Norwegian University of Life Sciences

Masteroppgave 2022 (Høst) 30stp
Fakultetet for realfag og teknologi

An Interactive Optimal Path Planning App for Farming Machinery by GIS Approach

Mesenbet Alemu Delele
Instituttet for Geomatkk

Preface

This thesis is to conclude a master degree in geomatic in the Norwagian University of Environment and Life Sciences, NMBU.

First of all, I thank my almighty God who gave me the strength and converting my endeavour to success. I would like to thank my supervisors Seyed Hossein Chavoshi from the Faculty of Science and technology and Nils Bjugstad from dept. of mathematical science and technology for being of friendly helpers especially in proofreading and feedback , general and specific comments without it this thesis would have not been coming to a reality.

I would also like to thank my family for being considerate and let me work harder and for being thoughtful for me in each and every step of my journey and special thanks to my children who were my sources of smile even in the impossible times.

Sammendrag

Det “coverage path planning problem”, CPP er en av hovedforskningslinjene innen robotikk og landbruk. Det er mange vellykkede verk i litteratur i denne forbindelse. Imidlertid er hovedmålene med dette arbeidet først og fremst å foreslå alternativ offline løsning for dekningsveiplanlegging (CPP) problem som en installerbar gratis app kalt PlowPlaner gjennom integrering av kunnskap fra GIS- og Python-biblioteker. Og for det andre å introdusere en ny tilnærming for å finne optimalisert retning kalt minimum rotert rektangel (MRR) i splitting av region av interesse (ROI) i celler (underformer) og ved utforming av optimaliserte dekningsbaner for underformene.

For å utvikle denne app-ideene fra tidligere CPP-relaterte studier, var GIS- og geomatikkkonsepter, spesielt transformasjon av geografiske objekter, kjernekonseptene. Åpne gatekart som basiskart, og Python-biblioteker er integrert.

Det er mange kommersielle online og offline-baserte apper og verktøy der ute, og disse er utviklet med tanke på at de vil være nyttige for større eiendomseide og mer utdannede og teknologibevisste bønder. Det er imidlertid behov for en enklere og gratis app som hver skalabonde kan ha tilgang til. PlowPlaner, en alternativ CPP-app, foreslått for å tjene alle typer bønder i planlegging av optimalisert dekningsbane offline, ble vellykket utviklet gjennom denne metoden.

Suksessen til PlowPlaner har implikasjonen at den alternative løsningen gjennom å integrere kunnskap for CPP-problemet er lovende og bør utnyttes mer i prosessen med å finne løsninger på dette og relaterte problemer.

Abstract

The coverage path planning problem, CPP is one of the main research line in Robotics and Agriculture. There are many successful works in literature in this regard. However the main aims of this work is firstly to propose alternative offline solution for coverage path planing (CPP) problem as an installable free app called PlowPlaner through the integration of knowledge from GIS, and Python libraries. And secondly to introduce a new approach of finding optimized direction called minimum rotated rectangle (MRR) in splitting of region of interest (ROI) into cells (sub-shapes) and in designing optimized coverage paths for the sub-shapes.

To develop this app ideas from earlier CPP related studies, GIS and geomatics concepts especially transformation of geographic objects were the core concepts. Open street map as base-map, and Python libraries have been integrated.

There are many commercial online and offline based apps and tools out there and these are developed having in mined that they will be useful for bigger estate owned and more educated and technology aware farmers. However there is a need for more simpler and free app that every scale farmers can have access. PlowPlaner, an alternative CPP app, proposed for aiming to serve all kinds of farmers in planing optimized coverage path offline was successfully developed through this method.

The success of PlowPlaner has the implication that the alternative solution through integrating knowledge for CPP problem is promising and should be exploited more in the process of finding solutions in this and related problems.

Table of Contents

Table of Figures

List Of Tables

List Of Equations

List of abbreviations

Base Map : It is a layer with geographic information that serves as a background.

cells/sub shape/Sub-shape : the smallest striped shapes resulted from a splitting function after it has splitted the free space into optimized cells

CPP: Coverage Path Planning : It is a navigation planning for a robot inside a free space

CSV : comma-separated values

Free space : It is a apart of the ROI after the Prohibited areas/holes are subtructed.

GIS : Geographic Information System

GPS : Global Positioning System

GUI : stands for Graphical User Interfaces

hole/prohibited area: It is an area inside the ROI which is inaccessible/will not be tilled because of some natural/man-made structures positioned in that specific area.

Incomplete path space/IPS: is a strip that remained after a sub-shape is covered with paths==> a path has a width equal to the width of the machine but incomplete paths are having a smaller width than the width of the machine they are small stripes remaining after the sub-shape covered by paths

IPS/incomplete path space: is a strip that remained after a sub-shape is covered with paths==> a path has a width equal to the width of the machine but incomplete paths are having a smaller width than the width of the machine they are small stripes remaining after the sub-shape covered by paths

Lat/Lon: the latitude and longitude of a point near, on, or around the earth.

MRR : Minimum Rotated Rectangle

NCAP : stands for None Convex Angle Points

OSM : Open Source Map

Path: It is an area/strip having the same width as the machine's effective working area it is a line that the machine follows during the activity in the field

PD : Path distance it is the length of

Project area/ROI : Region of Interest

Reference Measurement: It is any type of known measurement used to evaluate a proposed measurement through comparison.

ROI/Project area : Region Of Interest

Shapely: It is a python library concerned with shapes. It enables transformation and other manipulation of geographic objects.

Sub-shape/sub shape/cells : the smallest striped shapes resulted from a splitting function after it has splitted the free space into optimized cells

TkinterMapView: It is a python Library that can facilitate OSM into tkinter.

TPD : Total Path Distance it is the sum of the lengths of all paths in a sub-shape or ROI

Turning Distance: It is an unworkable distance in a full path, during which the machine turns to change its direction/path to proceed with its next movement.

WKRD: It is a well-known distance used as a reference, In this work, this word is used to say the TPD of a sub-shape whose path direction is found by selecting the best direction out of 180 degrees with every 1-degree interval. i.e. The TPD for every 180 degrees with a 1-degree interval is collected and the minimum of all the collected TPD will be considered as WKRD.

CHAPTER 1. INTRODUCTION

1.1 Problem Statement

In practical agriculture, tractors and self-propelled farming machines are traditionally driven by human drivers. Human drivers solely design the driving strategy for every single field without any assistance. They choose\their strategies based on the type of tasks, type of machine, and their experience [1]. This might seem to be an effortless task if the given field shape is rectangular, or, any other convex shape with manageable size and without inaccessible areas inside. However, in practice fields are more complex and often sizable non-convex shapes with many obstacles inside. The best driving strategy can be achieved by automating the entire process for any type of field of any shape and size. By minimizing human intervention, the best result would be planning optimized paths with fewer turnings that cover the entire field. Realizing automation for field agricultural machines is required to tackle many current problems namely food insecurity around the world [2], expensive costs for farming chemicals, driving staff, acute labor, and daily wages [3], excessive driving effects on the environment [4], the operators' limitations, and fatigue during the long and busy days in bigger harvesting projects [5].

There have been already attempts from different disciplines including agriculture domain to produce scientific solutions as supply to the outstanding demand agricultural machine's driving strategy in fields [6], [7]. Precision agriculture, also known as site-specific farming, is being developed and put into practice thanks in large part to scientists working in the field of the combination of Global Positioning System (GPS) and Geographic Information Systems (GIS). The primary approach to solving the issue is to find ways to computerize and make agricultural machinery intelligent through the integration of technologies. This strategy has so far led to the development of numerous automated solutions algorithms and approaches.

According to Pham [8], an automated system is typically a system that should determine the following key aspects: what is the task that needs to be completed, what are the steps that need to be taken to make it completed, where the task needs to be performed, where the machine is currently located, what other machines are in the field that need to be coordinated with, and what is the path through the field that needs to be taken. Agricultural equipment must follow the planned paths to carry out tasks in the field significantly better than those designed by operators without any prior planning.

A planned CPP helps reduce turnings and creates longer paths, which intern helps accomplish optimized field tasks. Additionally, this facilitates operator productivity, saves time and fuel, and protects the environment. In general, it is either exceedingly difficult or overly expensive to integrate every computerized planning tool directly into agricultural machinery, and such expensive systems are not affordable for all farmers. Simple pre-planning programs mix route-optimizing computer algorithms and other useful features into a practical graphical user interface (GUI) are more beneficial for everyone and economical at the same time.

Since many studies on precision agriculture rely on coverage path planning (CPP), there are two strategies in CPP to proceed with [9]. 1) Either the modern tiling or combining machines must have

every computer tool onboard to plan navigation strategy in the field. This is mostly performed through detecting things using sensors around and planning the averting strategy using its autonomous system online/directly in the field. This strategy does not usually require the full knowledge of overall shape/size for the Region of Interest (ROI). Or 2) There should be some solutions in terms of offline planning tools which require prior knowledge of the project area/the free space in addition to the algorithm enabling to program prior paths plans.

There are many offline and online algorithms developed by researchers however there are only few tools developed to practically solve problems. According to [10] a research on the spread of agricultural apps conducted in 2015 and 2016 shows quite few free online agricultural precision apps found in the whole Europe.

The geo-spatial arable field optimization service (GAOS) according to [11], is among the widely known and quite cleaver web application developed solely to help the farmers to plan agricultural activities like coverage planning. It has a very advanced and has a relatively user friendly GUI.

The other very promising effort in this regard is Fields2Cover [12] open source coverage path planning library. This is an open-source coverage path planning library for unmanned agricultural vehicles. Such libraries gives a wider range services by being bench mark tools in the process of creating specific tools related to coverage path planning.

There are currently commercial methods for precision agriculture (PA) that are often used on large farms with significant capital expenditures and only for farmers with more experience with information technologies. However, there are still insufficient substitutes for low-cost, simple-to-use procedures and techniques that can be used by small- and medium-sized farms. Simple installable desktop apps that combine the CPP methods, algorithms, and techniques with some useful GIS products like open source maps (OSM) base-maps are not much common and known at least locally. These planning tools can be considered as input in finding the above-mentioned alternative solutions for path planning. Such tools build a bridge for all farmers to easily connect with the improvement in current agricultural technology and enable them to profit from all its advantages.

1.2 Objective

The main goal of this thesis is to develop an offline, free, installable app that can compute agricultural machine's optimal coverage paths. Alternative GIS approach focused on transformation of geographic objects will be used as a chief splitting of ROIs and coverage path designing. The main goal has been subdivided to the following specific objectives.

1) Develop the CPP algorithm:

Develop a function to decompose Region of Interests, in short ROIs into long sub-shapes.

Develop a function to produce optimum CPP for each sub-shapes.

2) Define an optimum direction for the CPP algorithm based on the method called aspect direction of minimal rotated rectangle (MRR)

3) Develop a graphical user interface (GUI)

Incorporate various base maps such as Open Street Map (OSM)

Develop an interactive and user-friendly data input features to transfer the necessary inputs to the algorithm.

Develop features to output the CPP's result

Enrich the GUI with extra tools such as area and length measure

4) Evaluate the CPP results with those of another technique from literature

5) Examine the applicability of the proposed app for various ROIs with different shapes, size and with/out inaccessible areas (i.e., holes)

1.3 Background

Most modern agricultural machines have automated systems making the planning and task-performing process easier by equipping them with the needed tools to make automatic and semi-automatic planning and performing solutions through automated guidance [13]. The two steps that must be combined with the vehicle's GPS to achieve the required automation are agricultural mission planning and CPP determination. To plan optimized routes and path-planning coverage in the field, however, efficient, and effective algorithms, as well as online or offline planning tools, are required. Until now, various methodologies have been proposed in different studies.

The term "offline" is used in this work to refer to apps using offline algorithms as well as apps that assist users or farmers in doing CPP tasks using a personal computer while they are not utilizing an agricultural machine. The development of digital technology has made it easier to create digital tools as solutions for numerous problems in our daily lives. Without the help of tools like marketing

proposals for agricultural products, practical camera- and GPS-based sensors [14], and mobile apps [15] for analysis and general information about plants, animal diseases, soil, pesticides, and precision agriculture, agricultural tasks have never been easier or more effective. Even though the use of apps to agricultural tasks is becoming so popular [16] [17], there is still a lack of pervasive availability and distribution of them [10].

The majority of developed mobile apps are camera-based sensor utilities, according to a survey [10]. More varieties and straightforward agricultural apps are required to encourage more farmers to adopt digital technologies in their daily agricultural operations.

The development of a CPP application with a CPP algorithm consist of two key functions is covered in this thesis. 1) Splitting function, 2) CPP function, as well as a number of other helpful features and a user-friendly 2D CPP planning environment. The creation of computer algorithms incorporated in the app stems from previous works and techniques. The two main cellular decomposition techniques 1) Boustrophedon cellular decomposition, and 2) Trapezoidal cellular decomposition techniques combine to make a new computer splitting algorithm. To realize a simple CPP computer app other helping tools and environments be included. Open source maps (OSM) taken as advantage to give a convenience to the user to plan and show the planned CPP path on map. Python is used as the language to write all the codes needed and different well known GIS libraries is utilized as well.

The common method for splitting a complex-shaped field with inaccessible regions into simpler/convex none-overlapping shapes/cells is cellular decomposition. The criteria for the division are that internal parts of adjacent cells should not have touched each other and all the cells from the division should cover the original field [18],[19]. They should be optimized in terms of being the best candidates to be covered by lots of longer pathway.

Several coverage path planning algorithms proposed in many previous works are based on cellular decomposition planning techniques [9]. Cellular decomposition is a technique used to decompose the effective ROI i.e., the field area excluding the obstacle areas which is usually called ‘free configuration space’. Trapezoidal cellular decomposition is a modified form of cellular decomposition technique which is the division of the free configuration space into trapezoids or similar shapes followed by the merge and searches i.e., re-aggregation of them into some simpler and bigger usually convex-shaped polygon [1],[20].

According to the tools used in computer science and mathematics, live or online computation is typically referred to as local, whereas pre or offline computation is global because it is based on a field area whose shape, size, and inaccessible portions inside the main field are known. While the pre-computational/offline strategy involves the division of the field area into straightforward, optimized, and convex-shaped polygons, live-computational approaches, for example, use cellular grid division and base their computation on grid cells surrounding their present positions [21]. In both approaches, algorithms have often been developed using a variety of related methodologies.

In [1], the coverage path planning algorithm has been performed as a combination of two algorithms; First a higher-level algorithm was employed to split a complex shaped field into smaller parts. Then an algorithm used as a bottom-to-top approach to sequentially and recursively create paths for the split shapes based on the adjacency graph.

The splitting algorithm is based on trapezoidal cellular decomposition. Any shaped field with obstacles can use this algorithm. Regional restrictions are described as prohibited driving routes. In the second algorithm, namely, the pass designing algorithm, the already-operated portion of the field is removed from the field area and the process is continued until the plot for the entire field is complete. They compute the efficiencies of each potential path using a simulator, and the most efficient path is chosen. A heuristics approach was employed to limit the search space. The method performs well for fields with straight edges, according to the outcome. It also showed that it works for curved edges even though it is not so efficient.

However, trapezoidal decomposition yields a large number of cells because the neighboring sides of each cell are projected straight from all of the vertices of the provided field and vertices from obstacle bounds. This needs a further task for re-merging them or springs in many cells which must be covered with paths separately. The separate path designing for every cells cause many incomplete path spaces to be occurred (fig 12). The number of incomplete path is directly proportional to the number of cells. See the left side of (Figure 1). This has been tried to be overcome in boustrophedon [22] [23].

The ancient Greek word “boustrophedon” which has the meaning “the way of the ox” shows the tillage path of the oxen. Although the method is similar to that of trapezoidal decomposition, in boustrophedon decomposition only two critical points on the boundaries of the obstacles are crossed; this occurs typically when the obstacle alters connectivity with the sweeping line, unlike in trapezoidal decomposition where the adjacent cells are created from projections of each vertex. Because it lowers the cost of re-merging the cells and creates a more continuous path than trapezoidal decomposition, which creates as many discontinuous paths as the number of cells, in this case makes the Boustrophedon preferable (Figure 1).

