- Mechanized planting.
- Guiding machinery within the field.
- Fertilization.
- Weed identification.
- Calculation of reseeding percentages.
- Harvesting operations, especially at night.

2.1.2. Colombian sugarcane industry

The agroindustrial sector of Colombia sugarcane takes place in the Cauca River valley between latitudes 2° 49′ and 5° 13′ North and between longitudes 76° and 75° West, at altitudes between 920 meters and 1150 meters above sea level (m.a.s.l), with 240,000 hectares planted. It is ranked as the most productive per unit area in the world but represents only 1.1% of world sugar market. However, this sector is one of the most

important for the Colombian economy, generating approximately 190,000 direct jobs, with a yearly production of 24.3 million tons (Mt) of sugarcane, 2.4 Mt of sugar, 406 million liters (Ml) ethanol, 0.28 Mt of molasses, and 215 MW power cogeneration, among other products and services. Strategic spatial location has special conditions for agronomic labors such as planting, harvesting, fertilizing sugarcane crops throughout the year due to agroclimatic advantages.

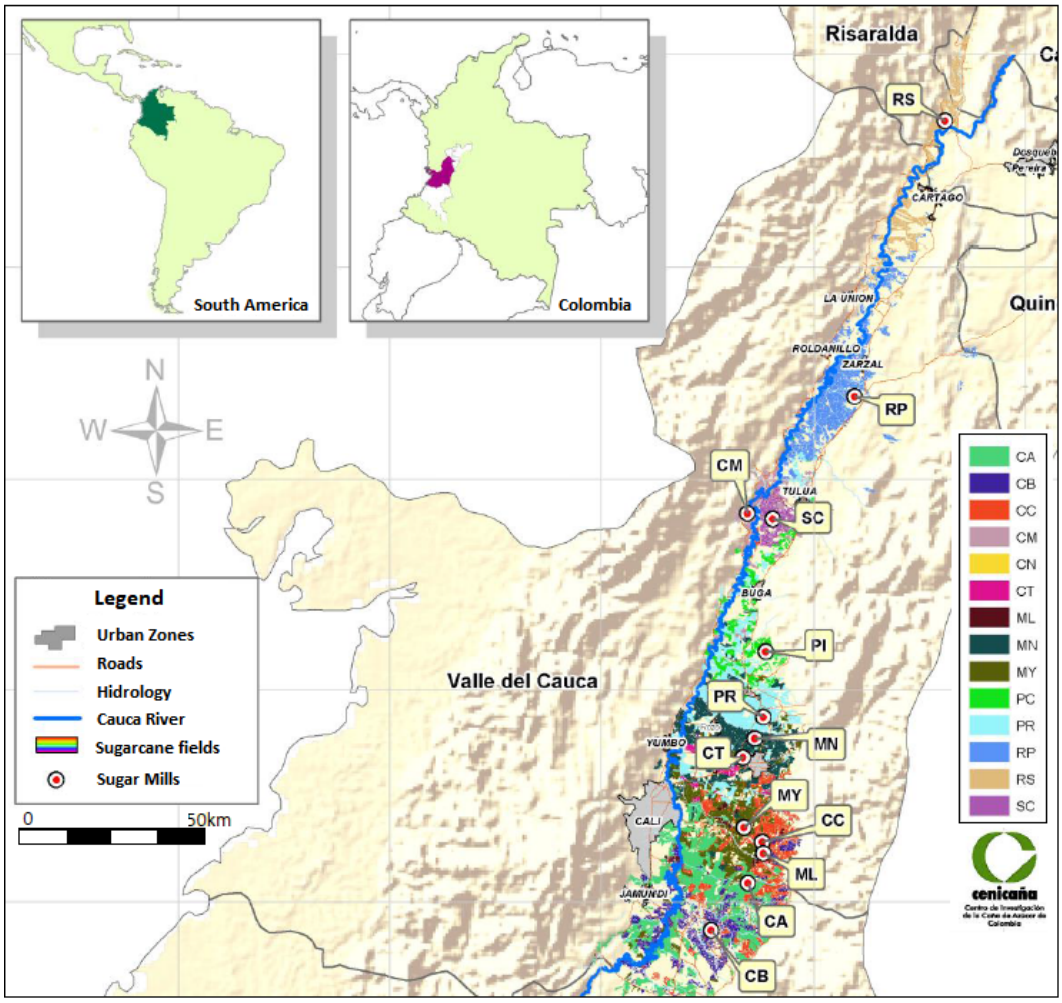

Figure 2.2.: Colombian Sugarcane industry spatial location at the geographic valley of Cauca river (Source: Cenicaña 2017).

2.2. Methods for Crop Rows Generation

There are two common methods where georeferenced crop rows could be generated.

2.2.1. Mechanized method

In a mechanized method, a precision equipment is required at the field such as a (GNSS-RTK) devices attached to a tractor. Figure. 2.3 shows an illustration of tractor moving around a field on a crop row.

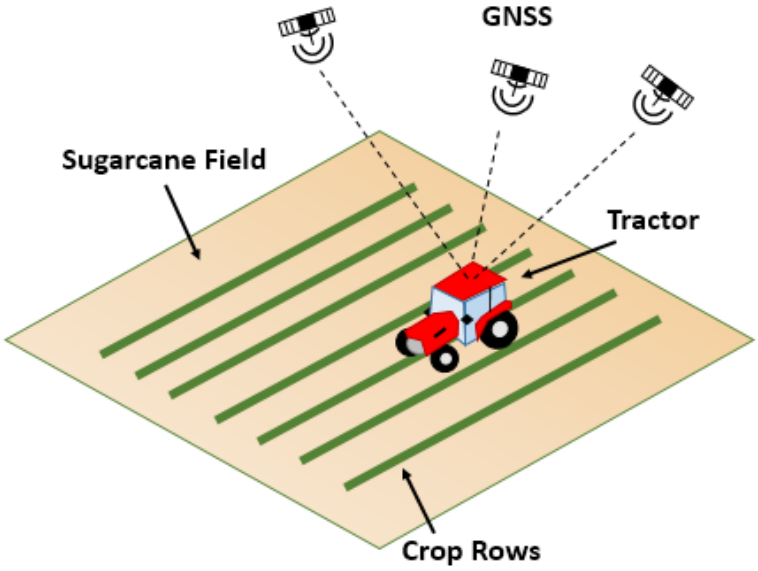

Figure 2.3.: Crop Rows generation by an instrumented tractor with GNSS device.

A tractor driven by an operator moves throughout a whole field on every crop row. Once, all the field was covered, the information collected on the onboard computer should be downloaded and processed at an office to obtain georeferenced crop rows.

2.2.2. Manual method

Digitizing in GIS is the process of converting geographic data from Raster orthomosaic image into vector data by tracing features. During the digitizing process, features from the traced map or image are captured as coordinates in line format. Crop Rows Manual

digitizing is the method of tracing geographic features from another dataset (usually an aerial image) directly on a computer screen through a GIS software. Figure. 2.4 represents a flow diagram for crop rows generation by doing a manual digitizing process.

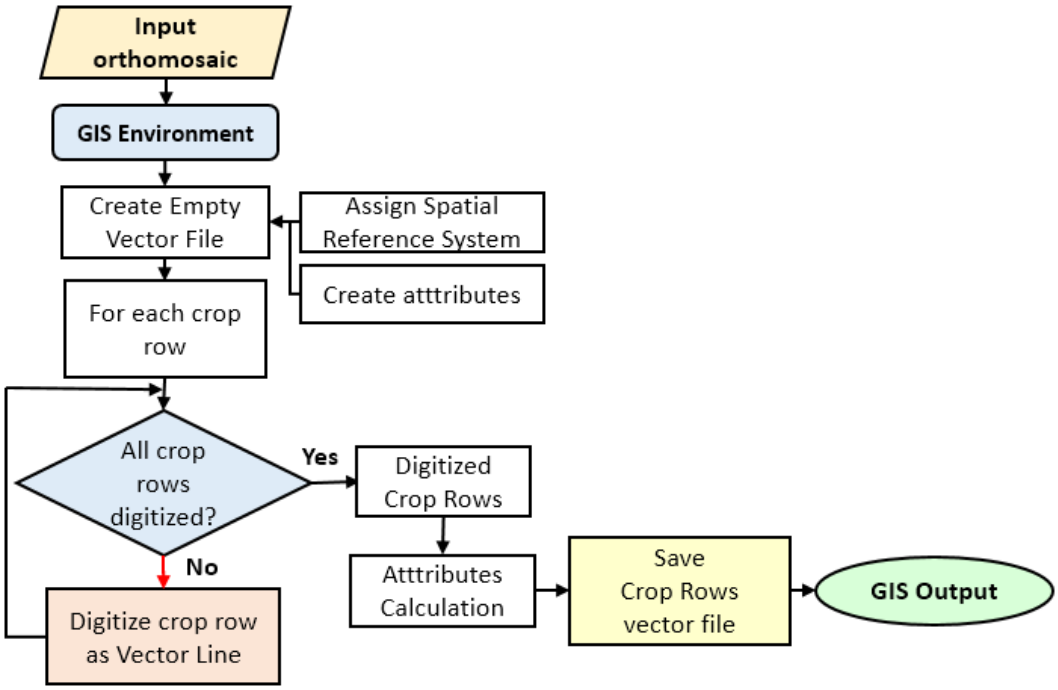

Figure 2.4.: Flow diagram for Crop Rows generation by traditional manual method. Digitizing vector lines in a GIS software.

The GIS Output is the result of the crop rows generation by the traditional method and corresponds to a georeferenced vector file of each Crop Row.

2.3. Precision Agriculture

Precision agriculture (PA) is a concept, also a management strategy, and even a philosophy [94]. As a concept, PA offers the promise of increasing productivity while decreasing production cost and minimizing environmental impacts. As a management strategy, PA matches resource applications and agronomic practices with soil properties and crop requirements as they vary across a site. As a philosophy, PA is the ability to manage square meters of land instead of square miles and is changing the farmer's relation to the land. Several technologies have been developed trying to obtain healthier agricultural products

and lower environmental impacts. The concept of PA provides a valuable framework to achieve the above mentioned [95]. PA focuses on a level of detail smaller than the size of an individual field. It can be described as a new concept for sustainable utilization of agricultural resources, and this can be achieved by managing agricultural systems based on information and knowledge [78]. Figure 2.5 shows a set of concepts part of PA derived from different geo-technologies sources.

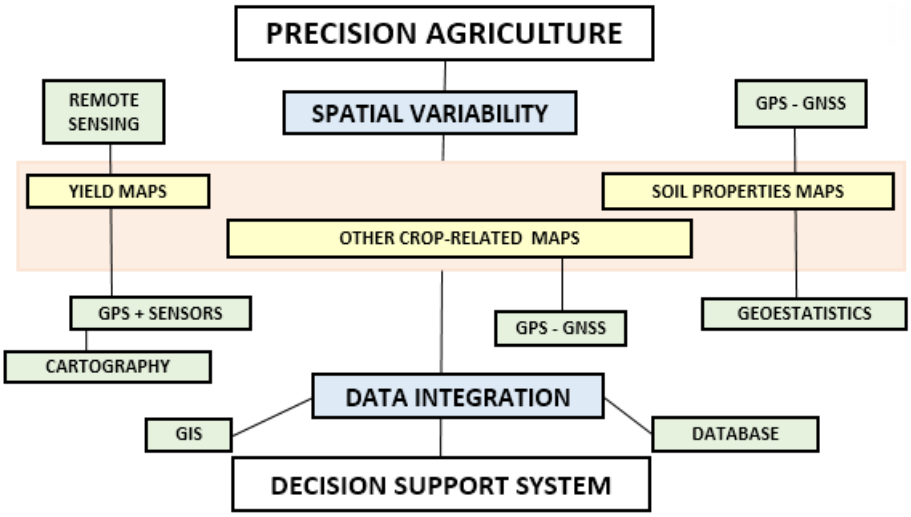

Figure 2.5.: Several geo-technologies are part of PA (adapted from [78]).

PA practices in agricultural field operations could positively contribute to global Greenhouse Gas (GHG) emission reduction due to [10]:

- The enhancement of the ability of soils to operate as carbon stock reserve by less tillage and reduced nitrogen fertilization (direct GHG decrease).
- The reduction of fuel consumption through fewer in-field operations with the tractor (direct GHG decrease).
- The reduction of inputs for the agricultural field operations (indirect GHG decrease).

PA practices improve farm productivity by optimizing agricultural inputs producing higher or equal yields with lower cost than conventional practices.

2.3.1. Machine guidance

Machine guidance (MG) refers to the applications of GNSS for steering and guidance though two main systems [10].

- **Driver assistance:** help the operator to keep drive over the crop line through add-ons that are not integrated into the tractor system and can be simply installed.
- **Machine auto-guidance:** are integrated in tractor hydraulics and can directly take over steering operations.

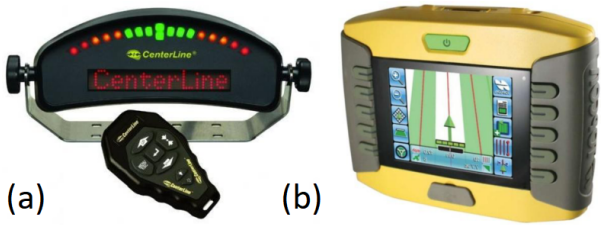

Figure 2.6.: A light bar control panel (a) and an auto-guidance control panel (b) (adapted from [80]).

Figure 2.6(a) shows a light bar which is a device with a horizontal series of Light Emitting Diodes (LEDs) located in front of the operator. If a light is at the centreline of the light bar, the machine is on target, while if a bar of light extends to one side, the machine is off the path and has to be steered. An advanced device is shown in Figure 2.6(b) wherein systems are coupled to onboard computers that allow for headland steering, section control and that accept drive-maps (routing) and task maps to operate implements.

2.3.2. Drones for precision agriculture applications

Satellite images have been used as primary source of information for analyzing crop status in PA. However, obtaining up-to-date aerial photography is expensive, image quality is variable, and data processing is also intensive [92]. Moreover, satellite and aerial remote sensing can be severely affected by cloud cover, and they may not be available at desired times. In the last decade, the development of drones, known as unmanned aerial vehicles

(UAV) platforms has offered a new solution for crop management and monitoring, since they are capable of timely provision of high-resolution images, especially where small productive areas have to be monitored [54]. Drones typically fly at low altitudes to acquire remotely sensed data (LARS). The most agricultural drones are a mini model (Figure 2.7) fixed-wing airplanes or rotary-winged helicopters of low cost, low speed, low ceiling altitude, lightweight, low payload weight capability, and short endurance [99]. Drones have the capability to provide remote sensing data in near real time in a field, rather than the resulting lag from satellite and aircraft-based imagery. For PA purposes drones should fly around 10 to 152 m high, above 152 m high, a special authorization in Colombia is required [2].

Figure 2.7.: Common aerial platforms used in agriculture contexts. (A) multirotor, (B) helicopter, (C) fixed-wing.

2.3.3. Precise positioning

Applications over a field are based on precise GNSS. There are two satellite constellations of Global Navigation Satellite Systems (GNSS) that are fully operational and commercially available to provide all-weather guidance virtually 24 h a day anywhere on the surface of the earth. GNSS are the collection of localization systems that use satellites to know the location of a user receiver in a global (Earth-centered) coordinate system and this has become the positioning system of choice for precision agriculture technologies. At present North American Positioning System known as Navigation by Satellite Timing and Ranging Global Position System (NAVSTAR GPS or simply GPS) and Russian Positioning System known as Global Navigation Satellite System (GLONASS) both qualify as GNSS. Two other satellite localization systems, Galileo (European Union) and Compass

(Chinese), are expected to achieve full global coverage capability by 2020[69]. Precision GNSS enables accurate mapping and system control of various field activities. Not all tasks, that have to be performed in precision agriculture, require the same level of positioning accuracies. Some precision agriculture operations such as yield monitoring, soil sampling or variable rate applications, can be performed using submeter accuracy differential GPS (DGPS) as errors below 1 m are acceptable. Tasks like mechanical intra-row weed control, thinning of crop plants, precise planting or autonomous navigation within tight rows demand decimeter or even centimeter level accuracy. A solution to this demanding requirement can be found in real-time kinematic GPS (RTK)[69]. High-precision or real-time kinematic (RTK) has become well adopted into autoguidance technology for tractors and harvesters to improve machine control and to accurately maintain spacing for adjacent passes [46]. Precision GNSS enables accurate mapping and system control of various field activities.

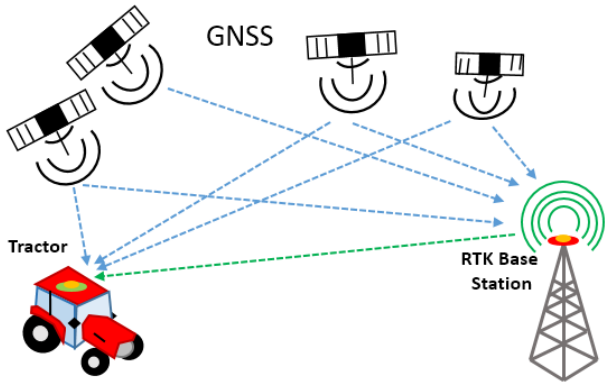

Figure 2.8.: GNSS enabled tractor.

A GNSS enabled tractor, in Figure 2.8, can automatically guide farm implements within a field. Farm machinery can be operated at higher speeds, day and night, with increased accuracy. This increased accuracy saves time and fuel and maximizes the efficiency of the operation. GNSS-based applications are used to support farm planning, field mapping, soil sampling, tractor guidance, and crop assessment. More precise application of fertilizers, pesticides, and herbicides reduces cost and environmental impact. Operator safety is also increased by greatly reducing fatigue.

2.4. Geographic Information System

A Geographic Information System (GIS) is a computer-based system which provides a set of capabilities for collecting, storing, retrieving, transforming, and displaying all kinds of spatial or geographical data. A GIS system stores spatial data in a digital mapping of the environment. GIS provides the capability to integrate diverse datasets. A GIS integrates five key components: hardware, software, data, people, and procedures or methods.

- **People:** are the most important part of a GIS, them must define and develop the procedures used by a GIS.
- **Data:** is the information used within a GIS, multiple sources may be incorporated into it.
- **Software:** provides the tools to manage, analyze, and effectively display and disseminate spatial data and spatial information.
- **Hardware:** is the computer system on which a GIS software operates.
- **Procedures/Methods:** are the steps according to a well-designed implementation plan and business rules.

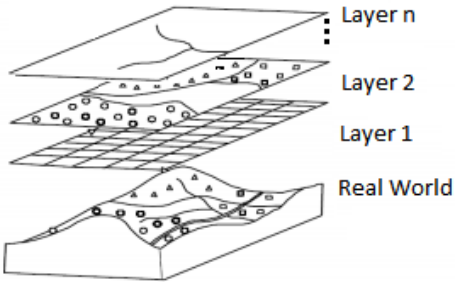

Figure 2.9.: GIS abstraction of real-world objects into a set of layers.

A GIS stores information about the world as layers of spatial features that can be overlaid with data or other layers of information onto a map in order to better viewing and understanding physical features and relationships as seen in Figure 2.9.
The geographical space can be represented by entities which are distributed with specific locations and attributes of these entities.Raster data model and Vector data model are the

ways used to represents the real world into discrete objects in GIS environment. GIS technologies can be used for scientific investigations, resource management, and agriculture among others.

2.5. Georeferenced data

Georeferenced data is spatial data that is referenced to a location on the earth's surface. Common frames of reference and coordinate systems have been set up [65] for referencing a location.

2.5.1. Vector data model

A vector object is defined as an array of connected vertexes with X, Y and Z coordinates. This model is also called an object-oriented model. Since the model assimilates the cartography and the relational databases. It uses the three entities of any map of objects: points, lines, and polygons, as seen in Figure 2.10. One of the main defining parts of vector data is attribute data which is information, able to describe different characteristics, and parameters of objects.

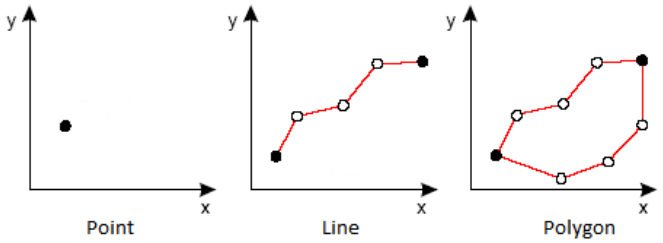

Figure 2.10.: A point, a line and a polygon represented in vector data model.

2.5.2. Raster data model

Raster data is defined as a matrix of cells, (Figure 2.11) called pixels, organized into rows and columns (or a grid) where each pixel containing attribute data of area respective to it. The fineness of the data is limited by the cell size. Each pixel is a square with a size

preselected when created, thus a single pixel can cover from tiny to a huge area. Raster data model are useful for storing data that varies continuously, as in an aerial photograph, a satellite image. Raster is convenient for storage and manipulation of region-type features and information from remotely sensed; image data is extracted efficiently. However, the processing speed can be extremely slow. In order to achieve high resolution, cell size have to be very small, and this may result in large data sets which are time-consuming to process.

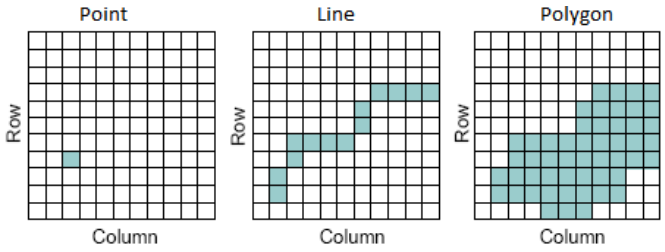

Figure 2.11.: A point, a line and a polygon represented in raster data model.

2.5.3. Coordinate systems

Maps are in two dimensions whereas the earth's surface is a three dimensional ellipsoid. Every map has a projection and scale. In GIS, there are many types of coordinate systems. In Figure 2.12, are shown a geographic (3D) and projected (2D) coordinate systems which are the most used.

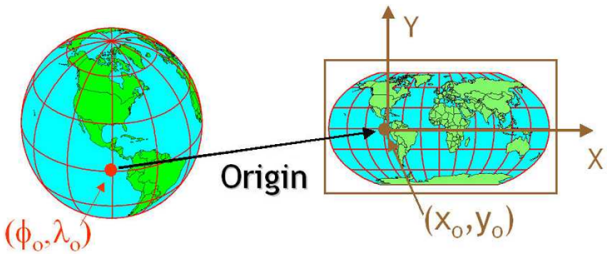

Figure 2.12.: Geographic(GCS) and projected (PCS) coordinate systems.

- **Projected coordinate system (PCS):** is a flattened version of a 3D coordinate system. It is the most localized coordinate system that utilizes zones which are fine-tuned for the highest accuracy on the ground. Projection is a mathematical

transformation used to project the real three dimensional spherical surface of the earth in two dimensions on a plane.

- **Geographic coordinate system (GCS):** uses a grid on the surface of a 3D globe. Graticule lines are not parallel to each other because they are defined using angles from the centre of the globe, not linear units (e.g. meters) on a flat surface. The Global Positioning System uses the World Geodetic System (WGS84) as a reference coordinate system.

2.5.4. GIS file formats

Georeferenced data is created, shared, and stored in many different formats. The field of GIS includes a large number of file formats. A list of common GIS file formats is presented below.

Vector Data File Formats

- **Shapefile:** Is the most common geospatial file type. All commercial and open source applications accept shapefile as a GIS exchanging vector format. Three files are mandatory: .shp is the feature geometry, .shx is the shape index position and .dbf is the attribute data [29].

- **GeoJSON:** Is a form of JSON developed for representing vector features. The GeoJSON spec gives some basic examples of how different entities such as point, lines, and polygons are structured [44].

- **KML:** Spatial data file format developed by Google Earth using XML notation [64].

Raster Data File Formats

- **GeoTIFF:** Is a format completely open, public domain, non-proprietary which provides geographic information to associate with the image data [61].

2.6. QGIS Software

QGIS is the abbreviation of Quantum GIS [76] and is a user-friendly Open Source Geographic Information System (GIS) released under the GNU license. QGIS supports vector, raster, and database formats and it runs in different platforms. The software provides a lot of tools for creating maps which geographic data is given. Likewise, QGIS permits to solve complex problems. With all advantages and actual demands, QGIS is applied in many research areas most of all for data viewing, editing, and analysis. QGIS also provides a plugin architecture for developers to enlarge the ability and functions of the system to support scientist community to developing and sharing their tools to interact with the geographic data. Open Source software can provide a feature-complete alternative to proprietary software in most system designs.

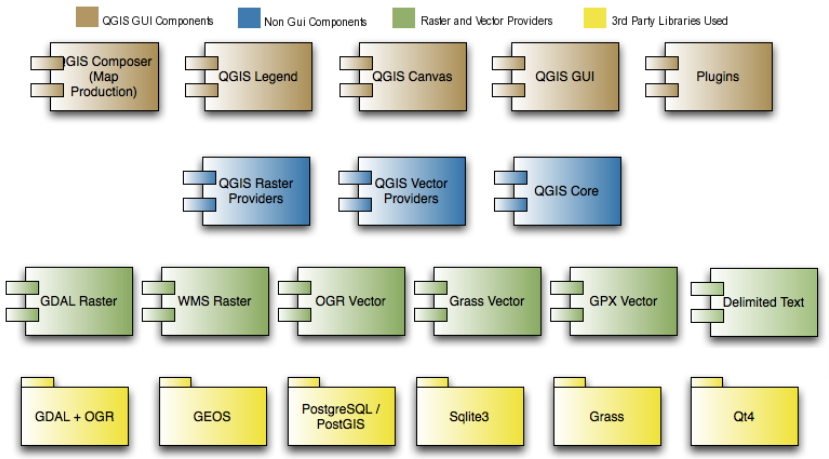

Figure 2.13.: QGIS high level architecture (adapted from [76]).

QGIS is build of multiple modules that deliver specific functionality to the whole system as seen in Figure 2.13. The whole application is managed by five main GUI components: Map composer, Legend, Canvas, GUI, and plugins. The main GUI components are independent and do not directly provide GIS capabilities. QGIS raster providers, QGIS vector providers, and QGIS core are non-GUI components. The raster and vector providers function as an abstraction layer between the data and the application, thus allowing the use of a unique API for all data types and the QGIS core performs all the GIS work tasks[32].

2.6.1. QGIS plugins

QGIS plugins are modular components that can be added to the main program by the user depending on its necessities.This allows many new features/functions to be easily added. The plugins can be programmed using C++ or Python. Quick development and release cycle are possible since Python is an interpreted language and does not require compilation. Distributing a Python is as simple as uploading a zipped version to a central plugin repository via a web interface [49].

There are two groups of plugins:

- **Core Plugins** are maintained by the QGIS Development team and there is automatically part of every QGIS distribution.
- **External Plugins** are stored in external repositories and maintained by the individual authors. They can be added to QGIS through the Python plugin installer.

2.7. High Performance Computing

High Performance Computing (HPC) is the recently developed technology in the field of computer science, which evolved due to meet increasing demands for processing speed. HPC refers to the computing system, including several processors as part of a single machine or a cluster of several computers as an individual resource. HPC brings together several technologies such as computer architecture, algorithm, programs and system software under one canopy to solve complex problems, quickly and effectively. This technology focuses on developing and implementing methods like parallel processing, cluster processing and distributed processing for solving problems. Therefore the main methodology that is currently applied to HPC is parallel computing resources.

There are two models for task execution in HPC environments:

- **Single Instruction Multiple Data (SIMD)**: the same computing instructions and operations are executed across multiple processes at the same time.
- **Multiple Instruction Multiple Data (MIMD)**: multiple processors are used to asynchronously control multiple instructions, achieving space parallelism.

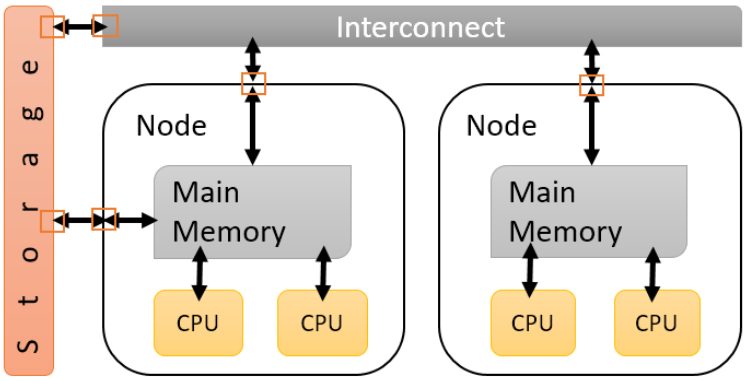

Figure 2.14.: Five major parts in a High Performance Computing system (adapted from [35]).

Five elements of a HPC system are illustrated in Figure 2.14, and the relation among: CPUs, memory, nodes, inter-node network and non-volatile storage (disks, tape) [35].

2.7.1. Parallel processing

Parallel processing is a computing approach which performs a simultaneous processing of the same task on two or more microprocessors in order to obtain faster results.
Parallel processing is particularly useful in projects that require complex computations. In image processing and computer vision algorithms, although task-level parallelism may exist, data-level parallelism can be most efficiently exploited for the following reasons. Firstly, comparing to task-level parallelism, data-level parallelism is present more often in image processing algorithms. Secondly, even if task-level parallelism exists, it only offers limited opportunities for parallelization (i.e. limited processing speed improvement). A significant increment of image resolution increases the computational requirements of

most image processing algorithms rapidly, which only can be compensated by exploiting data-level parallelism [101].

2.8. Image Processing

Image processing or digital image processing is a technique to improve image quality by applying mathematical operations on it, in order to get an enhanced image or to extract some useful information from it. Image processing deals with manipulation and analysis of images by using computer algorithms, such as to improve pictorial information contained in an image or perform image analysis to extract hidden information for better understanding and clarity. A digital image is a matrix, of square pixels (picture elements) arranged in columns and rows, as shown in Figure 2.15. Where each number represents the brightness at regularly spaced points or very small regions in an image. Mathematically, an image may be defined as a two dimensional function, $f(x, y)$, where x, y are spatial (plane) coordinates and the amplitude of f at any pair of coordinates (x, y) is called the intensity of the image at that point. A RGB image has 3 planes representing three primary colors namely Red, Green and Blue color as components of an image.

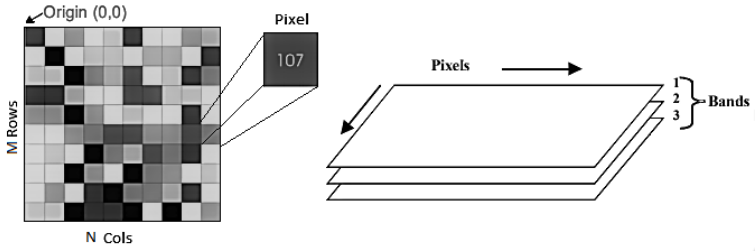

Figure 2.15.: An image is a matrix of pixels arranged in columns (N) and rows (M), each pixel has a value depends on the color depth of the image (K bands).

Image processing operations can be divided into two levels [57]:

- **Low level image processing:** contrast enhancement, noise reduction, and noise removal are a few examples of low-level image processing operators. Which can be classified as point operators, neighborhood operators, and global operators, with respect to the way that output pixels are determined from the input pixels.

- **High level image processing:** they are used to interpret the image content such as classification and object recognition. These operations work on graphs, lists and relations among regions/objects to derive some decision.

Figure 2.16 shows a generic block diagram of fundamental steps in image processing.

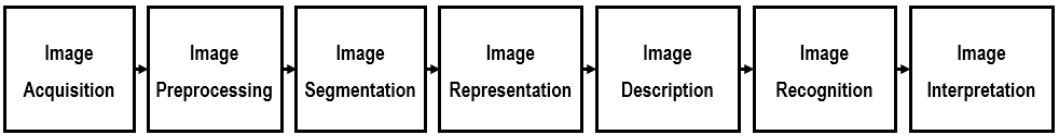

Figure 2.16.: Fundamental steps in image processing.

- **Image acquisition:** a digital image is produced by one or several image sensors.
- **Image preprocessing:** in order to increase the chances for success of the other processes an image has to be improve in this step.
- **Image segmentation:** is the process that subdivides an image into its constituent parts or objects.
- **Image representation:** to convert the input data into a form suitable for computer processing.
- **Image description:** to extract features that result in some quantitative information of interest or features that are basic for differentiating one class of objects from another.
- **Image recognition:** to assign a label to an object based on the information provided by its descriptors.
- **Image interpretation:** to assign meaning to an ensemble of recognized objects.

2.8.1. Image data structure

Computer images are comprised of a set of points or picture elements, usually referred to as pixels, stored as an array of numbers. Images are spatial data indexed by two spatial coordinates; typically the variables x and y refer to the horizontal and vertical axes of an image. Pixel value represents the color or the intensity at each pixel, and the placement of pixels within the matrix corresponds to their placement within the image. If more

than one value is required to encode pixel information, an image is often represented by a multidimensional matrix. For example, an RGB encoding of an image would contain 3 matrices: one each for red, green and blue intensity as shown in Figure 2.17. In other terms, each pixel represented in the matrix has a value that is encoded as either a scalar (in the case of gray-scale) or a vector (in the case of color). In Figure 2.17, the RGB value (0,255,0) produces a green pixel.

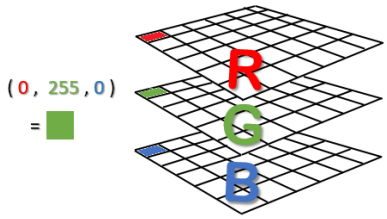

Figure 2.17.: RGB color (Green) image represented by three matrices.

2.8.2. Color space

Because manipulation of color is possible through additive properties (for example, red and blue produce purple) a wide range of colors is generated from a choice of three primary colors. A red, blue and green (RGB) color space is commonly used in modern displays like computer monitors, televisions and digital cameras. Each primary color has a range of values dependent on bit resolution.

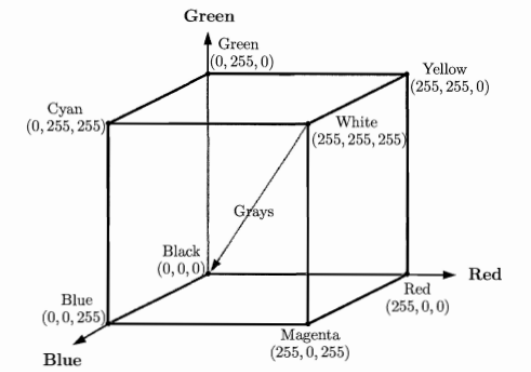

Figure 2.18.: RGB color space represented by a cube.

The RGB color space is represented via a three-dimensional cube as seen in Figure 2.18. The coordinates at each point inside the cube represent values of each primary color.

2.8.3. Spatial resolution

The detail discernible in an image is dependent on the **spatial resolution** of the sensor and refers to the size of the smallest possible feature that can be detected. Spatial resolution of passive sensors depends primarily on Instantaneous Field of View (**IFOV**). The IFOV is the angular cone of visibility of the sensor Figure 2.19 (A) and determines the area on the Earth's surface which is "seen" from a given altitude at one particular moment in time Figure 2.19 (B). The size of the area viewed is determined by multiplying the IFOV by the distance from the ground to the sensor Figure 2.19 (C). This area on the ground is called the resolution cell and determines a sensor's maximum spatial resolution.

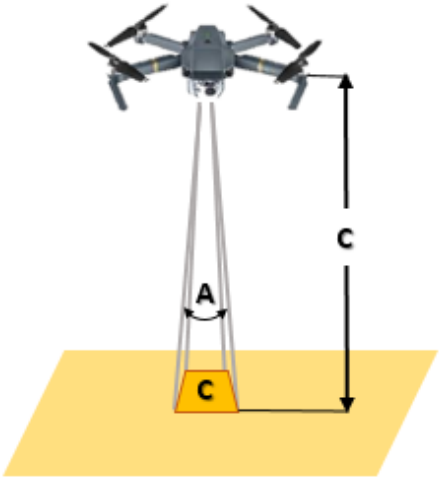

Figure 2.19.: Size of the smallest possible feature that can be detected by the sensor.

For a homogeneous feature to be detected, its size generally has to be equal to or larger than the **resolution cell**. If the feature is smaller than this, it may not be detectable as the average brightness of all features in that resolution cell will be recorded. Spatial resolution can be defined as the smallest discernible detail in an image. Or in other way we can define spatial resolution as the number of independent pixels values per inch. For example, Figure 2.20 shows the different image resolutions. Figure 2.20(a) is a 18 x 18 pixel image which shows only basic structure. The crop rows features are indecipherable and lineal objects identification of the image is impossible. Figure 2.20(b), a 62 x 62 pixel image, begins to show more detail, but identification of the crop rows is still difficult.