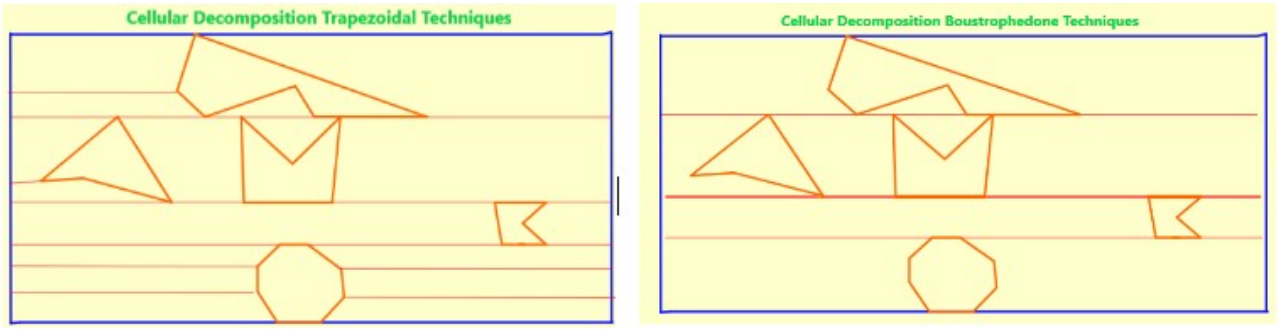

Figure 1: Comparison Trapezoidal vs Boustrophedon cellular decomposition techniques.

1.4 Scope of The Topic

This work is to make an agricultural machinery path planning app that can compute an optimized coverage path performed by the machine. The app incorporates a computer algorithm to compute the optimized path for the agricultural machine.

The algorithm can work for any given shape and size field. The app have tools to import Latitudes, Longitudes (Lat, Lon) positions of vertices for the ROI, and the vertices of the inaccessible regions

in the ROI. Additionally, the app incorporate OSM and Google satellite maps as background maps with some map projection, transformation, and conversion functions under the hood. The GUI of the app should incorporate drawing and erasing tools to enable the user to delineate the field and obstacles directly on the OSM with enough accuracy instead of importing measured (lLat, Lon) from CSV-file format. The background map enables the user to visually check if the planned path resulted as expected or not. These features enable the app o be used by any user around the world (since the OSM covers the entire world). A feature is developed to export the computed path as: .shp, .json, and .csv file formats. The fields agricultural machine can read the path data directly from the file and follows the path to perform the given task. To achieve this app three steps are followed. 1) The Development of an algorithm for simplifying a more complex given ROI and the inaccessible regions in it, if any, into simpler shapes 2) Algorithm to create optimized tiling path for each new optimized sub-shapes resulted in step-1 is created and implemented 3) The supporting tools (drawing, editing, saving, importing and exporting functions) also be created and incorporated with a user interface environment and are made to an installable app.

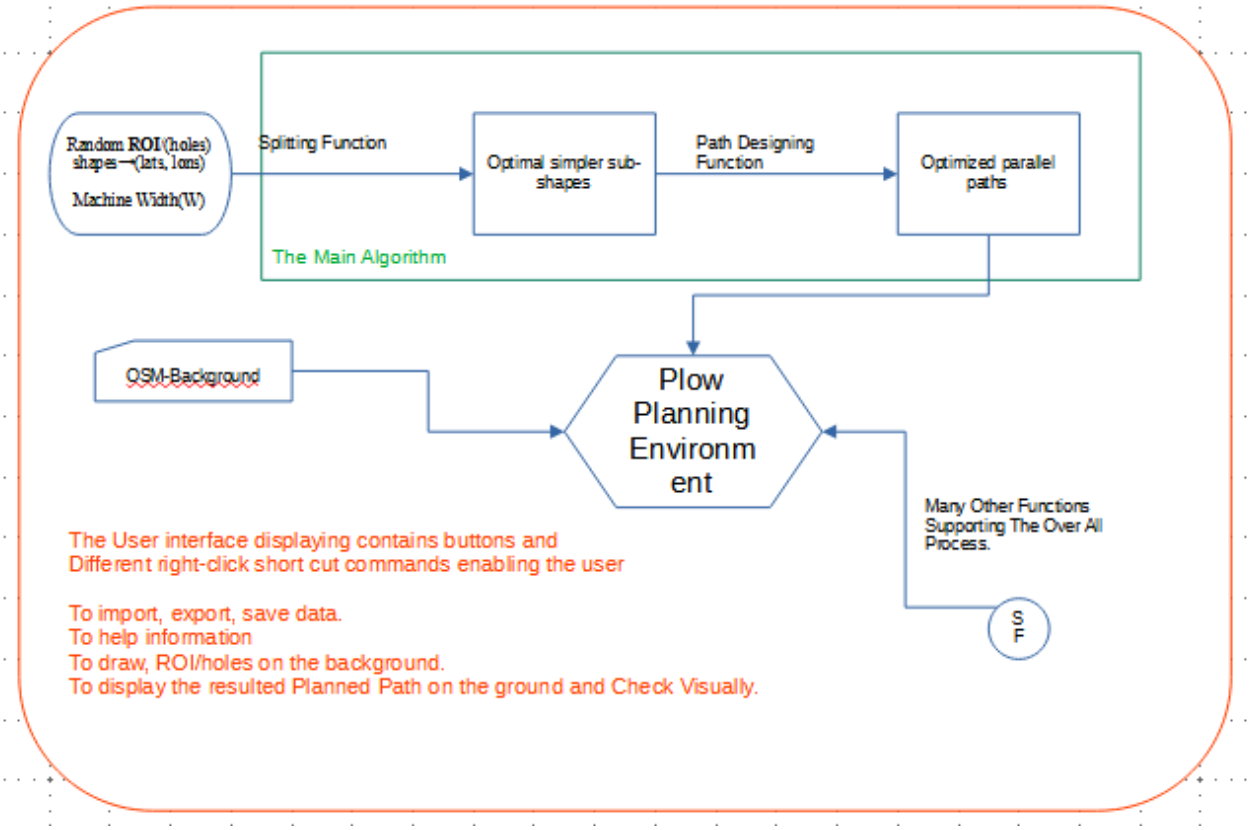

Figure 2: Flow Chart Showing The Over All Plan of The work

This app is created using the Python environment and numerous relevant Python libraries, including Shapely, tkintermapview, and others (Figure 2). The program require the Lat, Lon of the vertices of the ROI, its inaccessible sections, if any, and the width for the machine in meters as inputs. These vertices can be located by either directly delineating them on the OSM background map or provided machine's GPS file.

The ROI is assumed to have any regular/irregular shape, of a random size but is flat. Then the developed algorithm in the interface computes driving automated optimized patterns (parallel driving tracks) with a given machine width or the machine's swath width. The goal is to cover the specified field form with as few route lines as feasible, which translates to as few turns as possible.

CHAPTER 2. Theoretical Background

2.1 Geographic Objects

According to [24] [25] spatial object/geographic object can be defined as the digital representation of geographical entity or phenomenon which forms the basis for data management and analysis.

The main research purpose is to design and plan an efficient path for agricultural machineries including tractors carrying out specific agricultural tasks in a field. Proposed paths are represented as lines as one of the primary geographic objects. Geographic objects can be studied from different perspectives. A database perspective for geographic objects is detailed as below.

2.1.1 Geographic Objects in Database Designing Perspectives

Geographic objects should be taken into consideration at the conceptual level of database design. The theme is defined based on a collection of geographic objects which represent real-world entities. Based on the expressions in [24], The following characteristics of the geographic objects are significant in the database design process.

Each object should be accompanied by a list of attributes that describe the spatial object. For instance, a city's description is comprised of its name and population. Additionally, a spatial component is one of the important factors that must be considered in the design process. This represents both geometry (location in the underlying geographic space, shape) and topology (spatial relationships among objects, such as adjacency).

2.1.2 The Geographic Objects Structure

Geographic objects might have complex or atomic data structure. The geographical objects that incorporate other geographic objects are referred to as complex objects. For example, a map of Europe includes more intricate geographic features called countries in Europe, which may include (atomic geographic objects, their cities). While a complex object comprises of multiple geographic objects and their accompanying descriptions, a single geographic object represents only a simple object component and its attribute.

In GIS, a theme is defined as a set of homogeneous geographic objects. Geographic data types are often differ from those standard data types such as text or integer [24]. A suitable modeling of reality is derived from how precise and accurate geographical objects are stored in database.

Vector and raster data are the two primary geographic datatype used to represent geographic objects. In some cases, raster data will be converted to vector data types to ease some of the complex geo-computation. In the process of modeling spatial data, the three most prevalent vector data types are point (zero-dimensional object), line (1-dimensional object), and region (2-dimensional object). For example, cities are represented as points, roads as lines, and countries as regions. Vector space is used to define the spatial and topological features of geographic objects in digital representations.

2.1.3 Vector Space:

There are different perspectives to understand vector space. For instance, vectors are frequently employed in physics to describe forces or speed, although in general science, vectors are typically thought of as arrows with a length and direction. A vector may have two or more directions. While three dimensional vectors are represented in a three-dimensional environment, two dimensional vectors can be represented on a plane. Vector space is the collection of vectors.

In a two-dimensional space, a vector is represented by two coordinates , one indicates the tail of the vector which assumed to be the origin of the coordinate system in which the space of that specific vector falls in and the other coordinate defines the head of the vector. For instance, each of the vertices in a polygon represents a vector.

Vectors can be added, subtracted or multiplied. Multiplying the coordinate of the vector by a number will scale up or down the vector. An n-dimensional vector is defined by n-coordinates. The collection of these n-dimensional coordinate vectors represents n-space.

2.1.3.1 Properties of Vectors:

If $\vec{v},\vec{w}$ are vectors then The following properties should be satisfied:

additive: $\vec{v}+\vec{w}$ *is a vector* (1)

Commutative (*under addition*): $\vec{v}+\vec{w} = \vec{w}+\vec{v}$ (2)

Zero vector: $\vec{v}+\vec{0} = \vec{v}$ (3)

Identity element for vectors addition: $\vec{0}+\vec{v} = \vec{v} + \vec{0} = \vec{v}$ (4)

Inverses Vector: $\vec{v}+(-\vec{v}) = (-\vec{v})+\vec{v} = \vec{0}$ (5)

Associativity (*Under addition*): $\vec{v}+(\vec{w}+\vec{z}) = (\vec{v}+\vec{w})+\vec{z}$ (6)

Ditsributive: $c.(\vec{v}+\vec{w}) = c.\vec{v}+c.\vec{w}$ (7)

$$\textit{Ditsributive}: \quad (c+d).\vec{v} \;=\; c.\vec{v}+d.\vec{v} \tag{8}$$

$$\textit{Identity element for vectors multiplication is } 1: \quad 1.\vec{v} \;=\; \vec{v}.1 \;=\; \vec{v} \tag{9}$$

$$\textit{Associativity}\ (\textit{Under multiplication}): \quad \vec{z}+(\vec{d}+\vec{v}) \;=\; (\vec{z}+\vec{d})+\vec{v} \tag{10}$$

Scaling is accomplished by multiplying each vector's coordinate by a factor that is typically referred to as a scalar. Scalars provide vectors with more structure and features as a result.

Numerous operations, including geometric transformations, function compositions, matrix multiplications are used in abstract algebra and some other familiar arithmetic operations like additions, subtractions, multiplications, and division.

If V is a Vector and F is a scalar then:

$$v\in\vec{V}\wedge f\in F\Rightarrow f.v\in\vec{V}\,,\textit{a scaled vector} \tag{11}$$

$$f.(\vec{v}_1+\vec{v}_2)=f.\vec{v}_1+f.\vec{v}_2 \tag{12}$$

$$(f_1+f_2).v=f_1.v+f_2.v \tag{13}$$

$$f_1.(f_2.v)=(f_1 x f_2).v \tag{14}$$

$$\textit{Vector Matrices} \;=\; \begin{bmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \end{bmatrix} \tag{15}$$

$$\textit{Inverses Mtrix} \;=\; \begin{bmatrix} -a_{11}-a_{12}-a_{13} \\ -a_{21}-a_{22}-a_{23} \end{bmatrix} \tag{16}$$

$$\textit{Scalar Multiplication Mtrix} \;=\; c.\begin{bmatrix} a_{11}a_{12}a_{13} \\ a_{21}a_{22}a_{23} \end{bmatrix}=\begin{bmatrix} c.a_{11}c.a_{12}c.a_{13} \\ c.a_{21}c.a_{22}c.a_{23} \end{bmatrix} \tag{17}$$

The Expansion of Real Numbers $\mathbb{R}$

The set of real number is an example of a vector space

The set of real vectors of length n will be $\mathbb{R}^n$:

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{18}$$

Vector Addition:

A degree n polynomial vector space is created by adding another degree n polynomial vector space.

$$\mathbb{R}^n + \mathbb{R}^n = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} + \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} v_1 + u_1 \\ v_2 + u_2 \\ \vdots \\ v_n + u_n \end{bmatrix} \tag{19}$$

Multi-dimensional vector (mxn)

$$\mathbb{R}^{mXn} = \begin{bmatrix} v_{11} & v_{12} \cdots & v_{1n} \\ v_{21} & v_{22} \cdots & v_{2n} \\ \vdots & \vdots \ddots & \vdots \\ v_{m1} & v_{m2} \cdots & v_{mn} \end{bmatrix} \tag{20}$$

Multi-dimensional vector multiplied by scalar (c)

$$c.\mathbb{R}^{mXn}=\begin{bmatrix} c.v_{11} & v_{12}\dots & c.v_{1n} \\ c.v_{21} & v_{22}\cdots & c.v_{2n} \\ \vdots & \vdots\ \ddots & \vdots \\ c.v_{m1} & v_{m2}\cdots & c.v_{mn} \end{bmatrix} \tag{21}$$

Vector spaces can be more than just a set of vectors. Vector space can be defined as functions as it is needed. For instance vectors can be defined as a set of linear polynomials as $ax+b$.

Such vectors can be :

$$destributive: \quad c.(ax+b) = (c.a).x + (b.c) \tag{22}$$

$$Summed\ up: \quad (a_1x+b_1)+(a_2x+b_2) = (a_1+a_2)x+(b_1+b_2) \tag{23}$$

Despite the fact that the inputs are all vectors, the result will not be a vector space unless the closure conditions are satisfied. For instance

$$w=\left\{\begin{bmatrix} x \\ y \end{bmatrix} : xy\geq 0\right\} \tag{24}$$

This space can be closed under multiplication but not under addition. Because $xy\geq 0$ imp-lays that both x and y should be positive vectors or both negative vectors. i.e. That means the sum should always either give negative x and negative y or positive x positive y which is impossible. For instance the sum of the following vectors will not give both positive or both negative.

$$\begin{bmatrix}5\\5\end{bmatrix}+\begin{bmatrix}-3\\-7\end{bmatrix} = \begin{bmatrix}2\\-2\end{bmatrix} \tag{25}$$

The result $\begin{bmatrix}2\\-2\end{bmatrix}$ *does not satisfy the* $xy \geq 0$ *rule.*
Therefore this space is not close under addition.

2.1.4 Linear Transformation (L) and Its Application:

Linear transformations are vector space functions to transform a given vector V to a new vector u in this transformation. The input vector's type might not match that of the modified vector. For example, a vector of degree n can be changed into a new vector of degree m, or a vector can be changed into a scalar and vice versa.

A Linear transfromation can be denoted as: $L: V \rightarrow U$

$\mathbb{R}^n \xrightarrow{L} \mathbb{R}^m$ *Then its transformation Matr* i_x *will be an mxn Matr* i_x (26)

$L: \mathbb{R}^2 \rightarrow \mathbb{R}$ *A vector tto a scalar* (27)

$$L(\vec{v}) = A\vec{v} \tag{28}$$

A linear transformation is to map one vector space to another. The main properties of linear transformations are as follow:

$$L(c.\vec{V}) = c.L(\vec{V}) \tag{29}$$

$$L(\vec{V}+\vec{U}) = L(\vec{V})+L(\vec{U}) \tag{30}$$

$$L: \mathbb{R}^2 \rightarrow \mathbb{R}^3 \tag{31}$$

Cross − Checking Examples:

Checking the following based on $L : \mathbb{R}^2 \to \mathbb{R}^3$

$\vec{V} = \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \end{bmatrix}$ *then let our transformation formula be* $L(\vec{V}) = \begin{bmatrix} \vec{v}_2 \\ \vec{v}_1 + \vec{v}_2 \\ \vec{v}_1 - \vec{v}_2 \end{bmatrix}$

First Checking If $L(c.\vec{V}) = c.L(\vec{V})$

$$L(c.\vec{V}) = \begin{bmatrix} c.\vec{v}_2 \\ c.\vec{v}_1 + c.\vec{v}_2 \\ c.\vec{v}_1 - c.\vec{v}_2 \end{bmatrix} = \begin{bmatrix} c.(\vec{v}_2) \\ c.(\vec{v}_1) + c.(\vec{v}_2) \\ c.(\vec{v}_1) - c.(\vec{v}_2) \end{bmatrix} = c.\begin{bmatrix} \vec{v}_2 \\ \vec{v}_1 + \vec{v}_2 \\ \vec{v}_1 - \vec{v}_2 \end{bmatrix} = c.L(V)$$

Second Checking If $L(\vec{V} + \vec{U}) = L(\vec{V}) + L(\vec{U})$

Given $L(\vec{V}) = \begin{bmatrix} \vec{v}_2 \\ \vec{v}_1 + \vec{v}_2 \\ \vec{v}_1 - \vec{v}_2 \end{bmatrix}, \quad \vec{V} = \begin{bmatrix} \vec{v}_1 \\ \vec{v}_2 \end{bmatrix}, \quad \vec{U} = \begin{bmatrix} \vec{u}_1 \\ \vec{u}_2 \end{bmatrix}$ *, Then* $\vec{V} + \vec{U} = \begin{bmatrix} \vec{v}_1 + \vec{u}_1 \\ \vec{v}_2 + \vec{u}_2 \end{bmatrix}$

Applying our transformation formula with the above result:

$$\begin{bmatrix} (\vec{v}_2 + \vec{u}_2) \\ (\vec{v}_1 + \vec{u}_1) + (\vec{v}_2 + \vec{u}_2) \\ (\vec{v}_1 + \vec{u}_1) - (\vec{v}_2 + \vec{u}_2) \end{bmatrix} = \begin{bmatrix} (\vec{v}_2 + \vec{u}_2) \\ (\vec{v}_1 + \vec{v}_2) + (\vec{u}_1 + \vec{u}_2) \\ (\vec{v}_1 - \vec{v}_2) + (\vec{u}_1 - \vec{u}_2) \end{bmatrix} = \begin{bmatrix} \vec{v}_2 \\ \vec{v}_1 + \vec{v}_2 \\ \vec{v}_1 - \vec{v}_2 \end{bmatrix} + \begin{bmatrix} \vec{u}_2 \\ \vec{u}_1 + \vec{u}_2 \\ \vec{u}_1 - \vec{u}_2 \end{bmatrix} = L(\vec{V}) + L(\vec{U})$$

2.1.4.1 Application of Linear Transformation,

In geometric transformation linear transformations are needed in stretch, squish, reflect, rotate or translate the input geometry or coordinate System.

2.1.5 Dot and Cross Product of Vectors:

2.1.5.1 Dot Products

In mathematics, the dot product or scalar product is an algebraic operation that takes two equal-length sequences of numbers (usually coordinate vectors), and returns a single number. Even-though dot products produce scalar values, they can be converted to vectors by projecting the vector's unit vector onto them [26] (Figure 3). The following two methods are used to compute the dot products mathematically [26] [27].

$$Let\ \vec{A}=\begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad \vec{B}=\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\vec{A}.\vec{B}=A_x B_x+A_y B_y+A_z B_z. \tag{32}$$

$$\vec{A}.\vec{B}=|A||B|\cos(\theta), \quad Where\ \theta\ is\ the\ angle\ b/n\ the\ two\ vectors. \tag{33}$$

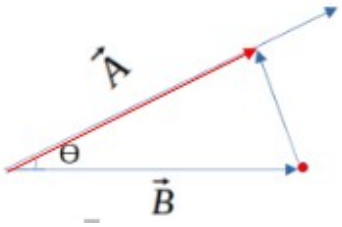

Figure 3: Linear Projection With Dot Product

Mostly vector tails are thought to be the coordinate system's origin. The second (formula 33) will be vital if a random point or random line should be projected to the other line (Figure 3).

If dot product of these two vectors divided by the length of one of them means the same as the length of the vector projected on its corresponding vector. For instance if their dot product is divided by length of vector A results the projected vector-B on vector-A (see formula 34).

$$This\ means: \quad \frac{\vec{A}.\vec{B}}{length\ of\ \vec{A}} = length\ of\ \vec{B}\ \ at\ \vec{A} \tag{34}$$

The sign for the dot product will be negative if the vector-B was projected in the opposite direction of vector-A and it would have been zero if the projected vector-B was perpendicular to vector-A. If the length of this vector would be doubled then the original dot product will be doubled.

$$i.e. \quad \vec{A}.(2\vec{B}) = 2(\vec{A}.\vec{B}) \tag{35}$$

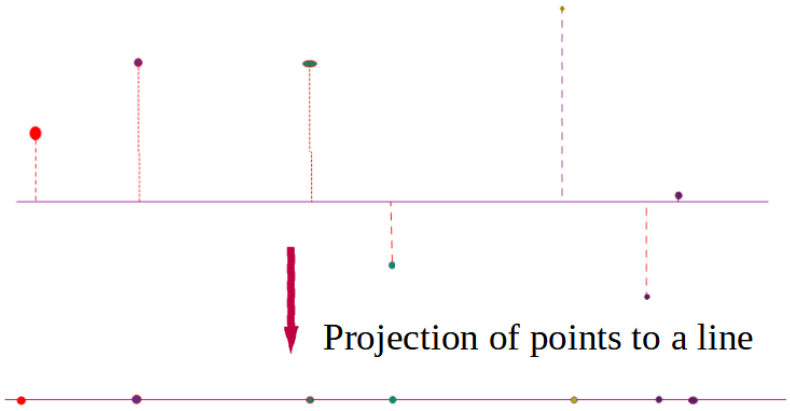

Figure 4: Linear Projection of Points on to a line Based On Dot Product

2.1.5.2 Cross Products

The cross product of two vectors represents the area/volume of the parallelogram spanned by the two 2D/3D vectors. The cross products are not scalars, and have direction. The direction of the cross product is determined by right hand rule. This value is defined as the scalar product of the absolute value of each of the vectors multiplies by the sine value of the angle between them The cross product's direction will always be perpendicular to the plane where the two vectors are located.

$$For\ instance: \quad If\ \vec{A}\ ,\ \vec{B}\ \ be: \quad \vec{A}=\begin{bmatrix} a1_i \\ a2_j \\ a3_k \end{bmatrix}, \quad \vec{B}=\begin{bmatrix} b1_i \\ b2_j \\ b3_k \end{bmatrix}$$

$$\vec{A}\,x\,\vec{B} = |\vec{A}||\vec{B}|\sin\theta\,\hat{d} \tag{36}$$

$$where:$$

$\hat{d}$ *is defined as a negative* $\vee$ *a posetive unit vector by righthand rule.*

$$|\vec{A}|=\sqrt{a1^2+a2^2+a3^2}\ ,\ |\vec{B}|=\sqrt{b1^2+b2^2+b3^2}$$

$\vec{A}\,x\,\vec{B}$ *can Also be deffined as*:

$$\vec{A}\,x\,\vec{B} = (a2b3-a3b2)i+ (a3b1-a1b3)\,j+ (a1b2-a2b1)k \tag{37}$$

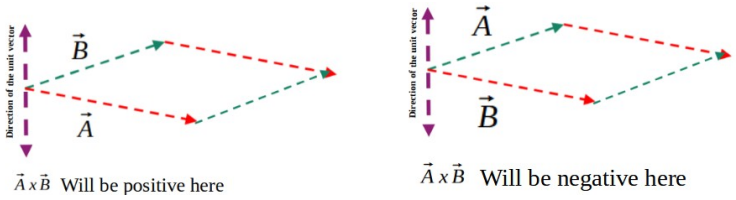

Figure 5: Positive vs Negative Cross Product, (negative=downward), Positive(upward)

Finding the determinant can be referred to as computing the area in 2D vectors or volume in 3D vectors without multiplying the vectors with one another for the 2D and 3D cross product respectively. The two above matrices (formula 36) can be converted to a linear transformation matrix as follows:

$$\begin{bmatrix} a1_i & b1_i \\ a2_j & b2_j \\ a3_k & b3_k \end{bmatrix}$$

$$\begin{bmatrix} a1_i & b1_i \\ a2_j & b2_j \end{bmatrix} = a1_i b2_j - b1_i a2_j \tag{38}$$

$$\begin{bmatrix} a1_i \\ a2_j \\ a3_k \end{bmatrix} x \begin{bmatrix} b1_i \\ b2_j \\ b3_k \end{bmatrix} = \begin{bmatrix} a2_j b3_k - b2_j a3_k \\ b1_i a3_k - a1_i b3_k \\ a1_i b2_j - b1_i a2_j \end{bmatrix} \tag{39}$$

This can be rewriten as:

$$\begin{bmatrix} a1_i \\ a2_j \\ a3_k \end{bmatrix} x \begin{bmatrix} b1_i \\ b2_j \\ b3_k \end{bmatrix} = \begin{bmatrix} i & a_1 & b_1 \\ j & a_2 & b_2 \\ k & a_3 & b_3 \end{bmatrix} = (a_2b_3-b_2a_3)i+(b_1a_3-a_i)\,j+(a_1b_2-b_1a_2)k \quad \textit{which is the as } eq36$$

This determines the direction for the unit vector of the area computed from the cross product of the two vectors which is defined as the area of the parallelogram (Figure 5) spanned by the given vectors.

2.2 Spatial Data Types Transformations

The two most popular forms of spatial data transformation will be covered in this section namely geometric transformation and coordinate transformation in map projection

2.2.1 Geometric Transformation

Transformation can generally be defined mathematically as a function of both the one to one and onto relationships between two sets $T: A \rightarrow A'$ [28]. A transformation preserving distance can be called isometry.

Geometry transformation is a sort of motion of the plane (or of space) carrying each point A into a new point A' such that the distance between any two points A and B is equal to the distance between the points A' and B' into which they are carried [29].

In general, transformations are required to change a graphic image by rotating, reflecting, contracting, expanding, shearing, or projecting it. The most common types of geometry linear transformations will be discussed in this section. Transformations are functions that a geometric object should go through when viewing or having them in other sizes, shapes, or changing their position. Transformations on geographic objects or their coordinate systems are usually involved on their vector representations in the form of Matrix algebras. Geometric transformations can be rigid or non-rigid. Rigid transformations are transformation types where the structures of the input geometry will be preserved [30].

2.2.1.1 Rigid Transformations

Rigid transformation is mostly mentioned in relation to congruence. For instance according to some references it is better to simply say that two figures are congruent if one of them can be transformed into the other by a transformation that preserves the size and shape of the figure [30]. Suggestions from [28] show that rigid motions and similarity transformations can be used to deduce the two mathematical concepts congruence and similarity.

Thus Rigid transformation is a mathematical transformation in which length and angles are preserved. Therfor, the relative distances or angles (i.e. the amount of angle between any three points on the object) before transformation will remain the same after transformation. Rigid transformations include translations, rotations, and reflections.

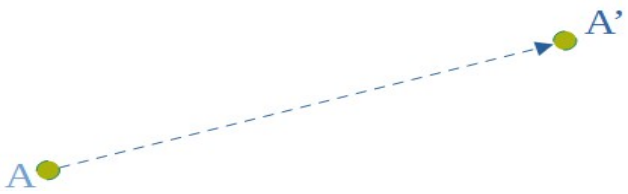

Figure 6: Translation vector translates A to A'

2.2.1.2 Translation:

A translation is an example of a transformation of the plane that carries each point A into some other point A' along a translation vector $\vec{AA'}$. The translation vector will have the same length as the distance $\overline{AA'}$ Clearly, no point is left in place by this transformation; in other words, a translation has no fixed points it carries no point in itself/there is no point that does not change its position [29]. Translating a point (x, y) along vector (a, b) will then be (x+a, y+b). This is one of the transformation type in which every points of the given spatial object mapped/get a new position change by moving to a given direction and distance. For instance, if one spatial object to be transformed into a region, then, every point on that region will be moved to a given direction and distance i.e. every vertices of the region and every points inside it will be shifted by some given x-displacement to some given y- direction.

$$\text{If the translation should be } A \rightarrow A'$$

$$\text{Given point } A = \begin{bmatrix} a_x \\ a_y \end{bmatrix}, \quad \vec{V} = \begin{bmatrix} a \\ b \end{bmatrix} \text{ be a translation vector} \tag{40}$$

$$\text{Then } A' = \begin{bmatrix} a_x + a \\ a_y + b \end{bmatrix}$$

2.2.1.3 Rotation:

It is another form of transformation in which every point/part of the given object is rotated by a given value in specific direction either clock-wise or counter clock-wise and relative to a fixed reference point. After the rotation, every point in the object will have equal distance to this fixed point of reference called center of rotation.

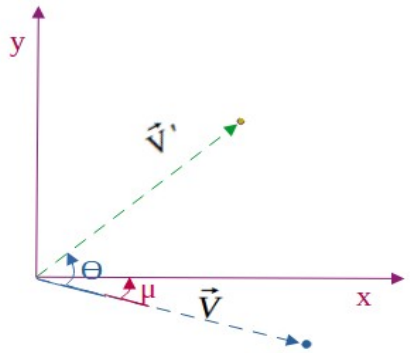

Figure 7: Rotating vector V to vector V′ by angle theta

$$\vec{V}'_x = V\cos(\theta-\mu) \qquad \vec{V}'_y = V\sin(\theta-\mu)$$

$$\vec{V}'_x = V(\cos(\theta)\cos(\mu)+\sin(\theta)\sin(\mu)) \qquad \vec{V}'_y = V(\sin(\theta)\cos(\mu)-\cos(\theta)\sin(\mu))$$

$$\vec{V}'_x = V(\cos(\theta)\cos(\mu)+V\sin(\theta)\sin(\mu)) \qquad \vec{V}'_y = V\sin(\theta)\cos(\mu)-V\cos(\theta)\sin(\mu)$$

$$\vec{V}'_x = V_x\cos(\mu)+V_y\sin(\mu) \qquad \vec{V}'_y = V_y\cos(\mu)-V_x\sin(\mu)$$

Convering $\vec{V}'_x$ *annd* $\vec{V}'_y$ *toa matrx form*

$$\begin{bmatrix}\vec{V}'_x\\ \vec{V}'_y\end{bmatrix} = \begin{bmatrix}\cos(\mu) & \sin(\mu)\\ -\sin(\mu) & \cos(\mu)\end{bmatrix}\begin{bmatrix}V_x\\ V_y\end{bmatrix} \tag{41}$$

For example, if the point of rotation is on the object to which the transformation is applied, that specific point of the object will not be shifted, whereas the rest point on the object will be rotated in the given direction by the given amount of rotation (Figure 7). Trigonometry matrices are commonly used to represent rotation vectors. These matrices are of two types: one represents clockwise rotation and the other represents counter-clockwise rotation.