Figure 2.20(c) is a 626 x 626 pixel image with a much higher level of detail. While the image still suffers from pixelization, the crop row is identifiable and some details such as leafs are recognizable.

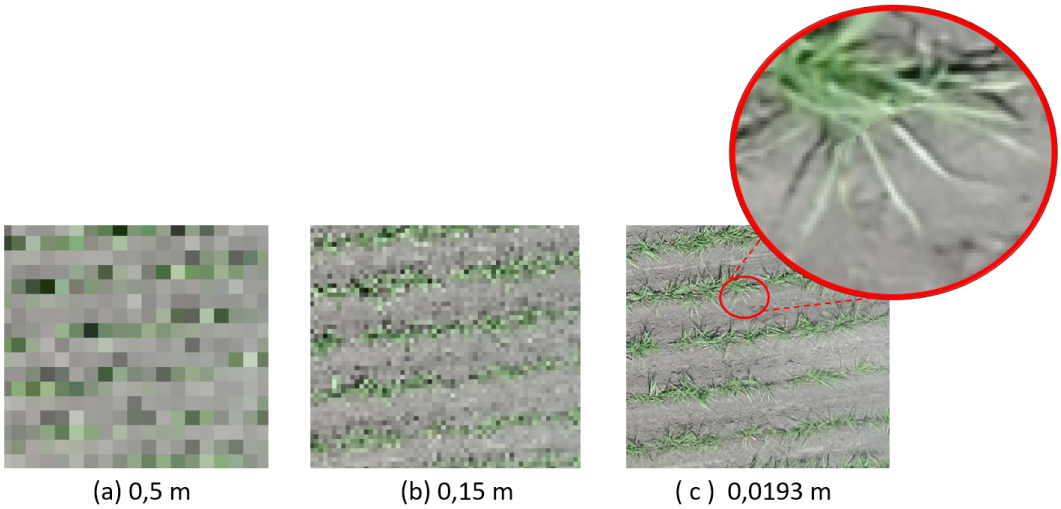

Figure 2.20.: Spatial resolution example in a crop rows image.

2.8.4. Image acquisition

In this step, the image is captured by a camera or sensor and digitized. An image has to converted to numerical form before processing. An image captured by a sensor is expressed as a continuous function $f(x, y)$ of two co-ordinates in the plane. Image digitization means that the function $f(x, y)$ is sampled into a matrix with $Mrows$ and $Ncolumns$.

2.8.5. Image preprocessing

The principal objective of the image preprocessing or enhancement is to process or manipulate an image in order to more suitable results than the original one for specific applications, a processed image is better viewed than the original image. During image preprocessing, there may be artifacts in images that should be corrected prior to feature measurement and analysis (sensor corrections, lighting corrections, noise, geometric corrections, color corrections, among others). Enhancements are used to optimize for specific feature measurement methods, rather than fix problems. Some image enhancements that

can be perform are (scale-space pyramids, illumination,blur and focus enhancements).

2.8.6. Image segmentation

Categories	Methods	Functional Properties
Region-based	Thresholding	Separate the foreground from background by assuming some threshold value.
	Texture	The texture of a region is used to group like textures into connected regions.
	Region operating	Divide the image according to given criteria based on gray scale, color, texture.
Edge-based	Histogram	Obtain histogram is calculated from the entire pixel in the image and detect edges.
	Intensity Gradient	
Specific theory-based	Fuzzy	Partition the image into some clusters depending upon color, intensity and location.
	Neural network	Realize small area of an image using an ANN and then decision-making mechanism marks the area in the image as per requirement.
Model-based		Knows the exact geometrical shape of and object in the image.

Table 2.1.: Classification of image segmentation methodologies (adapted from [52]).

Image segmentation subdivides an image into constituent regions or objects (sets of pixels, also known as super-pixels). Image segmentation is a classification problem and therefore is one of the most important and difficult tasks in image processing. Segmentation is the process of partitioning an image into mutually exclusive regions based on pixel characteristics. The goal of image segmentation is to assign each pixel to a group or class with similar characteristics. Ideally, classes with similar characteristics correspond to similar objects in an image. In many cases image segmentation remains an unsolved problem [105]. The simplest form of segmentation is binarization. Binarization is the process of segmenting an image into two classes, such as foreground and background, thus distinguishing the object(s) of interest from the rest of the image. The binary image is referred to as a mask when it is used to exclude pixel data from calculations. The segmentation objective is to simplify and change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation algorithms are generally based on one of the two basic properties of intensity value: discontinuity (isolated points,

lines and edges of the image) and similarity (thresholding, region growing, region splitting and merging) [103]. There are some main categories present for the classification methods for image segmentation, as shown in Table 2.1.

2.8.7. Image representation

The results of segmentation is a set of regions. Regions have then to be represented and described. There are two ways to represent a region: The external representation is used when the primary focus is on shape characteristics (its boundary). The internal representation is used when the primary focus is on regional properties such as color and textures (its internal pixels). In fact, sometimes it may be necessary to use both ways. Shape features are extracted from the contour only or are extracted from the whole shape region. Under each class, the different methods are further divided into structural approaches and global approaches. A whole hierarchy of the shape representation and description techniques is shown in Figure 2.21.

Texture features

Texture is a very useful characterization for a wide range of images. It is generally believed that human visual systems use texture for recognition and interpretation. In general, color is usually a pixel property while texture can only be measured from a group of pixels. A large number of techniques have been proposed to extract texture features [9]. Based on the domain from which the texture feature is extracted, they can be broadly classified into spatial texture feature extraction methods and spectral texture feature extraction methods [71].

Color features

Color is one of the most important features of images. Color features are defined subject to a particular color space or model. A number of color spaces have been used in literature, such as RGB, CMYK, HSV, YCrCb, LAB among others . Once the color space is specified, the color feature can be extracted from images or regions [71].

Shape features

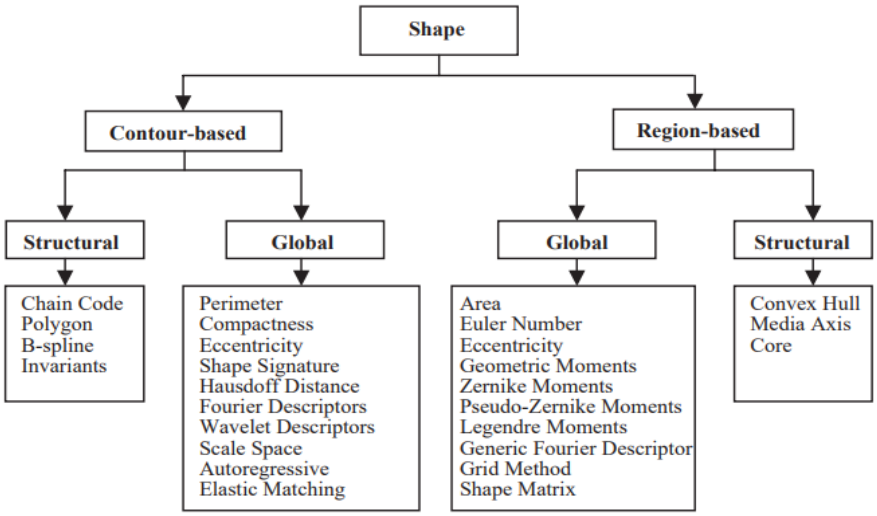

Figure 2.21.: Shape representation and description techniques (adapted from [106]).

2.8.8. Image description

Feature extraction describes the relevant shape information contained in a pattern such as a task of classifying the pattern is made easy by a formal procedure. The main goal of feature extraction is to obtain the most relevant information from the original data and represent that data in a lower dimensionality space. Feature extraction is an important step in the construction of any pattern classification and aims at the extraction of the relevant information that characterizes each class. In this process, relevant features are extracted from objects/ alphabets to form feature vectors. These feature vectors are then used by classifiers to recognize the input unit with a target output unit. Image description is a special form of dimensionality reduction.

2.8.9. Image recognition

It is the process that assigns a label to an object based on information provided by its descriptors. Classification is a usual process used to recognize the image. The descriptors from the image data stored in the database are compared with the descriptors from the

query image. The closer gap within those descriptors is then chosen to appoint the query image to be in which class. Artificial neural network (ANN) and fuzzy logic are the most common techniques used in classification.

2.8.10. Image interpretation

Interpretation is the processes of detection, identification, description, and assessment of signs of an object and pattern image. The method of interpretation may be either visual or digital or combination of both. Interpretation is used to assign meaning to an ensemble of recognized objects.

2.8.11. Vegetation Indices

Vegetation Indices (VIs) are quite simple and effective algorithms for quantitative and qualitative evaluations of vegetation cover, vigor, and growth dynamics, among other applications. There are a variety of vegetation indices that have been developed to help in the monitoring of vegetation. These indices have been widely implemented within remote sensing applications using different airborne and satellite platforms even with recent advances using drone technologies [67].

Name	**Equation**
Excess Green VI (**ExG**)	$2G - R - B$
Color index of vegetation (**CIVE**)	$0.441 * R - 0.881G + 0.385B + 18.78745$
Vegetetiven (**VEG**)	$G/R^a B^{1-a}$ with a=0.667
Excess green minus excess red (**ExGR**)	$ExG - 1.4R - G$
Combination (**COM**)	$0.25ExG + 0.3ExGR + 0.33CIVE + 0.12VEG$

Table 2.2.: Vegetation indices from the RGB spectral bands (adapted from [98])

Formulas to compute various VIs from the visible spectral bands RGB are related in Table 2.2.

2.9. Software engineering

2.9.1. Software development process model

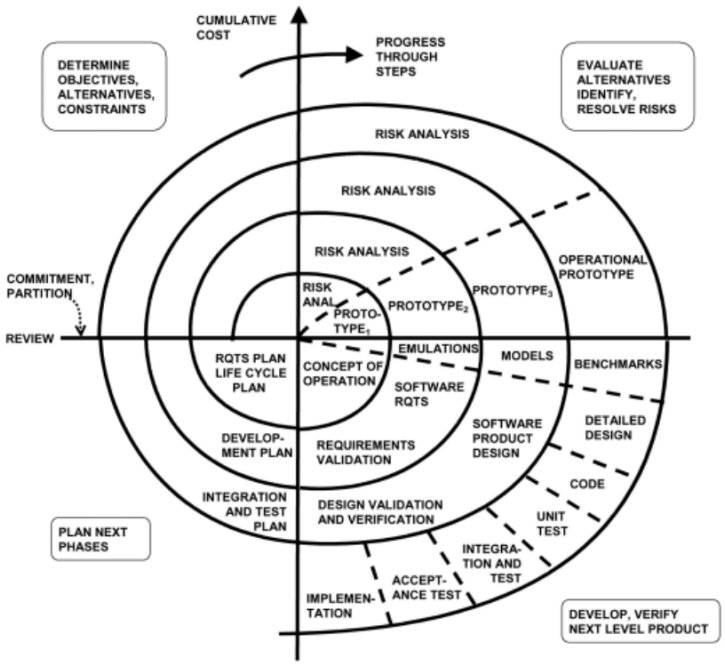

Figure 2.22.: The Spiral process (retrieved from [11])

The spiral model is an evolutionary software development life cycle model that aims to combine the sequential approach of waterfall with prototyping from iterative models [72]. The aim is to develop software through iterations that start small and simple and grow more complex with each new cycle. The spiral model aims to take a risk-driven approach to software development instead of a documentation or code-driven approach [11]. The model can be divided into four quadrants that each iteration cycle passes through:

- Defining objectives, alternatives, and constraints
- Analyzing and solving risks
- Developing and testing product
- Spiral Planning next iteration

The phases and detailed activities within them are shown in Figure 2.22.

3. Related work

This section provides related work of the crop row detection and mapping which aims to analyze the main characteristics of publications on the subject and discuss trends for future research directions.

3.1. Protocol

The methodology adopted in this section consists of six steps:

1. Select databases.
2. Search and filter the results.
3. Read the abstract to confirm the relevance of the documents.
4. Filter papers according to eligibility criteria.
5. Grouping papers following categorization criteria.
6. Review the relevance of documents, contributions, conclusions and verify if there are open questions.

3.1.1. Databases

The following databases, Scopus [84], Web of Science [102], Google Scholar [37], Scielo International [82], and Science Direct [83] were used to perform the searches.

3.1.2. Filters

Results were filtered using terms such as Satellite, UAV, drones, RPAS, crop furrows, crop rows, computer vision and precision agriculture, surco, surcado, respectively in their topic, title, abstract or keywords. The terms within each group were combined with 'OR', and three groups were linked with 'AND'. Time restriction filters were not used. Relevant studies were identified based on a combination of a group of terms.

3.1.3. Retrieved publications

After filtering results obtained during full searches, thirty-five publication entries were returned that had to be reviewed for quality. Publication entries retrieved are shown in Table 3.1.

Publication Type	Reference	Total
Books	-	0
Articles/Journals	[19], [23], [25], [40], [77], [43], [50], [55], [73], [86], [21], [7], [16], [22], [26], [28], [39], [41], [42], [47], [53], [59], [60], [68], [79], [85], [88], [91], [93], [97], [107], [108], [109], [87]	34
Letters,notes and patents	-	0
Books chapters	-	0
Thesis	[4]	1
Total documents	-	35

Table 3.1.: Shows the full results after applying step 3 (relevance of the papers).

3.1.4. Questions

- What type of aerial platform was used in the study?
- What kind of methodologies for the detection of crop rows were used?
- What kind of images were used in the study?
- Are the methodologies used in other crops can be extrapolated to sugarcane?

3.1.5. Eligibility criteria

The platform from where the images were acquired to develop the article or research was the main selection criterion. In this case, platforms such as tractors, robotic implements, and manual methods were excluded. Only aerial platforms such as drones, planes or satellites were included. Also, articles published before the year 2000 were excluded. Table 3.2, shows the publications and the reason for article selection.

Total retrieved		**34**
Reason	**Publication**	**Total**
Image processing through HPC	-	0
Row detection	[77], [40], [21], [43], [23], [25], [50], [55], [86], [73], [4], [87]	12
Row mapping	[23], [25]	2
Case of use	-	0
Methodology proposal	[50], [86], [77], [43], [21], [40], [25]	7
Application	[25]	1
Sugar cane crop	-	0
Other crops	[77], [40], [21], [43], [23], [25], [50], [55], [86], [73], [4], [87]	12
Discarded	[19], [7], [16], [22], [26], [28], [39], [41], [42], [47], [53], [59], [60], [68], [79], [85], [88], [91], [93], [97], [107], [108], [109]	23
Total retained	[77], [40], [21], [43], [23], [25], [50], [55], [86], [73], [4]	**12**

Table 3.2.: Publications that were included and excluded from the retrieved publications

After applying the eligibility criteria previously explained a group of publication entries was discarded. Table 3.2 shows zero articles where any related HPC approach in image processing is applied or discussed. Twelve articles, where described method crop rows detection, are obtained but only in two of them a crop rows mapping is related. In seven of them, authors propose and adjust methodologies. In no one of the articles found a sugarcane crop was used as a target crop. Twenty-three articles were discarded due to them did not pass the eligibility criteria established. Finally, according to the proposed methodology, twelve articles are relevant to review.

3.1.6. Categorization criteria

After applying the eligibility criteria, eleven relevant papers were selected. There are summarized in Table 3.3 by the following characterization.

- Type of input image.
- Year of publication.
- Purpose of the application.
- HPC approach or CV techniques.

3.2. Previous studies

All these works are supported by image processing using computer vision techniques widely addressed in different contexts. For all of the cases, inputs are always high-resolution raster images obtained from different aerial platforms through diverse on-board sensors. In Table 3.3 briefly shows each of the selected related articles.
In [87], plantation rows were identified on UAV imaged coffee crop fields. For the rows identification purposes, authors use images sensed by UAV mounted RBG camera from up to 100m above the plantation level. In that work, in order to make it feasible, authors explored the application of Hough Transform to extract plantation lines. The input image was first divided in a set of partially overlapped tiles. In doing so, each plantation line segment can be roughly approximated to a locally straight line. Very short lines which approximate quite precisely real plantation lines are obtained. These lines were considered and treated in this work as noise. Subsets of those lines present a high level of overlapping allowing subsequent partition and by means of interpolation. The authors highlight that each window is independent of one another making its proposal algorithm ideal to be implemented as a divide and conquer approach.

Input	Application	Techniques	Platform	Year	Reference
RGB Image	• Crop Lines Identification	• Image processing • K-Means clustering • Hough transform	UAV	2018	[87]
RGB + NIR Images	• Crop and weed classification	• Image processing • Geometric Features • Random forest • Hough transform	UAV	2017	[55]
RGB Image	• Plant definition and missing plant characterization in vineyards	• Image processing • Segmentation	UAV	2017	[73]
RGB Image	• Crop row detection	• Image processing • Unsupervised clustering • Hough Line Transform	UAV	2016	[77]
Very high resolution satellite images of WorldView-2	• Crop row detection	• Image processing • Line Segment Detector(LSD) algorithm	UAV	2016	[4]
RGB and Multispectal Images	• Weed Mapping	• Image processing • Hough transform	UAV	2015	[40]
RGB Image	• Vine parcel detection and boundaries extraction	• Image processing • Fast Fourier Transform	Ultra Light Aircraft	2015	[25]
Visual/near-infrared (VNIR)	• Vineyards water management	• Image processing • Unsupervised detection • Skeletonisation	UAV	2015	[21]
Pancromatic	• Crop rows orientation	• Image processing • Mathematical morphology	Satellite Formosat-2	2014	[86]
IKONOS and WorldView-1	• Automatic detection of field furrows	• Image processing • Hough transform • Fourier Analysis	Satelital	2013	[43]
Multispectral Image	• Crop row characterization for site-specific weed management	• Image processing • Object-based image analysis	UAV	2012	[50]
Very high-resolution image	• Vine field monitoring	• Image processing • Active Contour Network	Plane	2006	[23]

Table 3.3.: Document review - relevant papers filtered

In [55], a vegetation detection, plant-tailored feature extraction, and a classification to obtain an estimated distribution of crops and weeds on the field were performed in a group of fields of sugar beets crop. The principal contribution addressed is a vision-based classification system for identifying crops and weeds in both, RGB-only as well as RGB combined with near infra-red (NIR) imagery of agricultural fields captured by a UAV. The method used is capable of detecting plants in such images only based on their appearance and is be able to exploit the geometry of the plant arrangement without requiring a known spatial distribution to be specified explicitly. In order to normalize intensity values on a global scale and detect the vegetation in each image a preprocessing is applied. Using a combination of an object-based and a keypoint-based approach features only for regions and vegetation are extracted for exploitation of plant arrangement and for exploiting crop rows. A multi-class Random Forest classification is applying and obtain a probability distribution for the predicted class labels (crop/weed). Authors use the supposition that in most agricultural field environments, the plants are arranged in rows, which share a constant inter-row space. They employ the ***Hough Transform*** on the vegetation mask and analyzed the Hough space and perform the following three steps. a) Estimating the main direction of the crop rows. b) Estimating the crop rows as a set of parallel lines. c) Refitting the best line set to the data. The proposed method provides good results under challenging conditions such as overlapping plants and is capable for weed detection in intra-row space.

In a tomato crop field, a rows detection was achieved in [77], authors proposed a spectral-spatial method. Where a K-means unsupervised algorithm was used for spectral clustering together with a spatial method for mathematical morphology and geometric operations. ***Hough transform*** is used to detect lines. Several lines more than the number of rows is detected. The lines are aggregated using a authors technique which consists of taking the middle row with the pixels 240 and traversing each other to arrive at the number of rows. Authors results could be extended to detect rows in a vast stretch of crops by analyzing mosaicked images of larger crops due to its simplicity in the implementation and reported

results obtained.

In [4], a linear feature extraction(edge detection) for boundary delineation and crop row detection was performed. Linear features are extracted from remote sensing imagery using the ***LSD algorithm*** considering intensity differences between neighboring pixels. The LSD algorithm starts by calculating level-line angle for each pixel based on pixel gradient.In this research, parameter tuning was performed by varying one parameter and keeping the others fixed. Level-line angles are used to form line-support regions. Crop rows detection is based on dominant orientation of crop rows was derived from the orientation of the automatically detected lines (or specifically from the gradient orientation of pixels). The difference in row spacing between the reference and the automatically detected datasets ranged from 0.03 m to 0.93 m. One possible cause for this wide range could be the distribution of rows. Though most crop rows in a subset followed similar patterns and assumed to have the same spacing, there were some irregularities on the row spacing. Assuming a fixed distance is a problem, usually the work of furrowing in the field is realized with tractors and its trajectory may vary a bit due to the conditions of the terrain. Authors propose that these results could be improved by combining the procedure with other techniques present in literature such as ***Fast Fourier Transform***.

In [40], authors proposed a system for weed mapping in sunflower crops. The system uses the imagery of the field provided by the UAV and relies on very little information provided by the user: a set of labeled patches for each class and the set of parameters for the different algorithms. For the experimental stage, aerial images were taken at 30, 60 and 100m. The methodology is based on the following steps: partition of the experimental field into different subimages, computation of vegetation indexes and binarisation of these, detection of crop rows and, finally, training of a classification model for the data provided. The detection of crop rows is based on the ***Hough transform***. Additionally, a comparison of three machine learning paradigms (unsupervised, semi-supervised and supervised learning) was performed to study the influence of labeled and unlabeled

examples in the quality of the obtained classifiers.

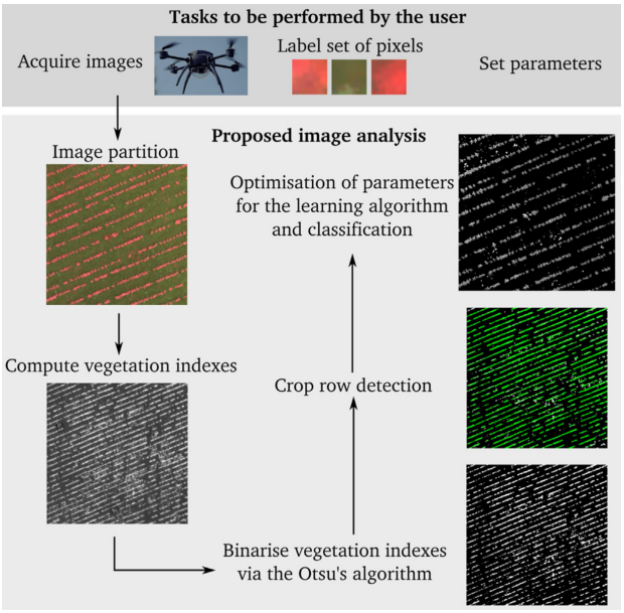

Figure 3.1.: Image analysis steps proposed in paper [40].

As shown in Figure 3.1, this proposal is potentially parallelizable by using of an HPC approach and gives the possibility of exploring different machine learning methods and techniques in order to robustness the crop row detection over different field conditions such as sunny days, shading days, the presence of post-harvest residues and different soils conditions. In [40], authors let some open questions and suggestions for future work in order to improve the work developed. Firstly, the paradigm of object-based image analysis could be explored for image segmentation. Another point that could be studied in future work is the use of big data or large-scale techniques for the processing step in such a way that each model is not only trained in a sub-image but also complemented with the whole image information.

In [25], authors develop a comprehensive and automatic tool for vineyard detection, delineation and characterization using aerial images of very high spatial resolution. This tool has been implemented in a completely automatic way and exports results into GIS format (.shp) with an associated database containing characteristics such as area, perimeter, inter-row width, row orientation, missing vine plants rate and cultural practices. For

vine row extraction, Authors takes advantage of the crop patterns periodicity, applied a ***Fourier Transform*** to characterize already delineated vine blocks on 25 cm resolution images. This approach also enables the accurate estimation of inter-row width and row orientation, which can be used to easily extract and characterize each vine row, contrary to the complex and time-consuming classical methods. The article is not totally clear, due to authors generalize the description of the method which is relatively long, and they have chosen to present only the best results, obtained with the red channel of the image.

In [21], an automated algorithm that uses ***skeletonisation*** techniques to reduce the complexity of agricultural scenes into a collection of skeletal descriptors is described. The algorithm has been applied to a high-resolution aerial orthomosaic and has proven its efficiency in unsupervised detection and delineation of vine rows in a commercial vineyard. The image processing algorithm, consists of three main steps; 1) histogram filtering, 2) skeletonisation and 3) vine row identification. The proposed method reduces the complexity of agricultural scenes into a collection of skeletal descriptors that enable the application of geometric and spatial constrains to identify vine rows accurately.

In [86], authors present a technique developed for the retrieval of the orientation of crop rows. High spatial resolution satellite images acquired by Formosat-2 in panchromatíc and multispectral modes is used for the proposed technique. A method developed is based using directional filters and operators taken from binary mathematical morphology. The technique proposed by the authors could be extrapolated to another kind of images, due to, the use of a succession of morphological operations (erosion. geodesic reconstruction. opening and skeletoning). Due to operations that are performed its implementation should be simple.

In [43], authors proposed an algorithm which allows estimating, at the field scale, of the main furrow directions. This algorithm uses a segmentation of a whole image in order to label each agricultural field individually. Then, for each field, mathematical morpho-

logical operations are applied in order to enhance image contrast. Then, an automatic thresholding process is applied to create a binary image of furrows or other features in the field. In order to reduce the noise due to the overlap between the classes of furrow and background, furrows are straight lines exhibiting regular patterns is introduced as apriori information. The ***Hough transform*** is applied to detect straight lines and furrow directions as accumulations on the angle axis in the Hough space. A ***Fourier Transform*** is applied on the binary image of the furrows in order get the maximum contrast enhancement.

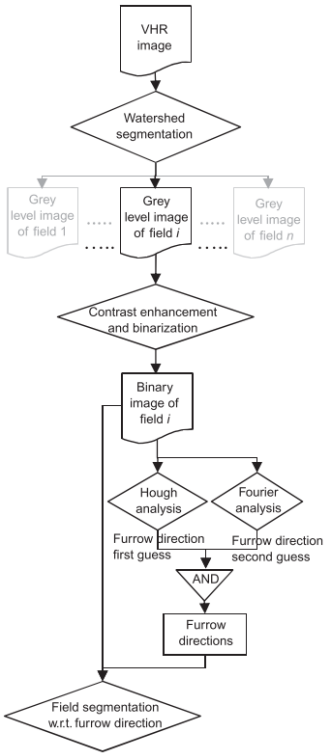

Figure 3.2.: Flow chart for estimate furrows direction proposed in paper [43].

An ***object-based image analysis*** (OBIA) procedure described in [50], identifies and classify the crop rows within a maize crop-field. This procedure was developed by combining several scene, contextual, hierarchical and object-based features in a looping structure (Figure 3.3). The procedure integrates several features from the crop-field patterns: 1) field structure, such as field limits and row length, 2) crop patterns, such as row orientation and inter-row distance, and 3) plant (crop and weeds) characteristics, such as spectral

properties and plant dimensions; as well as 4) hierarchical relation based on different segmentation scales, and 5) neighboring relationships based on distance, position, and the angle between objects. The OBIA procedure for the identification and classification of crop rows was developed by using the commercial software eCognition Developer 8 [96], therefore an implementation in other contexts could be limited.

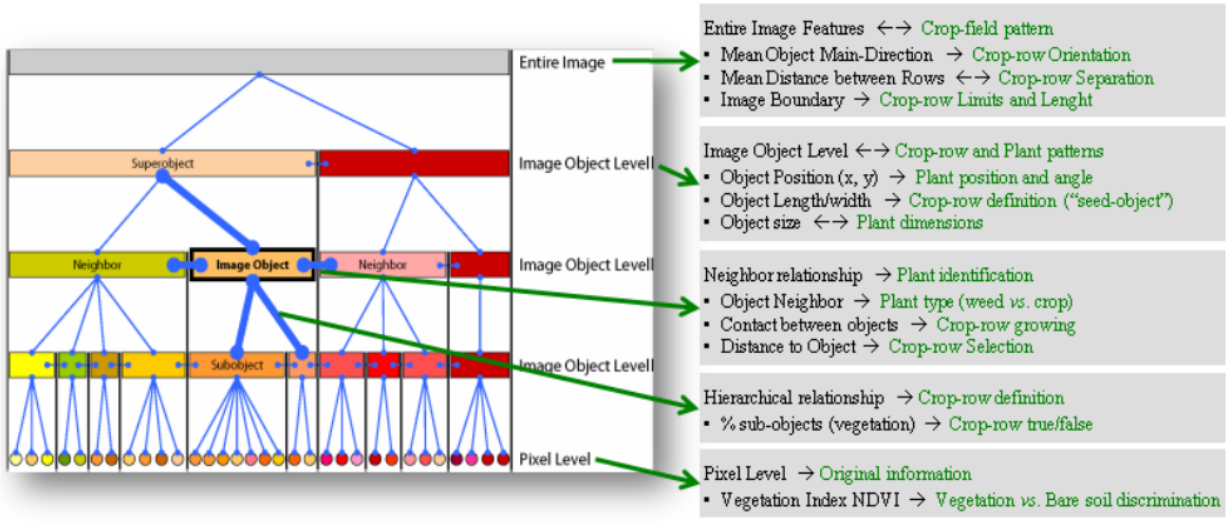

Figure 3.3.: Field and crop features involved in rule-set algorithm developed in paper [50].

Described in [23], a framework for crop rows segmentation and characterization in remote sensing images has been developed and validated for vineyard monitoring. This framework operates on any high-resolution remote sensing images since it is mainly based on the geometric information. The framework consists in several steps. First, the segmentation step allows the delineation of the parcel under consideration. A region-growing algorithm, based on the textural properties of row crops, was developed for this purpose. Once the parcel under consideration is delineated, a boundary smoothing process is applied and the row detection process begins. Row detection operates by means of an active contour model based on a network of parallel lines. In order to retrieve the rows with high accuracy, authors implement an **active contour network**, which aims at fitting a line to each row through a global convergence process. The process consists in minimizing an appropriate energy function in order to converge quickly towards the optimal line network. In this framework, the segmentation step requires very low input from the end-user who just needs

to draw a small rectangle inside the field which wants to survey. Authors leave open a prospective work concerns the reduction of the processing time for the segmentation step and the implementation of a new network model considering row curvature in order to strengthen slope variation robustness.

3.3. Final Remarks

Several strategies have been proposed for crop row detection. They share similar characteristics in various stages of image processing using methods based on the Hough transformation, blob analysis, and frequency analysis. Image processing techniques are used to enhance agricultural image content and improve accuracy and consistency of processes while reducing manual and hard job processing. Different aerial platforms and sensors are used for crop row detection, crop row mapping or related applications. Based on the reviewed and synthesized articles, listed in Table 3.3, sugarcane crop was not found. None of the reviewed articles released a piece of software that can be used for crop row detection, and all remain as academic exercises. In terms of the validation of obtained results, the authors used only a visual inspection. Moreover, no none constructs or uses metrics to validate the accuracy and precision of crop row lines. In this regards, a ground truth validation is essential in order to contrast the manual with an automatic generated crop rows. Although in Table 3.3, platforms such as satellite images were considered in the review, with these kinds of images a crop row mapping is not accurate due to pixel size and the platform limitations. The results of our work intents to be useful for high-precision guidance in agriculture were tractor systems have achieved steering accuracy of less than 2.5 cm. An automatic crop row identification and mapping in different field conditions could be feasible by getting high-resolution images at low altitude in an aerial platform and using a combination of several of the revised proposals and assembling a new one to sugarcane field conditions. In order to handle high-resolution images a divide and conquer approach by parallelizing some process may be used.

4. Crop Rows Generator

This section describes the process, conducted during the research project for **Design and build a method for generating georeferenced sugarcane crop rows using high-resolution aerial images**. The protocol included as an Appendix A was followed for acquiring aerial images using (RGB) cameras on-board Drone platforms over sugarcane fields.

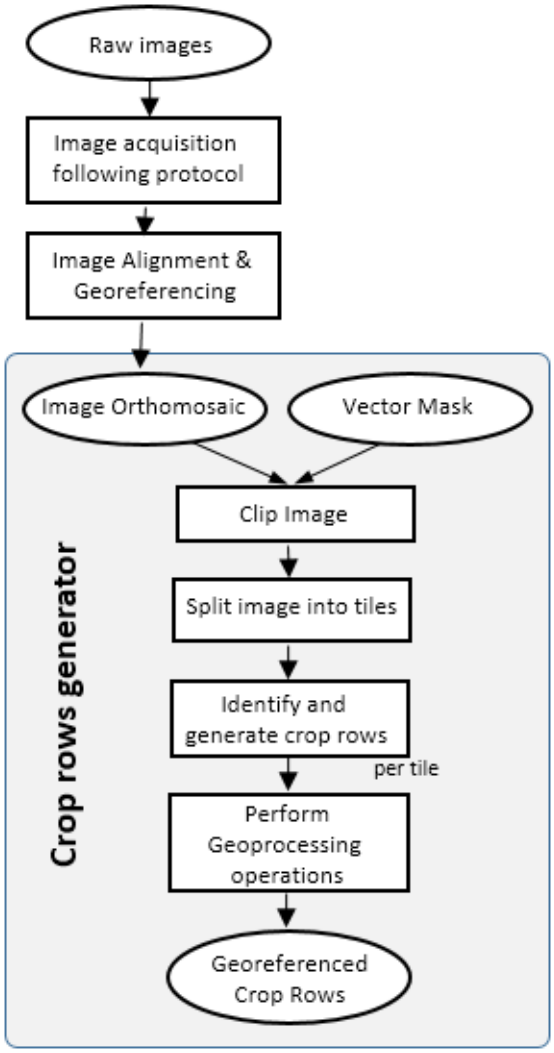

Figure 4.1.: High-level diagram for crop rows generation process

In Figure 4.1 is shown highlighted a high-level flow diagram of the developed process for the crop rows generator. The input data of the process are georeferenced images and outputs are vectorial georeferenced crop rows lines. Input images shall be stitched into a single orthomosaic Raster image using specialized software for that purpose.

4.1. High Resolution Georeferenced Image Mosaic

An orthomosaic is the result of the processing of many individual images through a mosaicing process. Orthomosaic is composed of image preprocessing, image registration and image fusion [48].

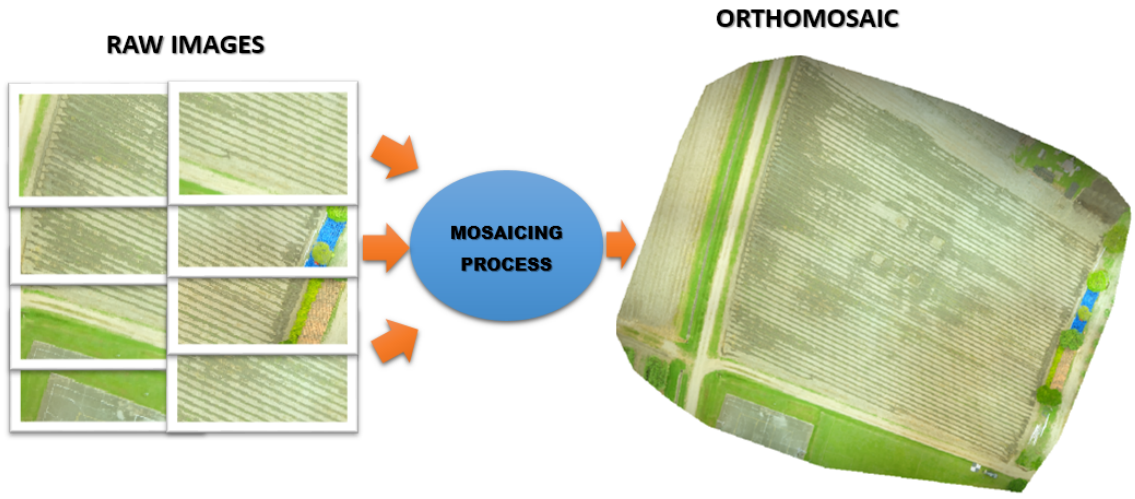

Figure 4.2.: Orthomosaic file generated by mosaicing process

Parameters referenced in the protocol such as drone flight height, crop field area, image capture resolution, sensor characteristics among others they affect directly an orthomosaic file size. Orthomosaic file size resulting could be a few hundred megabytes (MB) or even gigabytes (GB) of data. It means that processing this data can take a few minutes or even hours.