If the rotation should be $A \rightarrow A'$

Fram equatiion 40 *We have the following.*

Given point $A = \begin{bmatrix}a_x\\ a_y\end{bmatrix}$, $\vec{V}_{Clockwise} = \begin{bmatrix}\cos(\theta) & \sin(\theta)\\ -\sin(\theta) & \cos(\theta)\end{bmatrix}$, $\vec{V}_{Counter-clockwise} = \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}$ *be the rotation vector*

Then $A'_{ClockWise} = \begin{bmatrix}a_x\cos(\theta)+a_y\sin(\theta)\\ -a_x\sin(\theta)+a_y\cos(\theta)\end{bmatrix}$ *While,* $A'_{Counter-clockWise} = \begin{bmatrix}a_x\cos(\theta)-a_y\sin(\theta)\\ a_x\sin(\theta)+a_y\cos(\theta)\end{bmatrix}$

2.2.1.4 Reflection:

A reflection is defined as a linear transformation of a 2-dimensional vector space $\mathbb{R}^2$ -- (the x-y-plane). It is done of parameters as a form of a matrix-vector as a standard base and a line formula as a reference. Therefore the reflection of an object is the translation of every point on the object by an amount of the shortest, perpendicular, and equal distance of itself from a mirror/reference line to the other side of the mirror line (Figure 7). In reflection the mirror line does not alter by the transformation.

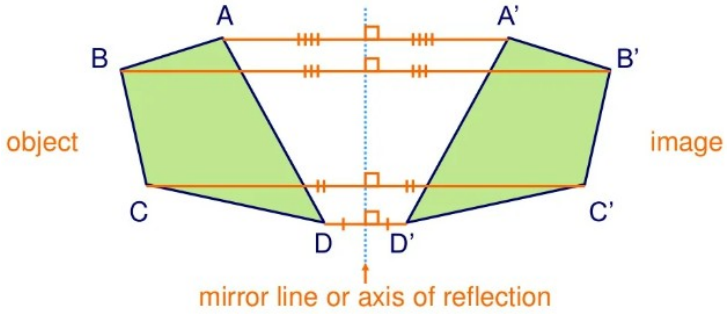

Figure 8: Reflection of An Object (Object vs it's reflected image)

2.2.2 None Rigid Transformations

Individual rigid parts of an object may move independently of one another, and such motion of a part of the object only, which does not include the entire object, disturbs relative relationships (relative distances or directions to one another) between other parts of the object. Nonrigid transformation is the type of transformation that results in a distorted transformed image. This type of transformation is also known as elastic motion because it is nonrigid motion that conforms to a certain degree of continuity or smoothness where it is required. Some examples include dilation and shear, as well as scaling. This transformation type will change the size, shape, or both object's dimensions.

$$\begin{aligned}
&\textit{Given: a function } y = f(x)\\
&y'_{vertical} = cf(x) \quad \textit{if } c>1/\ 0<c<1, \quad \textit{if c is outside of function.}\\
&y'_{horizontal} = f(cx) \quad \textit{if } c>1/\ 0<c<1, \quad \textit{if c is inside of function.}
\end{aligned}$$

These produces a vertical stretch/shrink annd a horizontal stretch/shrink respectively. Such a transformation is non−rigid.

2.2.3 Geographic Data

Basically, geographic data/information may be derived/collected primarily from the field in which the determination of point locations on Earth is often done with the help of the Global Positioning

System (GPS) of satellites. Remote sensing techniques other ways of derivation which produce satellite images (usually Landsat and Spot), aerial photographs (with the help of photogrammetry). The problems in data collection include delimiting geographic objects (the boundary may be fuzzy, for example, in the case of continues data such as soil type).

Data collected from a map may create problems due to maps are created by integrating several existing digital data sources. For example, analog maps and other cartographic documents can be digitized using manual, automatic, or semiautomatic techniques. Therefore data collected from a map may have different quality and accuracy.

On the other hand to represent geographic coordinates (Lon, Lat) on a 2D map information about the source map is required, such as the reference system on which the map is based and the map projection system used (e.g., Mercator, Universal Transverse Mercator.).

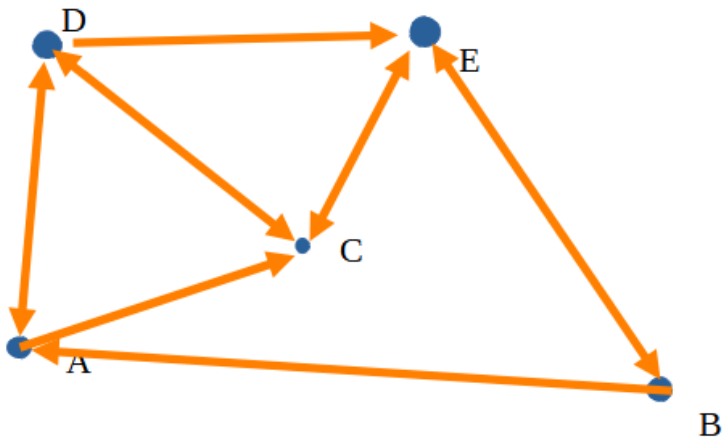

Figure 9: Creating Data Structures

2.2.4 Data Structures

Data structure is the concept by which the data it is stored in a computer memory. There are various methods for storing data on a computer. This can cause issues if the data is not properly stored. For example, having the (Lat, Lon) positions of objects in (figure 9) can reveal their absolute positions. However, it is impossible to know which streets connect the objects to and where the relative locations of those streets are. As a result, a method for properly storing street information is required. For example, all possible paths connecting those objects can be stored in array or list format. It is important to note that these paths can be bidirectional or single directional. A method for storing bi-directional paths is required, so that both directions can be stored and accessed separately. Figure (9) shows from object A to object D or from object D to object A for bi-directional paths or from object A to object C only for single direction. Accordingly, to get to A from C, the best path, C, E, B, A, should be saved as a path. Another method for storing those paths is to select one of the given objects as the initial point and record the number of objects accessed

from that initial object (for example from object C to object B, to object A, to object D...). This is an example of the a hash table or hash map data structure.

2.2.5 Geodetic Coordinate Systems

A Coordinate System is defined as "a system for specifying points using coordinates measured in a specific manner" [31]. The realization for actual usage of digital spatial data is approved by the proper understanding of coordinate systems. The transformation of coordinate systems solves many difficult problems arising in geomatics[32].The proper understanding of coordinate systems validates the realization for actual use of digital spatial data. Positions representing information or objects may require new positions in a different coordinate system. That's where coordinate system transformation is required.

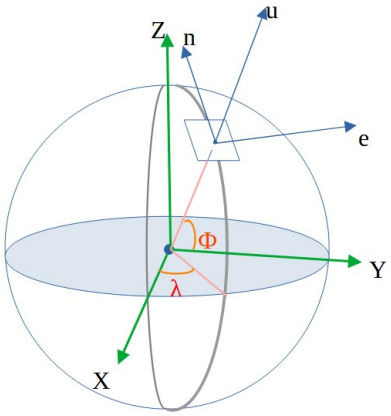

Figure 10: Transformation Between Coordinate Systems

2.2.6 Applications of Coordinate Systems

The four commonly used Coordinate Systems in Geomatics are: I) The geocentric Cartesian Earth-Centered, Earth-Fixed (ECEF), II) local horizontal Cartesian, III) geodetic curvilinear, and IV) Projected planimetric Cartesian. ECEF is mainly used in Global Navigation Satellite System (GNSS), astronomy, astrophysics, geodesy, Geo-dynamics, and climate change studies. The geocentric coordinate system is used internally as short-term system to calculate frameworks as part of several geographic (datum) transformation methods. Geodetic curvilinear or geodetic coordinates including the latitude, longitude, and ellipsoidal heights (length, breadth, and height (LBH) are established by imaginary grid lines over a given earth model usually the spheres/spheroids called graticules. The verticals (all have equal circumference) are the longitude lines while the horizontals (variable in circumference) are the latitude lines. This are coordinate systems used to define positions on the surface of the earth. Projected coordinate systems are flat 2D coordinate systems with constants areas and angles across the all areas. These coordinate systems constructed based on sphere/spheroid geographic coordinate systems. Local coordinate systems based on coordinates

Easting, Northing and Up (ENU) are projected type coordinate system used in large scale mappings their origins are arbitrarily established on the earth's surface rather than the earths center.

2.2.6.1 Coordinate Transformations

The coordinate transformation can happen in different ways. Conversion is a transformation based on a set of formulas that are given in advance, without adaption as in transformation, for instance a conversion can be feet to meter or from degrees to radians from geocentric coordinate to geodetic coordinates (latitude/longitude). Coordinate transformations can also happen if there is a need to transform a 2D/3D position vector from one reference frame to another. For instance, a position vector from ED50 to EUREF89 or from ITRF to EUREF89.

Coordinate transformations within the same reference frame have no effect on accuracy. In contrast, transformation between different reference frames does.

The European Petroleum Survey Group codes/EPSG-codes are crucial in coordinate transformation or conversion. These are the codes used by this group to create a database containing coordinate system information as well as some excellent related documents on map projections and datum. These are codes made up of 4-5 digit numbers that represent definitions for the coordinate reference system (CRS). The WGS84 geographic (latitude, longitude) coordinate system, for example, has an EPSG-code of 4326.

2.2.6.2 Coordinate Transformation From LBH to ENU

Transformation from geographic coordinates i.e., (LBH) to ENU is a two steps process. In which first the LBH should be transformed to ECEF (X, Y, Z) coordinates and the second step will conclude the transformation from X, Y, Z to ENU.

$LBH \Leftrightarrow XYZ$
$LBH \rightarrow XYZ$

$N = a/\sqrt{(1-\epsilon^2 \sin^2\phi)}$, *is the radius of curvature*

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} (N+h)\cos(\phi)\cos(\lambda) \\ (N+h)\cos(\phi)\sin(\lambda) \\ ((1-\epsilon^2)N+h)\sin(\phi) \end{bmatrix} \quad (42)$$

The reverse transformation will be using Bowring's noniterative procedure as follows:
$p = \sqrt{X^2+Y^2}$

$$\lambda = \arctan(Y/X) \qquad k = \arctan\left(\frac{Z}{(1-f)\sqrt{(X^2+Y^2)}}\right) \qquad \phi = \arctan\left(\frac{Z+(\epsilon')^2 b \sin^3 k}{p-\epsilon^2 a \cos^3 k}\right) \qquad h = \frac{p}{\cos(\phi)} - N(\phi) \tag{43}$$

where a, b, f *are the semi−major axis, semi−minor axis ,flatening of the reference ellipsoid respectively of the geodetic system. while* ϵ, ϵ' *first, second eccentricities respectively.*

Even-though both the ECEF and ENU are Cartesian coordinate systems based their origins are quite different. The origin of ECEF is the center of the spheroid while the origin for ENU is on the surface of the earth therefore the transformation from ECEF to ENU needs a translation(from center to the surface) and rotation(to align the ECEF axes to ENU, usually the x-axis with East, the y-axis with North and z-axis normal to the reference ellipsoid[33]) transformations. The most common transformation matrix transforming between these two coordinate systems is shown as follows.

$$\begin{bmatrix} e \\ n \\ u \end{bmatrix} = \begin{bmatrix} -\sin\lambda_0 & \cos\lambda_0 & 0 \\ -\sin\phi_0\cos\lambda_0 & -\sin\phi_0\sin\lambda_0 & \cos\phi_0 \\ \cos\phi_0\cos\lambda_0 & \cos\phi_0\sin\lambda_0 & \sin\phi_0 \end{bmatrix} \cdot \begin{bmatrix} X-X_0 \\ Y-Y_0 \\ Z-Z_0 \end{bmatrix} \tag{44}$$

Where λ_0, ϕ_0 = *geodetic coordinate for the origin of the coordinate System at which the new* e, n, u *system is established.*

X_0, Y_0, Z_0 = *The ECEF Coordinates for* λ_0, ϕ_0.
X, Y, Z = *The ECEF coordinate for the point being transformed.*

2.2.7 Map Projection

Map projection is to be perceived as a special form of conversion. In map projection, a mathematical formula is used as a method that "transfers" points from the ellipsoid, often within a limited area, to points in the map plane [34].

Map projection is a transformation of geographic (λ, ϕ) coordinates/geographic system to Cartesian coordinates/Cartesian system (x, y) ,where height will not be considered, by graphical or mathematical means[32]. The parallels and meridians is also projected as a graticule in the process of the projection while the map projection grids are formed by lines parallel to x, and y axis of the projecting coordinate system. The projection is the forward mapping process while the inverse mapping is backward even-though reverse process is impossible in the mathematical definition of 'Projection'. "No information is lost or gained in the map projection."[32].

Regular surfaces that can be converted to planes and are preferred as projection surfaces are referred to as developable/topologically planar surfaces. Among others, map projection can be classified as cylindrical planar or conic based on the developable surfaces used in the process. The names of map projections typically refer to the developable surface and the alignment of the surface

in relation to the semi minor axis of the ellipsoid used in the process, which can be normal, transversal, or oblique. The name transversal Mercator projection, for example, indicates that the axis of the cylinder used in the process is transversal with the minor axis of the ellipsoid used in the process

2.2.7.1 Universal Transverse Mercator (UTM)

UTM is a map projection system dividing the whole earth into 60 zones of each spans 6^o latitude with some exceptions while the longitudinal band is from south to north split by equator i.e. 60 zones at the north and another 60 zones at the southern hemisphere. Each zone has a name of its zone-number and a zone letter from C at south-extent to X at north-extent. *The zonenumber starts fromm* 180^oW-174^0W (180^oE-186^0E) *the easting continue upto* 0^oE-6^oE *which is zone* 30. *Therefore smallest easting zone number is* 30.

[1]utm zones

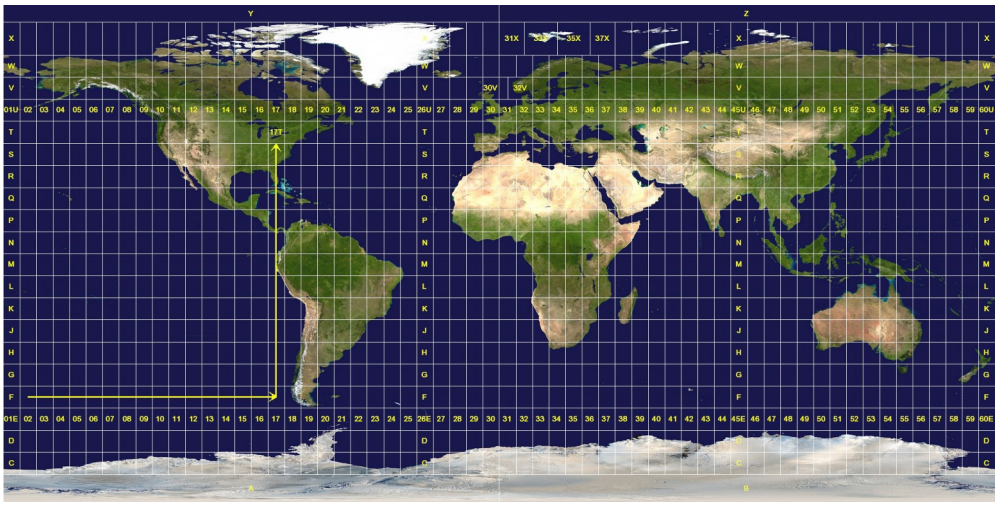

Figure 11: The UTM Zones.

In the UTM representation as it is shown at (figure 11) each grids are a gain have their own coordinate systems. Each of the small coordinate system in the UTM has origin with a false easting coordinate of 500000 while the northing coordinate for the southern hemisphere starts from 0 at south and ends 1000000 at the equator and for the northern hemisphere it starts 0 at the equator and continues up to 1000000 towards the north. All coordinates are in terms of metric unit.

2.2.8 Topology

Today, topology in GIS is generally defined as the spatial relationships between adjacent or neighboring features. Mathematical topology assumes that geographic features occur on a two-dimensional plane. “Topological relationships among spatial objects are those relations that are invariant under topological transformations”[24].

The knowledge for topology can be important in spatial functions like spatial networks establishment of nodes etc... The data integrity in spatial datasets and their quality enhancement can

1 The figure 11‘ utm zone’ taken from https://upload.wikimedia.org/wikipedia/commons/e/ed/Utm-zones.jpg

be assured with this knowledge. The accurate establishment of topological relationships results in keeping the datasets be synchronized in a GIS or other spatial data systems. Spatial datasets can be created with knowing the rules for topology.