4.1.1. Orthomosaic characteristics

GeoTIFF format

GeoTIFF is a standard open file format which is widely used as an interchange format for georeferenced raster imagery. GeoTIFF [36] format uses numerical codes to describe coordinate systems, projection types, datums, ellipsoids, among others.

```
C:\ >gdalinfo testimage1.tif
Driver: GTiff/GeoTIFF
Files: testimage1.tif
Size is 17992, 23231
Coordinate System is:
PROJCS["WGS 84 / UTM zone 18N",
    GEOGCS["WGS 84",
        DATUM["WGS_1984",
            SPHEROID["WGS 84",6378137,298.257223563,
                AUTHORITY["EPSG","7030"]],
            AUTHORITY["EPSG","6326"]],
        PRIMEM["Greenwich",0],
        UNIT["degree",0.0174532925199433],
        AUTHORITY["EPSG","4326"]],
    PROJECTION["Transverse_Mercator"],
    PARAMETER["latitude_of_origin",0],
    PARAMETER["central_meridian",-75],
    PARAMETER["scale_factor",0.9996],
    PARAMETER["false_easting",500000],
    PARAMETER["false_northing",0],
    UNIT["metre",1,
        AUTHORITY["EPSG","9001"]],
    AUTHORITY["EPSG","32618"]]
Origin = (358624.438760828290000,376855.999522842180000)
Pixel Size = (0.018439902478200,-0.018440140341400)
Metadata:
  AREA_OR_POINT=Area
Image Structure Metadata:
  INTERLEAVE=PIXEL
Corner Coordinates:
Upper Left  (  358624.439,  376856.000) ( 76d16'21.50"W,  3d24'31.14"N)
Lower Left  (  358624.439,  376427.617) ( 76d16'21.48"W,  3d24'17.19"N)
Upper Right (  358956.209,  376856.000) ( 76d16'10.75"W,  3d24'31.15"N)
Lower Right (  358956.209,  376427.617) ( 76d16'10.73"W,  3d24'17.20"N)
Center      (  358790.324,  376641.808) ( 76d16'16.11"W,  3d24'24.17"N)
Band 1 Block=17992x1 Type=Byte, ColorInterp=Red
  Mask Flags: PER_DATASET ALPHA
Band 2 Block=17992x1 Type=Byte, ColorInterp=Green
  Mask Flags: PER_DATASET ALPHA
Band 3 Block=17992x1 Type=Byte, ColorInterp=Blue
Filesize = 1.672.074.844 testimage1.tif
```

Figure 4.3.: Example results of gdalinfo command for GeoTIFF image

gdalinfo command [33] is used for raster data to get information about a spatial dataset, including spatial reference information (e.g., the projection), a summary of grid cell values. Figure 4.3 shows the result of gdalinfo command execution. As an example, a georeferenced RGB image(tesfile1.tif) is 1.6 GB file size and corresponds to sugarcane field about of 7.8 ha. with 417.972.152 pixels to process per band, the pixel size is 0.018 meters.

4.2. High-Performance Computing Approach

Multiprocessing is a way to improve the performance. In parallel computing, many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved in parallel [45].

Parallelization strategy for performance

- Break problem down into tiles.
- Execute tile operations in parallel.
- Reassemble output of tiles into a result.

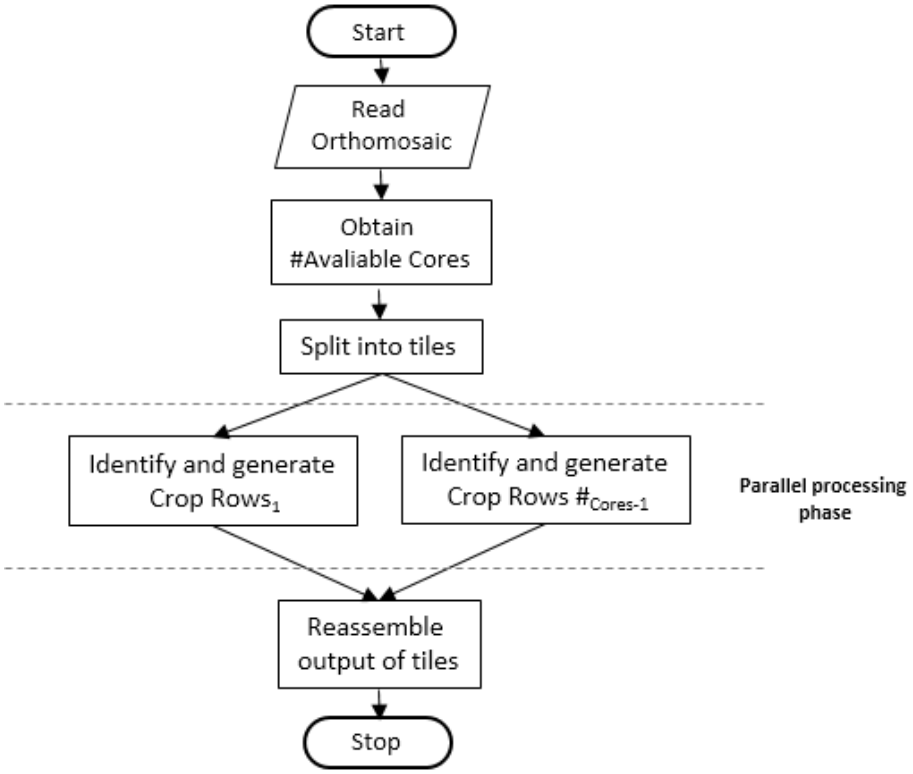

Figure 4.4.: Parallelization strategy overview workflow

A parallel processing environment can process tasks faster when programs are designed to use parallel processing. CRG uses a divide and conquers strategy in order to improve the image data processing time. The original orthomosaic is dividing into a set of adjacent tiles of the same size. An overview of the parallelization strategy is shown in Figure 4.4.

4.2.1. Split orthomosaic image into single tiles

The proposed approach exploits a decomposition technique of the image domain into tiles. The solution can be computed by applying image processing on each tile in a parallel way and combining the partial results corresponding to the different blocks of variables through a proper interconnection rule.

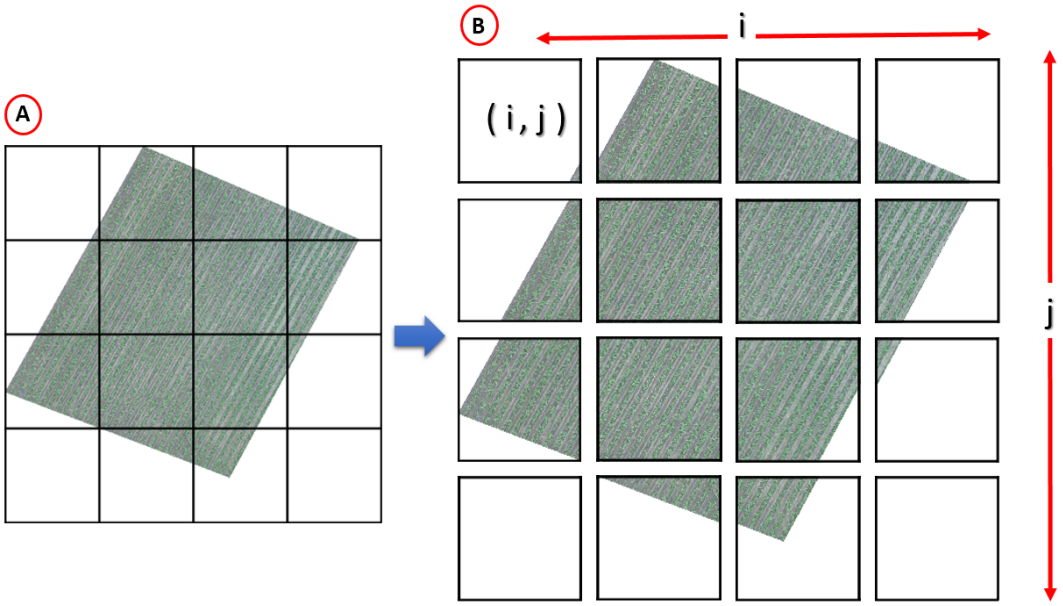

Figure 4.5.: Break orthomosaic down into separate georeferenced tiles

Figure 4.5 (A) corresponds to the original image (orthomosaic). Figure 4.5 (B) corresponds to the image tiles generated in the process. Each tile can be located by an i, j index. Each tile is stored in an individual raster file in a compressed image format JPG. Georeferenced information of each tile is stored in a world file. The world file describes the height and the width represented by each cell/pixel and the coordinate position of the top left cell of the image data. The filename extension of the world file is based on the raster file's extension (e.g jpg) with an append letter "w" at the end of the raster filename: **tile-1-1.jpg** − >**tile-1-1.jpgw**. A graphical view of a world files parameters and computed values of the four first upper left tiles of an image are represented in Figure 4.6.

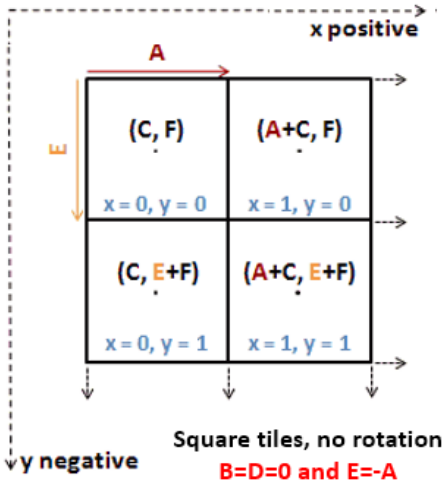

Figure 4.6.: Graphical view of world files parameters.

The world file gives the information with six numeric parameters, each in its own line. The meaning of the six parameters in a world file are [30]:

- Line 1: **A**: pixel size in the x-direction in map units/pixel
- Line 2: **D**: rotation about y-axis
- Line 3: **B**: rotation about x-axis
- Line 4: **E**: pixel size in the y-direction in map units
- Line 5: **C**: x-coordinate of the center of the upper left pixel
- Line 6: **F**: y-coordinate of the center of the upper left pixel

Image origin is located at the upper-left corner, whereas the origin of the map coordinate system is located at the lower-left corner. Row values in an image increase from the origin downward, while y-coordinate values in the map increase from the origin upward.

```
0.01856
0.0
0.0
-0.01856
353487.7774311414
378995.839178871
```

Code block 4.1: World file contents

4.2.2. Tile size selection

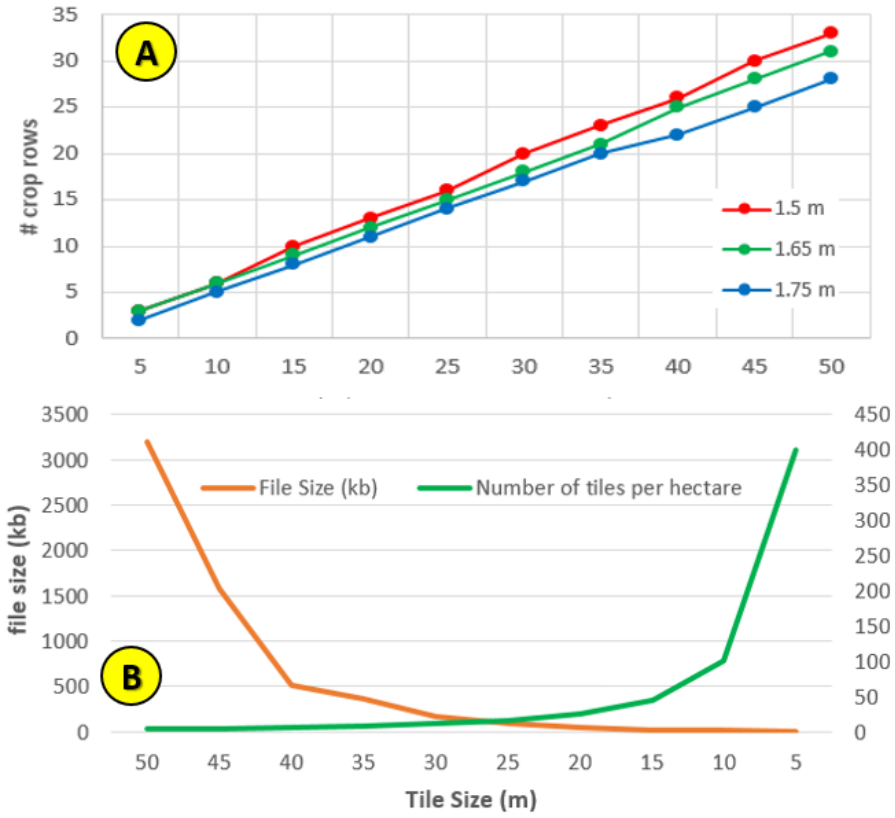

Figure 4.7.: Graph of crop rows per tile, tiles per hectare and file size per tile

Figure 4.7 (A) shows a graph of crop rows per tile for three distances of crop rows spacing also in Figure 4.7 (B) a graph of a number of tiles per hectare and the average file size of the tile is shown.

- Large tiles covers a major area, although, the disk storage size of each tile is higher.
- Getting many tiles means less disk storage for each tile, but more tile images to be processed.
- Number of possible crop rows to be detected is linear and depends on the size of the tile and it is independent of the crop rows spacing.

The size of the tile was defined as a functional requirement for the application as an optional parameter at a user needs. The size of the tile is 20 meters by default.

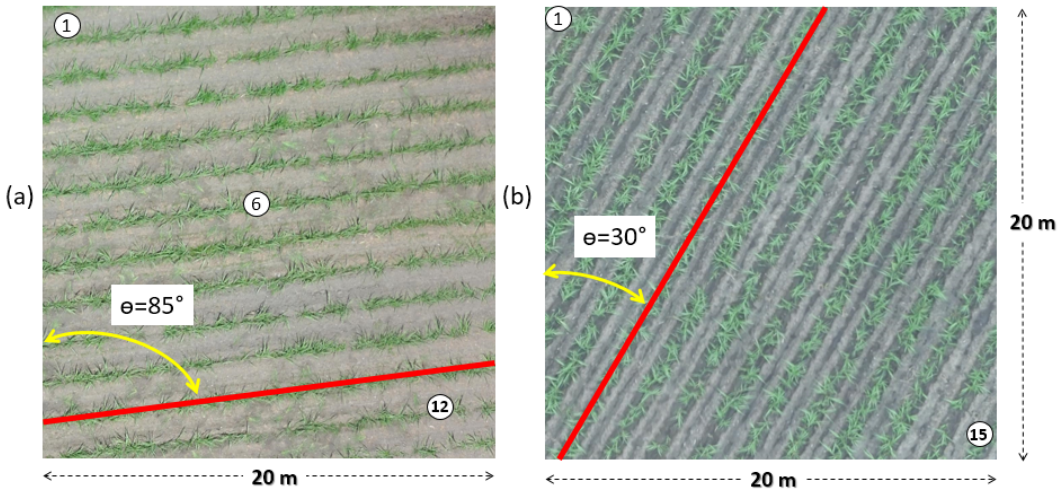

Figure 4.8.: Two georeferenced square tiles of 20 m with crop rows in different orientations

In a 20 meters x 20 meters tile, twelve crop rows in the sugarcane field are clearly identified in Figure 4.8 (a) where crop rows are oriented N 85° E. Notice that the number of crop rows can also vary depending on the row orientation and row spacing at the field as seen in Figure 4.8 (b) where crop rows are oriented N 30° E.

Python multiprocessing module

Python has a multiprocessing [75] module built into the language. This package contains multiple strategies to improve speed by running functions in parallel. The multiprocessing library uses separate memory space, multiple CPU cores. Python provides a Pool class for multiprocessing. Pool class allows doing multiple jobs per process. Pool in multiprocessing was used to parallelize tasks. The pool will distribute those tasks to the worker processes(usually same in number as available cores) and collects the return values in the form of a list and pass it to the parent process. When a Pool object is created, a new Python run-time is started for each core. Once the pool is allocated a group of worker threads that can be processed in parallel. There are multiple places in the code where the multiprocessing package is used and a Pool object instantiated and every function that requires it creates a Pool object as needed and then joins and terminates before exiting. Code block 4.2 is a portion of source code were a parallelized strategy is implemented for cutting the raster orthomosaic image in small pieces (tiles).

```
processingCores = (multiprocessing.cpu_count())
crutils.printLogMsg("Number of CPU cores: %s" % (str(processingCores)) )
processingPool = multiprocessing.Pool(processes=(processingCores))
results = [processingPool.apply(self.cropTileImageWorker,
           args = (i,j,pixelTileSize,processingImageFile))
           for i in range(nCols) for j in range(nRows)]
processingPool.close()
processingPool.join()
crutils.printLogMsg("All Crop Tile Image Jobs Finished" )

def cropTileImageWorker(self,iCol,jRow,pixelTileSize,processingImageFile):
    name = multiprocessing.current_process().name
    crutils.printLogMsg('Process name: %s' % (name))
    crutils.printLogMsg('Parent processs: %s' % (str( os.getppid() )) )
    crutils.printLogMsg('Process id: %s' % (str( os.getpid() )) )
    x2 = 0 + (int(iCol) * pixelTileSize)
    y2 = 0 + (int(jRow) * pixelTileSize)
    x1 = x2 - pixelTile
    y1 = y2 - pixelTile
    # create a box to crop the input orthomosaic raster image
    boxToCrop = (x1,y1,x2,y2)
    rasterImage = Image.open(processingImageFile)
    rasterImage.load()
    rasterCroppedRegion = rasterImage.crop(boxToCrop)
    rasterImage.close()
```

Code block 4.2: Pool object with multiprocessing library in python

multiprocessing.cpu_count() returns the number of available virtual or physical CPUs on the running system. A processing pool contains worker processes ***cropTileImage-Worker***. Each Process instance has a name with a default value. processingPool.close() is called when the parallelizable part of the program is finished. Then the worker processes will terminate when all work already assigned has completed. processingPool.join() wait for the worker processes to terminate. cropTileImageWorker implements image a crop method for cropping the input image in small tiles defined by a bounding box.

4.3. Identify and Generate Crop Rows

Computer vision techniques and a GIS approach using geoprocessing strategies are used to carry out the crop rows generation tasks.

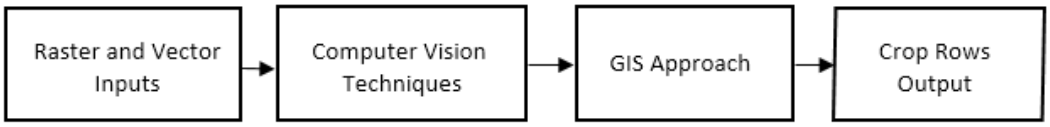

Figure 4.9.: Block diagram of crop rows identification and generation

In order to identify and generate georeferenced crop rows through computer vision techniques and a GIS approach, a block diagram in Figure 4.9 shows how are combined together. All the conducted process performed are described in this section.

4.3.1. Computer vision techniques

Characteristics of the processing tile

The image contained in the tile shall be present the characteristics listed below in order to be processed.

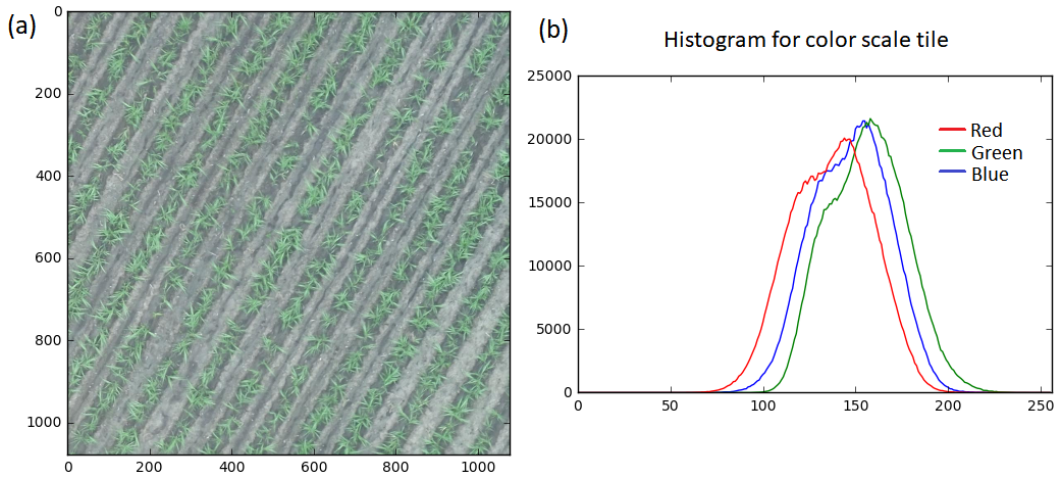

Figure 4.10.: RGB image (a) and histograms (b) for each color component (R, G, and B) of the tile image

- Sugarcane crop rows have to exist.
- Crop rows have to be parallel.
- Uniformity in the crop row.
- There are enough plants to configure a crop row shape.
- Crop rows have be oriented in the same direction.

Avoid tiles with the following characteristics

With the condition that an image capture protocol (included as an Appendix A) is followed, all these cases, listed below, shall not be presented.

- Tiles with blurry images.
- No crop rows in tile.
- Crowding crop rows.
- High water content in the field.
- Non-parallel crop rows.
- Crop rows oriented in different direction.
- High content of post-harvest residues.
- High weed content.
- Crop in inappropriate age to be identificated.
- Orthomosaic not well generated.

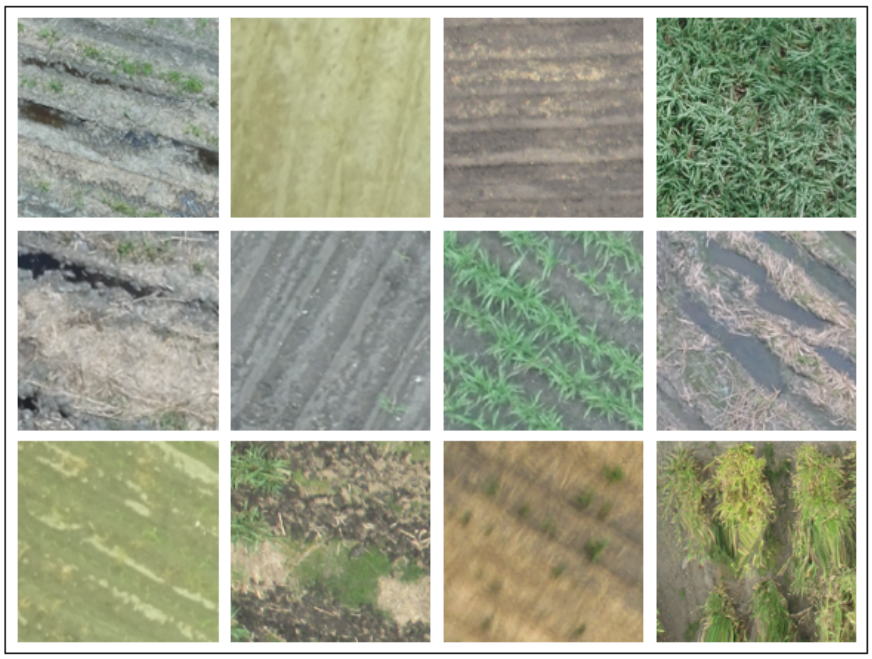

Figure 4.11.: Image tiles in not suitable conditions to be processed

In Figure 4.11, some of the tile image cases to be avoided are shown. Sometimes in a crop field, more than one of these cases may occur. In the case that one of these images is present in the tile, the processing algorithm is not able to identify the crop rows.

Tile Preprocessing

The goal of tile preprocessing is to improve a image quality in order to enhance the perception of crop rows in an image (Figure 4.12(a)). Image preprocessing involves image enhancement and noise removal.

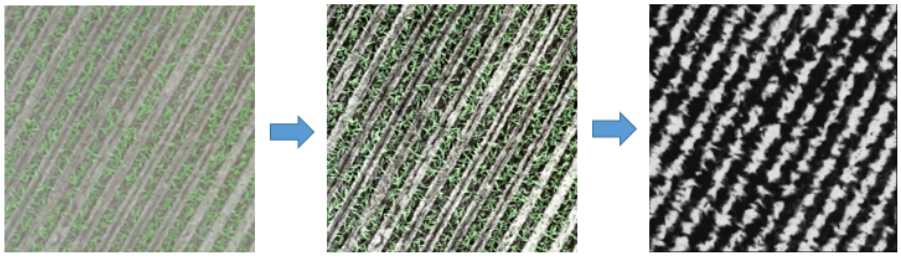

Figure 4.12.: Image preprocessing steps for each georeferenced tile

Image enhancement includes contrast enhancement, intensity manipulation and sharpening, and filtering. An adaptive median filter was selected for performs spatial processing to preserve detail and smooth non-impulsive noise. Adaptive median filters adapt to the properties of an image locally and selectively removes noise from an image while preserving edges. Thus, the adaptive median filter seeks to preserve detail while smoothing noise. Adaptive median filtering is suitable for enhancing crop rows images. An enhanced RGB image (Figure 4.12(b)) is remapped to grayscale by calculating the excess green vegetation index (ExG) where is highlighting green regions of interest (crop rows). In Figure 4.12(c) background is darkened and the crop rows are highlighted.

$$ExG(x,y) = (2 * G'(x,y) - R'(x,y) - B'(x,y)), \tag{4.1}$$

where:

ExG = Excess Green vegetation index,

$R' = \frac{R}{R+G+B}$: Normalized Red channel,

$G' = \frac{G}{R+G+B}$: Normalized Green channel,

$B' = \frac{B}{R+G+B}$: Normalized Blue channel.

Among several existing indexes [79] [6] [81] [92], ExG is widely used in problems within the precision agriculture domain [58] [8] [56] [104] [60] [55] [42] [63] [12] [22].

Tile segmentation

Segmentation process for each tile is follow through in two steps:

- Reduction of a gray level image to a binary image.
- Removing the imperfections of a binary image.

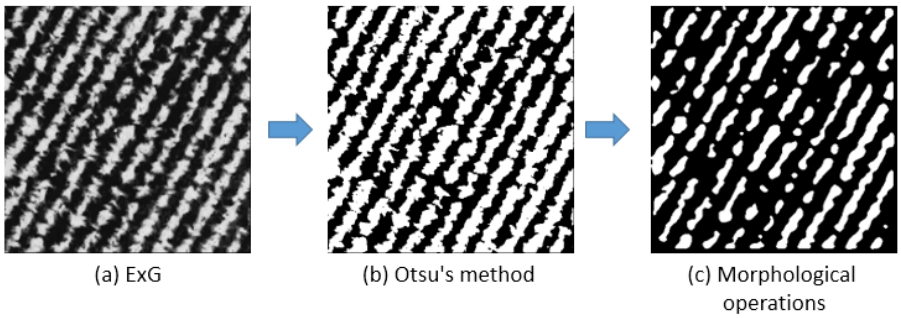

Figure 4.13.: Image segmentation steps for each georeferenced tile

The Otsu's threshold method separates green pixels (sugarcane plants and weeds) from the rest (soil, residues, stones, and others). It automatically calculates a threshold value from image histogram for a bimodal image [13]. *ExG* image contains two classes of pixels (foreground pixels and background pixels). Otsu's is used to automatically perform clustering-based image thresholding and calculating the optimum threshold separating the two classes [66]. Morphological operations such as dilation and erosion are used in combination to remove the imperfections in the structure of a binary image. In Figure 4.14 operations such as hole filling, extraction of connected components, thinning and thickening are performed.

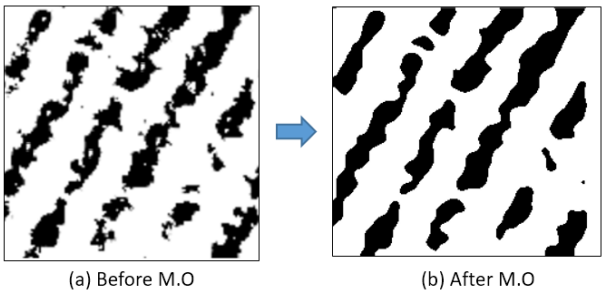

Figure 4.14.: Morphological operations (M.O) applied

Build contours

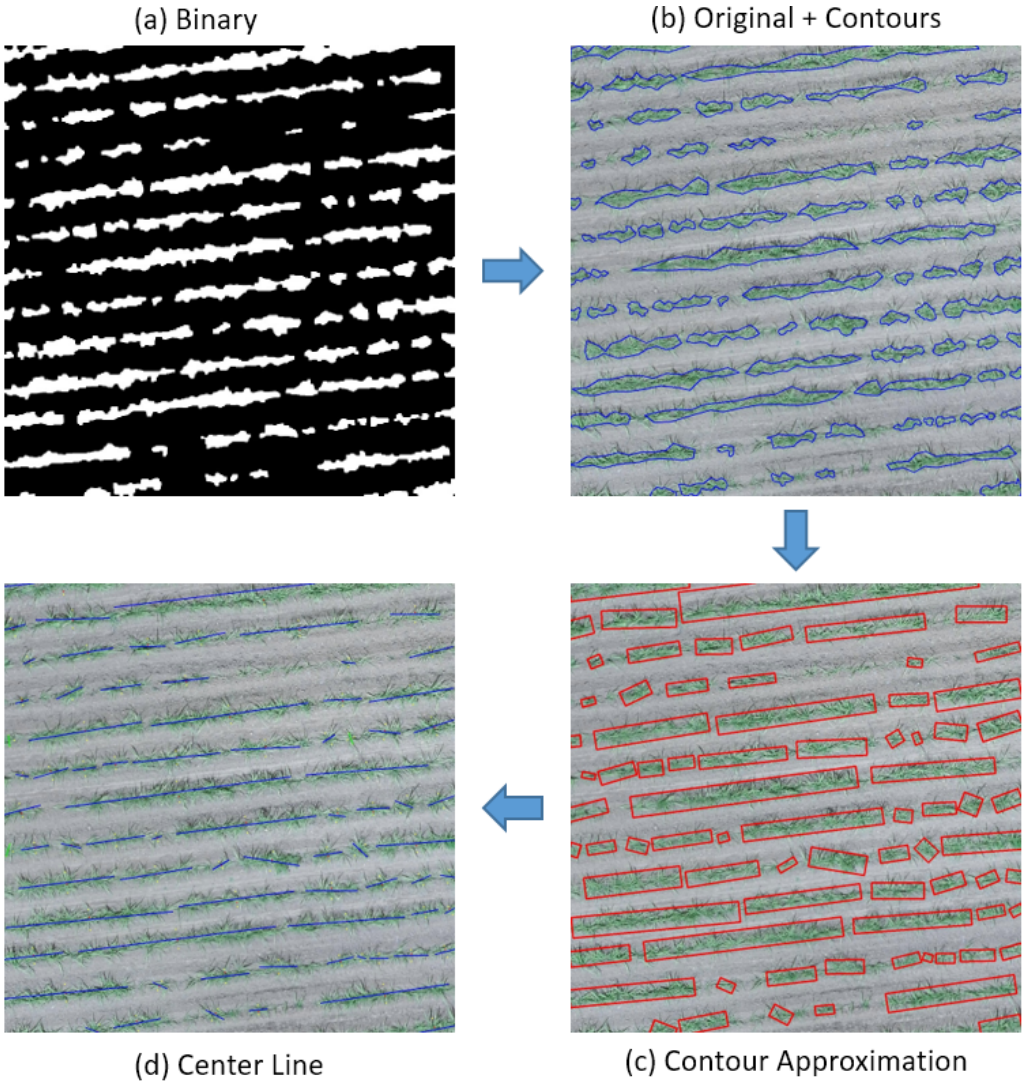

Figure 4.15.: Steps to find contours on each georeferenced tile

A binary (crop/soil) image is obtained once contours are performed in Figure 4.15 (a) result is shown. Contours are closed curves obtained from edges of the crop rows. In Figure 4.15 (b) shows contours found, overlapped with the original image in order to get the clearest visualization. A bounding rectangle is obtained with the minimum area of the contour. In Figure 4.15 (c) rotated rectangles are the result of the previous step. In Figure 4.16 (a) shown an example of a rotated rectangle which contains the following details: center x, y, width, height, an angle of rotation.

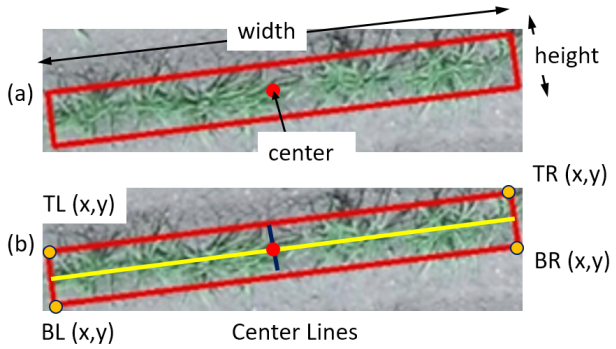

Figure 4.16.: Zoom to a rotated rectangle

To obtain centerlines coordinates in Figure 4.16 (b). The longest line is constructed for each rotated rectangle from the midpoint between coordinates of (Top Left (TL) and Bottom Left (BL)) and (Top Right (TR) and Bottom Right (BR)) in the same way, the shortest line is constructed for each rotated rectangle from the mid point between coordinates of (Top Left (TL) and Top Right (TR)) and (Bottom Left (BL) and Bottom Right (BR)). The midpoint between two points is calculated with the equation 4.2.

$$(x, y)midpoint = (\frac{x1 + x2}{2}, \frac{y1 + y2}{2}). \tag{4.2}$$

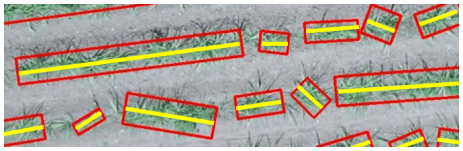

Figure 4.17.: Orientation of the longest centerlines

For each oriented rectangle, the longest line generated is selected. Lines information for each tile is stored to perform later statistical calculations. Figure 4.17 shows the orientation of the lines, they can vary by a few degrees, but some lines with high variations are presented inside the tile.

Crop rows orientation in tile

In order to find the average orientation of all the crop rows on a tile, center lines are used to perform a statistical search.

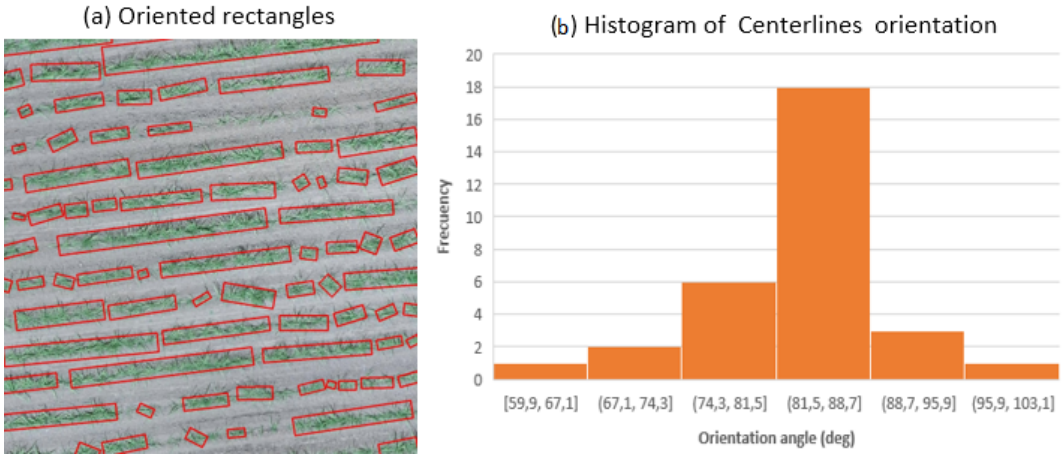

Figure 4.18.: Histogram of lines orientation

The histogram, in Figure 4.18 (b), represents the frequency distributions of the detected crop rows orientations in a single tile.

The "seed" parameter

A parameter called "***seed***" is introduced with the aim of synthesizing the search of the average angle inside a tile. The seed is provided by a user and it corresponds to one of four possible orientation patterns.

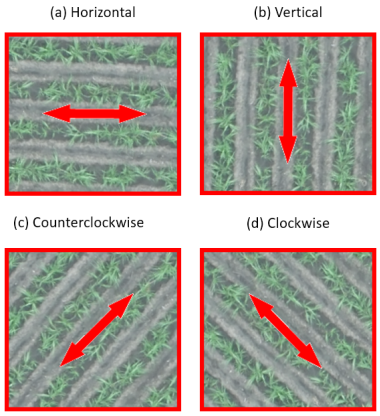

Figure 4.19.: Crop rows orientation patterns

In general crop rows are oriented in the patterns shown in Figure 4.19, orientation is given from the North and varies depending on the configuration of the crop field. Figure 4.19 shows (a) Horizontal pattern, (b) Vertical pattern, (c) Counterclockwise pattern and

(d) Clockwise pattern. A user has to provide the most similar pattern to the crop rows direction information contained in an image. For each tile, a global crop rows orientation is calculated. Crop rows orientation angle in a tile is given in decimal degrees.