2.2.8.1 Spatial Relationships

A spatial relation describes how an object is in space in relation to another object. Any of the relationships contained by, covered by, disjoint, equal, inside, overlap, and touch, as well as their derivations, are all types of spatial relationships.

2.2.8.2 Spatial Predicates

Spatial predicates are a series of spatial relations between two or more spatial features.

2.2.8.3 DE-9IM

The Dimensionally Extended 9-Intersection Model (DE-9IM) is a topological model and a standard used to describe the spatial relations of two regions (two geometries in two-dimensions, R2), in geometry, point-set topology, geo-spatial topology, and fields related to computer spatial analysis. The spatial relations expressed by the model are invariant to rotation, translation and scaling transformations.

Spatial Relationships can happen as one of the elements of the following $3\,X\,3$ *matrx.*
This matrx shows the Dimentionally Extended 9 *Intersections Model known by The Name* $DE\,9\,I_M$.

$$DE\,9\,I_M(a,b)=\begin{bmatrix} dim(I(a)\cap I(b)) & dim(I(a)\cap B(b)) & dim(I(a)\cap E(b)) \\ dim(B(a)\cap I(b)) & dim(B(a)\cap B(b)) & dim(B(a)\cap E(b)) \\ dim(E(a)\cap I(b)) & dim(E(a)\cap B(b)) & dim(E(a)\cap E(b)) \end{bmatrix} \tag{45}$$

Where a,b *be the regions be envolved into the relationship.*
I,B,E *Stands for Interior Border* $\wedge$ *Exteriors of the regions involved into relationship.*

This is how the spatial relationship b/n *regions is decided tobe a touch, an overlap, adjecency, contiguity* $\wedge$ *proximity.*

2.3 Computational Geometry

“Algorithmics is the branch of computer science that consists of designing and analyzing computer algorithms”[24].

Computational geometry is a branch of algorithmic dealing with computation on geometric objects. Now a days computational geometries used in many applications such as spatial databases, GUI, robotics, and computer-aided design and manufacturing (CAD).

2.3.1 Algorithms:

In mathematics and computer science, an algorithm is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing

Here they are generally identified as abstract machines, mathematical models of computers, sometimes idealized by allowing access to “unbounded memory” and all this definitions for algorithm has been discussed briefly in [35]. But in this work the simpler definition for algorithm is a set of instructions operating on different data structures and executing them according to the instruction.

Given the example presented above (Figure 9), an algorithm is needed as a set of instructions that operates on the data structure to select the best path.

The proposed algorithm should traverse all possible paths connecting the objects and select the best path based on the parameters provided. In this process, the given data structure and all necessary instructions should be provided in such a way that the computer understands what needs to be. The diagram below illustrates how a set of systematic instructions for determining the shortest distance between two objects can be created.

. Find the starting object

. Find the possible paths from the starting object.

. Keep track of the distance traveled as go through each possible paths

. Repeat this process until the satisfying result will be gained.

Different algorithms can be established to solve the same problem with different considerations.

2.3.1.1 Algorithmic Strategies

Most algorithms rely on well-known general (algorithmic) strategies as incremental, divide and conquer, and sweep line.

In an incremental algorithm first, a subset of the input is tried in a small enough amount so that the problem is easily solved, and then to add, one by one, the remaining elements of the input while maintaining the solution at each step. Divide and conquer is a strategy in which the bigger problem is divided in to sub-problems to avoid pushing against the hardest directly while it could be converted to easier by dividing it. In this strategy the bigger problem will be sub-divided and the solutions for the smaller problems are recombined as a solution for the bigger problem. The rule in

this strategy is the sub problems should always be the same type as the bigger. For instance if a bigger sorting problem is posed the sub-problems shall also be sorting but divide and conquer doesn't mean dividing a bigger task in to smaller different tasks.

2.3.2 Motion Planning:

Motion planning is a planning related to the property of trajectory followed by a robot (Velocities/acceleration) performed by the robot. While path planning is focused only on the generation of path followed by a robot. Some of the application of path planning are: Navigation, Mapping and coverage. In robotics the effective space in a given ROI, the ROI space without the subspace occupied by the obstacle usually called a free space (FS).

$$FS = W \setminus U\, C_i \tag{46}$$

$$Where\, W = the\, robot's\, work\, pace, C_i = i^{th}\, obstacle \wedge FS = The\, robot's\, free\, sapce$$

The algorithms which are related to coverage path planning(CPP) problem known by variants of traveling cells man problem[36], lawn mower problems[37], piano mowers problem[38],[39]the art gallery problem[40] and the watchman route problem[41] are all examples of real problems of kind difficult to solve.

2.3.2.1 Cellular Decomposition

The cellular decomposition strategy is a strategy to decompose polygon into simple polygons

"It follows that a key strategy for solving problems on simple polygons is to decompose them into simpler polygons."[24].

The trapezoidal decomposition is the simplest cellular decomposition technique specially used to create an offline CPP[19] [42].

In [1], an algorithm proposed based on the trapezoidal decomposition, to decompose the agricultural field into trapezoids. It is a two steps algorithm that starts with a trapezoidal decomposition of the field followed by a cell merging procedure. According to [24] In trapezoidal decomposition process the given free space (FS), will be decomposed into trapezoids and this trapezoids will also be re-union as much as possible to make them bigger a gain, so that they will be better suited for coverage path planning.

CHAPTER 3. Methodology

3.1 Introduction

The main aims of this work are firstly to develop a coverage path planning (CPP) app, entitled 'Plow Planner', free installable app that can help in planning CPP to support farmers in their tasks in agricultural fields by uploading the proposed planned paths into their machines. Well developed methods, algorithms, open sources from the earlier works of GIS, Agriculture, and Python libraries has been adopted for the success of this aim. Secondly to introduce a method for defining optimized direction for the splitting and coverage paths called MRR, since the methods known for finding optimized directions so far contributes in costs of slowing the algorithm performance.

The main idea is that farmers will uploading the proposed planned paths from the app into their machines. Then, agricultural machine can read the paths, and navigate throughout the field Region of Interest (ROI) (see figure 23). This app has evolved through several steps. These steps will be described below.

Plow Planner incorporates:

1. The primary CPP algorithm

2.The interface that can instruct the user graphically on how to use the CPP algorithm.

3. Tools for importing/exporting (Lat, Lon) coordinates and exporting planned paths as shape and geojson files.

4. Drawing and erasing tools which assist users in drawing/removing ROI/inaccessible areas (i.e.,holes) inside/across ROI.

5. Leveraging different base maps including Open-source maps (OSM) to support users in drawing and contextualizing the planned path.

6. Help button to provide the user with instructions on how to use the app.

7. Clearing tool for removing all drawings from the background.

8. Exit button to exit the app

9. The app will be converted to an installable exe file. So that the app can be distributed, installed to any computer, and used by any user/farmer.

All of the code required to implement this proposal is written in Python. Shapely, tkintermapview, and utm libraries are used in the coding process. Shapely incorporates many ready-made shape manipulation methods, whereas tkintermapview is a widget to display tile based maps like OpenStreetMap or Google Satellite Images. TkinterMapView is a tile based interactive map renderer widget for the python Tkinter library. Utm is another library to make the projection transformation for the input/output tools, such as converting (Lat, Lon) to UTM coordinates (Easting, Northing). Each of the aforementioned steps will be described as follows:

3.2 The Proposed Coverage Path Planning (CPP) Algorithm

The foundation of the proposed algorithm is inspired by previous algorithms and techniques in the literature and it is developed in Python.

3.2.1 Relevant Literature on Decomposition Techniques

The well-known cellular decomposition techniques will be used in this process; namely i) Boustrophedon cellular decomposition and ii) Trapezoidal cellular decomposition (trapezoidal) techniques. These techniques have been employed in many studies on CPP and motion planning.

These are techniques widely used in connection with decomposing the field area of any given shape and size to optimal sub-shapes suited for to plan coverage path on each sub-shape. The optimized paths are those proposed long paths with minimum number of turnings for the machine.

Each of these methods has advantages and disadvantages. When compared to the Boustrophedon technique, the decomposed sub-shapes derived from the given region of interest (ROI) in the trapezoidal technique are usually many in numbers. This will increase the number of incomplete path spaces (IPS) (see figure 12) created for each sub-shape. This means that the more shapes produced, the more IPS will be produced. This IPS produces some narrow strips within the ROI. As a result, the machine should cover these narrow strips separately.

Figure 12: Normal machine width(w) vs incomplete path space in path planning of a sub-shape.

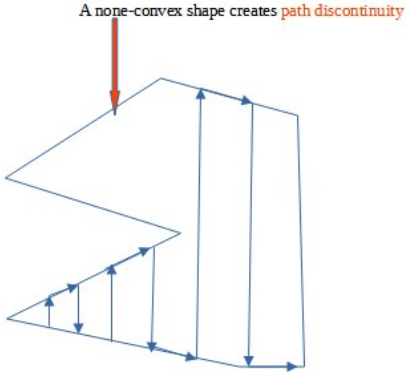

Figure 13: None-Convex polygons are polygons whose maximum of internal angles is greater than 180 degrees.

The trapezoidal technique has this advantage to produce more convex polygons (see figure 14). Paths are easier to design in convex polygons than in non-convex polygons. In contrast to convex polygons, non-convex polygons can cause path discontinuity. This path discontinuity prevents the shape from being completed as a continuous path. As a result, the machine must return and complete the tilling process on the portion of the shape that has not been tilled due to this circumstance (see figure 13). The Boustrophedon technique occasionally produces non-convex shapes, which can result in path discontinuity. Boustrophedon and Trapezoidal techniques are used in conjunction in this work to take advantage of the best features of both.

The proposed CPP algorithm has two primary functions. 1) A function that divides a given ROI into sub-shapes and 2) A function that creates a coverage path that explicitly covers each sub-shape. The proposed algorithm works for any given ROI' shape and size with any number of holes within that defined ROI. The algorithm needs the ROI vertices positions in (lat, lon). The vertices of the holes, if any, within the ROI is considered separately. The machine turning types as (U, omega, straight AB) and the effective machine working width are also the needed inputs for the algorithm. The proposed CPP algorithm only works for the 2D ROI and does not consider the landscape or topography of the ROI. In the process of splitting and designing the path coverage for the sub-shapes, the input (Lat, Lon) are transformed to UTM coordinates (Easting, Northing).

3.2.2 The Splitting Function in the Proposed Algorithm

The split function divides the given ROI's shape with possible holes and finds the best direction to make a sub-shape/strip so that longer paths can be designed in each sub-shape. Based on the literature, if the paths are longer, the number of turns will be less which results in turn optimized CPP, where all other conditions are constant. Therefore, determining the suitable direction to split the given ROI into sub-shapes or strips is significant.

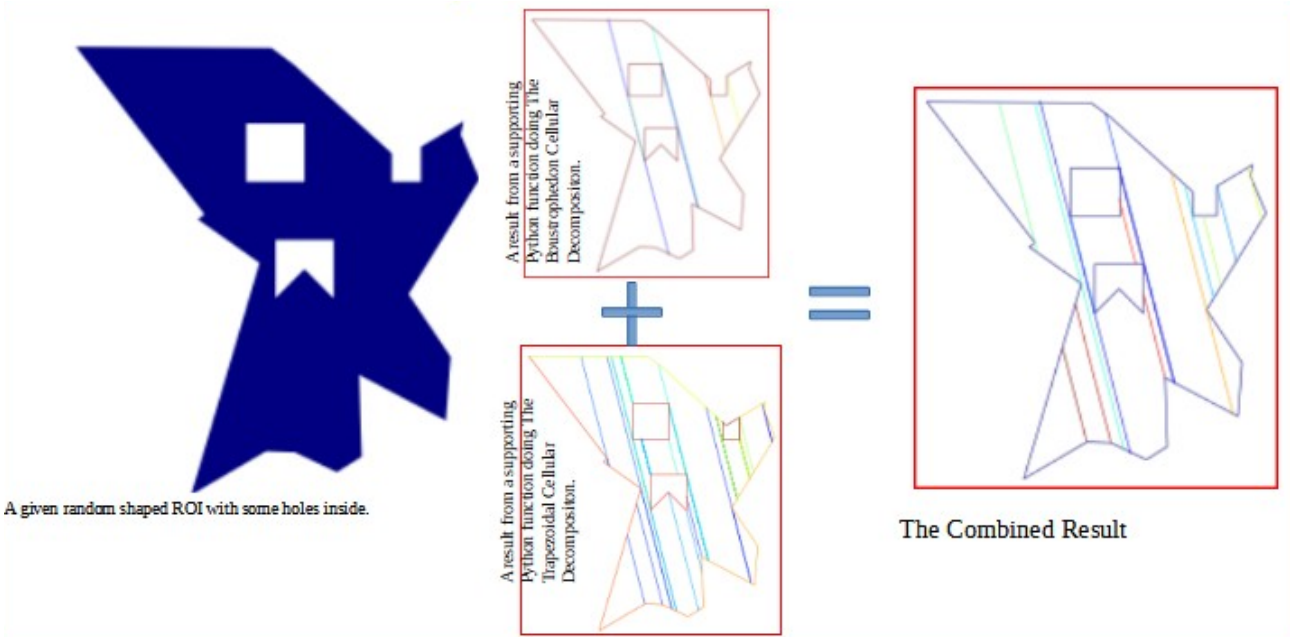

Figure 14: Combining Best behaviors From Both of The Boustrophedon and Trapezoidal Cellular Decomposition Techniques

To optimize the splitting process, the underlying ideas used in each of these techniques are taken into considerations and suitable features from both Boustrophedon and Trapezoidal cellular decomposition has been taken as advantage. The goal is to find optimized sub-shapes without an extra steps to further merge some of the sub-shapes. Because the trapezoidal cellular decomposition results in many sub-shapes/cells, re-merging some adjacent cells will be required to find fewer optimized sub-shapes. Whereas the Boustrophedon produces a result with fewer sub-shapes, however, some of them are non-convex shapes that may cause discontinuity in proposed paths. As

example is given in figure 15 in which the sub-shapes/cells are more optimized and convex compare to those of results from either techniques in figure 14.

Figure 15: The Combined results from Both of the cellular decomposition techniques will be converted to real sub_shapes/Strips with the Shapely Python library.

3.2.2.1 The Direction of Splitting

Another critical step in successfully splitting a given ROI is determining the direction of splitting. Detecting the general aspect of the given ROI shape is a common process for determining the optimal direction. The minimum rotated rectangle (MRR) that circumscribes the given ROI shape can indicate the ROI's general aspect. As shown in figure 16, the MRR for the given ROI provides a hint on which to base the optimal direction for splitting a given ROI. Defining the direction for the aspect of a ROI based on (ROI-MRR) is considered to select an optimized direction to split the given ROI.

Splitting the ROI with the resulted direction from ROI-MRR technique outcomes longer sub-shapes/stripes, which lead to longer paths.

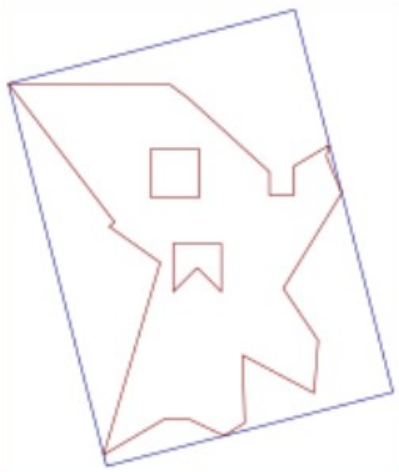

Figure 16: The minimum rotated rectangle circumscribing the polygon indicates the general aspect of the given ROI.

3.2.2.2 The Inaccessible Areas or Holes

Before the splitting process begins, the holes/inaccessible areas within the ROI are subtracted from the given ROI. These holes should always be polygons; for example, if an electric line runs through the given ROI, the hole will be defined as a long rectangle that crosses the ROI with some buffering area of the electric line. The same concept is applied for roads or rivers. When such issues arise across the ROI, it is worthwhile to plan two separate ROIs, one on each side of the crossing line. Because the algorithm's chosen aspect is optimized if they are considered separately rather than as one in whole as represented in (figure 17).

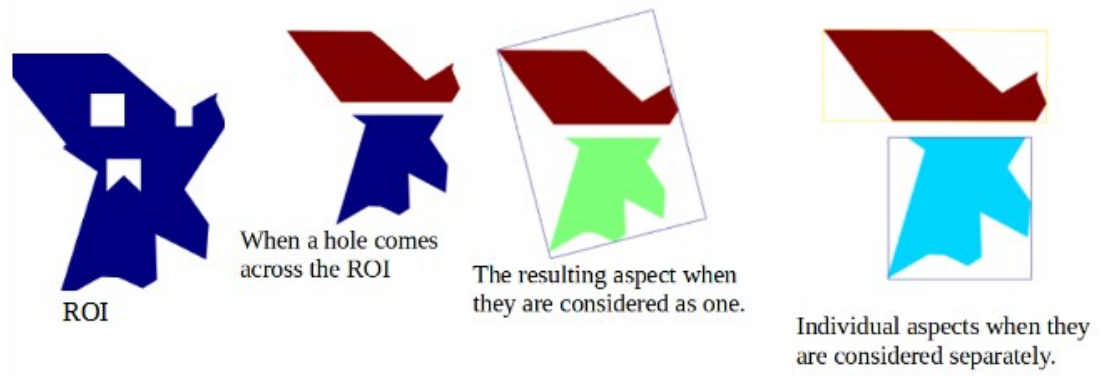

Figure 17: If a ROI is crossed by a hole coming across the ROI, it will be preferable to consider the parts of the ROI separately than considering the crossed ROI as a single ROI. It is because the algorithm can define the splitting process according to the optimized direction of each one of them that will give a better result that considering the crossed ROI as one.

3.3 The Proposed Algorithm.

3.3.1 The Splitting Function

The function dedicated to decomposing/splitting the given ROI is the main part of the proposed app. The splitting process begins with determining the best direction with the defined function. This thesis proposes the Minimum Rotated Rectangle (MRR) method to define an optimized decomposing function. This is explained in the section 'The Direction of Splitting'.

The decomposition function uses a hybrid approach to combine the two well-known exact cellular decomposition techniques, namely the Boustrophedon and Trapezoidal cellular decomposition. As explained earlier, the idea is to take advantage of both algorithms' strong sides.

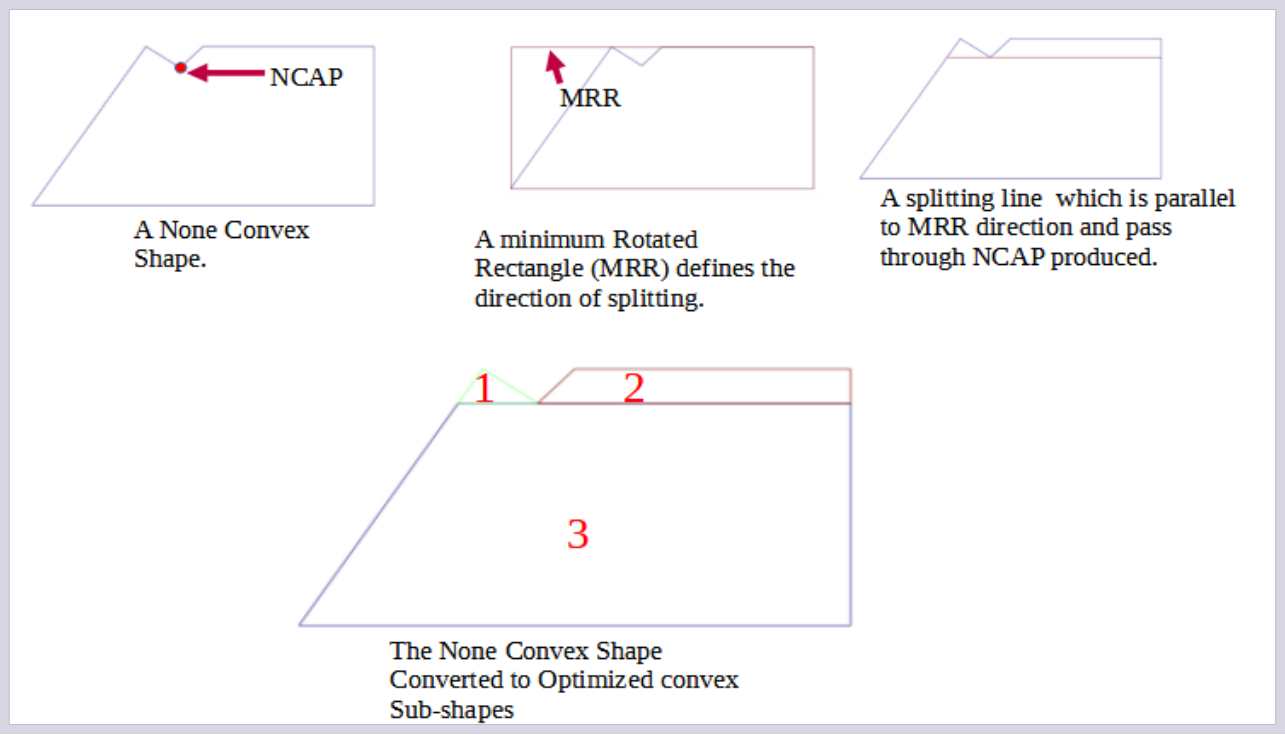

Figure 18: From left to right showing how the splitting function of the main algorithm transforming the non-convex shapes to convex sub-shapes.

Internal angles are the primary characteristics that distinguish convex shapes from non-convex shapes. To be classified as a non-convex shape, at least one internal angle must be greater than 180o. Converting a non-convex shape to a convex shape means reducing the larger angle to less than 180o. This is accomplished in this work by splitting the non-convex shape right at the Non-Convex Angle Point (NCAP) where the larger internal angle occurs. A supporting Python function

for detecting NCAP was developed as part of this process. Another supporting function that creates a splitting line that goes through NCAP and parallels to the MRR direction is required. The splitting function divides the given shape into convex sub-shapes. A single process may not always produce the expected all convex-sub-shapes.

Another supporting function was required to detect the remaining non-convex shapes from the first output and bring them through a new process until all became convex sub-shapes. This process has been developed into two distinct cases, namely when there is no hole and when there is/are hole/s in (figs. 18, 19).

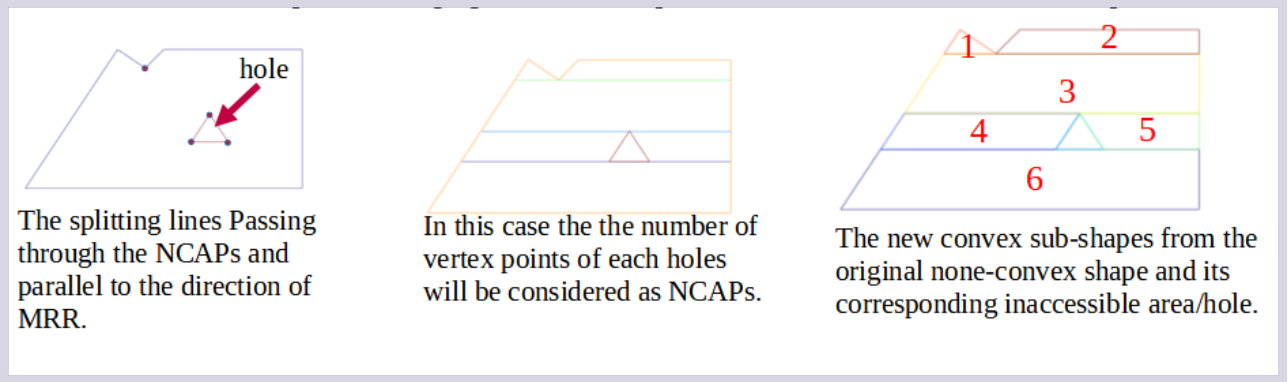

Figure 19: From left to right showing how the splitting function of the main algorithm is splitting the None-convex shapes to Optimized Convex Shapes when there is/are hole/s.

3.3.2 The Coverage Path Function.

The splitting function produces convex sub-shapes. This function iterates through all of the sub-shapes, placing the optimized paths that cover each sub-shape. Individual paths must be as long as possible in order to reduce the number of turns [12]. However, the task of path coverage design includes not only length optimization but also head land angle optimization. As a result, the task of path design is to find the best direction that meets the following two conditions: i) Path length condition which results long paths based on the determined direction ii) Condition of the headland angle varies from few-degrees to 90-degrees, and the best result is found when it is 90). The lower this angle, the worse the result in terms of turning distance. This procedure can be seen in: (figure 21 and formula 48).

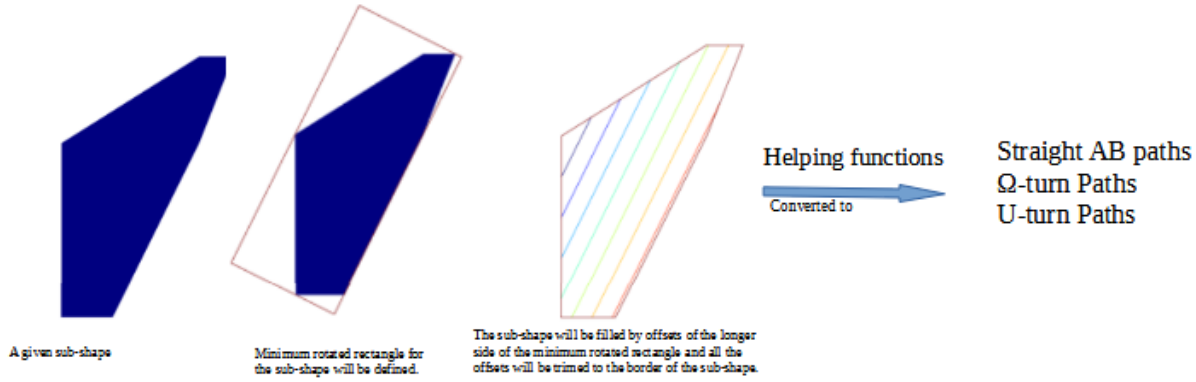

Figure 20: Coverage Path Designing Process

Failure to find the optimal path direction has the potential to increase the overall cost of conducting agricultural tasks. This direction will be determined in this assignment using the same technique that earlier used to determine the splitting direction for the ROI. It is parallel to the minimum rotated rectangle circumscribing the sub-shape (sub-shape MRR) (figure 5). The optimality for the two previously mentioned conditions can be demonstrated by comparing the total sum of distance traveled by the machine to complete a task in each ROI. As the chosen path direction fulfills the maximum optimality, the sum of the path is the smallest. The following formula shows the sum of all distances of parallel paths at a single sub-shape i.e., the total path distance of the sub-shape (TPDsubshape). This consists of the total sum distance of every path(li) and the turning distance at the end of every path(tdi).

$$TPDj_{subshape} = \sum_{i=1}^{m} (l_i) + \sum_{i=1}^{m-1} (td_i) \tag{47}$$

Where, $TPDj_{subshape}$ = *total coverage distance of the* j^{th} *subshape* $\in$ *the ROI.*
l_i = *the* i^{th} *path length.*
td_i = *the* i^{th} *turning distance see the next* (*turnning distance formla*).

3.3.3 Turning Distance

A turn is the unworkable distance that the machine travels between turning points/headlands. A path is defined in this work as a strip covered by the machine swath width (the effective operating width of the machine) as the machine moves in a particular direction. The proposed app gives the users with three options to choose for turnings, which are as follows: 1) A Ω-turn is a type of turn produced by the machine as it attempts to follow the next adjacent paths without creating a gap between them. 2) A U-turn is a type of turn that the machine makes at the headland while passing

through all other paths with a gap of one or more paths. In contrast to the Ω-turn, this turn type makes turning most machine types very easy and efficient because the gap creates enough space to turn the machine. 3) The straight AB path patterns are just paths put adjacent to each other without the turning part. These patterns give ease to the user to select whatever it works for the machine since it is unrestricted by a turn type. In this work, this distance has been explained as it is shown in (figure 21 and formulas 48, and 49)

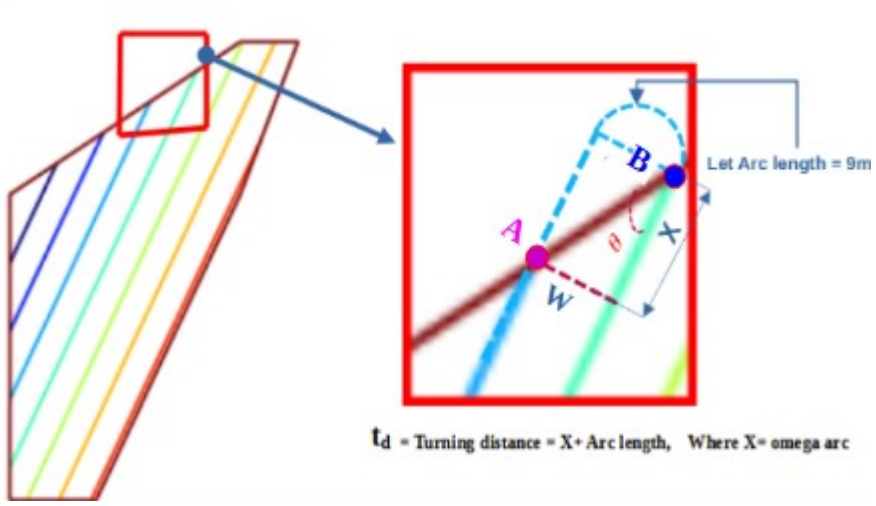

Figure 21: The Turning Distance Calculation

We know the coordinates of ends of lines ***A*** *og* ***B****. This makes possible the distance definition between two end points. Then X can be defined with one of the following two ways:*

$$X = AB^2 - w^2 \tag{48}$$

$$\begin{aligned} X &= \arctan(\theta) * W \\ PD_i &= l_i + \arctan(\theta_i) + arc\,length \end{aligned} \tag{49}$$

Assuming a subshape has m paths. The i^{th} *turning distance* td_i *will be calculated as:*

Where θ *= The angle made by the path against the edge at the headland* $(headlandangle)$.
l_i *= the part of the path without the turning distance.*
W *= width of the machine.,* X *= a part of the turning distance.*
PD_i *= The total of a path.*

The total distance covered by the machine in a single sub-shape and for a selected direction is shown by (formula 50): This is a formula re-defined from (formula 47 and 48) as:

$$TPDj_{subshape} = \sum_{i=1}^{m} l_i \; + \; \sum_{i=1}^{m-1} \left(W * \arctan(\theta_i) \; + \; arc\,length \right) \tag{50}$$

The total distance covered by the machine to fulfill the whole coverage of a ROI can be computed from the base of (formula 47 and 49) as follow:

Assume a ROI consists of n sub-shapes.

$$TPD_{ROI} \; = \; \sum_{j=1}^{n} TPDj_{subshape} \tag{51}$$

Where $TPDj_{subshape}$ = *The total distance the machine traveled for covering the* j^{th} *subshape*, $\in$ *the ROI*.
TPD_{ROI} = *The total distance covered by the machine tocover the whole ROI, Asuming a ROI has n subshapes*.

From the (formula 50 and 51), the (formula 51) can be re-written as:

$$TPD_{ROI} \; = \; \sum_{j=1}^{n} \left(\sum_{i=1}^{m} l_i \; + \; \sum_{i=1}^{m-1} \left(W * \arctan(\theta_i) \; + \; arc\,length \right) \right)_j \tag{52}$$

(formula 52) is developed to compute TPD_{ROI}.