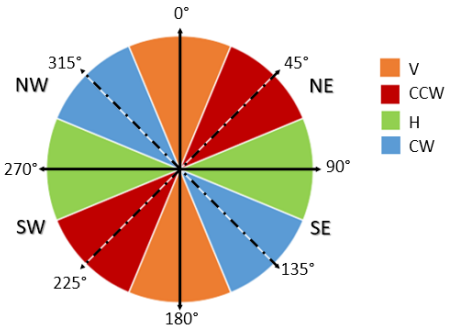

Figure 4.20.: For each "seed" value an angle range is valid

Average crop rows orientation angle value for each tile is processed statistically, outliers are removed. Standard deviation has to be less than three degrees. Figure 4.20 illustrates the angle ranges in which outlier filtering is performed, e.g a seed value corresponding to Counterclockwise(CCW), all lines with an angle out of (22.5 ° to 67.5 °) or (202.5 ° to 247.5 °) are discarded. For each tile, statistics such as the maximum, minimum, average and standard deviation of the orientation of lines are calculated and stored.

Global crop rows orientation

Once angle statistics per tile has been calculated, the global orientation of the whole image is calculated using equation 4.3. Without exception, tile orientation with a standard deviation lower than three degrees is computed. Selected tiles are designated as filtered tiles.

$$A = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + \cdots + a_n}{n}, \tag{4.3}$$

where:

A = Global orientation for all crop rows in image,

a = Average crop rows orientation value in the tile,

n = Number of filtered tiles.

Oriented centerlines

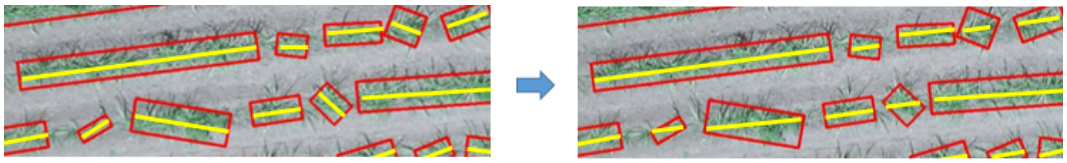

Figure 4.21.: Centerlines with a global orientation

As seen in the Figure 4.21 (b) new oriented centerlines are generated with the global angle found. For each of the tiles, oriented centerline generation process is performed. The midpoint(cx,cy) of the centerlines is used as an anchor point to rotate centerlines given the global angle. A new point (x',y') can be obtained by rotating an existing point (x,y) θ radians around the point (cx, cy) using:

$$\begin{aligned} x' &= ((x - cx) * cos(\theta) + (y - cy) * sin(\theta)) + cx, \\ y' &= (-(x - cx) * sin(\theta) + (y - cy) * cos(\theta)) + cy. \end{aligned} \tag{4.4}$$

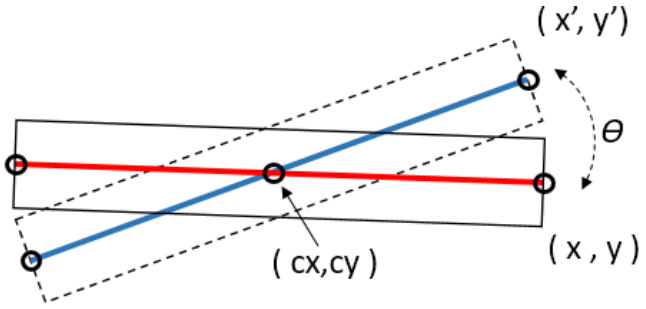

Figure 4.22.: Rotate centerlines around their midpoint

The coordinates of the starting point and end the point of centerlines are calculated per line within a tile.

A bottleneck arises, with oriented centerlines generated for each one of the tiles (Figure 4.23). How is possible to stick each one of the generated lines with lines in the neighbor's tiles? A GIS approach was developed to perform the proceeding.

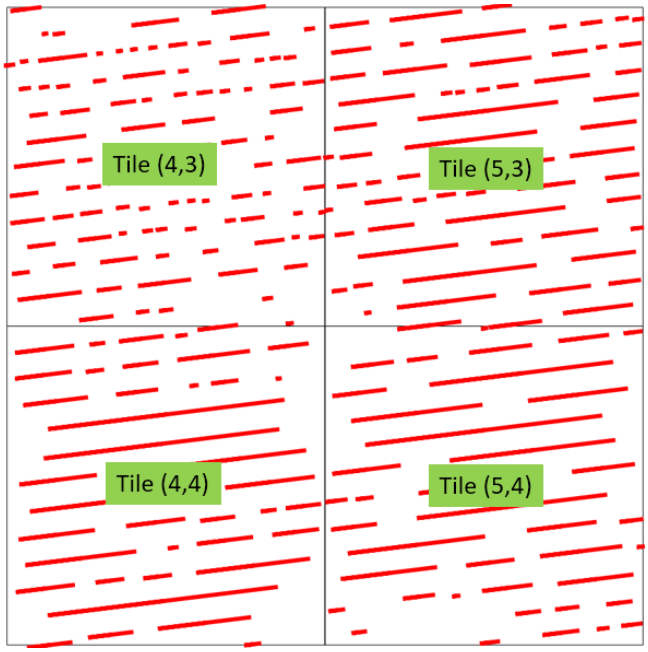

Figure 4.23.: Four tiles with calculated oriented centerlines

Different strategies can be used to perform "stick the lines together" task and keep just the crop rows. However, a GIS oriented strategy is proposed since the center lines can be represented as geographic objects and through geographic data treatment and geo-processing operations is possible to stick each one of the generated lines with lines in the neighbor's tiles. Lines in each tile are individual objects that have intrinsic characteristics such as orientation and length.

4.3.2. GIS approach

Centerlines are encoded to a standarized GeoJSON [44] format which is widely supported by geospatial libraries and frameworks. Spatial operations are performed by each of the tiles and the results are then later joined in a single result. Centerlines are represented by a *LineString* geometry type. Coordinate reference system of centerlines is the "*crs*" object. A centerline of the tile col: 2 row: 3, encoded in GeoJSON format, is shown in Figure 4.24. Geometry bounding boxes (BBox Tile) for each tile is also stored in GeoJSON.


```
{
  "type": "FeatureCollection",
  "crs": {
    "type": "name",
    "properties": {
      "name": "urn:ogc:def:crs:EPSG::32618"
    }
  },
  "features": [
    {
      "type": "Feature",
      "properties": {
        "len": 144.21656885420393,
        "col": 2,
        "row": 3,
        "generator": "Crop Rows Generator CLI"
      },
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            353537.39956,
            378931.475488
          ],
          [
            353537.728385,
            378932.058287
          ]
        ]
      }
    }
  ]
}
```


Figure 4.24.: A centerline encoded in GeoJSON format

Bounding Box (BBox) also known as the Minimum Bounding Rectangle (MBR) or Envelope, BBox is specified by two coordinates: xmin, ymin and xmax, ymax. BBx is a rectangle, oriented to the x and y axes.

Geoprocessing operations

Once the centerlines contained in each tile have been encoded to GeoJSON the following geoprocessing tasks (Figure 4.25) are performed.

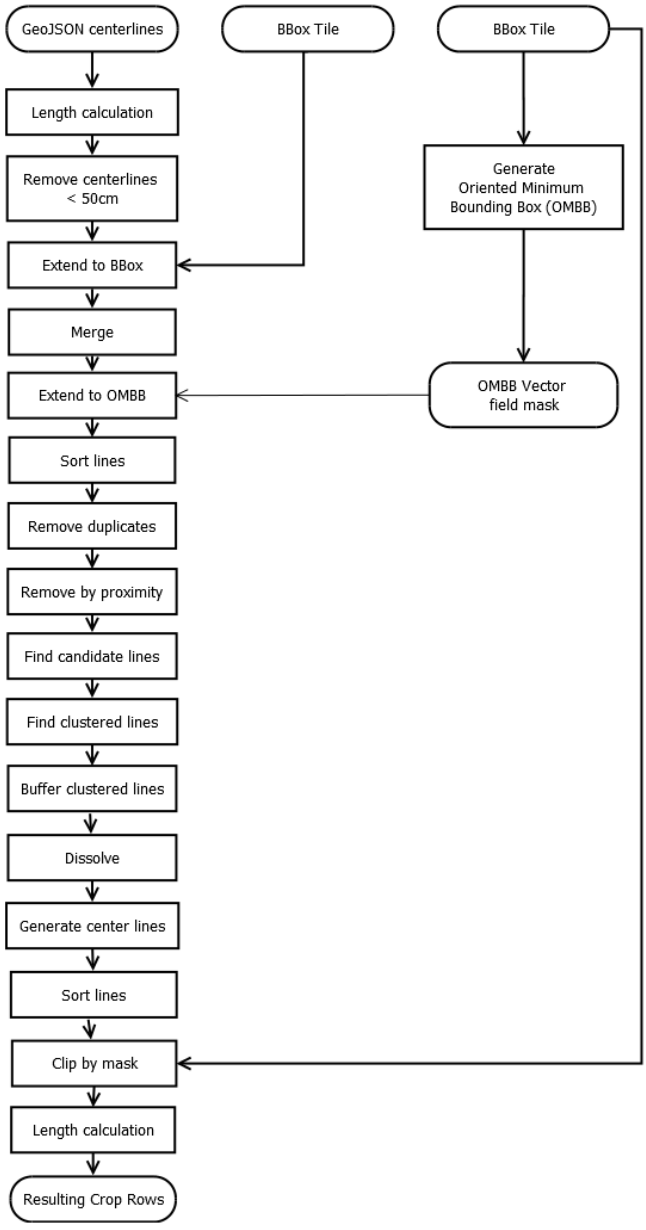

Figure 4.25.: Geo-processing operations for crop rows generation

GeoPandas[34] library was used to perform geometry operations and geoprocessing tasks. Geoprocessing operations are described throughout this section.

Length calculation

Length in meter units is calculated for each line segment in a tile. A first filter is performed: Lines with a length less than 50 cm are removed from all tiles.

Extend to BBox

For a tile, a line is extended to the bounding box. In Figure 4.26 is shown the result of centerlines extended to BBox.

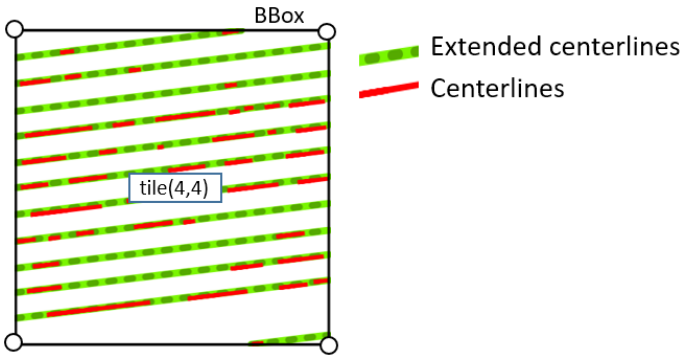

Figure 4.26.: Example of centerlines in tile (4,4) extended to BBox

Merge all lines

The aim of the merge extended lines process is to combine multiple extended tiles into a single, new output dataset. Merge lines are performed by a parallelized process, as well.

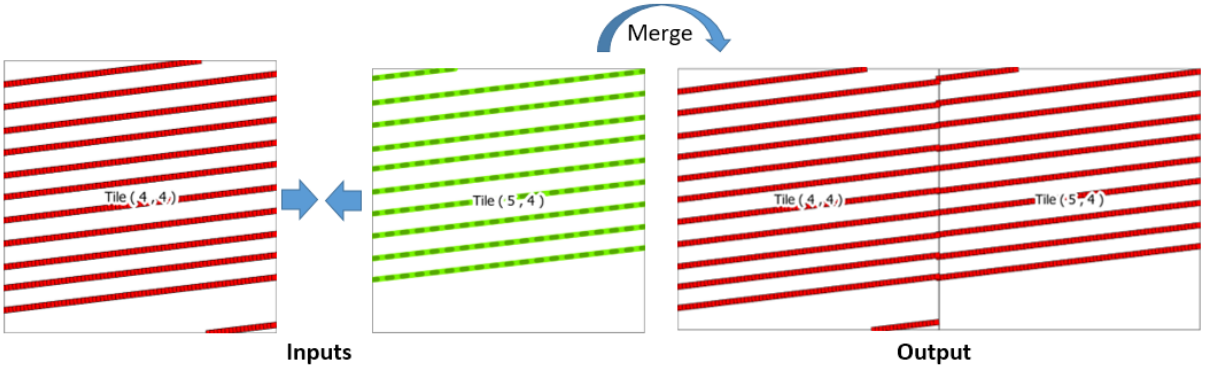

Figure 4.27.: Merge process combines multiple input datasets into a single

Extend to Oriented Minimum Bounding Box (OMBB)

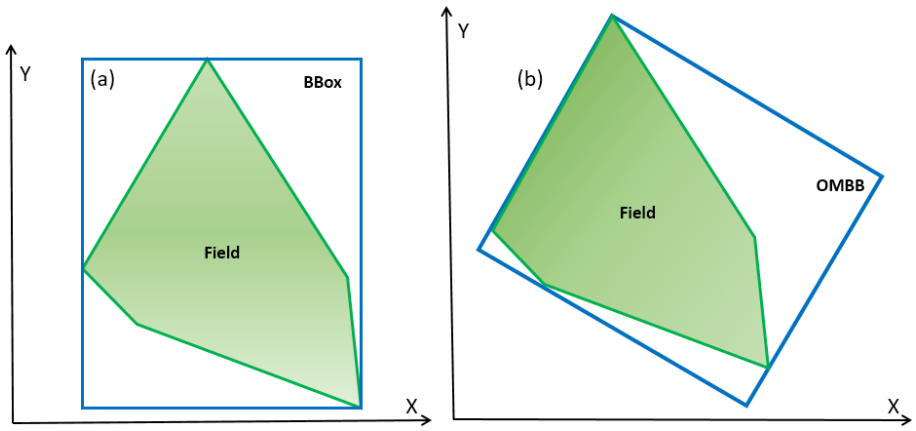

Figure 4.28.: Differences between a BBox (a) and OMBB (b) of an arbitrary polygon in two dimensions

The centerlines are extended to the Oriented Minimum Bounding Box (OMBB) polygon in order to normalize them to the same length (Figure 4.29). Python code [17] is used for OMBB implementation in the current project.

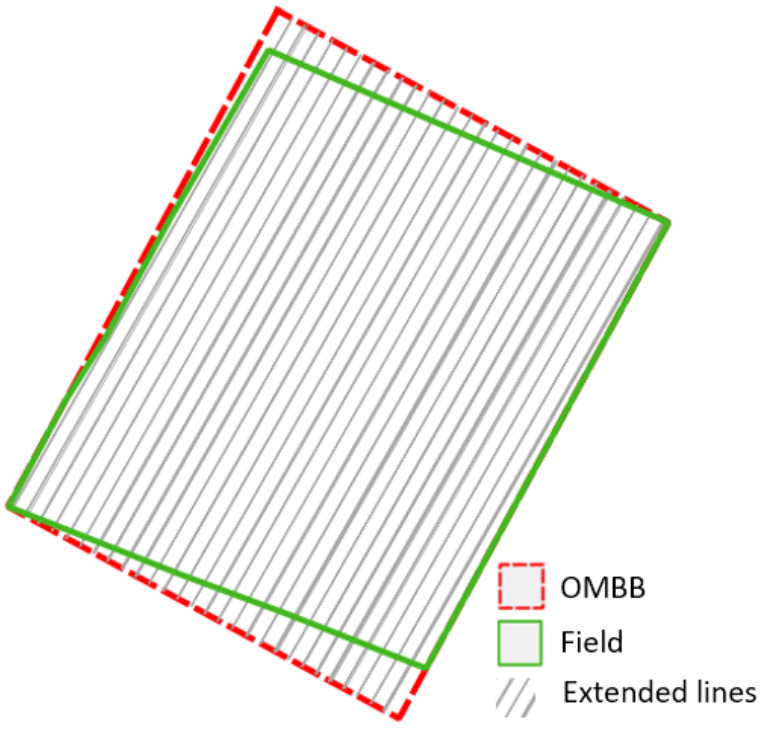

Figure 4.29.: Extended centerlines to OMBB

Operations with extended lines

A set of operations are performed on the resulting lines with the aim of reducing the number of possible not valid lines.

- Remove repeated lines
- Remove closest lines
- Remove multi-lines

The extended lines are ordered by geographic order, where the line number one corresponds to the first line found (reference line), which is referred to a built parallel line with the same orientation angle at a point of arbitrary external origin. The reference line is used as a reference axis to calculate distances to each line.

Find candidate lines

All the lines that reached this point are "candidate lines" and they are susceptible to be a line corresponding to the crop row.

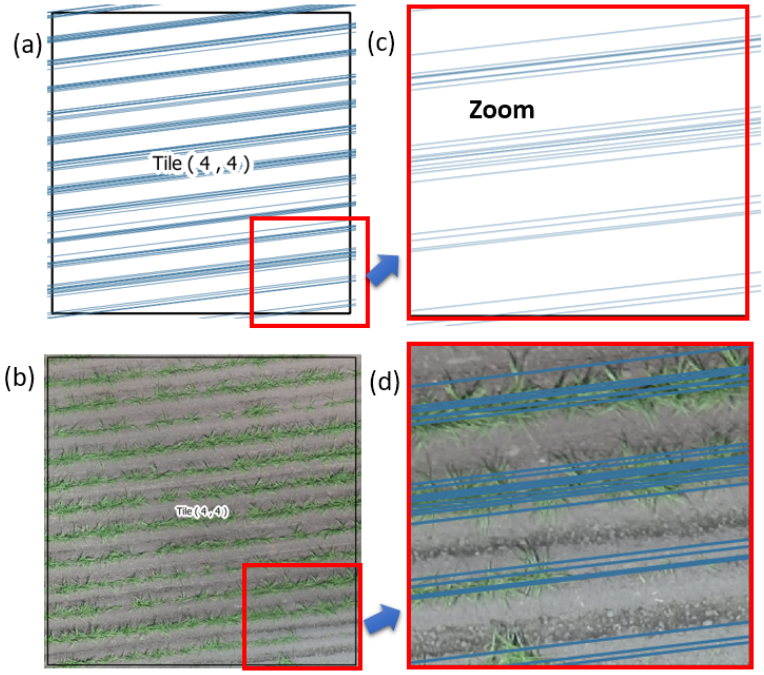

Figure 4.30.: Potential candidate lines to be a crop rows

In Figure 4.30 (a) candidate lines are shown, a zoom approximation is highlighted in Figure 4.30 (c), Figure 4.30 (b) is an image tile. Figure 4.30 (d) is an approximation of the image where candidate lines are overlayed.

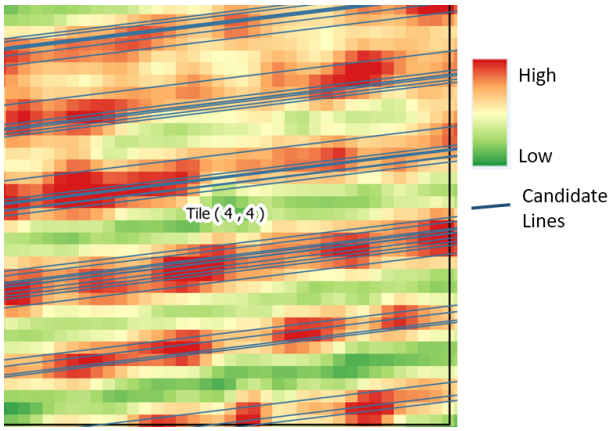

Figure 4.31.: Candidate lines over a hotspot visualization

Overall candidate lines, only candidate lines that are outside the cluster groups are removed. A hotspot strategy is used to perform this step, as seen in Figure 4.31.

Generate center lines

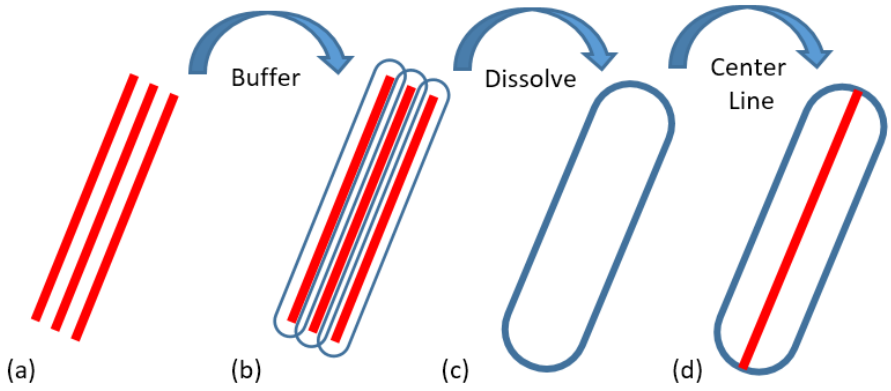

Figure 4.32.: Geoprocessing flow for center lines generation

Figure 4.32 shows the geoprocessing tasks carried out to generate the central lines. A buffer operation is applied to the resulting candidate lines (a) at a small distance of 30 cm, it is assumed that there are no crop rows spaced at distances less than 30 cm.

Resulting buffered polygons (b) are dissolved by spatial location into a single one polygon (c). Center line processing is achieved for the resulting dissolved features (d).

Resulting crop rows

Generated center lines are clipped by the vector field mask. Clipping mask may be present a complex geometry shape. As seen in Figure 4.33, the crop field is a polygon with an irregular shape.

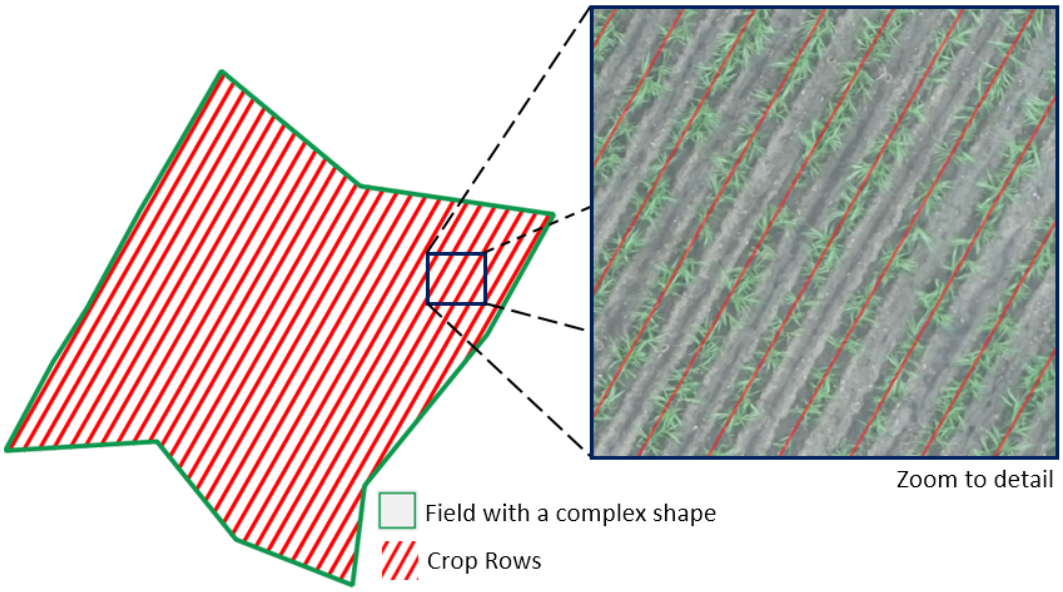

Figure 4.33.: The resulting Crop Rows clipped by a complex mask

Extra outputs

Crop rows, processing tiles, centerlines and intermediate processed data among others are stored for later analyses. All the data are georeferenced. All data are stored in vector formats. Each line has a unique identifier and its associated length. The resulting lines are also exported to formats with associated geospatial reference systems enabled for later use in navigation systems.

4.4. Open Source Application

4.4.1. Software engineering phase

This section illustrates the system development process that was carried out along the project.

Description of iterations

A detailed description of each of the iterations carried out in the current project is listed below.

Code	IT-1	**Name**	Iteration 1
Description	OpenCV Library for Python programming language was proposed to be evaluate and implemented in the first iteration. The development of a plugin for QGIS 2.8 was also proposed.		
Start Date	01/02/2018	**End Date**	15/02/2018

Table 4.1.: Iteration No.1 in software development process

Code	IT-2	**Name**	Iteration 2
Description	In this iteration was planned to read an orthomosaic image and a different methodologies for image preprocessing were explored.		
Start Date	16/02/2018	**End Date**	28/02/2018

Table 4.2.: Iteration No.2 in software development process

Code	IT-3	**Name**	Iteration 3
Description	In this iteration, different options for parallel processing in Python were explored. The algorithm for tile generation was implemented. Image enhancement functions were implemented.		
Start Date	01/03/2018	**End Date**	15/03/2018

Table 4.3.: Iteration No.3 in software development process

Code	IT-4	**Name**	Iteration 4
Description	In this iteration, the first version of the QGIS plugin was written and a client-server architecture was decided to explore later. The main functions for file management and input data processing were also written in this iteration. A simple user interface was proposed to be implemented for CRG-QGIS.		
Start Date	16/03/2018	**End Date**	20/04/2018

Table 4.4.: Iteration No.4 in software development process

Code	IT-5	**Name**	Iteration 5
Description	In this iteration, a new graphic interface of the plugin was schematized in the QT Designer tool.		
Start Date	20/04/2018	**End Date**	31/05/2018

Table 4.5.: Iteration No.5 in software development process

Code	IT-6	**Name**	Iteration 6
Description	In this iteration, the seed parameter was implemented to perform the search for the average angle in tiles.		
Start Date	01/06/2018	**End Date**	15/06/2018

Table 4.6.: Iteration No.6 in software development process

Code	IT-7	**Name**	Iteration 7
Description	In this iteration, the development of the (CRG-API) and (CRG-CLI) application is started with technologies such as Flask and Docker.		
Start Date	18/06/2018	**End Date**	30/06/2018

Table 4.7.: Iteration No.7 in software development process

Code	IT-8	**Name**	Iteration 8
Description	In this iteration, Geoprocessing functions were implemented using the GeoPandas library. Functions to export the results to KML and SHP files were implemented.		
Start Date	03/07/2018	**End Date**	19/07/2018

Table 4.8.: Iteration No.8 in software development process

Code	IT-9	**Name**	Iteration 9
Description	In this iteration, a unit tests and functionalities were performed for CRG-API, CRG-QGIS Plugin and CRG-CLI in Jupyter Notebook.		
Start Date	23/07/2018	**End Date**	31/08/2018

Table 4.9.: Iteration No.9 in software development process

Code	IT-10	**Name**	Iteration 10
Description	Tests of the application in operation were performed in this iteration.		
Start Date	03/09/2018	**End Date**	28/09/2018

Table 4.10.: Iteration No.10 in software development process

4.4.2. Requirements gathering

Requirements engineering is an essential part of building a software application. It includes the specification of services which are required from the system as well as defines the constraints on the system's operation and development. User requirements are clear statements, made using the natural language together with the diagrams which describe the services of that the system is expected to provide to the user. System requirements are more detailed description of the software system's functions, services, and operational constrains. The different types of requirements are used for better connection and understanding between different types of readers [89].

The development began with meetings with the stakeholders. A list of stakeholders interested in different areas within the project is presented in Table 4.11. The topics of

conversations were around how the crop rows generation process is realized and what they expect the new system to do. The current method to generate crop rows involves spent time digitizing vector lines in a GIS software or moving a tractor along the crop rows within the field.

Stakeholder	Position	Interest
Juan Valencia	GIS Analyst	Crop rows generation
Cesar García	Remote Sensing Analyst	Image processing
David Montero	Remote Sensing Analyst	Image processing
Mario Soto	GIS Analyst	Crop rows generation
Jeffrey Ospina	GIS Analyst	Usability, Crop Rows generation
Milton Ceballos	Precision Agriculture Analyst	Usability, Crop Rows generation
María Trujillo	Full time teacher	Image processing
Mauricio Cabezas	Full time teacher	Image processing, Software engineering

Table 4.11.: Primary stakeholders identified

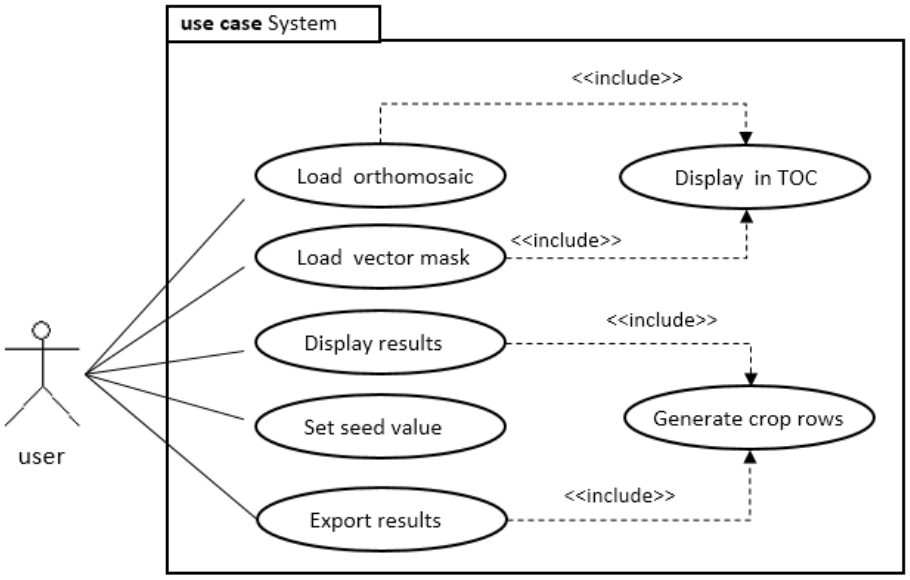

Figure 4.34.: Use cases diagram for basic requirements of CRG

The system is meant to be more of a proof-of-concept model rather than a product meant for day-to-day operation. This means that the importance of non-functional requirements

such as security, maintainability and reliability is lower. The basic requirements of the application were obtained and they are referenced in the diagram of use cases shown in Figure 4.34 for a particular user.

Software system requirements

The process of defining the software system requirements is included in the development part of the system. It allows the developers to interpret the client's needs and expectations for the future software. It is done by specifying the set of features and functions that the system must have [15]. One way of categorizing them is described as the FURPS+ model [38], using the acronym FURPS to describe the major categories of requirements with subcategories as shown below. **F**unctionality, **U**sability, **R**eliability, **P**erformance and **S**upportability.

Functional requirements

Function requirements describe an objective of the system by defining the main behavior of software at the specific conditions. They describe the abstract way what the system's functions should be. Functional requirements shall not include the technical details about the software. Non-functional requirements are responsible for it [89]. Functional requirements for crop rows generator system is presented in Table 4.12 . Overall there are five requirements presented for the software.

Code	Description
CRGFUN-01	The system shall have an ability to identify sugarcane crop rows.
CRGFUN-02	The system shall have an ability to detect sugarcane crop rows.
CRGFUN-03	The system shall have an ability to georeferencing sugarcane crop rows.
CRGFUN-04	The system shall have an ability to view the georeferenced sugarcane crop rows in a GIS format.
CRGFUN-05	The system shall have an ability to export the georeferenced sugarcane crop rows in a GIS format.

Table 4.12.: Functional requirements

Non-functional requirements

Non-functional requirements are requirements which specify how the system performs a certain function of its own. It describes the system behavior from the technical point of view. Also, it outlines the limitation of the system's functionality. Often the non-functional requirements specify the system's quality attributes or characteristics [89]. There is a considerate amount of sub categories of non-functional requirements. The most essential of its type are: usability Table 4.13, reliability Table 4.14, performance Table 4.15, and supportability Table 4.16.

Code	**Description**
CRGUSA-01	The system has to guide users through an interface based on end concepts.
CRGUSA-02	The system has to be easy to learn and does not distract the users' understanding of the system.

Table 4.13.: Usability requirements

Code	**Description**
CRGREL-01	The system has to give stability and possibility to reuse the georeferenced crop rows results in the future.
CRGREL-02	The system has to be able to tolerate and operate the errors.

Table 4.14.: Reliability requirements

Code	**Description**
CRGPER-01	The system has to be designed so that background tasks can continue while the user performs foreground tasks.
CRGPER-02	The system shall be able to respond to the user with not more than 1 minute.

Table 4.15.: Performance requirements

Code	**Description**
CRGSUP-01	The system shall be able to support various operation systems such as Linux and Windows.
CRGSUP-02	The system shall be able to be extended in the future.
CRGSUP-03	The system shall be able to be maintainable.

Table 4.16.: Supportability requirements

Activity diagram

The diagram in Figure 4.35 shows the workflow for georeferenced crop rows generation using orthomosaic images in sugarcane fields.

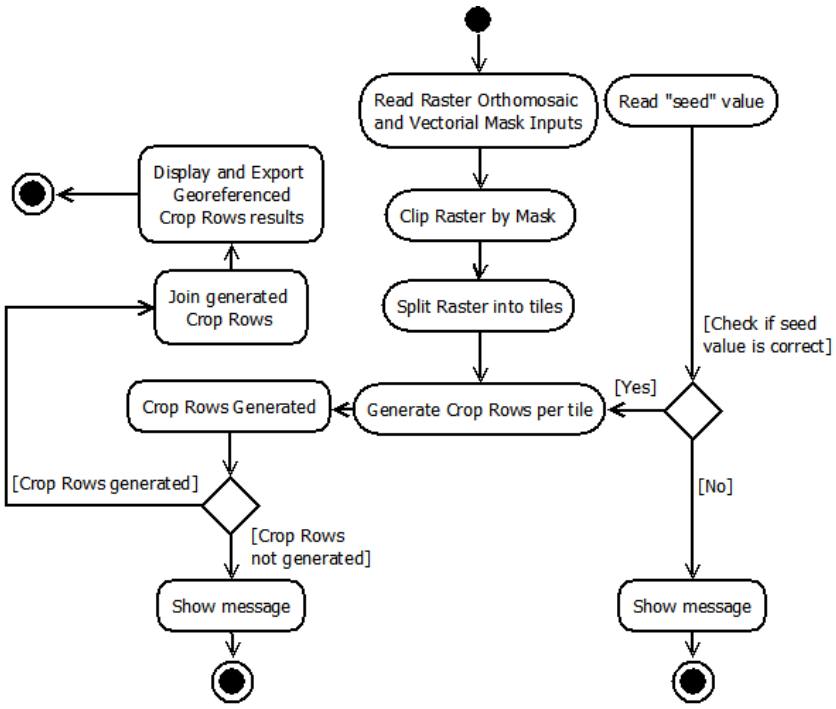

Figure 4.35.: Activity diagram

Firstly, raster orthomosaic, vectorial mask, and seed value must be provided by the user. The raster is clipped by the mask. Once is clipped a new raster file is generated. The new raster file is split into square tiles. If the seed value is not correct, the system shows a message to the user. A correct seed value is used to perform computer vision task to generate Crop Rows per tile. In the case, where no Crop Rows are detected and generated the system shows a message to the user. Generated Crop Rows per tile are joined through geospatial operations. Georeferenced crop rows results are exported for later use in file formats enabled for interoperability between GIS software and auto-guidance control panels. Furthermore, the system provides the user with the possibility to view the georeferenced Crop Rows results.

4.4.3. CRG-QGIS Plugin UI

A user of Crop Rows Generator shall see the main and configuration pages when he/she opens the application, as seen in Figure 4.36 and Figure 4.37 .The graphic user interface mockup for main window CRG-QGIS Plugin contains the elements listed below. GUI mockup generated by [1] tool.

1. Selector for orthomosaic input (raster layer).
2. Selector for field mask input (vector layer).
3. User options for choosing a crop rows orientation seed value.
4. Button for executing crop rows generation task.

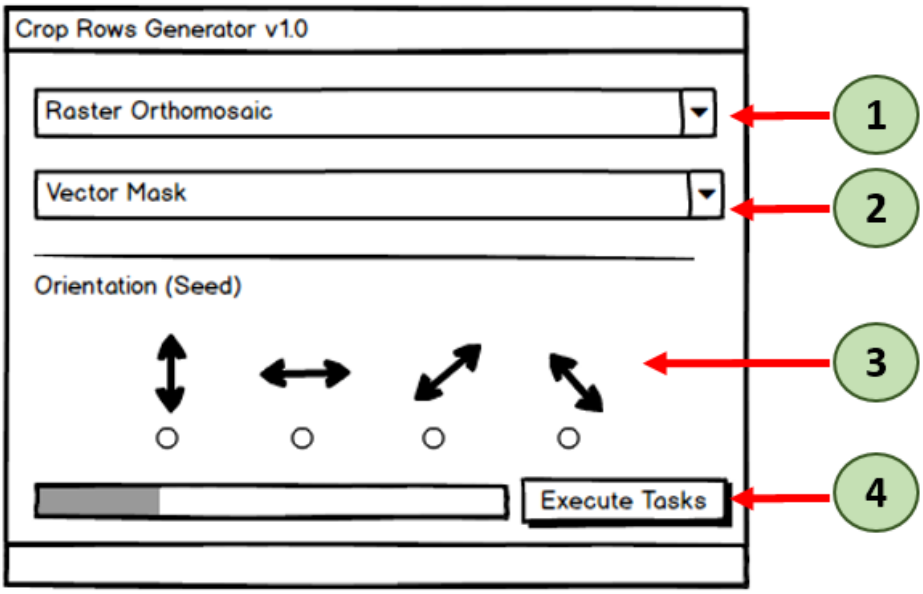

Figure 4.36.: User interface mockup for CRG-QGIS Plugin

The graphic user interface mockup for configuration window the CRG-QGIS Plugin contains the elements listed below.