3.3.4 The Plow Planner's Implementation

Based on the aforementioned algorithm, an application called Plow Planner is developed for the CPP. The app requires tools for the users to enable them to manage and manipulate inputs and outputs to facilitate communication between them and the algorithm. At this stage, only some basic tools are developed, however, more implementations needed from other interested researchers to work towards a full-fledged app that can fully support the users. The following are some of the current features of the application.

3.3.5 The Main Features of The App.

An import & export tool is developed to import coordinates for the ROI in CSV format and export the resulted paths in different formats including geojson file and shape file. Data transformations may be required during the process. To measure the length of a line or the area of a shape, for example, the available data (i.e.,Lat, Lon) must first be converted to an UTM coordinate (Easting, Northing). In this work, a Python library called utm was developed in collaboration with some of the functions. These functions, written for this app based on the utm library, assist the user in easily navigating the forward/backward conversion process.

Additionally, two drop-down menus are developed to choose a base-map from a selection of maps, as shown in (figure. 23) and the second one includes options to choose a machine turning types such as U-shape, omega-shape and straight line. This will assist the user in providing alternatives for selecting turning types based on the available machine type. An input box incorporated to enter a

value for machine width in meters. The explanations for all the functionalities is available in (table 1)

3.3.6 The Minimum Rotated Rectangle Method Evaluation

The general aspect direction for a minimum rotated rectangle circumscribing a shape (MRRD) is used to: i) The splitting direction to split the ROI into sub-shapes (see figure 16). ii) To determine the direction of the optimized parallel paths for every sub-shape (see figure 20).

How can evaluate whether the direction yield from the MRRD is the optimum direction?

This can be evaluated by comparing the TPD_{ROI} with a well-known reference distance (WKRD), if any. The WKRD can be found from: i) A manually measured results, or with results of other well-known software. ii) it is known that the best-optimized direction is between 0° and 180° taking as many samples of directions as possible between these values. The samples should be as uniform as possible. For instance, we can take the samples from 0o to 180° with 1° intervals. Having this sample direction all the possible $TPDj_{subshape}$ can be produced (see $nabla_{j1\text{-}180,}$ formula 54). The WKRD for the given sub-shape will be the minimum of all $nabla_{j1\text{-}180}$ values of every sample direction. The direction of the path resulting the minimum sum is the optimized direction for the given sub-shape. Finally, the WKRD and its corresponding direction for every sub-shape will be defined the same way. The optimized paths of sub-shapes selected from this process will generally be taken as optimized paths. Therefore TPD_{ROI} (from Formula 50) will be used to sum all the WKRDs of sub-shapes in the ROI. And this sum can be referred as WKRD for the given ROI ($WKRD_{ROI}$).

The following set builder notation (formula 53 and 54) is trying to explain the set of all 180 possible WKRDs for a sub-shape j.

$$\nabla_{jk} = \left(\sum_{i=1}^{m} \left(l_i + l_i + W * tanarc(\theta_i) + arc\,length \right) \right)_k \quad (53)$$

Where ∇_{jk} = *The sum of all the paths with a single direction* ∇_k *for the subshape* j

$$\nabla_{j_{1-180}} = \left\{ \nabla_{jk} \middle| \left(\sum_{i=1}^{m} \left(l_i + l_i + W * tanarc(\theta_i) + arc\,length \right) \right)_{\nabla_{jk}} , 0 < k < 180 \right\} \quad (54)$$

$$WKRD_{subshape} = min\left(\nabla_{j_{1-180}} \right) \quad (55)$$

The above set builder notation $(\nabla_{j_{1-180}})$ *represents the sum of a set of paths for every one of the* 180 *directions*. *i*. *e*. *every direction have a sum of all paths for that specific direction at subshape* j.

The $WKRD_{ROI}$ of any given ROI computed from this process can considered as a reference to compare with TPD_{ROI} derived from the proposed MRRD method. The following graph shows the result of the comparison of WKRDROI with TPDROI of each one of more than 1000 ROIs of different shapes and sizes each of which has three holes in them and a random area (between 20 and 60 hectares) following the same procedure of the above explanation for every one of the more than 1000 cases.

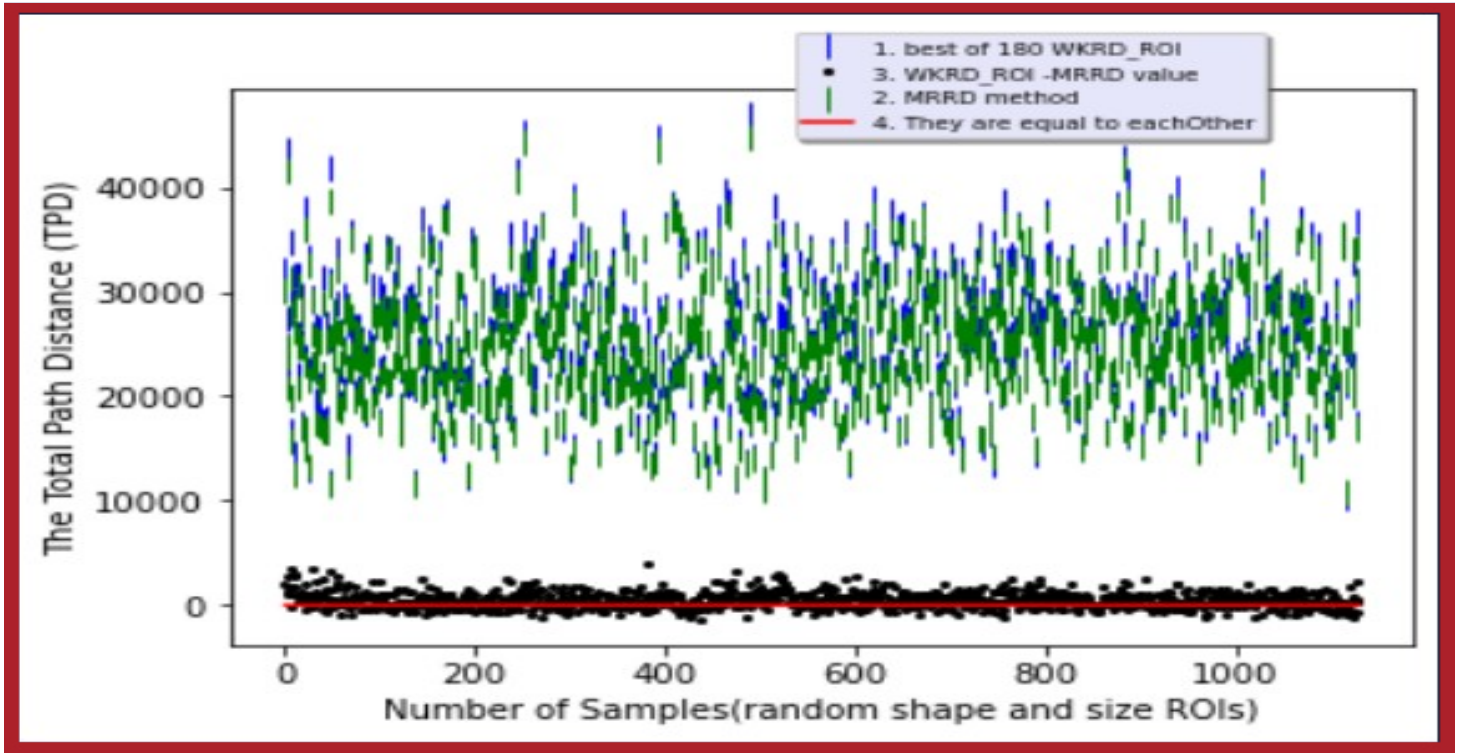

Figure 22: Comparison between TPDs values from WKRD_ROI and the proposed MRRD method. The blue-green sticks represent each of the compared values (Values from MRRD=Green, Values from WKRD_ROIs= Blue). The dot graph represents values TPDWKRD- TPD_{MRRD}.

Every combination of green and blue sticks represents the TPDROI the one from the reference(blue) and the one from the proposed MRRD method(green). This pair of sticks shows which one has a bigger total path distance (TPD). From the graph, it is revealed that the blue color is mostly at the top of the green this indicates the TPD from the reference is bigger than the TPD from the MRRD method. That means in most cases the MRRD method was more optimized (smaller in TPD) than the reference. At the bottom, the dot graph and the red line show the difference between the reference and the proposed method (TPDWKRD – TPDMRRD). The dots below zero show the number of times the reference was better than the proposed method. While the dots above the red line are showing the number of times the proposed method was better than the reference. Most number of the dots is shown above the reference line(the red line). This affirms that the proposed MRRD method was performing better than the reference method in most of the cases.

3.3.7 The Graphical User Interface (GUI)

The graphical user interface (GUI) is designed to ease access to functionalities of the app rather than visual aesthetic features. It is widely assumed that technical and professional users place greater emphasis on the app's functionality rather than its visual appeal. It is unarguable that an app with a suitable interface adds value and meaning to the app's functionalities. However, the overall aim of the app is to demonstrate how to combine some theoretical methods, algorithms, and GIS into a practically usable app.

This app's main visual interface consists of a window displaying the OSM base-map and three groups of different functionalities see (figure 11). i) The first group of functionalities are located on the left side of the window, ii) The second group of functionalities are located on the bottom side of the window, and iii) The third group of functionalities can be accessed by using the right mouse click on the base-map.

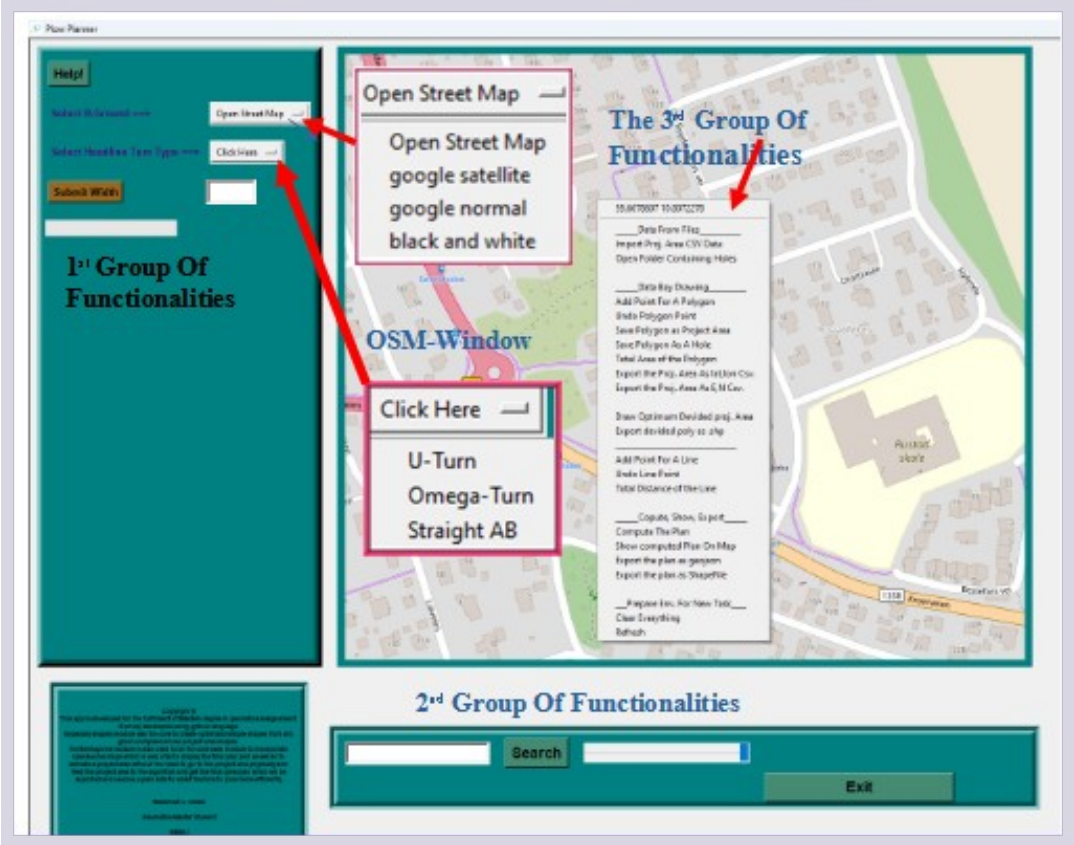

Figure 23: The Plow Planner's Graphical User Interface Environment

3.3.8 Button Nomenclature in the App

The button called by a numbering system of xy.z or xy.z.i, (x indicates functionality group number, y indicates the button group number and z indicates the button number). If the button has contents inside it, or has button sub group then those contents named as xy.y.i where 'i' indicates the content number. For instance the open street map will be called 11.2.1 where as the U-turn can be called 11.3.1 and so on.

For instance, the first functionality two drop-down menus (button 11.2 and button 11.3), the first drop-down menu has 11.2 this stands for functionality group-1, button group-1 and button number 2.

While 11.3 stands for functionality group-1, button group-1 and button number 3. Progress bar (11.4 i.e. button 4) and entry-box (11.5 i.e. button 5), and a Help Button (11.1 i.e. button 1). Users can click 'Help' button to get quick and short information about how to use the app.

One of the two drop-down menus (i.e. 11.2/11.3) helps the user to select the base-map. The second drop-down menu(i.e. 11.3) permits users to choose the turning types like Ω-turn(i.e. 11.3.1), U-turn (i.e. 11.3.2), or straight AB (11.3.3). In addition, users must submit the swath width of machine with two buttons I) the 'submit width' button (i.e. 11.4) and entry-bot to write in the swath width of the machine (11.5).

The second set of functions is located at the bottom of the window. There are 4 buttons, an address entry-box(i.e. 21.1), 'Search' button and a zoom-in/out bar (21.3) and 'Exit' button (21.4). Users can alternatively find a location using the address box to display the location in the center of window. The Exit button with Yes/No options to exit the process.

The third group of functionalities is a set of categorized buttons under a right-click on the map window. Right-clicking on the selected base-map reveals four categories (Those are 31,32,33,and 34). These categories perform similar functions, namely i) data from files (31), ii) data by drawing (32), iii) compute, display, and export (33), and iv) environment preparation(34) .

All the functionalities of buttons, drop-down menus entry boxes, and others that this app incorporates, are described below in (table 1).

Category	Button	Functionality
Data From Files (31)	Import Proj. Area CSV Data (1)	In this app the ROI can be imported as a CSV file.
	Open folder Containing Holes (2)	CSV files each containing vertices of a hole can be imported as a file input.
Data By Drawing (32)	Add Point For A Polygon (1.1)	A click on the map window followed by pressing this button results in one point of the ROI. The procedure should be repeated until the user finishes the drawing the polygon.

	Undo Polygon Point(1.2)	Clicking this button will undo a previous drawn point, Repeating clicks results in removing the drawn points one by one.
Data By Drawing(3 2)	Save Polygon As A Project Area (1.3)	This button should be clicked after the user finishes drawing the ROI. Then the algorithm understands that specific drawn polygon is a ROI.
	Save Polygon As A Hole (1.4)	This button should be clicked after the user finished drawing A hole. Then the algorithm understands that specifically drawn polygon is a hole.
	Total Area of The Polygon(1.5)	This button returns the total area for the polygon in square meters if the user clicks it after finishing the drawing of the polygon, Or right after the user imported the CSV file containing the vertices of the polygon as (lat, lon)
	Exporting The Proj. Areas As (lat, lon) CSV (1.6).	This button helps the user to export the vertices of the drawn ROI as a CSV file in (Lat, Lon).
	Exporting The Proj. Areas As (easting, Northing) CSV (1.7).	This button is to export the vertices of a drawn polygon as a UTM (Easting, Northing) coordinates in a CSV file.
	Draw Optimum Divided Proj. Area (1.8).	This button helps the user to show the optimally split Sub-shapes made by the algorithm. This button can be clicked once the user finishes drawing the ROI and holes /Project area or after importing their (lat, lon) positions of the vertices of the polygon and holes as CSV file.
	Export Divided Poly .shp (1.9).	This button is to export the divided/splitted project area/ROI as a shp file.
	Add Point For A line (2.1)	A click on the base-map followed by a press of this button results a point of a line. The procedure is repeated until the user has finished drawing the line.
	Undo Line Point (2.2)	This button undoes the previous drawn point of the line one by one as the user repeats the click.
	Total Distance of The Line (2.3)	This button returns the length of a drawn line in meters.