1. Textarea for set up where CRG-CLI processing server is running.
2. Textarea for set up where OSGeo4W are installed.
3. Textarea for set up where the shared folder is.

4. Button for applying configuration changes.
5. Textarea for set up No data value for raster mosaic clip processing.
6. Check boxes to load the results into QGIS data frame and TOC when processes are finished.

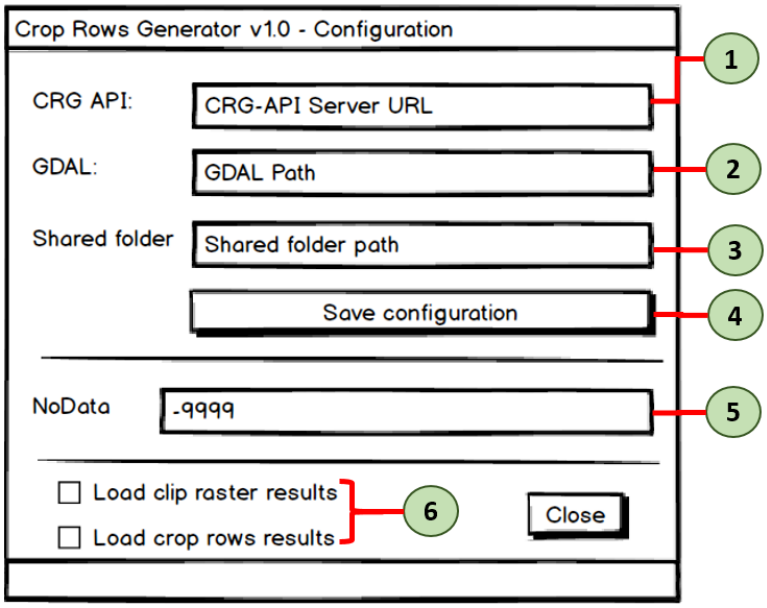

Figure 4.37.: User interface mockup for CRG-QGIS Plugin

4.4.4. Software Implementation

Developing the software is a time-consuming process which requires thoughtful planning of the design architecture and the result is achieved by try and error method. The ***Crop Rows Generator v1.0*** consists of three pieces of software. Figure 4.38 shows the two-tier architecture selected for software implementation.

- Crop Rows Generator - CLI a.k.a **CRG - CLI**.
- Crop Rows Generator - HTTP API a.k.a **CRG - API**.
- Crop Rows Generator - QGIS Plugin a.k.a **CRG - QGIS Plugin**.

CRG - CLI was implemented using the Python 3 [74], OpenCV 3.1.0-dev library [13] for the computer vision tasks for Crop Rows generation. GeoPandas 0.3.0 library [34] for

geoprocessing tasks. Joblib and multiprocessing 0.12.1 library [75] for parallel processing features.

CRG - API was implemented using the Python 3 [74], Flask 1.0.2 library [5] for HTTP communication interface API.

CRG - QGIS Plugin was implemented using the Python 2.7 [74], PyQT 4.10 library [70] and GDAL/OGR 2.2 library [33] for geoprocessing tasks.

4.4.5. Architecture

Crop Rows Generator uses two-tier architecture defined as client/server systems where the client requests resources and the server provide services to the client as requested, using its own resources. This means that the server does not rely on another application in order to provide part of the service. It runs the client processes separately from the server processes, usually on a different computer.

- **Presentation Layer**

  QGIS Desktop client and Crop Rows QGIS Plugin tightly connected (coupled)
- **Service Layer**

  Crop Rows-CLI runs on Server

  Separated from client

  Easy to scale
- **Business Logic and Data Access Layer**

  Heavy load on server

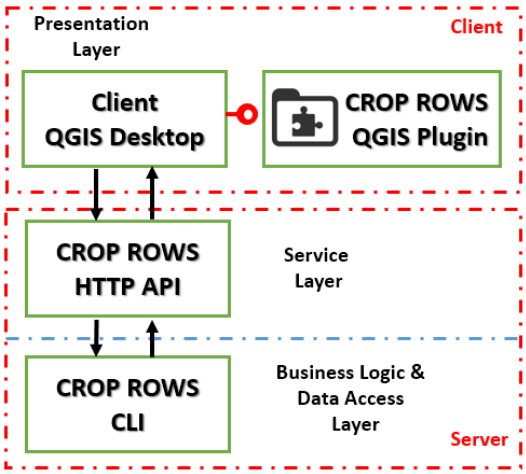

Figure 4.38.: Two-tier architecture

As seen in Figure 4.38, the presentation layer or interface runs on a client, and a service layer, business logic, and the data access layer are stored on a server. Separating these two components into different locations represents a two-tier architecture.

Server Side

CRG - CLI : runs inside a docker container that contains everything needed to run: computer vision tasks, geoprocessing tasks and so on. Containers are isolate applications from one another and the underlying infrastructure while providing an added layer of protection for the application. Containers provide a way to simulate isolation within the same machine or server.

Client Side

CRG - QGIS Plugin : runs inside the Quantum GIS environment as a Plugin with access to Geospatial libraries be able to perform geoprocessing task.

CRG - API : provides an interface API for communicating exposed resources from server to client through HTTP.

From the server side, new CRG-CLI instances are created from incoming requests through

CRG-API made by a client (CRG-Plugin) as seen in Figure 4.39.

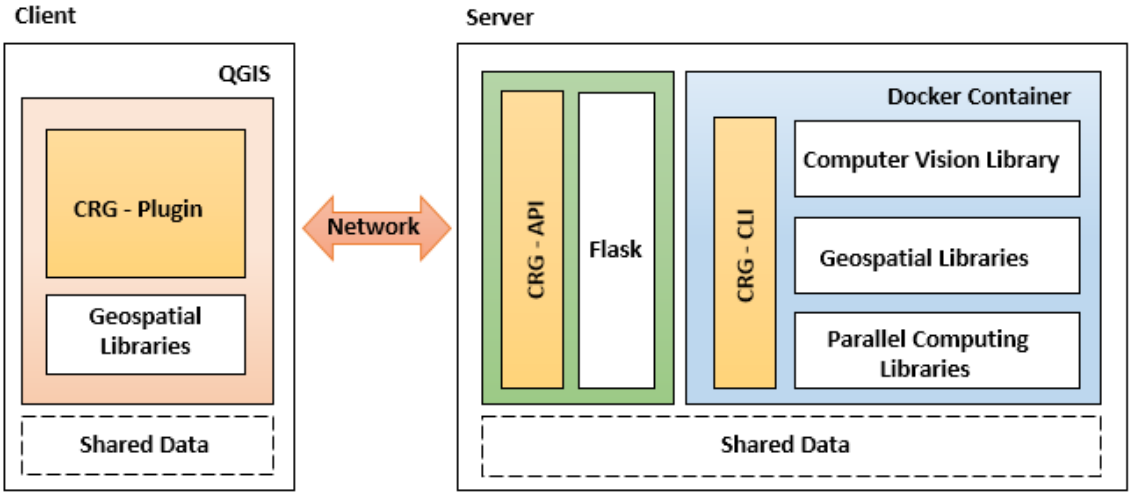

Figure 4.39.: Block diagram for Crop Rows Generator

CRG-API HTTP requests

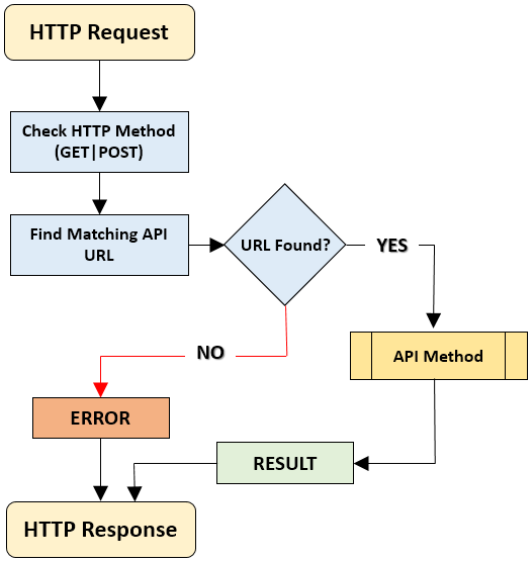

Figure 4.40.: CRG-API HTTP request scheme

Requests on the CRG-API are made through the HTTP method, if the requests are made correctly the user is notified with a successful response, otherwise, an error message will be received by the user as shown in Figure 4.40.

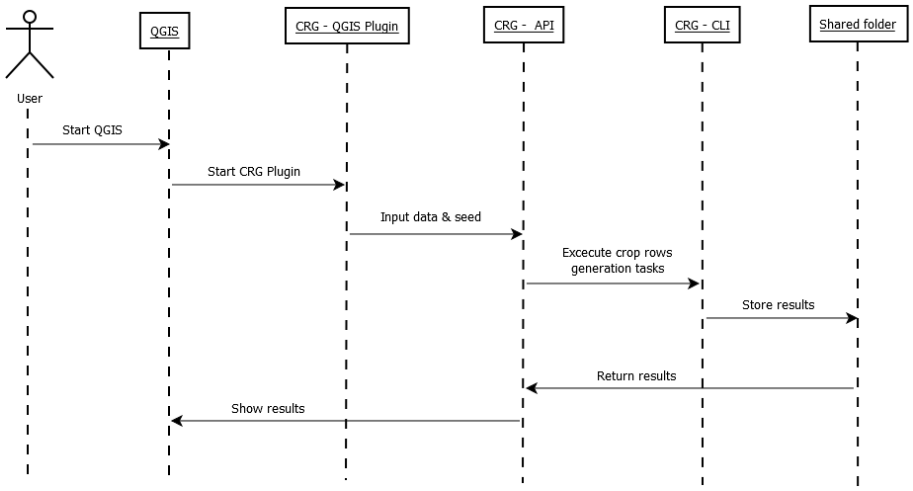

Figure 4.41.: Sequence diagram for Crop Rows Generator

The sequence diagram in Figure 4.41 specifically focus on the processes and objects that live simultaneously, and the messages exchanged between them to perform a function before the lifeline ends.

Shared folder

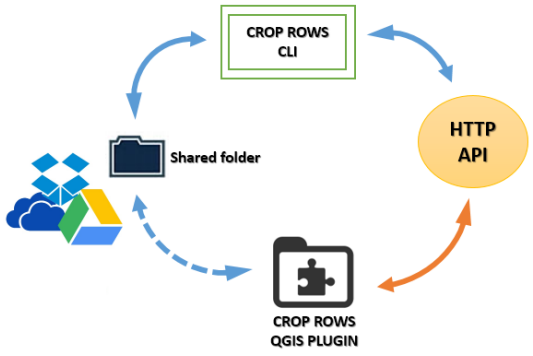

Figure 4.42.: Shared folder scheme

A shared folder is used to exchange resources (inputs and outputs) between the CRG-CLI and CRG-Plugin. A scheme is shown in Figure 4.42. The shared folder can be a local disk or a cloud storage service previously configured.

User interfaces implemented

In Figure 4.43 and Figure 4.44 there are the principal user interfaces that were implemented in CRG-QGIS Plugin, a complete listing conducted in the development stage are related in Appendix E for further details.

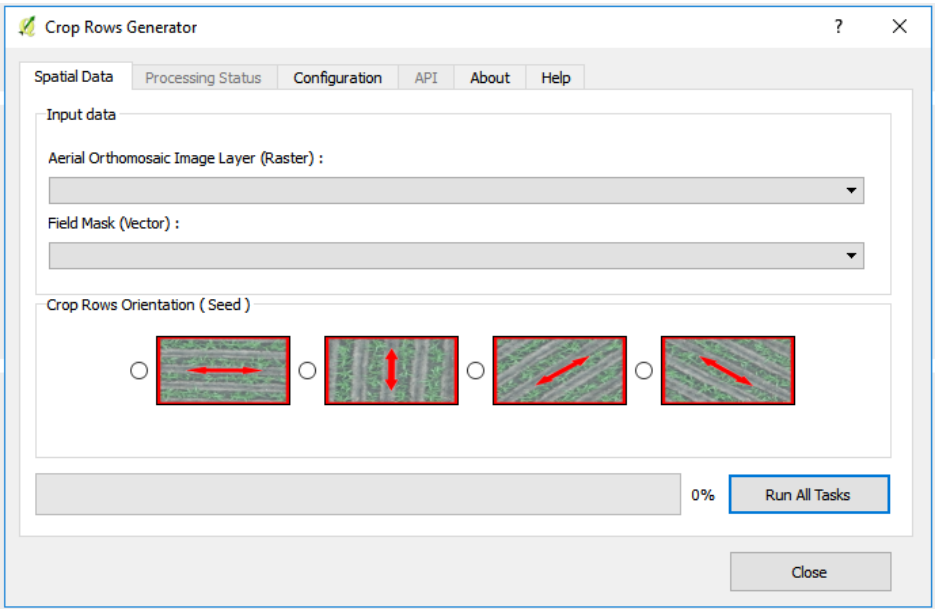

Figure 4.43.: Implemented spatial data input dialog window for CRG - QGIS Plugin

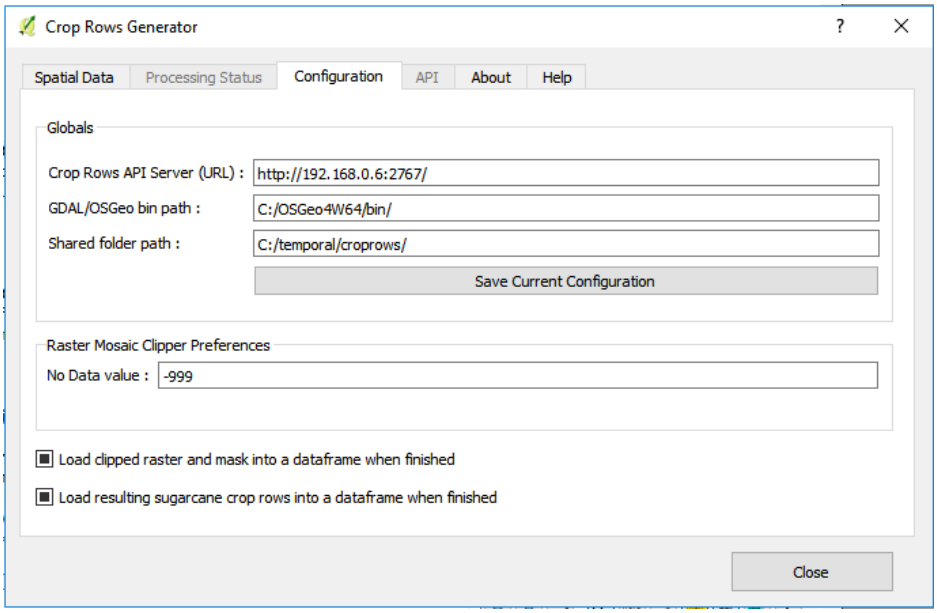

Figure 4.44.: Implemented configuration dialog window for CRG - QGIS Plugin

Crop Rows Project file

CRG - QGIS Plugin writes an exchange file in XML format that contains all the necessary data for *CRG - CLI* to perform the processing tasks.

```
<?xml version='1.0' encoding='utf-8'?>
<croprows>
<filename mask="maskfile_20180720-001914.shp" name="clipfile_20180720-001914.tif">
<image_dimensions>12369,11195</image_dimensions>
<image_origin>353510,378843</image_origin>
<pixel_size>0.01856,-0.01856</pixel_size>
<image_extents> 353510.402126,378843.153919,353739.970766,379050.933119 </
    image_extents>
<epsg>32618</epsg>
<projwtk>
PROJCS["WGS 84 / UTM zone 18N",GEOGCS["WGS 84",DATUM["WGS_1984",SPHEROID["WGS
    84",6378137,298.257223563,AUTHORITY["EPSG","7030"]],AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],UNIT["degree",0.0174532925199433,
    AUTHORITY["EPSG","9122"]],AUTHORITY["EPSG","4326"]],PROJECTION["
    Transverse_Mercator"],PARAMETER["latitude_of_origin",0],PARAMETER["
    central_meridian",-75],PARAMETER["scale_factor",0.9996],PARAMETER["false_easting
    ",500000],PARAMETER["false_northing",0],UNIT["metre",1,AUTHORITY["EPSG","9001"]],
    AXIS["Easting",EAST],AXIS["Northing",NORTH],AUTHORITY["EPSG","32618"]]
</projwtk>
<seed>3</seed>
<prj>croprows_20180720-00191</prj>
</filename>
</croprows>
```

Code block 4.3: A XML example of Crop Rows project file

This file contains parameters such as the name of the orthomosaic file, the vector file name, the pixel size, the spatial reference system, and the seed value selected by the user, among others.

Crop Rows results file

CRG - CLI performs all the necessary processes to obtain the main result (georeferenced Crop Rows) and some additional results. An exchange file in XML format is generated which *CRG - QGIS Plugin* reads and used to show the results in QGIS Desktop.

```
<?xml version='1.0' encoding='utf-8'?>
<croprows>
  <filename>
    <result>vectors/obj/croprows_lines.shp</result>
    <buffer>vectors/obj/croprows_lines_buffer.shp</buffer>
    <tile>vectors/tiles.geojson</tile>
    <export>export/croprows_wgs84.shp</export>
    <kml>export/croprows_wgs84.kml</kml>
  </filename>
  <metadata>
    <vectormask>vectormask.shp</vectormask>
    <epsg>32618</epsg>
    <candidatelines>117</candidatelines>
    <resultinglines>28</resultinglines>
  </metadata>
</croprows>
```

Code block 4.4: A XML example of Crop Rows results file

Generated result XML file contains the specific routes within the shared folder where the results are located. Results are geospatial information about crop rows in different formats and spatial reference systems supported by GIS software or auto-guidance control panels.

4.4.6. Testing and evaluation

Application testing

The test environment consists of a set of scripts running the tests. From Jupyter Notebook [51] software the tests are performed and the results are saved for debugging purposes and further analysis. The following steps are conducted to run a test:

- Fetch the source code according to the test run configuration.
- Run unit and integration tests.
- Run system tests and compare the output for algorithm verification.

Test environment runs manually. In Figure 4.45 a test case example is shown. A couple of test cases is referenced in Table 4.17.

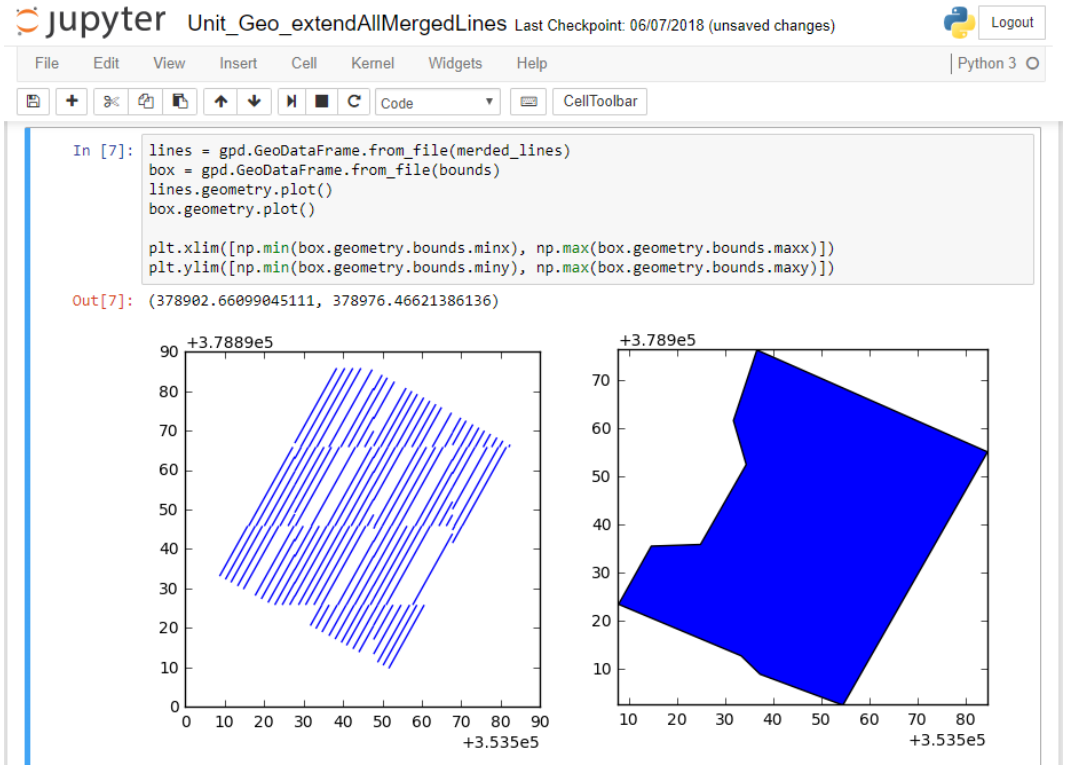

Figure 4.45.: Jupyter Notebook test environment for crop rows generator

Code	Scenario	Expected result	Status
CRGTEST-01	Read orthomosaic file image	Orthomosaic is read	Pass
CRGTEST-02	Read vector mask file	Vector mask file is read	Pass
CRGTEST-03	Start crop rows processing without setting a seed value.	Warning message	Pass
CRGTEST-04	Read a not georeferenced file image.	Error message	Pass
CRGTEST-05	Raster orthomosaic and Vector mask in different spatial reference systems.	Error message	Pass
CRGTEST-06	Clip orthomosaic by vector mask file.	Orthomosaic clipped by vector mask file	Pass
CRGTEST-07	Start crop rows processing without connection to CRG-API.	Error message	Pass
CRGTEST-08	Export crop rows result to KML file format.	Crop rows were exported in KML file format	Pass

Table 4.17.: Test cases for CRG

During a tests, no serious issues were been found. Out of the functional tests made, three different bugs were found in the test. Resulting bug report can be seen in Table 4.18.

Code	Bug
CRGBUG-01	When two CRG-Plugin instances have started, the application crash down.
CRGBUG-02	If input file names in the raster orthomosaic or in the vector mask contain special accents (i.e tildes) selectors does not display the option to be selected by a user.
CRGBUG-03	It is not possible to stop a processing task after it is started.

Table 4.18.: Bug report from functional testing

Release notes

The bugs found were solved in a new iteration. As a result, the application is working as designed, according to the functional and non-functional requirements. Even it is susceptible to improve towards new stakeholders needs and requirements.

4.5. Test the functionality and evaluate performance

4.5.1. Test functionality

The Crop Rows generator prototype was evaluated by eight persons with knowledge in GIS and precision agriculture. Even if the population was not that big, it allowed gaining a lot of useful information. Only the most revealing results were picked for this subsection but all results can be found in the Appendix F.

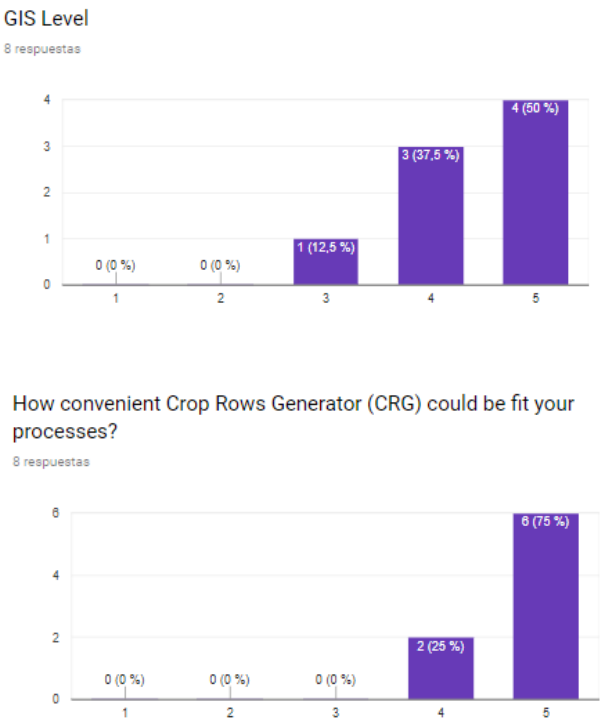

Figure 4.46.: Picked two of the most relevant results in the survey

Results for question *GIS Level* measured from (Newbie(1) to Expert(5)) is shown in Figure 4.46, likewise results for question *How convenient CRG could be fit your process?* where is measured from (Not convenient(1) to Very convenient(5)), In a general way, the users that tested the functionalities of the developed application has advanced knowledge in GIS issues and highlighted that CRG is very convenient and can be implemented in their processes within the companies where they work.

4.5.2. Test sites

High-resolution orthomosaic images of ten sites were used to evaluate the performance of the Crop Rows generation process. Sites for testing are located at department of the Cauca Valley at specific locations shown in Figure 4.47.

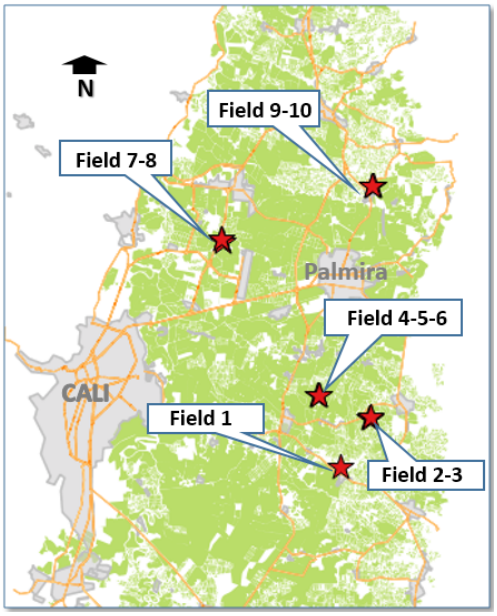

Figure 4.47.: Spatial location of the test sites

Test site	Area(ha)	Crop Rows	GSD
F1	8.1	232	1.8
F2	6.78	310	1.8
F3	3.09	128	2.0
F4	2.52	92	1.8
F5	2.2	64	1.8
F6	1.8	65	1.4
F7	1.72	71	1.6
F8	1.29	59	1.8
F9	1.1	37	1.8
F10	0.86	23	1.8

Table 4.19.: Test sites information and Crop Rows generated by CRG

GSD: Ground Sampling Distance(cm/pixel)

4.5.3. Performance Evaluation

In order to evaluate the performance of georeferenced Crop Rows Generation, crop rows detection performance has to be measured. A way to do it is using a binary classifier. Binary classifier performance can be described in terms of its sensitivity and specificity, quantifying its performance to false positive (FP) and false negative (FN) instances.

Confusion Matrix	**Positive**	**Negative**
Positive	True Positive	False Negative
Negative	False Positive	True Negative

Table 4.20.: Relation between, TP, TN, FP and FN — Confusion matrix

The confusion matrix in Table 4.20 indicates the relationship between different performance indices for binary classification.

Measure	**Definition**
True Positive (TP)	Crop row, which is also generated as a crop row by CRG.
False Positive (FP)	Crop row, which not generated as a crop row by CRG.
True Negative (TN)	Is not a crop row, which is also generated as is not a crop row by CRG.
False Negative (FN)	Is not a crop row, which is generated as a crop row by CRG.

Table 4.21.: Binary classification performance measures

For the computerized crop rows generation, the four performance indices (TP, TN, FP, and FN) in Table 4.21 are calculated by comparing the output lines from the CRG with the real Crop Rows determined by an expert. Using these four performance measures, relative measurements for binary classification can be calculated. Figure 4.48 is a graphical explanation of confusion matrix.

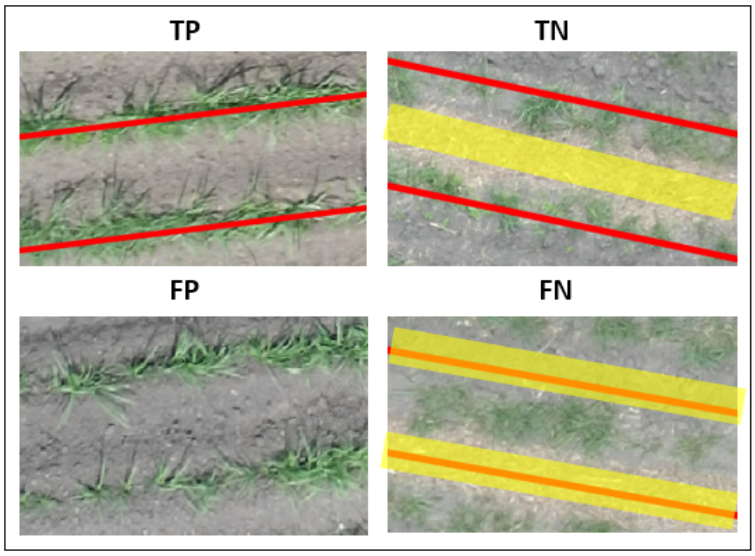

Figure 4.48.: Crop Rows confusion matrix

Sensitivity is defined as the ratio of crop rows which are generated as crop rows to all crop rows, given by the equation 4.5.

$$Se = \frac{TP}{TP + FN}. \tag{4.5}$$

Specificity is defined as the ratio of crop rows which are not generated as crop row, to all "is not crop rows", given by the equation 4.6.

$$Sp = \frac{TN}{TN + FP}. \tag{4.6}$$

The overall accuracy is the ratio between the total number of correctly generated crop rows instances and the test set size, given by the equation 4.7.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.7}$$

For each test site, a new polygon vector layer with random manual ground truth annotations is realized (expert criteria). Wherein the annotation with crop rows is tagged using

1 and no crop rows are tagged 0. The geospatial process implemented is shown in Figure 4.49. For each test site, sixty random manual ground truth annotations were realized. Union algorithm creates a layer containing all the features from both input layers (A: is a buffered crop rows layer generated by CRG) and (B: is a vector layer with manual ground truth annotations). The attribute table of the union layer contains attribute values from the respective input layer for non-overlapping features, and attribute values from both input layers for overlapping features. A set of classification rules are performed on attribute values of the resulting layer. Results of the geoprocessing are shown in the Table 4.22 with computed performance metrics.

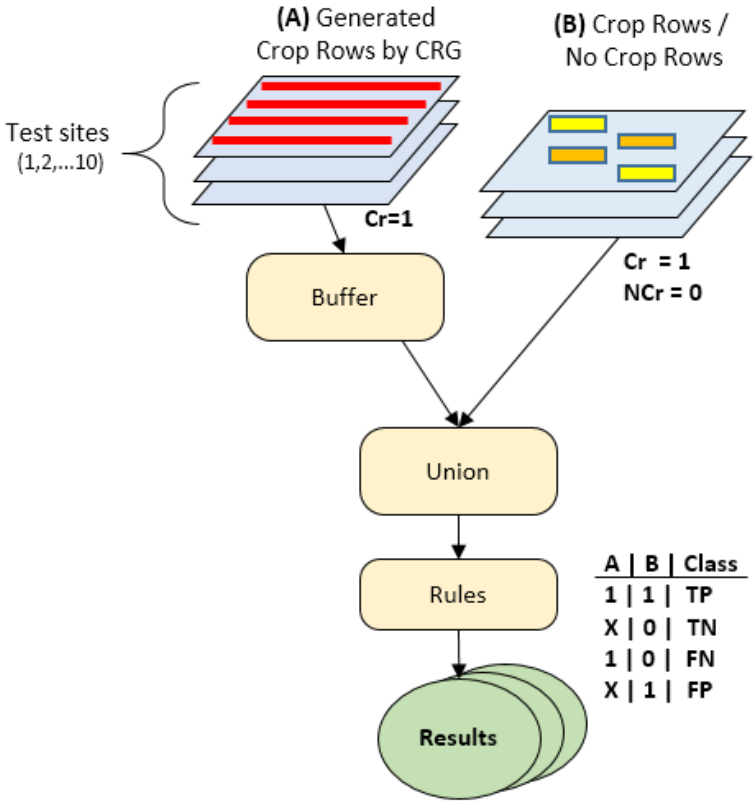

Figure 4.49.: Geoprocess for confusion matrix calculation of each test site

Buffer and Union are vector geoprocessing operations carry through with the generated crop rows by CRG and with the ground truth layer data.

	TP	FN	TP+FN	TN	FP	TN+FP	Se	Sp	Acc	FPF	TPF
F1	36	2	38	16	6	22	0.947	0.727	0.867	0.27	0.95
F2	25	5	30	26	4	30	0.833	0.867	0.850	0.13	0.83
F3	27	1	28	30	2	32	0.964	0.938	0.950	0.06	0.96
F4	32	0	32	28	0	28	1.000	1.000	1.000	0.00	1.00
F5	26	6	32	24	4	28	0.813	0.857	0.833	0.14	0.81
F6	16	2	18	42	0	42	0.889	1.000	0.967	0.00	0.89
F7	31	2	33	26	1	27	0.939	0.963	0.950	0.04	0.94
F8	38	0	38	22	0	22	1.000	1.000	1.000	0.00	1.00
F9	34	1	35	24	1	25	0.971	0.960	0.967	0.04	0.97
F10	42	0	42	18	0	18	1.000	1.000	1.000	0.00	1.00

Table 4.22.: Performance metrics for test fields

In order to visualize ROC curves of the binary classification performance, the performance metrics True Positive Fraction (TPF) and False Positive Fraction (FPF) can be computed using the equations 4.8 and 4.9.

$$TPF = \frac{TP}{TP + FN}. \tag{4.8}$$

$$FPF = 1 - \frac{TN}{TN + FP}. \tag{4.9}$$

Results of the computed performance metrics in the Table 4.22 shows that the CRG is capable to detect whit high accuracy crop rows. However it does not show how is the precision of detection can be. ROC graphs were generated by : Web-based Calculator for ROC Curves [27].

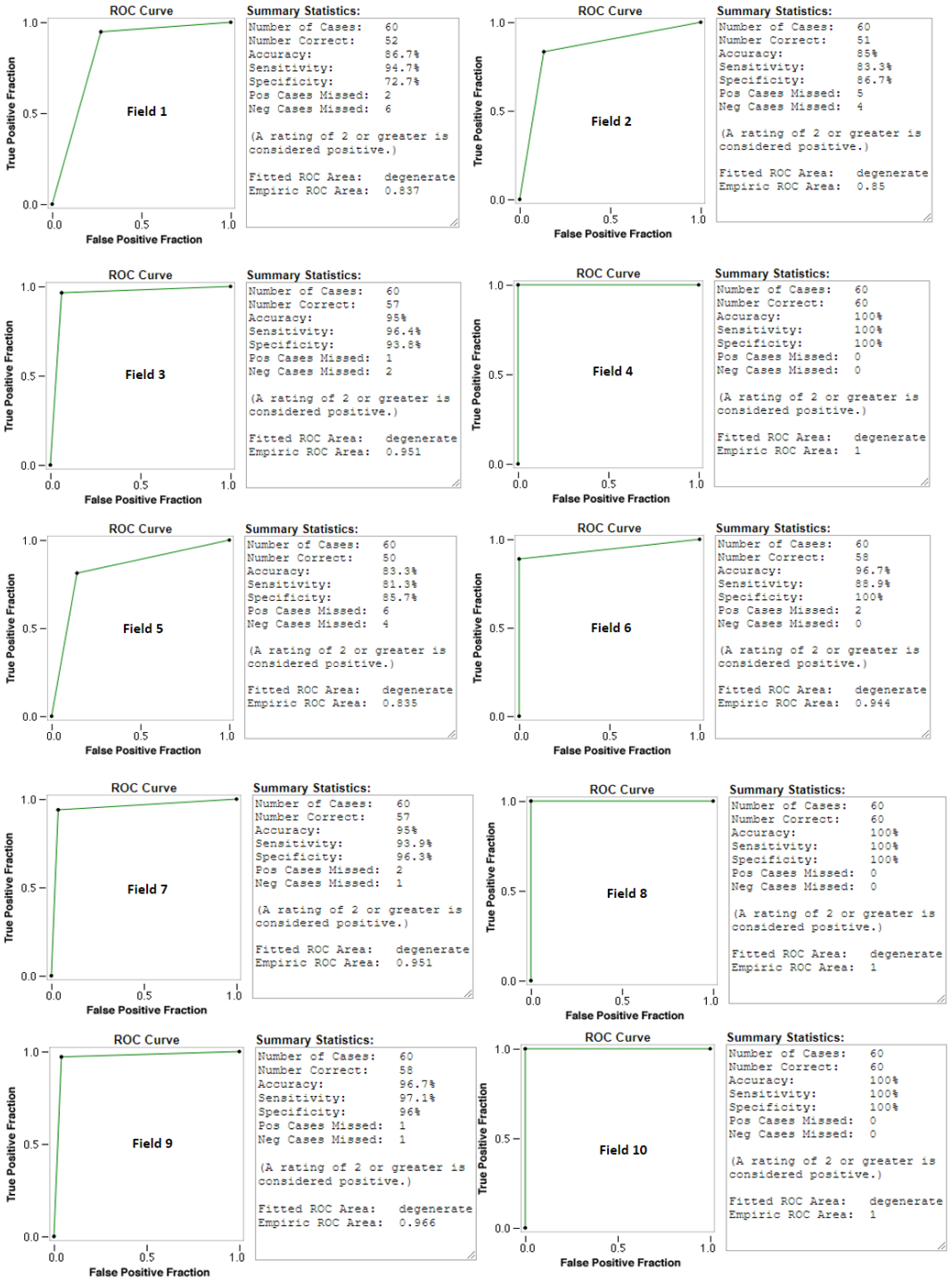

Figure 4.50.: ROC curves for the binary classification performance

4.5.4. Precision Evaluation

A metric of the precision of the developed system can be calculated by a measure of the horizontal error of the crop rows lines generated by CRG against ground truth crop rows. In order to obtain the horizontal errors, line distances between (georeferenced crop rows generated by CRG) and ground truth crop rows (generated by an expert user) were calculated along five transects (Figure 4.51). Horizontal errors are given in centimeters. Transects are perpendicular to the crop rows direction.

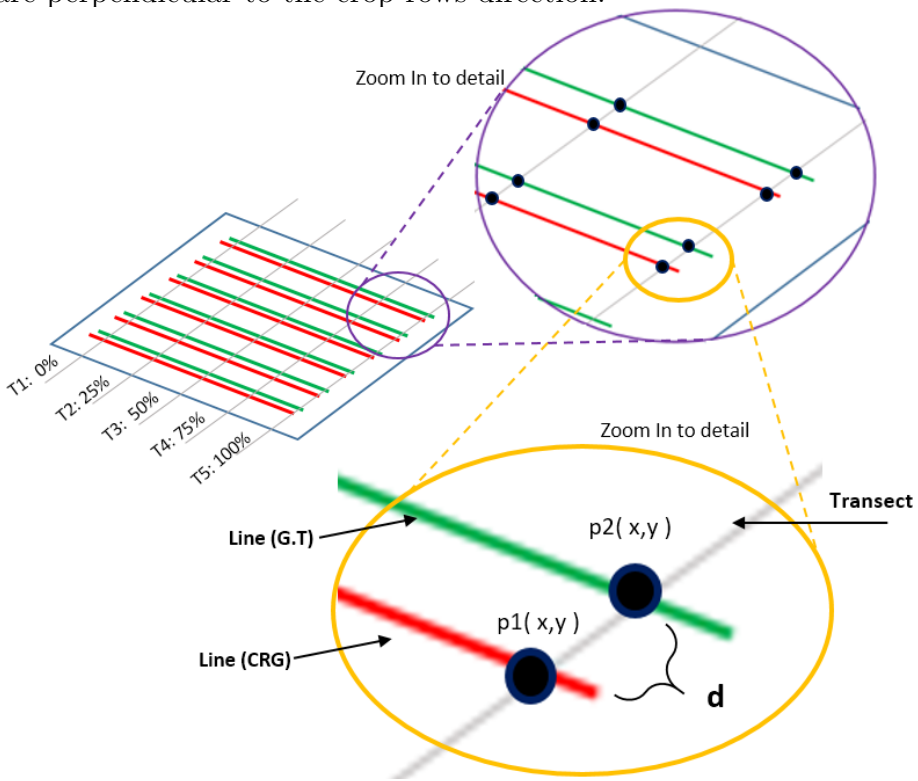

Figure 4.51.: The distance between a vertex of crop rows generated by CRG and vertex of ground truth crop rows to evaluate the precision

Transect lines are located at start(0%), First quantile(25%), middle(50%), third quantile(75%) and at the end(100%) perpendicularly along the crop rows. Geoprocessing routine described below was used to obtain the results.

Geoprocessing pseudocode

START

- Get crop rows orientation.
- Generate five lineal transects perpendicular to the crop rows direction.
- Generate vertexes by intersecting lines generated by CRG against line transects.
- Generate vertexes by intersecting ground truth (GT) lines against line transects.
- Calculate X and Y coordinates for each vertex.
- Calculate distances between GT and CRG vertexes.

END

The distance between vertex points from GT to CRG is given by the equation 4.10:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{4.10}$$

The resulting distances were plotted in the histogram shown in Figure 4.52. It shows how distant are the lines generated by CRG from the real ones and gives a general idea of the precision of the system.

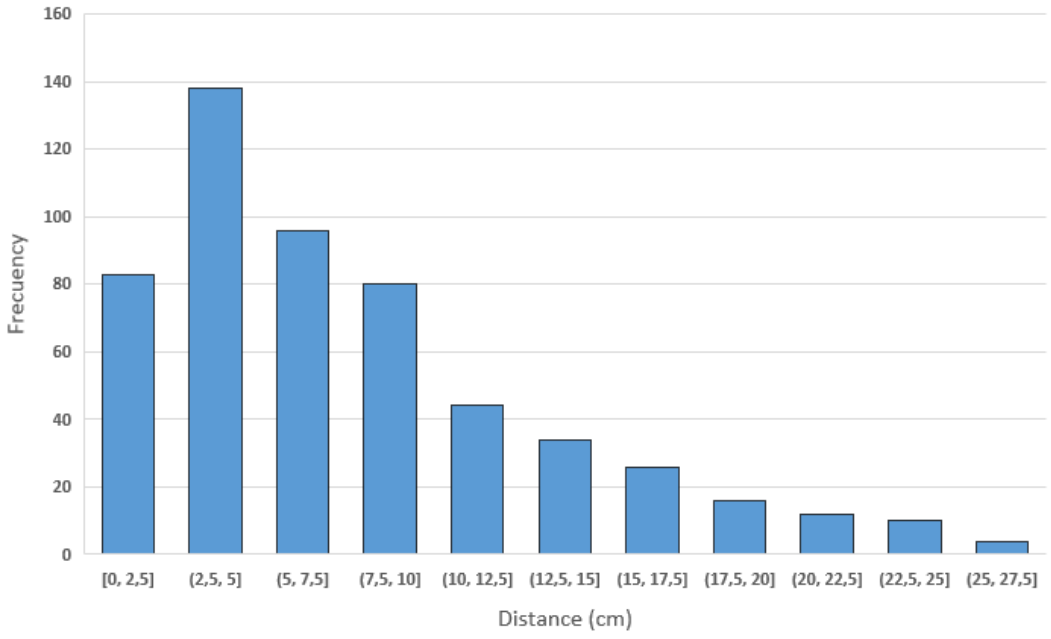

Figure 4.52.: Horizontal errors histogram result of the precision evaluation

4.6. Compare obtained results

In order to compare obtained results generated by CRG against the acquired by traditional methods. Hardware and software characteristics of the equipment used during the running test of the CRG for the server are described in the Table 4.23. Hardware and software characteristics for the client are described in the Table 4.24.

Processor	Intel(R) Pentium(R) CPU G640 @ 2.80GHz
RAM	4 Gb
Virtualization	VT-x
CPU(s)	2
Hard Disk	1TB
Operating system	Ubuntu 16.04 LTS GNU/Linux
Docker	Docker 1.13.1

Table 4.23.: Hardware and software on the server

Processor	AMD A6-7310 APU @ 2.0GHz
RAM	8 Gb
CPU(s)	4
Hard Disk	500 GB
Operating system	Windows 10
QGIS	Quantum GIS 2.18 LTR

Table 4.24.: Hardware and software on the client

Tests are conducted in order to determine the performance of the developed CRG compared to the traditional methods in terms of processing time. As seen in both tables above, hardware and software characteristics are very basic.

4.6.1. Lines generated by CRG vs manual digitizing

In the Figure 4.53 eleven volunteers users with experience in geographic information systems were selected to digitize crop rows lines following a simple process with orthomosaics images from three different sugarcane fields. They were trained in the procedure, spatial

data and instructions to carry out the task provided. Record formats, methodology, and metrics are described in the Appendix C.

Figure 4.53.: Volunteers from Fundamental of GIS course (2018-II) - Universidad del Valle

Composed evaluation metrics: Number of crop rows generated per minute (C_q) calculated by the equation 4.11 and Kilometers of crop rows generated per minute (C_k) calculated by the equation 4.12, were used to compare.

$$C_q = \left[\frac{n}{\frac{\sum_{i=1}^{n} x_i}{60}} \right], \tag{4.11}$$

where:

C_q = total crop rows per minute,

x_i = digitizing time for each i crop row,

n = total number of crop rows.

$$C_k = \left[\frac{\frac{\sum_{i=1}^{n} x_i}{1000}}{\frac{\sum_{i=1}^{n} t_i}{60}} \right], \tag{4.12}$$

where:

C_k = total kilometers of crop rows per minute,

x_i = length in meters for each i crop row,

t_i = digitizing time for each i crop row in seconds,

n = total number of crop rows.

Test site	**Area(ha)**	**Crop Rows**	**Time(s)**	**Length(km)**	C_q	C_k
F1	8.1	232	415	49.5	33.5	7.16
F2	6.78	310	360	40.8	51.7	6.80
F3	3.09	128	340	18.1	22.6	3.19
F4	2.52	92	192	14.3	28.8	4.47
F5	2.2	64	144	12.5	26.7	5.21
F6	1.8	65	147	10.2	26.5	4.16
F7	1.72	71	150	9.7	28.4	3.89
F8	1.29	59	133	7.0	26.6	3.15
F9	1.1	37	70	5.9	31.7	5.06
F10	0.86	23	55	4.3	25.1	4.69
				$minC_q$	**22.6**	
				$minC_k$		**3.15**

Table 4.25.: Metrics for Crop Rows generated by CRG

Metrics for Crop rows generated by CRG were calculated as the minimum value of (C_q) and (C_k) in ten sugarcane fields. $minC_k$ and $minC_q$ are given by the equations 4.13 and 4.14.

$$minC_k = \min(\sum_{i=1}^{n} C_k i). \tag{4.13}$$

$$minC_q = \min(\sum_{i=1}^{n} C_q i). \tag{4.14}$$

Metrics for Crop rows generated by ten humans who made a digitizing process were calculated as the maximum value of (C_q) and (C_k) in three sugarcane fields. $maxC_k$ and $maxC_q$ are given by the equations 4.15 and 4.16.

$$maxC_k = \max(\sum_{i=1}^{n} C_k i). \tag{4.15}$$

$$maxC_q = \max(\sum_{i=1}^{n} C_q i). \tag{4.16}$$

	Time(s)			Length(km)			C_q			C_k				
	F1	**F2**	**F3**	**F1**	**F2**	**F3**	**F1**	**F2**	**F3**	**F1**	**F2**	**F3**	C_qMax	C_kMax
H1	247	510	433	2.70	4.68	1.64	3.64	3.53	3.88	0.66	0.55	0.23	3.88	0.66
H2	252	463	426	2.70	4.68	1.64	3.57	3.89	3.94	0.65	0.61	0.23	3.94	0.65
H3	353	352	236	2.70	4.68	1.64	2.55	5.11	7.12	0.46	0.80	0.42	7.12	0.80
H4	478	707	244	2.63	4.68	1.63	1.88	2.55	6.88	0.33	0.40	0.40	6.88	0.40
H5	388	661	319	2.70	4.68	1.64	2.32	2.72	5.27	0.42	0.43	0.31	5.27	0.43
H6	293	322	217	2.70	4.68	1.64	3.07	5.59	7.72	0.55	0.87	0.45	7.72	0.87
H7	248	394	184	2.70	4.68	1.64	3.64	4.56	9.13	0.66	0.71	0.54	9.13	0.71
H8	650	659	503	2.70	4.68	1.64	1.39	2.73	3.34	0.25	0.43	0.20	3.34	0.43
H9	479	801	447	2.70	4.68	1.64	1.88	2.25	3.76	0.34	0.35	0.22	3.76	0.35
H10	224	781	251	2.70	4.67	1.63	4.01	2.30	6.69	0.72	0.36	0.39	6.69	0.72
H11	293	704	242	2.70	4.64	1.64	3.07	2.56	6.96	0.55	0.40	0.41	6.96	0.55
												$minC_q$	3.34	
												$minC_k$		0.35
												$maxC_q$	**9.13**	
												$maxC_k$		**0.87**

Table 4.26.: Metrics for crop rows generated by (H) human digitizing

Results of the metrics for crop rows generated by a human digitizing process are shown in Table 4.26.

4.6.2. Lines generated by CRG vs mechanized

For mechanized evaluation, a tractor movement along the crop rows was simulated with a constant speed of 12 km/h. Time movement per crop row was multiplied by the number of crop rows. Driving turns made at the end of the crop rows were not considered. Events such as contingencies and stops within the operation were not considered either. GPS/GNSS data downloading, verification and postprocessing were not considered. Results were generated for the same three sugarcane test fields digitized by users. Results of the metrics are condensed in Table 4.27.

	C_q	C_k
T1	3.40	0.20
$minC_q$	**3.40**	
$minC_k$		**0.20**

Table 4.27.: Metrics for crop rows generated by simulate tractor (T1) displacement at 12km/h

Comparison between CRG vs manual method and CRG vs mechanized method is shown in Table 4.28.

Method	C_q	C_k
Generated by CRG	22.3	3.15
Manual digitized	9.13	0.87
Mechanized (simulated)	3.14	0.20
CRG vs Manual	**248%**	**362%**
CRG vs Mechanized	**665%**	**1575%**

Table 4.28.: Comparison of the manual and mechanized method against crop rows generated by CRG

5. Summary, Conclusions and Future Work

5.1. Summary

The main objective of this project was *Design and build a method for generating georeferenced sugarcane crop rows using high-resolution aerial images* and it was covered in a successful way through **1)** Breaking the image down into square chunks and process each resulting chunk in a parallel way as a high-performance computing strategy for handle large and high-resolution aerial images. **2)** A hybrid methodology was proposed to identify and generate georeferenced crop rows and it involves computer vision techniques and spatial data geoprocessing. **3)** An open source software application *(Crop Rows Generator CRG)* was developed for crop rows generation. CRG consists of three components. **a)** As a user interfaces for data input a plugin (*Crop Rows QGIS-Plugin*) for QGIS software was developed. **b)** A server-side module for hard processing (*Crop Rows - CLI*) was developed. **c)** A communication interface between user and server over an HTTP REST API (*Crop Rows - API*) was developed. **4)** A group of tests was performed with data and real users and a couple of metrics were elaborated for testing the functionality and evaluate the performance in terms of accuracy and precision of the developed open source application. **5)** Obtained results by CRG were compared against the acquired by the traditional methods through comparable metrics elaborated by digitization process over a controlled area with a group of specialized users and a tractor displacement simulated as well. This research project was based on a real problem in the Colombian sugarcane agro-industrial sector and mainly focused on crop rows mapping for precision agriculture applications and it was tackled with geoinformatics techniques and methodologies.

5.2. Conclusions

- In this research, a method for generating georeferenced sugarcane crop rows using high-resolution aerial images was designed and built under a high-performance computing approach and computer vision techniques combined with a GIS oriented strategy for geographic data handling.

- The time for generating georeferenced crop rows using CRG is significantly lower than those achieved conducting similar tasks under field conditions or manual digitizing processes.

- Generate crop rows using CRG improves processing time above to 200% compared to a manual method.

- Generate crop rows using CRG improves processing time above to 650% compared to a mechanized method.

- The accuracy of the CRG is above to 83% for valid input orthomosaics.

- The precision of the CRG is in a range between $\pm$(2.5 to 10) cm for valid input orthomosaics.

- Breaking up tasks into individual parts that can be divided among several processor cores and reassembled as completed tasks, allows handling large file size images and high-resolution images in CRG.

- The CRG can be modified and adapted to the needs of other crops such as maize, wheat, pineapple, among others, even, crops arranged in not linear patterns.

- Tile segmentation by *Otsu's* thresholding method was adopted to binarize the index tonal images obtained from excess green vegetation index ExG and it allowed separate green pixels (sugarcane plants and weds) from the rest (soil, residues) efficiently.

- Searching the average angle inside a tile was synthesized by introducing the *seed* parameter which is supplied by an user input.
- Finding the oriented centerline of each rectangle in a tile was an efficient strategy without having to perform slow operations for line detection such as Hough transform feature extraction.
- The CRG could be part in a data processing workflow in precision agriculture departments within a sugarcane mill.

5.3. Constraints

- The protocol included as an (Appendix A) has to be followed for acquiring valid aerial images using (RGB) cameras on-board drone platforms over sugarcane fields.
- Only sugarcane crops in early stages (less than 100 days after seeding for plant cane and less than 75 days after harvest for ratoon crop) planted in rows pattern were contemplated in the project.
- In the developed software application and implemented methodology, only RGB imagery are allowed for crop row detection and mapping.
- Sugarcane crop rows present within the orthomosaic shall be contrasting and high quality and images capture from drone have to be realized corresponding to the established parameters in the protocol.
- In order to achieve a successful crop row detection process, a seed value needs to be provided by the user according to the crop row orientation pattern.

5.4. Future research

Even though the aim of the project was completed, there is still a lot of possibilities for future development.

- Image processing algorithms can be developed further so that the crop rows detection becomes generic including crops such as corn, wheat, rice, among others.
- Improve segmentation algorithms for sugarcane discrimination of other types of plants with a presence in the field such as the high weed incidence.
- Improve crop rows detection under variable environmental conditions, for example as when there are high humidity or very little solar radiation conditions.
- Include in the software application new options for using images obtained by sensors such as multi-spectral cameras which make plants discrimination much easier.
- Implement robust machine learning algorithms such as Convolutional Neural Networks (CNN) in order to not to depend on a strict image capture protocol.
- Implement an automatic crop row orientation finder feature in the software application in order to not to depend on the parameter (seed) entered by the user thus reducing possible user mistakes.

Bibliography

[1] Balsamiq Mockups. https://balsamiq.com/products/mockups/, 2008. [Online; accessed 7-April-2018].

[2] Aeronática Civil de Colombia. Circular reglamentaria Nº 002 requisitos generales de aeronavegabilidad y operaciones para RPAS (numeral 4.25.8.2). http://www.aerocivil.gov.co/AAeronautica/Rrglamentacion/RAC/Paginas/Inicio.aspx, 2015. [Online; accessed 19-October-2017].

[3] F. A. Al-wassai. Major Limitations of Satellite images. 2013.

[4] M. M. Alemu. Automated farm field delineation and crop row detection from satellite images. Master's thesis, University of Twente, the Netherlands, 2 2016. Faculty of Geo-Information Science and Earth Observation - Specialization: Geoinformatics.

[5] Armin Ronacher. Flask Python library. http://flask.pocoo.org/, 2010. [Online; accessed 6-March-2018].

[6] L. Armstrong. A survey of image processing techniques for agriculture. *Proceedings of Asian Federation for Information Technology in Agriculture*, pages 401–413, 2014.

[7] F. m. D. R. Audrey, D. F.-a. Fortin, K. Tanguy, X. Maldague, B. Panneton, and M.-j. Simard. Bayesian Classification and Unsupervised Learning for Isolating Weeds in Row Crops. *Pattern Analysis and Applications Industrial and Commercial Application*, 2012.

[8] X. D. Bai, Z. G. Cao, Y. Wang, Z. H. Yu, X. F. Zhang, and C. N. Li. Crop segmentation from images by morphology modeling in the CIE L Ã a Ã b Ã color space. 99:21–34, 2013.

[9] R. Bala. Survey on Texture Feature Extraction Methods. *International Journal of Engineering Science and Computing*, 7(4):10375–10377, 2017.

[10] A. Balafoutis, B. Beck, S. Fountas, J. Vangeyte, T. V. D. Wal, I. Soto, G. Manuel, A. B. Id, and V. Eory. Precision Agriculture Technologies Positively Contributing to GHG Emissions Mitigation , Farm Productivity and Economics. *Sustainability*, 9:1–28, 2017.

[11] B. Böhm and W. Hansen. The spiral model as a tool for evolutionary acquisition. 2010.

[12] I. Borra-serrano, J. M. Peña, J. Torres-sánchez, F. J. Mesas-carrascosa, and F. López-granados. Spatial Quality Evaluation of Resampled Unmanned Aerial Vehicle-Imagery for Weed Mapping. pages 19688–19708, 2015.

[13] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[14] T. Brosnan and D.-W. Sun. Inspection and grading of agricultural and food products by computer vision systems—a review. *Computers and Electronics in Agriculture*, 36(2):193 – 213, 2002.

[15] B. Bruegge and A. H. Dutoit. *Object-Oriented Software Engineering Using UML, Patterns, and Java*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2009.

[16] X. P. Burgos-artizzu, A. Ribeiro, and G. Pajares. Real-time image processing for crop / weed discrimination in maize fields. *Computers and Electronics in Agriculture*, 75(2):337–346, 2011.

[17] D. Butterworth. Find minimum-area bounding rectangle of a set of 2D points. https://github.com/dbworth/minimum-area-bounding-rectangle/, 2013. [Online; accessed 5-April-2018].

[18] Cenicaña. Colombian Sugarcane Research Center. http://www.cenicana.org/, 2017. [Online; accessed 10-December-2017].

[19] Y. Chen, J. Ribera, and C. Boomsma. Locating Crop Plant Centers From UAV-Based RGB Imagery. pages 2030–2037, 2016.

[20] I. Clavero-rumbao, A. Garcí, and F. Ló. Assessing Optimal Flight Parameters for Generating Accurate Multispectral Orthomosaicks by UAV to Support Site-Specific Crop Management. *Remote Sensing*, pages 12793–12814, 2015.

[21] G. Coast, A. P. Nolan, S. Park, S. Fuentes, D. Ryu, and H. Chung. Automated detection and segmentation of vine rows using high resolution UAS imagery in a commercial vineyard. pages 1406–1412, 2015.

[22] C. Correa and C. Valero. Row Crops Identification Through Hough Transform using Images Segmented by Robust Fuzzy Possibilistic C-Means Row Crops Identification Through Hough. (November), 2011.

[23] J. P. D. A. Costa, C. Germain, O. Lavialle, S. Homayouni, and G. Grenier. Vine field monitoring using high resolution remote sensing images. 1(figure 1):243–249, 2006.

[24] E. R. Davies. The application of machine vision to food and agriculture: a review. *The Imaging Science Journal*, 57(4):197–217, 2009.

[25] C. Delenne, S. Durrieu, G. Rabatel, M. Deshayes, C. Delenne, S. Durrieu, G. Rabatel, and M. Deshayes. From pixel to vine parcel : A complete methodology for vineyard delineation and characterization using remote-sensing data To cite this version : From pixel to vine parcel : a complete methodology for vineyard delineation and characterization using remote-sensing data. 2015.

[26] C. Delenne, G. Rabatel, M. Deshayes, C. Delenne, G. Rabatel, and M. Deshayes. Detection and Delineation in Remote Sensing detection and delineation in remote sensing. 2015.

[27] J. Eng. ROC analysis: Web-based calculator for ROC curves. http://www.rad.jhmi.edu/jeng/javarad/roc/JROCFITi.html, 2018. [Online; accessed 17-Jun-2018].

[28] A. English, P. Ross, D. Ball, and P. Corke. Vision Based Guidance for Robot Navigation in Agriculture. pages 1693–1698, 2014.

[29] Environmental Systems Research Institute, Inc. (ESRI). ESRI Shapefile Technical Description. https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf, 1998. [Online; accessed 5-March-2018].

[30] ESRI. FAQ: What is the format of the world file used for georeferencing images? https://support.esri.com/en/technical-article/000002860/, 2016. [Online; accessed 06-July-2018].

[31] Food and Agriculture Organization. Chapter 3. Furrow Irrigation . http://www.fao.org/docrep/s8684e/s8684e04.htm/, 2017. [Online; accessed 3-November-2017].

[32] S. Gary. *The Geospatial Desktop - Open Source GIS and Mapping*. Locate Press, 2012.

[33] GDAL Development Team. *GDAL - Geospatial Data Abstraction Library*. Open Source Geospatial Foundation, 2018. [Online; accessed 5-March-2018].

[34] GeoPandas developers. GeoPandas 0.3.0 library. http://geopandas.org/, 2018. [Online; accessed 6-March-2018].

[35] R. Gerber. Introduction to High Performance Computers . https://www.nersc.gov/assets/pubs_presos/IntroHPCSystems-NewUser.pdf/, 2012. [Online; accessed 1-November-2017].

[36] R. Giachetta. A Case Study of Advancing Remote Sensing Image Analysis. 22:57–79, 2015.

[37] Google Inc . Google Scholar . https://scholar.google.com/, 2017. [Online; accessed 1-July-2017].

[38] R. B. Grady. *Practical Software Metrics for Project Management and Process Improvement*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[39] M. Guijarro, M. Montalvo, J. Romeo, L. Emmi, A. Ribeiro, and G. Pajares. Automatic expert system based on images for accuracy crop row detection in maize fields.

[40] P. A. Gutiérrez, J. Torres-sánchez, and C. Hervás-martínez. A semi-supervised system for weed mapping in sunflower crops using unmanned aerial vehicles and a crop row detection method. 37:533–544, 2015.

[41] G. Halmetschlager, J. Prankl, and M. Vincze. Increasing the Precision of Generic Crop Row Detection and Tracking and Row End Detection. (i), 2014.

[42] E. Hamuda, B. M. Ginley, M. Glavin, and E. Jones. Automatic crop detection under field conditions using the HSV colour space and morphological operations. *Computers and Electronics in Agriculture*, 133:97–107, 2017.

[43] S. L. Hegarat-Mascle and C. Ottle. Automatic detection of field furrows from very high resolution optical imagery. *International and Sensing, Remote and Laboratoire*, 1(June 2015), 2013.

[44] Internet Engineering Task Force (IETF). The GeoJSON Specification (RFC 7946) . https://tools.ietf.org/html/rfc7946, 2016. [Online; accessed 5-March-2018].

[45] J. Ja. High performance computing algorithms for land cover dynamics. 21(6):1513–1536, 2000.

[46] F. Jhon. GNSS & Precision Farming:Higher accuracy,lower cost. *NovAtel ' s Thought Leadership Series*, pages 28–29, 2015.

[47] R. Ji and L. Qi. Crop-row detection algorithm based on Random Hough Transformation. *Mathematical and Computer Modelling*, 54(3-4):1016–1020, 2011.

[48] Y. Jia, Z. Su, W. Shen, J. Yuan, and Z. Xu. UAV Remote Sensing Image Mosaic and Its Application in Agriculture. 10(5):159–170, 2016.

[49] L. Joel. PyQGIS developer cookbook. I:86, 2016.

[50] M. Kelly. Object-based approach for crop row characterization in UAV images for site-specific weed management. (May 2011):426–430, 2012.

[51] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. E. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. B. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and et al. Jupyter notebooks - a publishing format for reproducible computational workflows. In *ELPUB*, 2016.

[52] V. B. Langote and C. S. Segmentation Techinques for image analysis. *International Journal of Advanced Engineering Research and Studies*, I(II):252–255, 2012.

[53] V. Leemans and M. Destain. Application of Hough Transform for Seed rows Localisation using Machine vision. pages 1–31, 2004.

[54] C. C. D. Lelong, P. Burger, G. Jubelin, B. Roux, S. Labbé, and F. Baret. Assessment of Unmanned Aerial Vehicles Crop in Small Plots. (May):3557–3585, 2014.

[55] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss. UAV-Based Crop and Weed Classification for Smart Farming UAV-Based Crop and Weed Classification for Smart Farming. 1(May), 2017.

[56] H. Lu, Z. Cao, Y. Xiao, Y. Li, and Y. Zhu. ScienceDirect Region-based colour modelling for joint crop and maize tassel segmentation. *Biosystems Engineering*, 147:139–150, 2016.

[57] A. R. Mamat. A Study of Image Processing in Agriculture Application under High Performance Computing Environment. 3(8), 2012.

[58] G. E. Meyer. Verification of color vegetation indices for automated ˜ Camargo Neto b. 3:282–293, 2008.

[59] P. Molin and P. S. Veiga. Spatial variability of sugarcane row gaps : measurement and mapping. 40(3):347–355, 2016.

[60] M. Montalvo, G. Pajares, J. M. Guerrero, J. Romeo, M. Guijarro, and Cruz. Automatic detection of crop rows in maize fields with high weeds pressure. 2011.

[61] Niles Ritter, Jet Propulsion Laboratory. GeoTIFF Format Specification. https://cdn.earthdata.nasa.gov/conduit/upload/6852/geotiff-1.8.1-1995-10-31.pdf, 1995. [Online; accessed 5-March-2018].

[62] M. Norremark, H. Griepentrog, H. Nielsen, and B. Blackmore. A method for high accuracy geo-referencing of data from field operations. *The Royal Veterinary and Agricultural University*, 56(July):463–467, 2003. AgroTechnology, 2630 Taastrup, Denmark.

[63] D. Oliva and Guadalaj. Unsupervised learning for crop / weeds discrimination in maize fields with high weeds densities. 1(January), 2015.

[64] Open Geospatial Consortium, Inc. (OGC). Keyhole Markup Language. http://www.opengeospatial.org/standards/kml/, 2018. [Online; accessed 5-March-2018].

[65] Ordnance Survey. Georeferencing. https://www.ordnancesurvey.co.uk/support/understanding-gis/georeferencing.html, 2018. [Online; accessed 5-March-2018].

[66] N. Otsu. A threshold selection method from gray-level histograms. 9:62–66, 01 1979.

[67] S. Panda, D. Ames, and P. Suranjan. Application of vegetation indices for agricultural crop yield prediction using neural network techniques. 2, 03 2010.

[68] J. M. Peña, J. Torres-sánchez, A. Serrano-pérez, A. I. D. Castro, and F. López-granados. Quantifying Efficacy and Limits of Unmanned Aerial Vehicle (UAV) Technology for Weed Seedling Detection as Affected by Sensor Resolution. 3:5609–5626, 2015.

[69] M. Perez-ruiz and S. K. Upadhyaya. GNSS in Precision Agricultural Operations. *New Approach of Indoor and Outdoor Localization Systems*, pages 3–26, 2012.

[70] Phil Thompson. *Python bindings for the Qt cross platform GUI toolkit*. Python Software Foundation, 2018. [Online; accessed 5-March-2018].

[71] D. ping Tian. A Review on Image Feature Extraction and Representation Techniques. *International Journal of Multimedia and Ubiquitous Engineering*, 8(4):385–396, 2013.

[72] R. S. Pressman and B. R. Maxim. *Software engineering: a practitioner's approach*. McGraw-Hill Education, international student edition, 2015.

[73] J. Primicerio, G. Caruso, L. Comba, A. Crisci, S. Guidoni, L. Genesio, D. R. Aimonino, A. Primicerio, P. Gay, S. Guidoni, L. Genesio, D. R. Aimonino, and F. Primo. Individual plant definition and missing plant characterization in vineyards from high-resolution UAV imagery. *European Journal of Remote Sensing*, 50(1), 2017.

[74] Python Core Team. *Python: A dynamic, open source programming language*. Python Software Foundation, 2018.

[75] Python Software Foundation. multiprocessing: Open source multiprocessing package for Python. https://docs.python.org/3.4/library/multiprocessing.html?highlight=Pool, 2017. [Online; accessed 16-March-2018].

[76] QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation, 2017. [Online; accessed 8-September-2017].

[77] K. Ramesh, N. Chandrika, S. Omkar, M. Meenavathi, and V. Rekha. Detection of Rows in Agricultural Crop Images Acquired by Remote Sensing from a UAV. *International Journal of Image, Graphics and Signal Processing*, 11(December):25–31, 2016.

[78] J. V. Rocha. *Precision Farming and Geographic Systems*, volume I. Encyclopedia of Life Support Systems (EOLSS), State University of Campinas (UNICAMP), Brazil, precision edition, 2016.

[79] J. Romeo, G. Pajares, M. Montalvo, J. M. Guerrero, M. Guijarro, and A. Ribeiro. Crop Row Detection in Maize Fields Inspired on the Human Visual Perception. 2012, 2012.

[80] J. Santana-ferná and J. Gó. Design and Implementation of a GPS Guidance System for Agricultural Tractors Using Augmented Reality Technology. pages 10435–10447, 2010.

[81] V. R. Sathuvalli, G. J. Vandemark, P. N. Miklas, A. H. Carter, M. O. Pumphrey, N. R. Knowles, and M. J. Pavek. Low-altitude , high-resolution aerial imaging systems for row and field crop phenotyping : A review. 70:112–123, 2015.

[82] SciELO.org. Scientific Electronic Library Online. http://www.scielo.org/, 2017. [Online; accessed 1-July-2017].

[83] ScienceDirect. Science Direct Web Search. https://www.sciencedirect.com/, 2017. [Online; accessed 1-July-2017].

[84] Scopus. Scopus web search. https://www.scopus.com/, 2017. [Online; accessed 1-July-2017].

[85] A. Sheykhi, P. Ahmadi, and H. Komarizade. Designing algorithm for detection of crop rows by using machine vision for use in variable rate systems. 4(2002):1150–1153, 2012.

[86] C. M. Sicre, F. Baup, and R. Fieuzal. Determination of the crop row orientations from Formosat2 multitemporal and panchromatic images, 2014.

[87] G. A. Soares, D. D. Abdala, and M. C. Escarpinati. Plantation Rows Identification by Means of Image Tiling and Hough Transform. 4(Visigrapp):453–459, 2018.

[88] H. T. Søgaard and H. J. Olsen. Determination of crop rows by image analysis without segmentation. 38:141–158, 2003.

[89] I. Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition, 2010.

[90] T. Stanhope. Applications of Low-Cost Computer Vision for Agricultural Implement Feedback and Control. 2016.

[91] D. Sudom, K. Runtz, and R. Palmer. Vision systems to track row crops for applying agricultural herbicides. pages 89–92, 1991.

[92] K. C. Swain and Q. U. Zaman. Rice Crop Monitoring with Unmanned Helicopter Remote Sensing Images. In D. L. Fatoyinbo, editor, *Remote Sensing of Biomass - Principles and Applications*, page 322. InTech, 2012.

[93] L. Tang and L. F. Tian. Plant Identification in Mosaicked Crop Row Images for Automatic Emerged Corn Plant Spacing Measurement Plant Identification in Mosaicked Crop Row Images for Automatic. 2008.

[94] E. Tayari, A. R. Jamshid, and H. R. Goodarzi. Role of GPS and GIS in precision agriculture. 2(3):157–162, 2015.

[95] A. Tellaeche, X. P. Burgos-artizzu, G. Pajares, and A. Ribeiro. A vision-based method for weeds identification through the Bayesian decision theory. 41:521–530, 2008.

[96] Trimble eCognition Suite. eCognition Developer. http://www.ecognition.com/, 2017. [Online; accessed 10-November-2017].

[97] C. Tu. An Efficient Crop Row Detection Method for Agriculture Robots. pages 655–659, 2014.

[98] G. Y. Tumlisan. Monitoring Growth Development And Yield Estimation Of Maize Using Very High-Resolution Uav-Images In Gronau, Germany. 2017.

[99] l. Ünal and M. Topakci. A Review on Using Drones for Precision Farming Applications. *Science, Technical Machinery, Agricultural*, pages 276–283, 2013.

[100] C. A. Viveros and H. Calderon. *El cultivo de la caña en la zona azucarera de Colombiana*, volume 1, chapter Siembra, pages 131–139. CENICAÑA, 1995.

[101] B. I. N. Wang. Pixel-Parallel Image Processing Techinques and Algoritms. *A Thesis Submitted To The University Of Manchester For The Degree Of Doctor Of Philosophy In The Faculty Of Engineering And Physical Sciences*, 1:142, 2015.

[102] Web of Science. Web of Science web search. https://webofknowledge.com/, 2017. [Online; accessed 1-July-2017].

[103] D. C. Wei. Image Segmentation,Representation and Description. *Graduate Institute of Communication Engineering*, page 36, 2009.

[104] M. Weyrich, Y. Wang, and M. Scharf. Quality assessment of row crop plants by using a machine vision system. pages 2466–2471, 2013.

[105] S. Yuheng and Y. Hao. Image Segmentation Algorithms Overview. 1, 2016.

[106] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37:1–19, 2004.

[107] Z. Zhang, Y. Lu, Y. Song, and H. Wang. A crop rows recognition based on hough transform using template matching. 44(2):172–178, 2012.

[108] S. Zhao and Z. Zhang. A new recognition of crop row based on its structural parameter model. pages 431–438, 2016.

[109] L.-y. Zheng and X. Jing-xue. Multi-crop-row detection based on strip analysis. pages 13–16, 2014. Proceedings of the 2014 International Conference on Machine Learning and Cybernetics.

A. Appendix: Image capture from a drone

Version No	Date	Author	Comments/Modifications
1.0	28-03-2018	Andres Herrera	Release Version

Abstract

Georeferenced crop rows are useful for certain precision agricultural (PA) applications. A flexible and inexpensive way for achieving this georeferenced lines is through image processing and certain computer vision techniques. This appendix presents a protocol for acquiring aerial images using simple (RGB) cameras onboard RPAS platforms over sugarcane fields. This protocol is divided in the following four steps: **1)** Scope and initial technical considerations, **2)** Preflight considerations, **3)** Flight considerations and **4)** Orthomosaic generation.

A1. Introduction

This section describes a image capture protocol which includes recommendations, characteristics and limitations for capturing high resolution aerial images used for crop rows generation in sugar cane fields. A set of valid images are needed for orthomosaic generation in an Agisoft PhotoScan software. Resulting orthomosaic are needed as an input for the developed *Crop Rows Generator (CRG) - QGIS Plugin*.

A2. System overview

In Figure A.1 shows a workflow diagram summarizing from flight planning, data acquisition in flight and post-processing captured data to delivery the orthomosaic as an end product ready to be processed for the *Crop Rows Generator (CRG) - QGIS Plugin.*

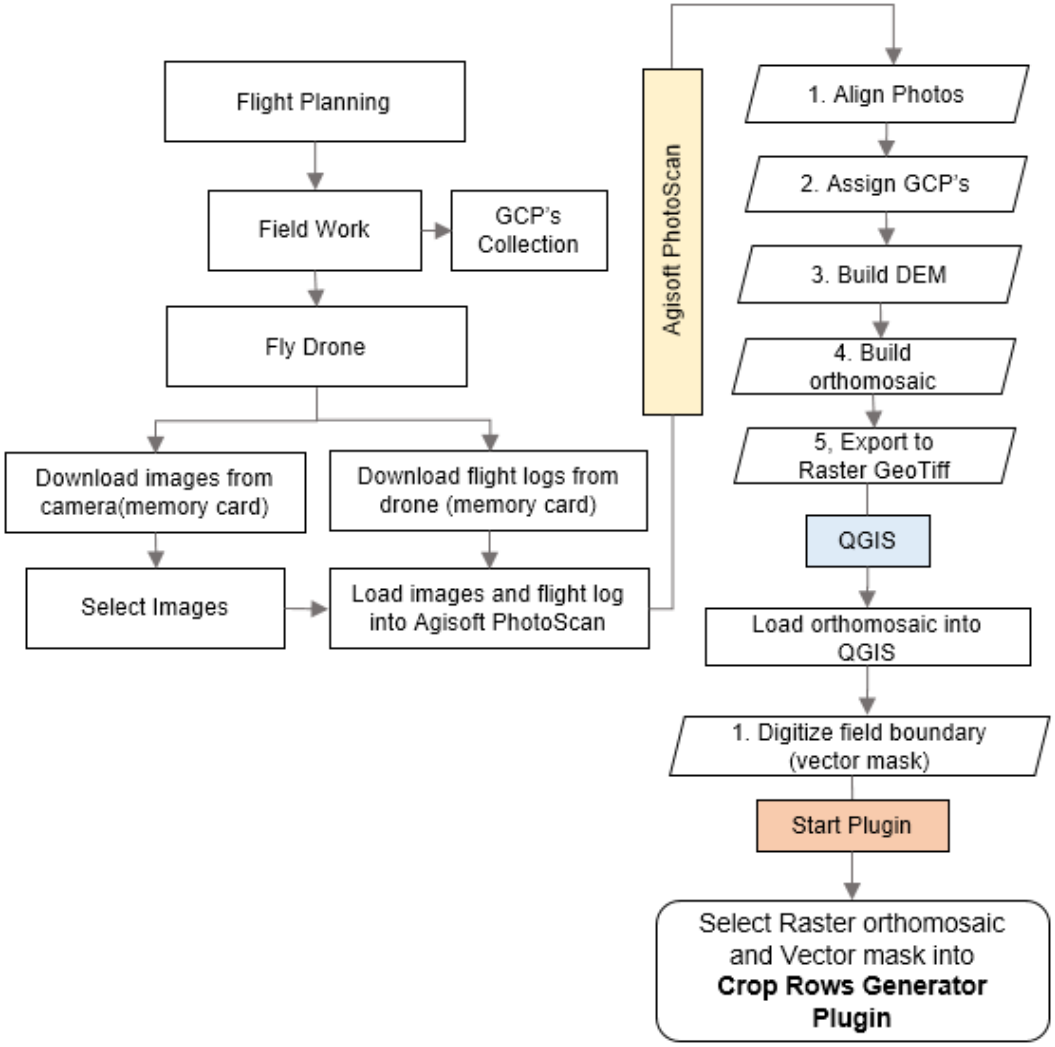

Figure A.1.: Workflow for high resolution raster orthomosaic image generation and vector mask required inputs for crop rows generator QGIS plugin

A3. Orthomosaic Generation Protocol

A3.1. Scope and Initial Technical Considerations

This section is based on technical considerations that have to be taken into account in order to have valid inputs that will later be used for crop rows generation process in the framework of the developed QGIS plugin. A list of technical considerations is listed below.

1. **Target Crop :** Sugarcane.
2. **Crop Variety :** Any.
3. **Growing Trait :** Ratoon crop or Plant cane.
4. **Crop Stage :** Early Stages.
5. **Image Sensor :** Any RGB Camera up to 12 Mega pixels.
6. **Image Scene :** Outdoors with normal light conditions.
7. **Imagery Type :** Nadir images
8. **Aerial Platform :** Any fixed-wing or rotary-wing.
9. **Orthomosaic Generation :** Agisoft PhotoScan Software.

A3.1.1 Growing Trait

The following Table A.1 was constructed based on criteria of experts consulted in the requirements analysis and only applies to sugarcane growing crop within *Colombia Cauca Valley* conditions and for the specific application of detection of crop rows.

Table A.1.: Appropriate scheduling for image capture in sugarcane

Growing Trait	Criteria	Min	Max
Plant Cane	Low germination ratio	>60 das	<100 das
Plant Cane	High germination ratio	>50 das	<90 das
Ratoon Crop	Low post-harvest residues incidence	>45 dah	<60 dah
Ratoon Crop	High post-harvest residues incidence	>60 dah	<75 dah

das: Days after seed, **dah:** Days after harvest

A3.2. Preflight Considerations

A3.2.1 Aerial Platform

Fixed-wing and rotary-wing are the mainly types of RPAS platforms used. In Table A.2 shows a few principal characteristics around them. Use of any of these aerial platforms are indifferent and does not affect the final result.

Table A.2.: Common RPAS platforms

Configuration	**Takeoff/Landing**	**Hover**
Fixed-wing	conventional	no
Multirotor	vertical	yes
Electric helicopter	vertical	yes
Gas helicopter	vertical	yes
Blimp	vertical	yes

The main criteria for a platform selection is given by the flight time *information not included in the table. Flight time and flight height limit the working area that could be cover by captured images.

A3.2.2 Parameters for flight planning configuration

Image frontal overlaps, side overlaps, and ground sampling distance parameters are listed below in order to get better results.

- **Frontal Overlap :** 80 %
- **Side Overlap :** 75 %
- **Ground Sampling Distance (GSD) :** Less than 3 cm/pix

Image overlaps and flight planning

The RPAS flight needs to cover the whole Area of interest (AOI) in a zig-zag pattern, as is shown in the Figure A.2. Flight path should be slightly larger than the intended data capturing area.

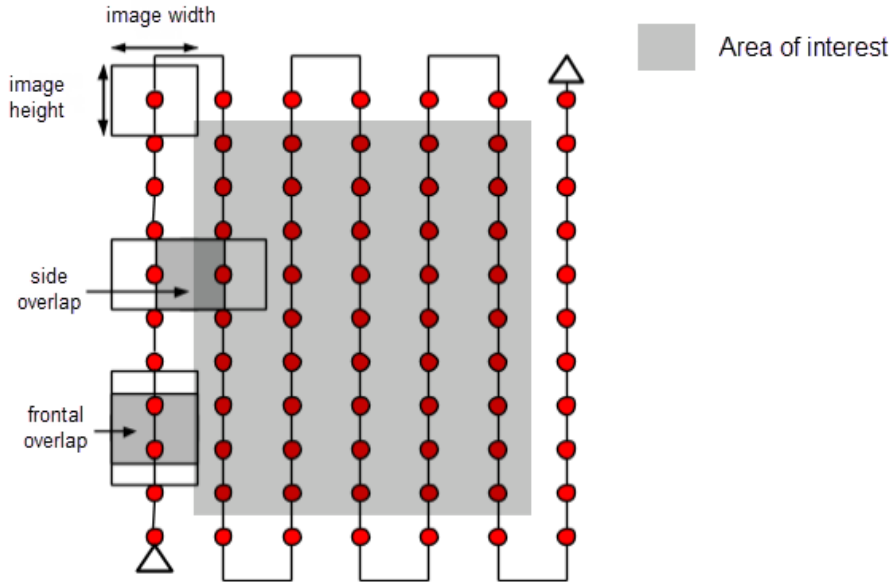

Figure A.2.: Image overlaps and flight acquisition route. (2018, February 02) [Digital Image]. Retrieved from https://support.pix4d.com

Ground Sampling Distance (GSD)

The Ground Sampling Distance (GSD) depends on the parameters of the camera (mainly camera resolution and focal length), the flight altitude and more generally the distance to the ground/target and Terrain surface uniformity as well as the presence of objects, trees or buildings as well.

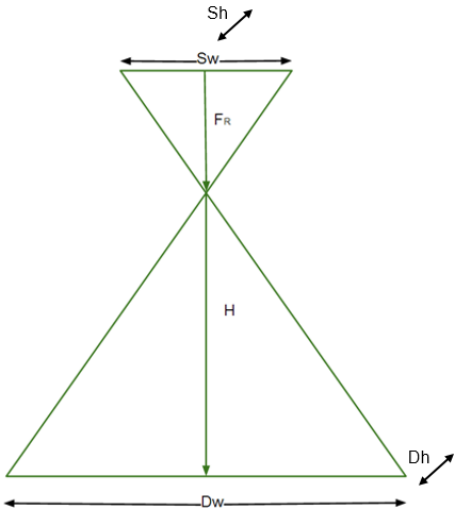

Figure A.3.: Ground Sampling Distance scheme

Sw : Sensor width of the camera (mm)

Sh : Sensor height of the camera (mm)

GSD : Ground Sampling Distance (mm/pixel)

Fr : Focal length of the camera (mm)

H : Flight height (meters)

Dh : Footprint height covered on the ground by one image

Dw : Footprint width covered on the ground by one image

For GSD calculation use this electronic calculator [1]

Table A.3.: GSD example for Canon S110 digital camera

Altitude [m]	**Resolution (GSD) [cm/px]**	**Ground photo width x height[m]**	**Ground photo area [m2]**
50	1.83	73 x 55	4.125
60	2.19	88 x 66	5.808
80	2.92	117 x 88	10.296
100	3.65	146 x 110	16.060
120	4.38	175 x 132	23.100
140	5.12	205 x 153	31.365

Table A.3 is a GSD example calculation for Canon S110 digital camera with 12 Mpx - Sw:7.6mm, Sh:5.7mm, FR:5.2mm, Iw:4000px, Ih:3000px

A3.3. Field Work

A3.3.1 GCP's Collection

GCPs increase the global accuracy of orthomosaic. Ground control points are large marked targets on the ground, spaced strategically throughout area of interest. First need to determine the GPS coordinates at the center of each. The ground control points and their coordinates are then used to help drone mapping software accurately position

[1]GSD Calculator https://support.pix4d.com/hc/en-us/articles/202560249-TOOLS-GSD-Calculator#gsc.tab=0

the map in relation to the real world around it. The GCPs and their coordinates are then used to help drone mapping software accurately position the map in relation to the real world around it.

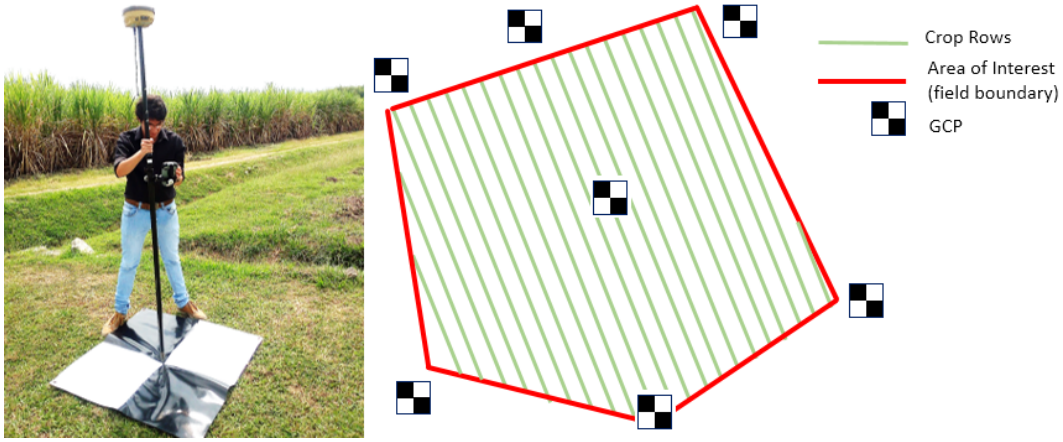

Figure A.4.: Ground control points (GCP) distribution out over the field. GCPs are typically measured with highly-accurate ground-based GPS equipment.

Some recommendations for correct GCP positioning in the field

- GCP have to be easily visible in the aerial imagery.
- Use a minimum of 5 large GCPs.
- Evenly distribute GCPs on the ground.
- Make sure the GCPs are unobstructed.
- Measure GCP Center with High Precision GPS/GNSS.

Measuring the Location of GCPs

Measure the GPS coordinates at the center of each ground control point. To do this, you need either a Real Time Kinematic (RTK) or Post Processing Kinematic (PPK) GPS receiver.

A3.4. Field Work

A3.4.1 Flight Conditions

The best weather for RPAs flying is when it is sunny, a reasonable temperature (25 degrees Celcius, for example), and little to no wind.

Table A.4.: General conditions in field

Wind	Precipitation	Cloud Cover	Visible Sats	K Index	TDF
0 - 24 km/h	No Precipitation	(0/8) or (1-2/8)	>10	<4	10 AM - 1 PM

Cloud cover : Describes the extent of cover by clouds in the portion of the sky visible from the observation point. It is estimated by the weather observer and given in eights by the synoptic service. The following classification is usually made.

Table A.5.: Cloud coverage amount

Cloud amount	Description
0/8	clear skies
1-2/8	fair
3/8	lightly clouded
4-6/8	cloudy
7/8	very cloudy
8/8	overcast

K-index : quantifies disturbances in the horizontal component of earth's magnetic field with an integer in the range **0–9** with **1** being calm and **5** or more indicating a geomagnetic storm. These disturbances are caused by solar flares so increased solar activity is likely to have a knock-on effect on your ability to fly in GPS mode. For Planetary K-Index check [2]

Time of Day to Fly (TDF) : the best hours for drone imagery collection are where there are less shadows over the terrain, in a tropical country such as Colombia is between

[2]NOAA planetary K-Index https://www.swpc.noaa.gov/products/planetary-k-index

10 AM to 1 PM.

A3.5 Orthomosaic image generation

Images will be stitched into a single orthomosaic Raster image using **Agisoft PhotoScan**[3] software." Agisoft PhotoScan is a stand-alone software product that performs photogrammetric processing of digital images and generates 3D spatial data ". Agisoft PhotoScan turns those simple images into three dimensional geographic data that can be used in combination with other geographic datasets. The workflow from images to products is very often just and easy-to-use black box processing standards. Orthomosaic has to be exported to an **RGB GeoTIFF** format. GeoTIFFs are TIFF files that contain spatial reference information.

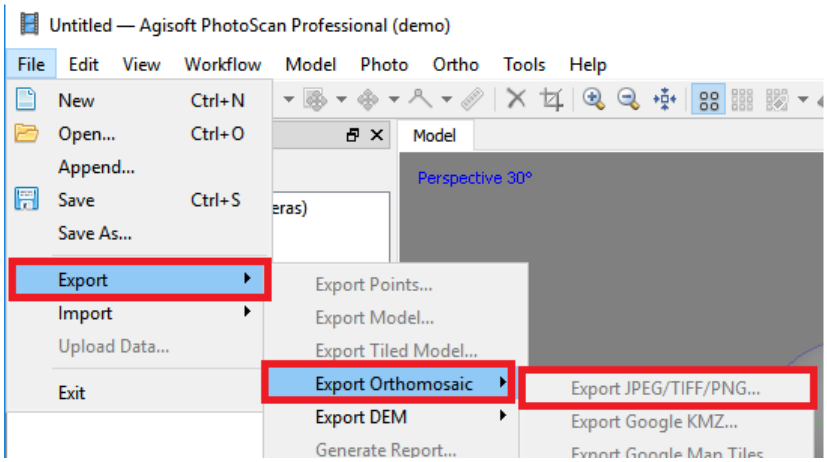

Figure A.5.: Agisoft PhotoScan Export Orthomosaic option

The easiest way is to export it as a geotiff is: File − > Export Orthomosaic − > Export Orthophoto − > Export JPEG/TIFF/PNG (Figure. A.5). Just make sure that you add ".tif" to the end of your filename. The geotiff format can be read directly by most GIS packages even QGIS. A valid coordinate system needs to be projected (such as UTM - where positions are measured in meters). Required projection can be referenced by the code EPSG:32618 (WGS 84 / UTM zone 18N) or EPSG:3115 (MAGNA-SIRGAS / Colombia West zone) for the Valle del Cauca Colombia West Zone.

[3]Avaliable at: http://www.agisoft.com/

B. Appendix: Generate a vector mask

Version No	Date	Author	Comments/Modifications
1.0	28-03-2018	Andres Herrera	Release Version

Abstract

This section describes a protocol for generate a required vectorial file mask for the *Crop Rows Generator QGIS Plugin.*

B1. Introduction

The *Crop Rows Generator QGIS Plugin* needs a Vector file mask to perform the action. This process is divided into two stages. 1) Load the raster file. 2) Draw the vector mask.

B2. Load Orthomosaic into QGIS

Navigate to and click on the menu entry Layer − > Add − > Add Raster Layer.

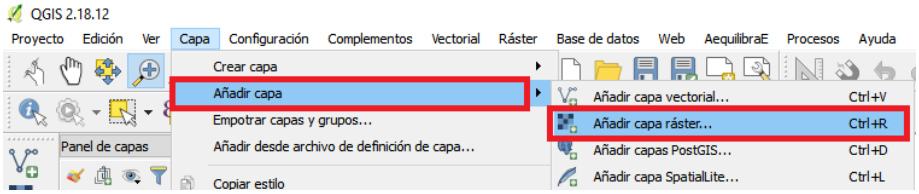

Figure B.1.: Toolbar navigation to Add a Raster layer.

Navigate to Orthomosaics folder. Select the desired file. Click Open. An image will load into the QGIS Data frame.

B3. Generate vector field mask

Open the New Vector Layer dialog that will allow you to define a new layer.

Navigate to and click on the menu entry Layer − > New − > New Shapefile Layer.

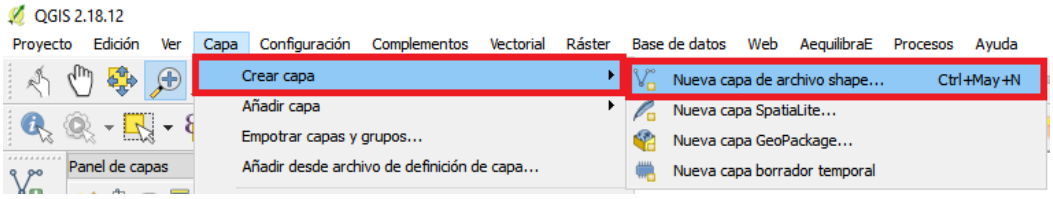

Figure B.2.: Toolbar navigation for new Shapefile layer creation.

Create a polygon dataset and specify the Coordinate Reference System, or CRS. A CRS specifies how to describe a point on Earth in terms of coordinates. The CRS supported for *Crop Rows Core* are based on Projected coordinate systems such as EPSG:3115 and EPSG:32618 among others.

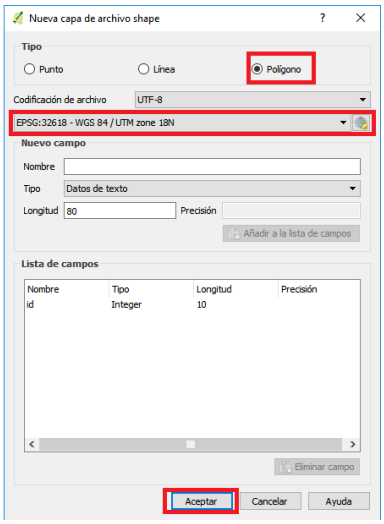

Figure B.3.: New vector layer dialog. Polygon selection and Coordinate reference system.

Save the new layer with a preferred name i.e *field_boundary.shp*.

For selected layer the Toggle Editing tool  provides beginning or ending an editing session.

Figure B.4.: Toggle Editing tool location on editing tool bar.

Polygons are created by left clicking with the Add Features tool  . To complete a polygon feature right click. After adding a feature you will be prompted to enter the attributes. Let this field null.

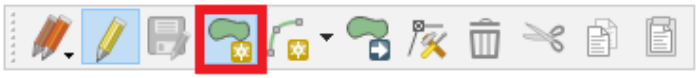

Figure B.5.: Add Features tool location on editing tool bar.

Digitize the boundary on new polygon layer created. Try to follow the contour as accurately as possible, using points (left-click) at any corners or turns.

Figure B.6.: Digitize sugar cane field boundaries over previous Raster mosaic loaded.

Once complete the polygon drawing. Left click in map. Then click in Toggle Editing tool  for ending an editing session.

C. Appendix: Evaluate manual Crop Rows generation.

Version No	Date	Author	Comments/Modifications
1.0	10-08-2018	Andres Herrera	Release version

C1. Introduction

This section describes a protocol to evaluate a manual Crop Rows generation trough digitizing in QGIS software.

C1.1 Requirements

Below there is a list of requirements necessary to evaluate the Crop Rows generation by digitization method.

- A laptop or desktop PC.
- An installed QGIS software.
- An orthomosaic image of a sugarcane field.
- A chronometer.
- A digital spreadsheet of the evaluation format.

C1.2 Protocol

Task 1. Load Orthomosaic into QGIS Software.

- Open QGIS
- Find and open Orthomosaic (**Add Raster layer**)
- Create a new vector layer type (**Line**)

  - Layer − > Create Layer − > New Shapefile Layer
  - Type: Line
  - SRC: Assign the same as the Orthomosaic
- Save in results folder with *"test.shp"* file name

Task 2. Digitize a crop row line

- Click on Toggle Editing tool 
- Lines are created by left clicking with the Add Features tool.  .
- Start Stopwatch.
- Left click on at the beginning of the crop row.
- Left click at the end of crop row.
- Right click to finish crop row digitizing process.
- In pop-up window: type crop row id.
- Finish the Stopwatch.
- Record time in the spreadsheet of evaluation format.

Repeat *Task 2* until finishing all the crop rows in the orthomosaic.

- Switch edition
- Save All Changes

Task 3. Calculate Crop Rows length

- Open table attributes of the layer *"test.shp"*
- Open Field Calculator
  - Create new field
  - Assign field name: length
  - Decimal format (real)
  - Use the following expression for calculating the length field: *geometry*− > $*length*

- Click on Toggle Editing tool.
- Save All Changes
- Record crop rows length column in the spreadsheet of evaluation format.

C1.3 Evaluation format spreadsheet

Crop Rows Digitizing Time

Digitizer Name:	

Raster Filename:	
Start time:	
End time:	

Crop Rows	Time (s)	Distance (m)
Line 1		
Line 2		
Line 3		
Line 4		
Line 5		
Line 6		
Line 7		
Line 8		
Line 9		
Line 10		
Line 11		
Line 12		
Line 13		
Line 14		
Line n		

Average		
Max		
Min		
Sd		
Sum		

Figure C.1.: Digitizing format spreadsheet example for n crop rows.

C1.4 Evaluation metrics

C1.4.1 Average time

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{C.1}$$

where:

$\bar{x}$ = the average time,

x_i = digitizing time for each i crop row in seconds,

n = total number of crop rows.

C1.4.2 Maximum time

$$X = \max(x_1, ..., xn), \tag{C.2}$$

where:

X = maximum digitizing time,

x_i = digitizing time for each i crop row in seconds,

n = total number of crop rows.

C1.4.3 Minimum time

$$X = \min(x_1, ..., xn), \tag{C.3}$$

where:

X = minimum digitizing time,

x_i = digitizing time for each i crop row in seconds,

n = total number of crop rows.

C1.4.4 Standard deviation time

$$Sd_{=}\sqrt{\frac{\sum_{i=1}^{n} (x_i - \overline{x})^2}{n-1}}, \tag{C.4}$$

where:

Sd = The standard deviation digitizing time of all crop rows,

x_i = digitizing time for each i crop row in seconds,

$\overline{x}$ = the average time, n = total number of crop rows.

C1.4.5 Total time

$$X = \sum_{i=1}^{n} x_i, \tag{C.5}$$

where:

X = total time in seconds,

x_i = digitizing time for each i crop row in seconds,

n = total number of crop rows.

C1.4.6 Average distance

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}, \tag{C.6}$$

where:

$\bar{x}$ = the average distance,

x_i = length in meters for each i crop row,

n = total number of crop rows.

C1.4.7 Maximum distance

$$X = \max(x_1, ..., xn), \tag{C.7}$$

where:

X = maximum digitizing time,

x_i = length for each i crop row,

n = total number of crop rows.

C1.4.8 Minimum distance

$$X = \min(x_1, ..., xn), \tag{C.8}$$

where:

X = minimum digitizing distance,

x_i = length in meters for each i crop row,

n = total number of crop rows.

C1.4.9 Standard deviation distance

$$Sd_{=}\sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}, \tag{C.9}$$

where:

Sd = the standard deviation of the length for all crop rows,

x_i = length in meters for each i crop row,

$\overline{x}$ = the average distance, n = total number of crop rows.

C1.4.10 Total distance

$$X = \sum_{i=1}^{n} x_i, \tag{C.10}$$

where:

X = total distance in meters,

x_i = length in meters for each i crop row,

n = total number of crop rows.

C1.5 Composed metrics

C1.5.1 Crop Rows Quantity per Minute

$$C_q = \left[\frac{n}{\frac{\sum_{i=1}^{n} x_i}{60}} \right], \tag{C.11}$$

where:

C_q = total crop rows per minute,

x_i = digitizing time for each i crop row,

n = total number of crop rows.

C1.5.2 Kilometers of Crop Rows per Minute

$$C_k = \left[\frac{\frac{\sum_{i=1}^{n} x_i}{1000}}{\frac{\sum_{i=1}^{n} t_i}{60}} \right], \tag{C.12}$$

where:

C_k = total kilometers of crop rows per minute,

x_i = length in meters for each i crop row,

t_i = digitizing time for each i crop row in seconds,

n = total number of crop rows.

D. Appendix: Crop Rows Generator User Manual

Version No	Date	Author	Comments/Modifications
1.0	24-07-2018	Andres Herrera	Release version

D1. Installation Process

This section describes the installation steps for *Crop Rows Generator - CLI* , *Crop Rows Generator - API* and *Crop Rows Generator - QGIS Plugin*, as well as main operations that can be performed.

D1.1 Crop Rows Generator - CLI

Docker installation is required and a container with a certain image which is provided in the attached DVD.

```
1. Update the apt package index
  $ sudo apt-get update
2. Docker installation
  $ sudo apt install docker.io
3. Once installed, enable the Docker daemon at boot.
  $ sudo systemctl start docker
  $ sudo systemctl enable docker
4. Restart the Docker daemon
  $ sudo systemctl restart docker
5. Deploy croprows container
  $ sudo docker load croprowscontainer.tar
6. Restart the Docker daemon again
  $ sudo systemctl restart docker
```

D1.2 Crop Rows Generator - API

```
1. Python installation
  $ sudo apt-get install python-setuptools python-pip python-dev build-essential
  $ sudo pip install --upgrade pip
2. Flask microframework installation
  $ sudo pip install Flask
```

D1.3 Crop Rows Generator - QGIS Plugin

- Download file **croprows-qgis-plugin-latest.zip** available on the Crop Rows Generator repository.
- Extract **croprows-qgis-plugin-latest.zip** into the .qgis2/python/plugins folder located in home directory.
- Start QGIS Desktop.
- Activate **Crop Rows Generator** Plugin in QGIS Desktop Plugin Manager. To open the Plugin Manager, click on the menu item Plugins − > Manage and Install Plugins ..

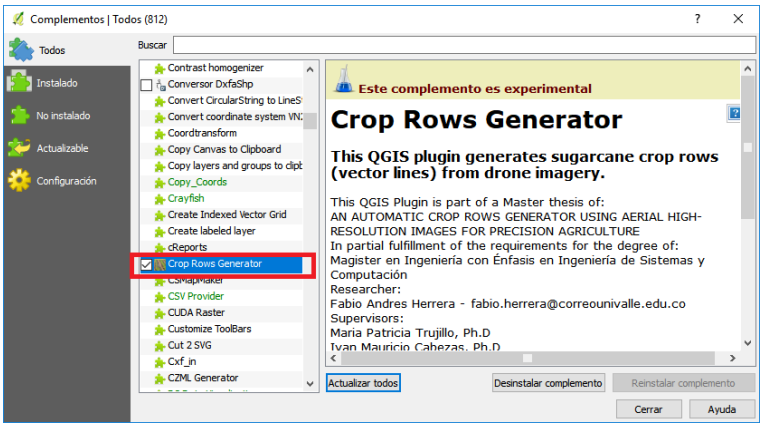

Figure D.1.: Dialog to activate *Crop Rows Generator* plugin in QGIS Desktop plugin manager.

Install OSGeo4W dependency

- Download the 32 bit or 64 bit OSGeo4W network installer avaliable at https://trac.osgeo.org/osgeo4w/
- Run the installer.
- Select Express Desktop Install, and Next.
- Pick QGIS, GDAL and GRASS GIS packages to install, and Next.

  The selected packages and their required sub packages will be downloaded and installed automatically.

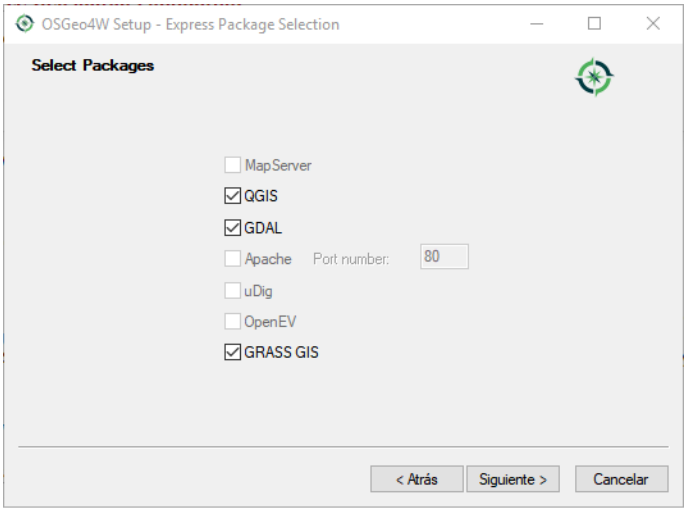

Figure D.2.: OSGeo4W setup dialog

D2. User manual

D2.1. Crop Rows Generator - CLI

```
1. Perform a crop rows processing task
  $ ./run_croprows-cli.sh croprowsproject.xml
2. Show help
  $ ./run_croprows-cli.sh -h
3. Perform a crop rows processing task in serial mode
  $ ./run_croprows-cli.sh croprowsproject.xml -s
```

4. Perform a crop rows processing task in parallel mode

```
$ ./run_croprows-cli.sh croprowsproject.xml -s
```

5. Perform a crop rows processing task with out debug messages

```
$ ./run_croprows-cli.sh croprowsproject.xml -debug
```

6. Perform a crop rows processing task with tile size 10 meters

```
$ ./run_croprows-cli.sh croprowsproject.xml -t 10
```

Figure D.3.: CRG-CLI help dialog

D2.2. Crop Rows Generator - API

1. Start server API

```
$ ./run_croprows-api.sh
```

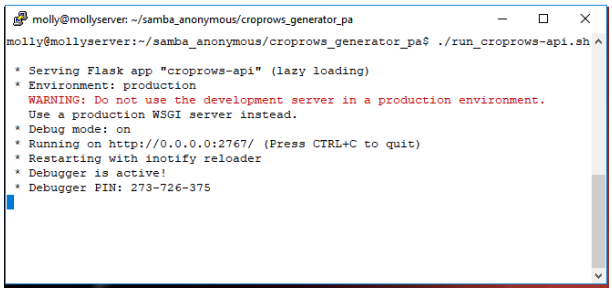

Figure D.4.: CRG-API Running on port: 2767

API Usage:

METHOD	URL	Response
GET	http://192.168.0.6:2767/imlive	Crop Rows API Status (JSON)
GET	http://192.168.0.6:2767/os	OS information (JSON)
GET	http://192.168.0.6:2767/croprows/**croprows-projectfile.xml**	Execute Crop Rows processing task for **croprows-projectfile.xml**
POST	http://192.168.0.6:2767/crimageuploader/	@image=filename.tif **Example:** curl -i -X POST "Content-Type: multipart/form-data" \ -F "image=@uploadfile.tif" http://192.168.0.6:2767/crimageuploader
POST	http://192.168.0.6:2767/crmaskuploader/	@shp=file.shp, @shx=file.shx, @dbf=file.dbf **Example:** curl -i -X POST "Content-Type: multipart/form-data" \ -F "shp=@filemask.shp" \ -F "shx=@filemask.shx" \ -F "dbf=@filemask.dbf" http://192.168.0.6:2767/crmaskuploader

Figure D.5.: CRG-API usage

D2.3. Crop Rows Generator - QGIS Plugin

Steps to generate sugarcane Crop Rows

The following contains a steps to generate geospatial vector files of sugarcane crop rows with *CRG - QGIS Plugin.*

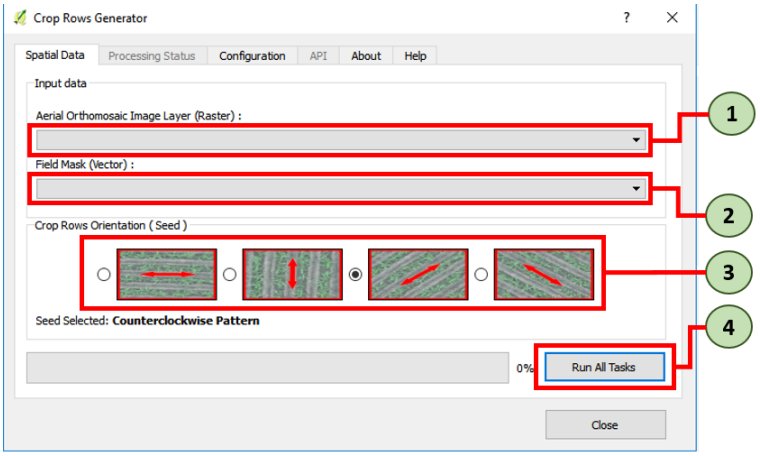

Figure D.6.: Spatial data tab of the Crop Rows Generator dialog window

Step 1

1. Select the **Raster** Image File previously loaded on QGIS TOC.
2. Select **Vector** Mask File previously loaded on QGIS TOC.
3. Select **Seed** value according to crop rows orientation.
4. Click on **Run All Task** button to perform the process.

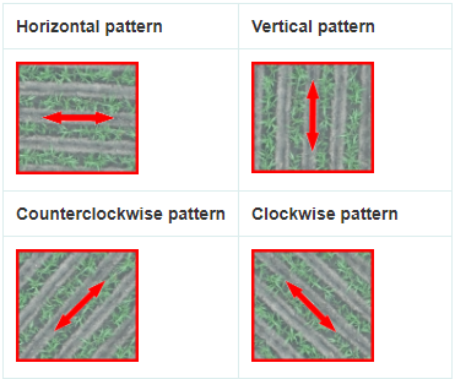

Figure D.7.: Crop rows orientation patterns

Step 2

Once you Click on **Run All Task** button in the popup dialog you must be click on **(yes/si)** button.

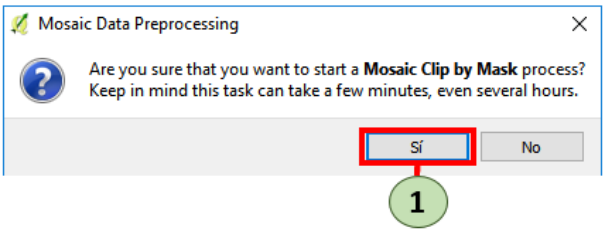

Figure D.8.: Mosaic Clip by Mask message in confirm dialog box

Note: Mosaic clipping process can take a few minutes, even several hours dependes of the size of the Raster Mosaic and the complex shape of the vector mask file.

Step 3

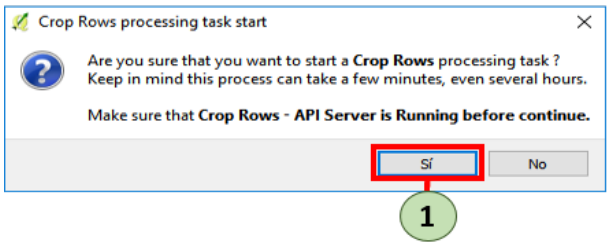

Figure D.9.: Processing task start message in confirm dialog box

Click on **(yes/si)** button to perform the task.

Note: Crop Rows processing task can take a few minutes, even several hours depends of the size of the Raster Mosaic and crop rows to detect. Also processing time may be affected by processing server where CRG-CLI is running.

Step 4

Tab page *Processing Status* will show the process of Crop Rows generation.

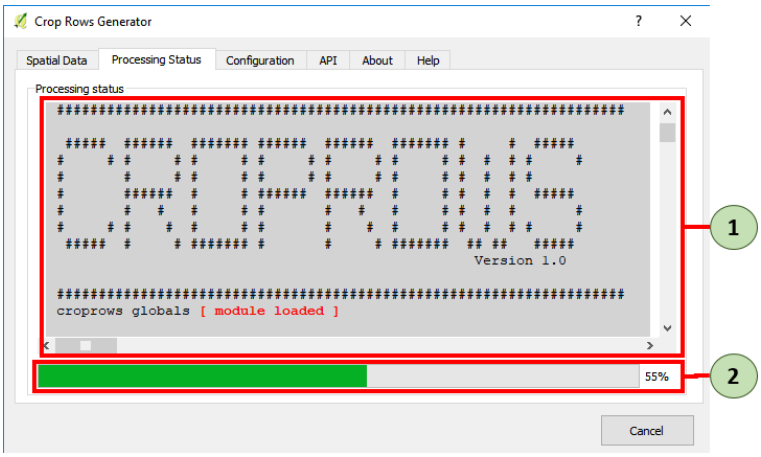

Figure D.10.: Processing status tab of the Crop Rows Generator dialog window

1. CRG-CLI standard output.
2. Status bar for Crop Rows generation process.

Note: Few minutes, even hours. This process can be take to complete.

Step 5

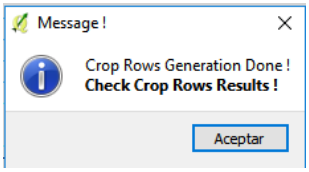

Figure D.11.: Process finished message in alert dialog box

Note: Once the process is complete, the resulting data will be loaded into the QGIS TOC.

D2.3.1. Results of Crop Rows generation

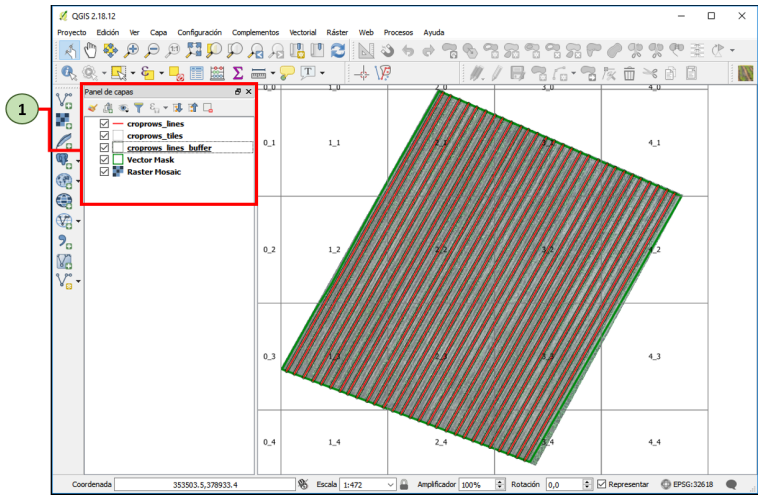

Figure D.12.: Crop Rows results loaded in QGIS TOC

1. A new vector file named **croprows_lines** is loaded on QGIS TOC.
2. A new vector file named **croprows_lines_buffer** is loaded on QGIS TOC.
3. A new vector file named **croprows_tiles** is loaded on QGIS TOC.

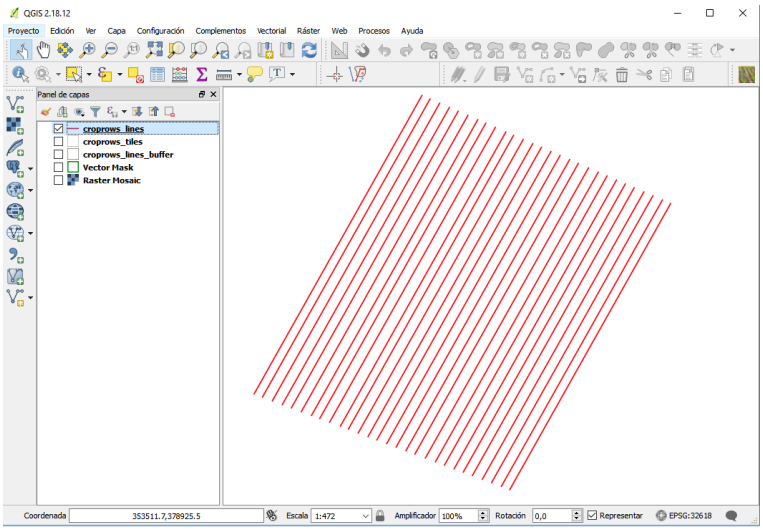

Figure D.13.: Spatial data with the croprows_lines file loaded in QGIS TOC

1. Generated **croprows_lines** has the same reference system as the input data.
2. Each of the detected lines (Crop Rows) is an independent entity.

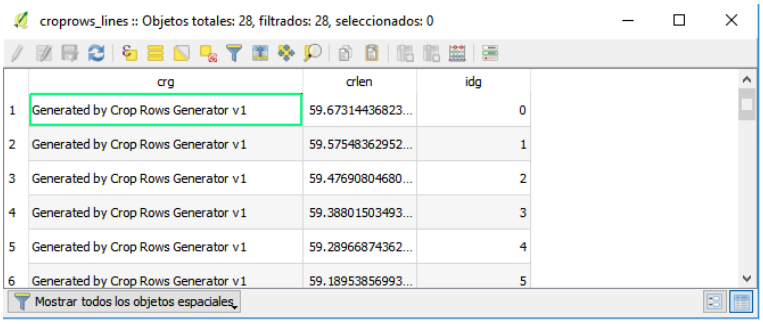

Figure D.14.: Alphanumeric data of the croprows_lines file displayed in QGIS

1. Generated **croprows_lines** has related alphanumeric data.
2. Each one of the tuples is a detected line (Crop Rows).

   - **idg**: unique line identifier.
   - **crlen**: length of detected line (meters).

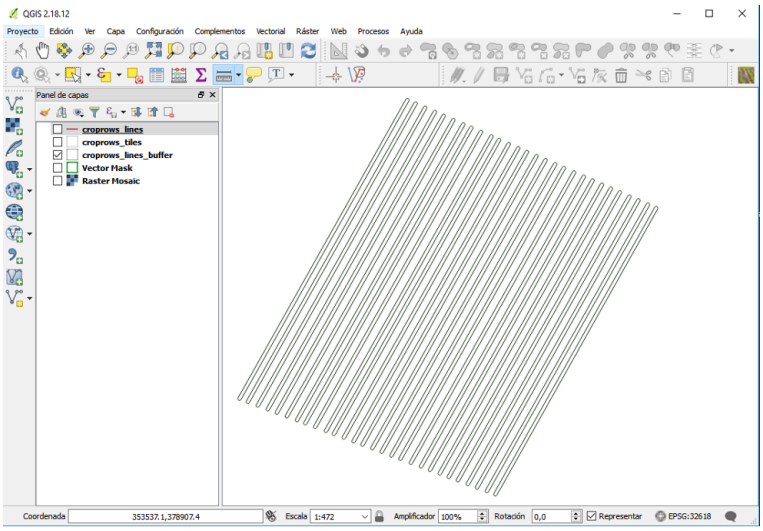

Figure D.15.: Spatial data with the croprows_lines_buffer file loaded in QGIS TOC

1. Generated **croprows_lines_buffer** has the same reference system as the input data.
2. Each of the polygon is an independent entity.
3. Corresponds to an influence area of 0.60 cm for each detected crop row.

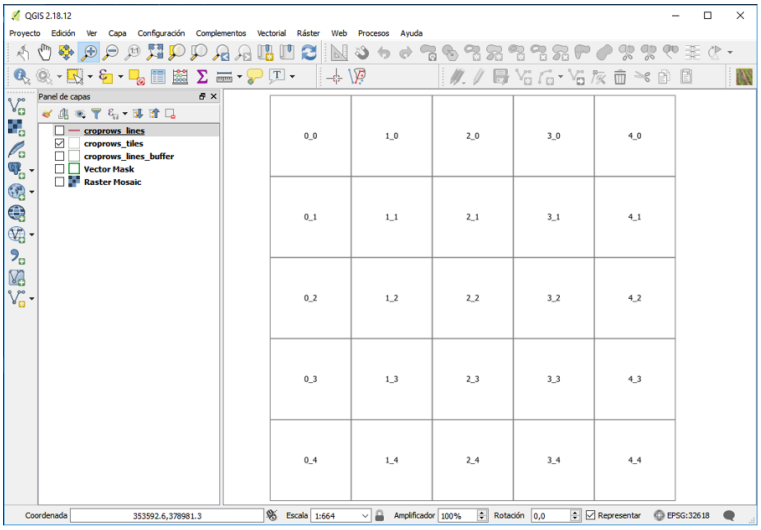

Figure D.16.: Spatial data with the croprows_tiles file loaded in QGIS TOC

1. Generated **croprows_tiles** has the same reference system as the input data.
2. Each one of the tile is an independent entity.
3. Each vector tile corresponds usually a processed tile of 20 m x 20 m.

D2.3.2. Results in Export Folder

1. A generated **croprows_wgs84.kml** file with a assigned geographic reference system (WGS84/EPSG:4326) is located. File supported by Google Earth software.
2. A generated **croprows_wgs84.shp** file with a assigned geographic reference system (WGS84/EPSG:4326) is located. File supported by most of the auto-guidance computer panels.

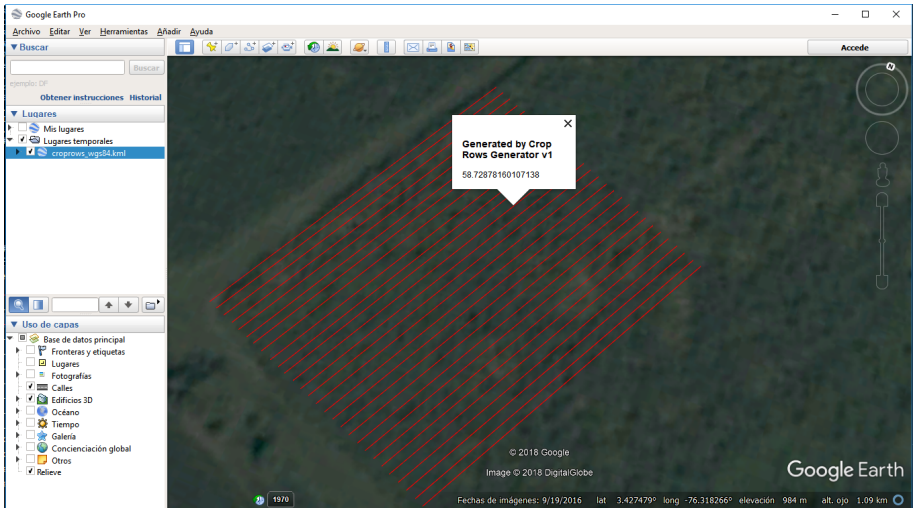

Figure D.17.: Crop Rows results displayed in Google Earth

D2.3.3. CRG Configuration Parameters

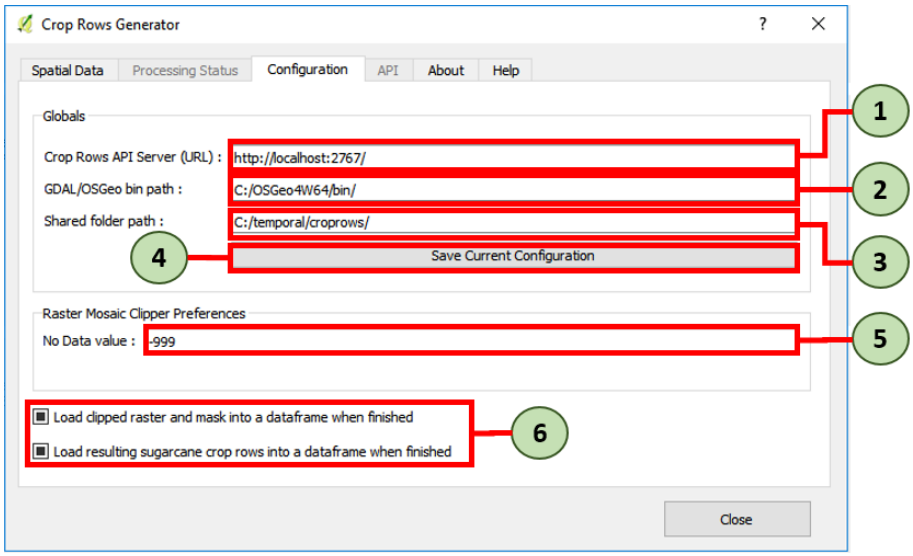

Figure D.18.: Configuration parameters tab of the Crop Rows Generator dialog window

1. Set where CRG-CLI processing server is running (port 2767 is set by default).
2. Set where OSGeo4W binaries are installed on host machine.
3. *Save Current Configuration* button for applying changes.
4. Set *No data value* for Raster Mosaic clipper processing (-999 value by default).
5. Check this options if you need to load the results into QGIS data frame and TOC when processes are finished (options are checked by default).

E. Appendix: Software Requirements Specification for Crop Rows Generator

Version No	Date	Author	Comments/Modifications
1.0	28-03-2018	Andres Herrera	Release Version

E1. Introduction

E1.1. Purpose

The purpose of this section is to present a detailed description of the open-source software **Crop Rows Generator v1.0** . It will explain the purpose and features of the software, the interfaces of the software, what the software will do and the constraints under which it must operate. This document is intended for users of the software and also potential developers.

E1.2. Document Conventions

This Document was created based on the IEEE template for System Requirement Specification Documents.

E1.3. Intended Audience and Reading Suggestions

- Advanced/Professional Users, such as engineers, cartographers, gis analyst or researchers, who want to use Crop Rows Generator for obtain georeferenced crop rows data in sugarcane crops.

- Programmers who are interested in working on the project by further developing it or fix existing bugs.

E1.4. Product Scope

Crop Rows Generator (CRG) v1.0 is a tool that specialized user can use to generate georeferenced lines from high resolution images obtained by a drone. (CRG) uses an orthomosaic and a field delimitation vector mask as an input. After processing CRG obtains at the output a set of files that can be used to be integrated into geographic information systems or to be provided as an input for an auto-guidance machinery. (CRG) is based on computer vision techniques and under a high performance computing approach is capable of processing high resolution and large images and on these images detect, generate and mapping a crop rows in sugarcane fields in an easy and accurate way,with a few clicks. CRG has a client interface (*CRG-QGIS Plugin*) integrated as a plugin into the geographic information system software (QGIS Desktop). A processing core that handles the most complex tasks and is accessed by a command line interface (*CRG-CLI*). A REST API (*CRG-API*) which exposes resources through on HTTP methods.

E1.5. Resources

- Crop Rows Generator (CRG) v1.0 website:
  http://54.186.237.120/croprowsgenerator/

- Crop Rows Generator (CRG) GitHub page:
  https://github.com/AndresHerrera/croprows_generator_pa

- Crop Rows Generator (CRG) Youtube Playlist:
  https://www.youtube.com/playlist?list=PL5Uf6W3KZ2JBKwHgeifJJUgeGkOZviSIK

- IEEE Template for System Requirement Specification Documents:
  https://goo.gl/nsUFwy

E2. Overall Description

E2.1 Product Perspective

CRG was developed for everyone who is interested in generate geospatial crop rows in sugarcane fields using high resolution imagery captured by a Drone. It is an open source project. It was developed to run on Windows and Linux Operating system.

E2.2 Product Functions

E2.2.1 Crop Rows Generator - QGIS Plugin

- Select Aerial Orthomosaic Image Layer
- Select Field Mask
- Run All Tasks
- Save Current Configuration
- Close

E2.2.2 Crop Rows Generator - CLI

- **-h** show help.
- **-p** processing crop rows in parallel mode.
- **-s** processing crop rows in serial mode.
- **-debug** disable debug messages
- **-t** change processing raster tile size.

E2.2.3 Crop Rows Generator - API

Crop Rows - CLI Operations exposed through HTTP REST API

- **http://server:2767/imlive** - GET
  Crop Rows API Status (JSON Response).

- **http://server:2767/os** - GET

  OS information (JSON Response).
- **http://server:2767/croprows/croprows-projectfile.xml** - GET

  Execute Crop Rows processing task for **croprows-projectfile.xml**
- **http://server:2767/crimageuploader/** - POST

  Upload orthomosaic file to the server.
- **http://server:2767/crmaskuploader/** - POST

  Upload vector mask file to the server.

E2.3. User Classes and Characteristics

- Advanced/Professional Users, such as engineers, cartographers, gis analyst or researchers, who want to use Crop Rows Generator for obtain georeferenced crop rows data in sugarcane crops.
- Programmers who are interested in working on the project by further developing it or fix existing bugs.

E2.4. Operating Environment

E2.4.1 Crop Rows Generator - QGIS PLUGIN

Operating System

- Windows 10

  QGIS Desktop
- QGIS 2.18 "Las Palmas"

E2.4.2 Crop Rows Generator - CLI and API

- **Operating System**

  Linux Ubuntu Xenial 16.04
- **Docker**

  Docker 1.13

- **Python**

  Python 2.7.12

- **Flask microframework**

  Flask 1.0.2

E2.5 Design and Implementation Constraints

Crop Rows Generator is developed in Python, it uses the following libraries.

- **CRG - Plugin**

  PyQt4

  qgis

  ElementTree

  urlib2

  socket

  functools

  ogr

  gdal

- **CRG - CLI**

  OpenCV

  SciPy

  NumPy

  gdal

  ogr

  pandas

  geopandas

  shapely

  joblib

  multiprocessing

- **CRG - API**

  Flask

E2.7 User Documentation

There is a quick start guide avaliable on the GitHub website of Crop Rows Generator: https://github.com/AndresHerrera/croprows_generator_pa

E2.8 Assumptions and Dependencies

CRG is developed in Python and therefore requires Python to be installed on the user's system. CRG requires Python version 2.7 or higher. This applies to Windows and Linux users.

E3. External Interface Requirements

E3.1 User Interfaces

1. Spatial data tab:

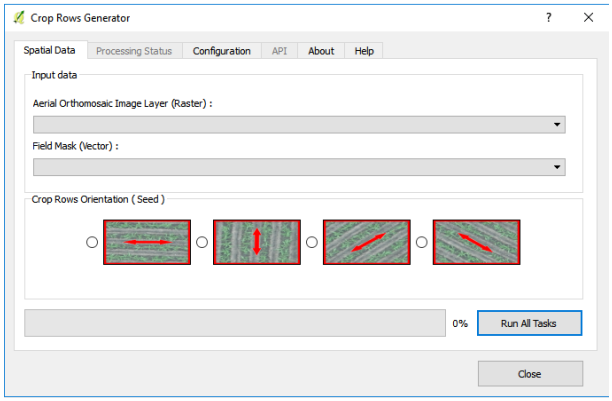

Figure E.1.: CRG-QGIS Plugin spatial data screen

3. Processing status tab:

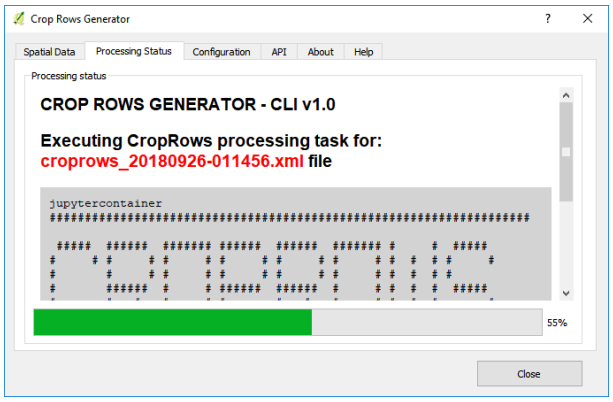

Figure E.2.: CRG-QGIS Plugin processing status screen

2. Configuration tab:

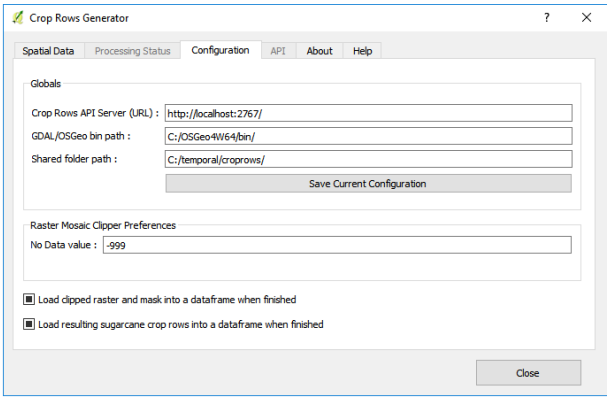

Figure E.3.: CRG-QGIS Plugin configuration screen dialog box

3. API tab:

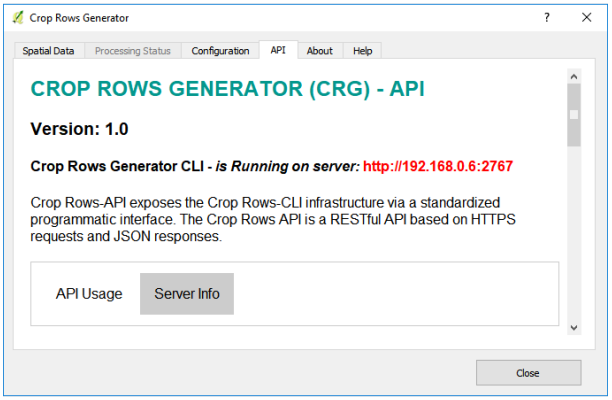

Figure E.4.: CRG-QGIS Plugin API screen dialog box

4. About tab:

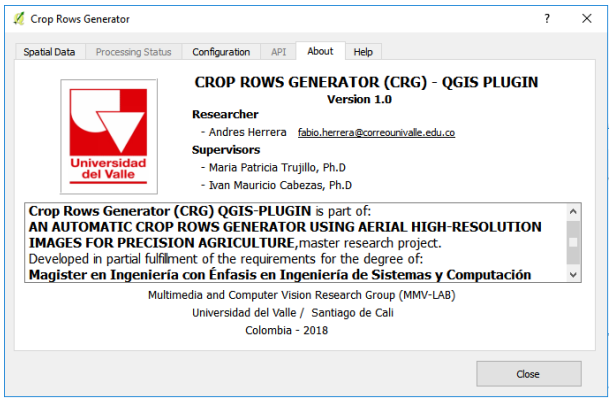

Figure E.5.: CRG-QGIS Plugin about screen dialog box

5. Help tab:

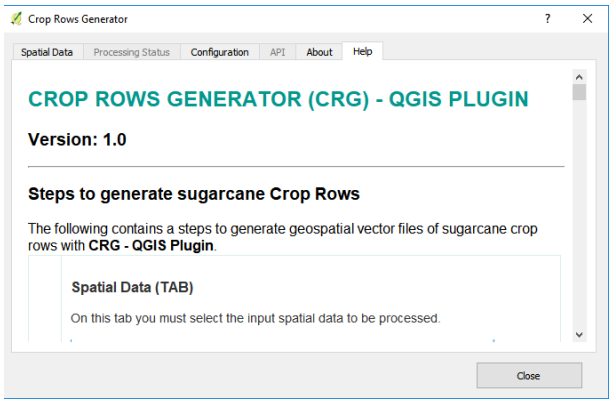

Figure E.6.: CRG-QGIS Plugin help screen dialog box

E3.2 Hardware Interfaces

The minimum hardware requirements of CRG are a 2.0 Megahertz CPU and 2 Gigabytes of RAM.

E3.3 Software Interfaces

CRG requires Python to be installed on the system, more specifically Python version 2.7.

E4. System Features

This section demonstrates CRG most prominent features and explains how they can be used and the results they will give back to the user.

E4.1 Generate Crop Rows

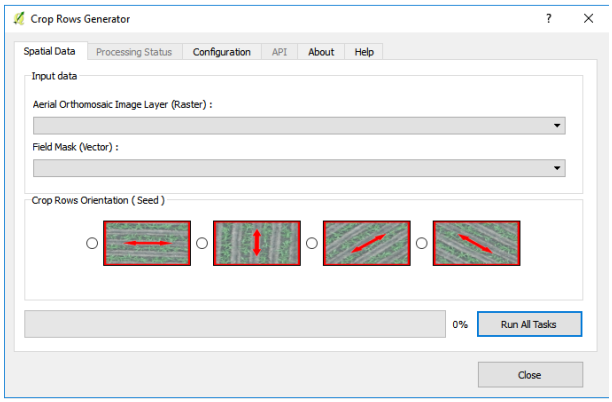

Figure E.7.: CRG-QGIS generate crop rows screen dialog window

For generate geospatial vector files of sugarcane crop rows with CRG - QGIS Plugin the following steps must be followed.

1. Select the **Raster** Image File previously loaded on QGIS TOC.
2. Select **Vector** Mask File previously loaded on QGIS TOC.
3. Select **Seed** value according to crop rows orientation.
4. Click on **Run All Task** button to perform the process.
5. **Wait for results ...**
6. Once the processes were complete, the resulting data will be loaded into the QGIS TOC.

E4.2 Configuration changes

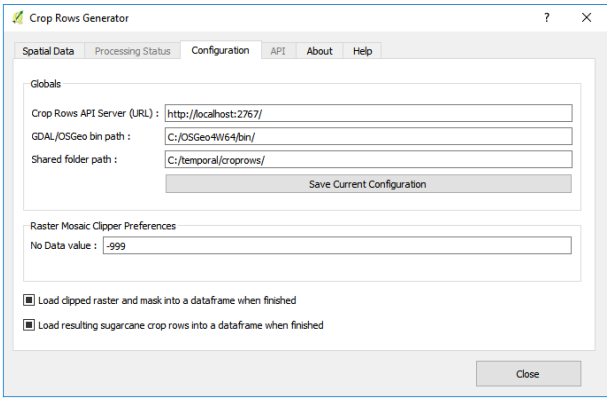

Figure E.8.: CRG-QGIS change configuration screen dialog box

If a change to the default configuration is required.

1. Edit values in the box of interest.
2. Click on **Save Current Configuration** button if changes are made.

E5. Other Nonfunctional Requirements

E5.1 Performance Requirements

CRG requires a system with at least a 2.0 Megahertz CPU and 2 Gigabytes of RAM. However, these requirements can support effectively processing small crop field areas. Performance depends on the orthomosaic size and as a result, the system requirements for bigger orthomosaics are more demanding.

E5.2 Security Requirements

CRG does not have any security requirements and thus any type of user can use it without any additional privileges.

F. Appendix: CRG v1.0 Survey

F1. Questions

1. Are you using your computer daily?

   ( ) Yes ( ) No
2. GIS Level

   (1) Newbie (2) (3) (4) (5) Expert
3. Do you often use QGIS ?

   (1) Never (2) (3) (4) (5) Often
4. Did you have troubles understanding the Crop Rows Generator (CRG) prototype?

   ( ) Totally complicated ( ) A little bit confused ( ) It is too easy to understand
5. Navigation is consistent and easy to use

   (1) Strongly Disagree (2) (3) (4) (5) Strongly Agree
6. The interface design is appropiate to the audience?

   (1) Strongly Disagree (2) (3) (4) (5) Strongly Agree
7. The images used to select the "seed" for Crop Rows orientation input are appropriate?

   (1) Not appropiate (2) (3) (4) (5) Very appropiate
8. Do you find useful to get the Crop Rows results in WGS84 CRS ?

   (1) Not useful (2) (3) (4) (5) Very useful
9. How convenient Crop Rows Generator (CRG) could be fit your processes?

   (1) Not convenient (2) (3) (4) (5) Very convenient
10. How likely is it that you will recommend Crop Rows Generator (CRG) to a friend or colleague?

    (1) Not likely (2) (3) (4) (5) Very likely

Form in digital version at Google Forms: https://goo.gl/forms/qaEzHt4YheSQvW3k1

F2. Results

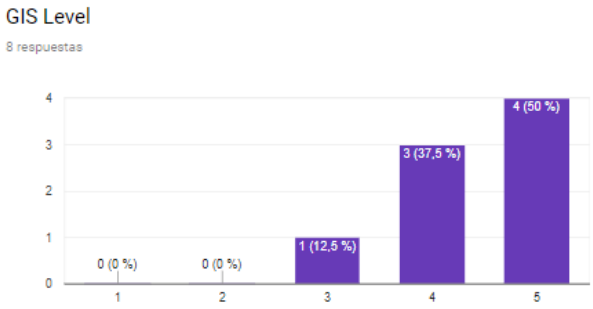

Figure F.1.: Answers to the question No.1

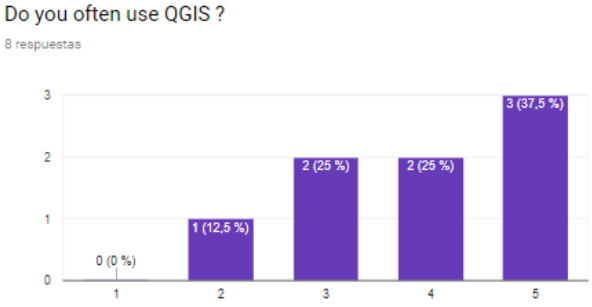

Figure F.2.: Answers to the question No.2

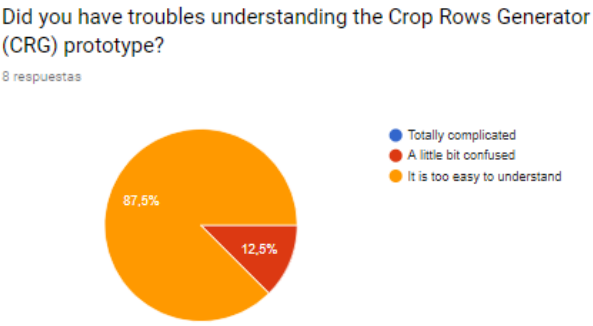

Figure F.3.: Answers to the question No.3

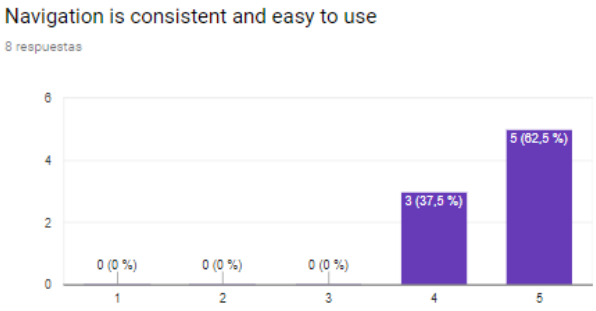

Figure F.4.: Answers to the question No.4

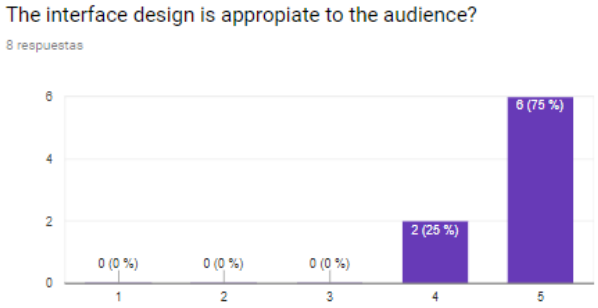

Figure F.5.: Answers to the question No.5

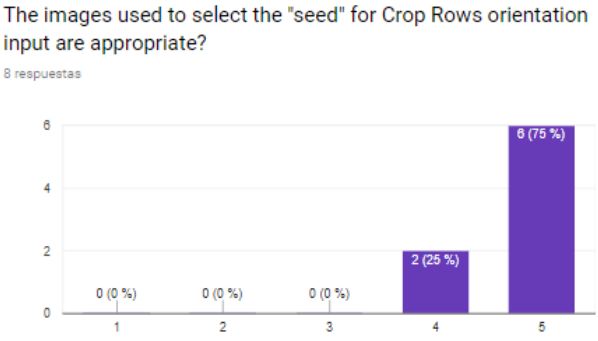

Figure F.6.: Answers to the question No.6

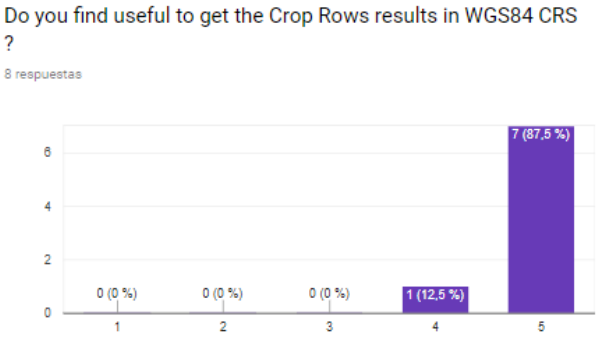

Figure F.7.: Answers to the question No.7

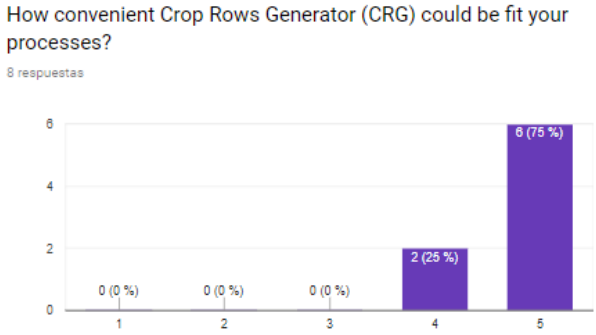

Figure F.8.: Answers to the question No.8

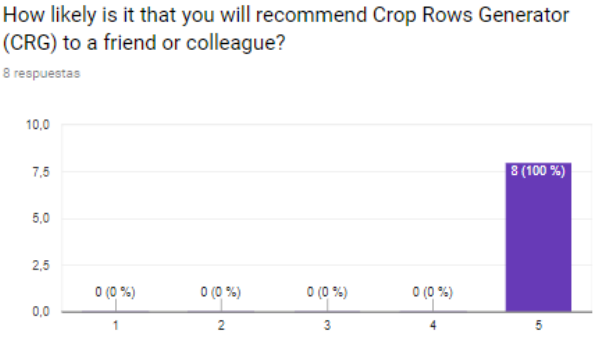

Figure F.9.: Answers to the question No.9

G. Appendix: Source Code and Demos

G1. Source Code

All of the source code of *Crop Rows Generator v1.0* is available together with the associated tools that have been used to develop this software on the GitHub repository of the project:

https://github.com/AndresHerrera/croprows_generator_pa/

G2. Demo Videos

Below are some videos that show how *Crop Rows Generator v1.0* works.

Demo 1: Crop Rows Generator (Simple Field) - **4:03**

https://www.youtube.com/watch?v=ymBDy9mDqbQ

Demo 2: Crop Rows Generator (Complex Field) - **3:36**

https://www.youtube.com/watch?v=8-V-RCDlD-k

Demo 3: Crop Rows Generator (Simple Field 2) - **4:23**

https://www.youtube.com/watch?v=VqmkviVmdPo