Compute, Show, Export(33)	Compute The Plan. (1)	This button should be selected after importing or drawing all of the data (vertices of the ROI/Project area or hole/s, if any). Following that, the swath width is entered and the type of turning/path is chosen. This button will start the algorithm, which will check all of the data submitted and begin splitting the ROI based on the data and designing the paths based on the path type selected. Depending on the size of the ROI, and the number of holes, this process can take from a few seconds to a few minutes. A ROI of 3000 hectares, with about 5 holes for example, takes about 1 minute.
	Show the computed plan on Map (2).	This button show the CPP after the computation process is finished.
	Export the Plan as geojson	This button helps the user to export the CPP as a geojson file.
	Export the Plan as Shape File (3).	This button helps the user to export the CPP as .shp file.
Prepare Env. For New Task (34).	Clear Everything (1)	This button assists the user to clear everything from the screen/background as well as from the temporary memory of the app. N: B, Be careful to not to press this button before you save things you need on the screen. Because this is irreversible erasing.
	Refresh (2)	The button will be functional to refresh the screen.

Table 1: Functionality Description Table

CHAPTER 4. Result and Discussion

This work's primary goal is to propose a solution for coverage path planning (i.e., CPP) problem. An application was developed to assist farmers in planing optimized paths for their agricultural machinery. Secondly, a minimum rotated rectangle (i.e., MRR) method for defining optimized directions for the splitting and coverage paths is introduced. In this section, the results of this work will be demonstrated and discussed as follows.

1) The algorithm is explained in detail in the Python environment. 2) The core computational function of CPP of the algorithm will be demonstrated. 3) The algorithm's applicability on curvy boundaries and their respective effects like(pathless gaps) will be demonstrated. 4) The proposed algorithm will be tested for applicability on various ROI shapes, particularly the challenging ROI shapes mentioned in the literature. 5) The algorithm's applicability will be tested using a real-world example of a random ROI, and the coverage path for each sub-shape from the ROI will be computed, first using the proposed app and then using a well-known method. The outcomes of the two methods will be presented and discussed. 6) Discussion about the speed of algorithm. 7) The app can design the paths according to turning systems(Ω-turn, U-turn, and straight AB), how the algorithm handles this turning system will be demonstrated and discussed.

4.1 Results

The outcomes are produced either by running the algorithm in a Python environment without a GUI (fig. 47) or by executing the app using the GUI (fig. 48).

4.1.1 The Proposed Algorithm in Nutshell

The proposed algorithm that computes the coverage path for a given ROI has been schematically illustrated in (fig. 24). From left to right, the first figure shows the area after the algorithm subtracts the holes, and other obstacle types from the given ROI. In the middle figure, the reference lines represent the striped area at every NCAP points with direction parallel to the derived general aspect of the minimum rotated rectangle circumscribing for the given ROI. These lines are replaced by the real sub-shapes represented in the rightmost figure followed by results of performing the application as of optimized path for two selected headland turning types. It should be noted that the app supports ROIs with curved boundaries. However, curves should be defined with as many points as possible based on the accuracy required for the project. Each of those points will be the ROI or hole's vertices. In figure 26, 34, 34, and 29, different examples of area with curved boundaries are represented. As can be seen, the shape of ROI has a significant impact on how the sub-shapes and CPP results form.

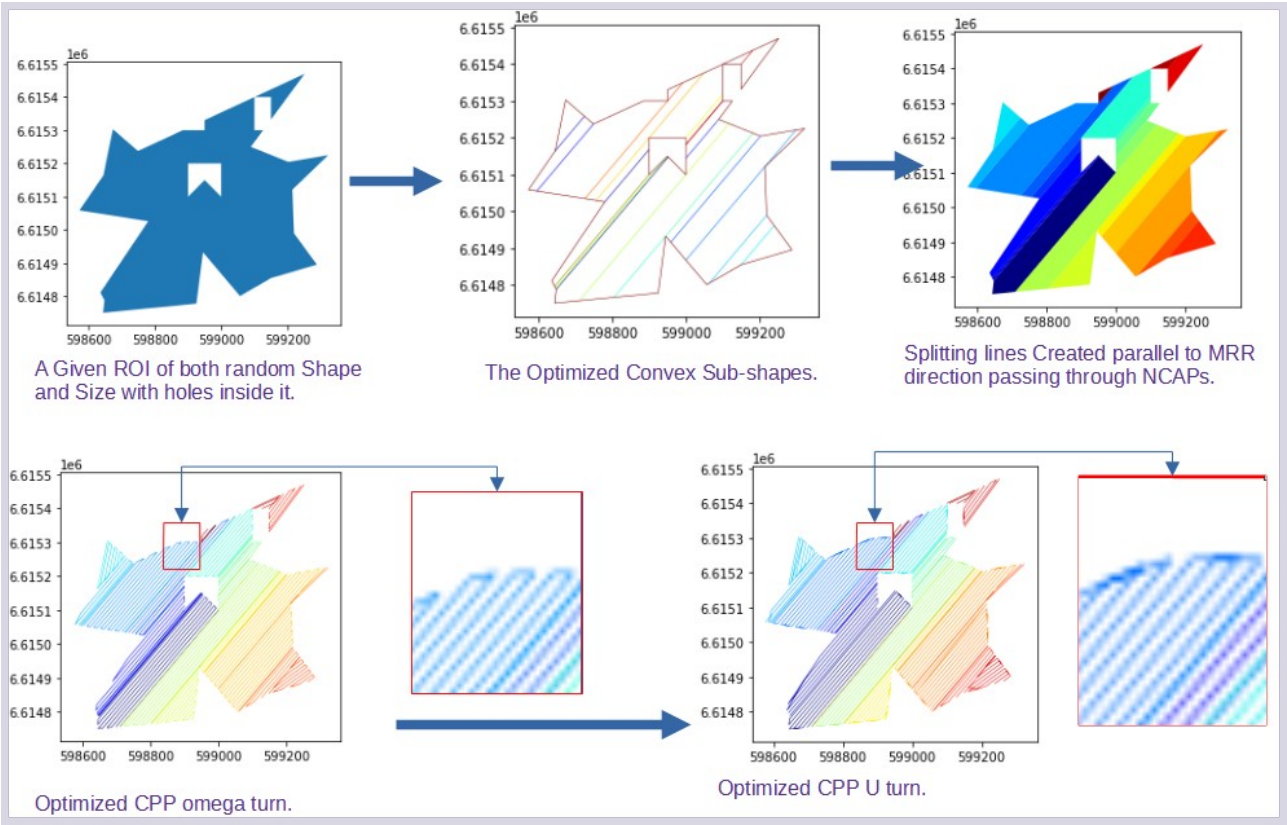

Figure 24: Schematic overview of algorithms for constructing CPP for a given ROI, Area: ***Ca: 30 hectare****.*

4.1.2 The Proposed App with a User Friendly Interface

An interactive application with a user friendly interface is developed to ease the production of optimized paths for the given ROI. In fig 25, as an example, a random ROI in Ås is selected as a project area followed by the computed CPP for that area. A variety of base-maps are provided (fig. 25) which help users to navigate through and in case, also to draw the ROI and possible holes just as shown steps (1 and 2), and other obstacle types on the map according to (fig. 25).

Step 3) The swath width will be submitted (NB: The swath width and the headland turning type can be selected before any step is started, but they do not have to be done at the start; they can be done right before the computation step or any time before the button 33.1 is clicked. Step 4) The button 33.1, i.e. “Compute,” will be selected. Step 5) Waiting until the progress bar (button 11.6) indicates that the computing process is complete. Step 6) When button 33.2, “Show Computed Plan On Map,” is selected, the computed optimized path coverage is displayed. (See Figure 25 for this process.)

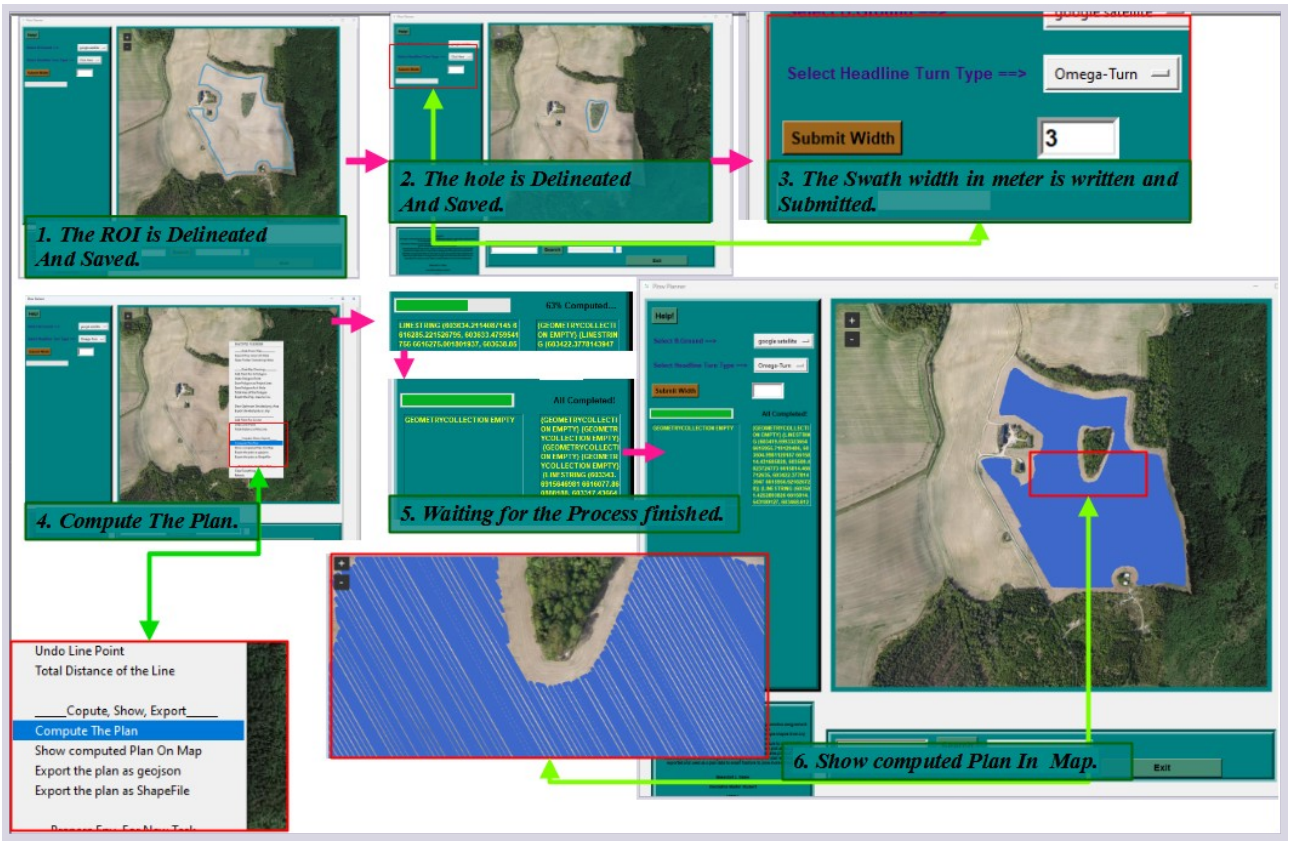

Figure 25: Illustrative example of the developed app to compute CPP for a given ROI.

4.1.3 ROIs With Curvy Boundaries

The app works for ROIs with curvy sides. However the curves should be defined with many points as possible according to the need of the accuracy for the curve. Each one of those many points will be vertices of the ROI/hole.

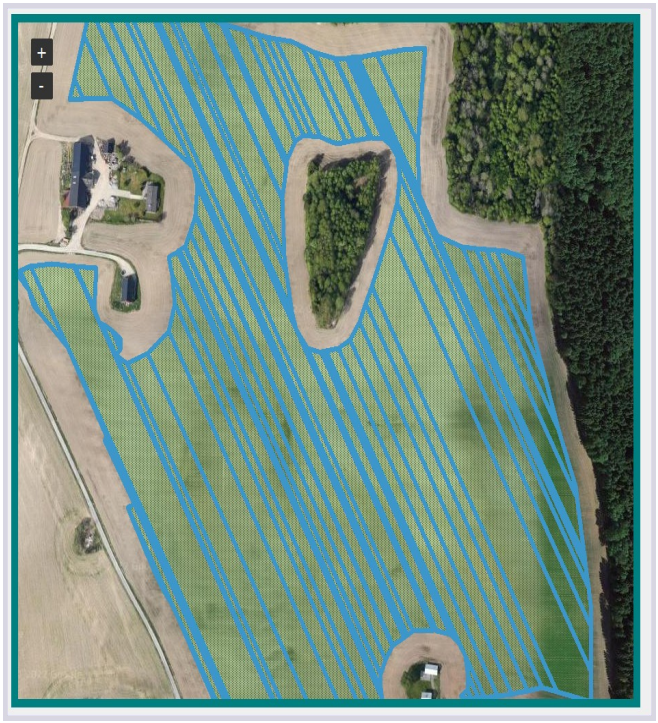

Figure 26: ROI with curved boundaries and the result of their split sub-shapes

Figure 27: Some gaps as Results of CPP for a ROI and its hole with many curved boundaries.

Figure 28: By converting the curved sides of the ROIs into straight lines with better size and smaller in number of sub-shapes will be produced. Which will have better sizes each to design many paths. The smaller they get in their number, the sub-shapes will create path discontinuity less often.

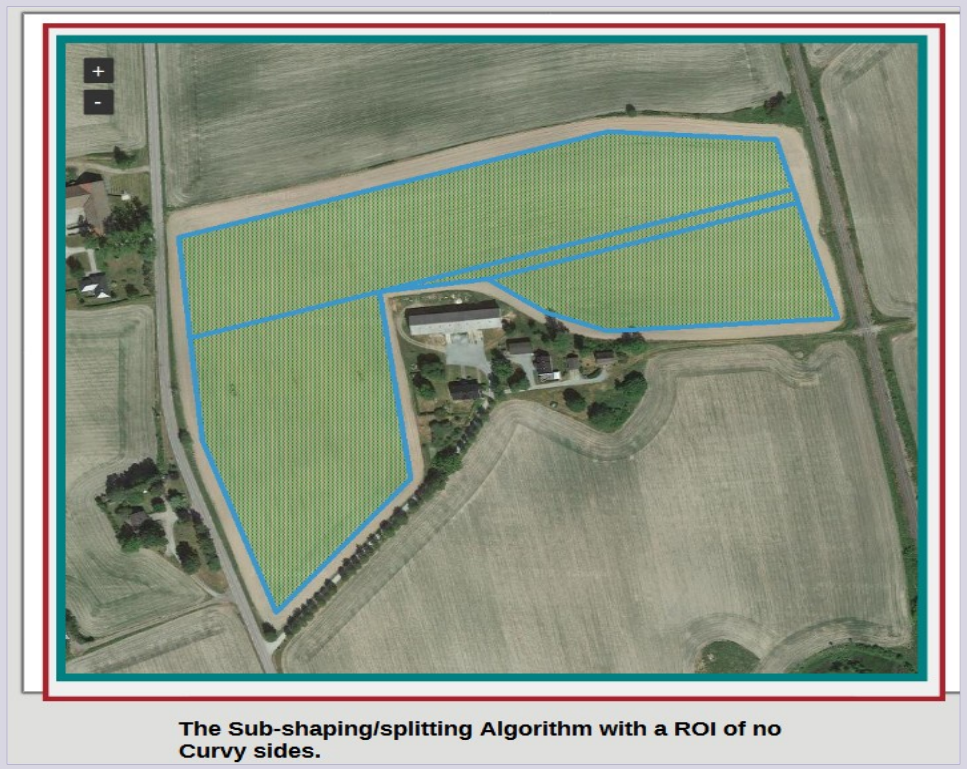

Figure 29: A ROI with straight sides and its sub-shapes output.

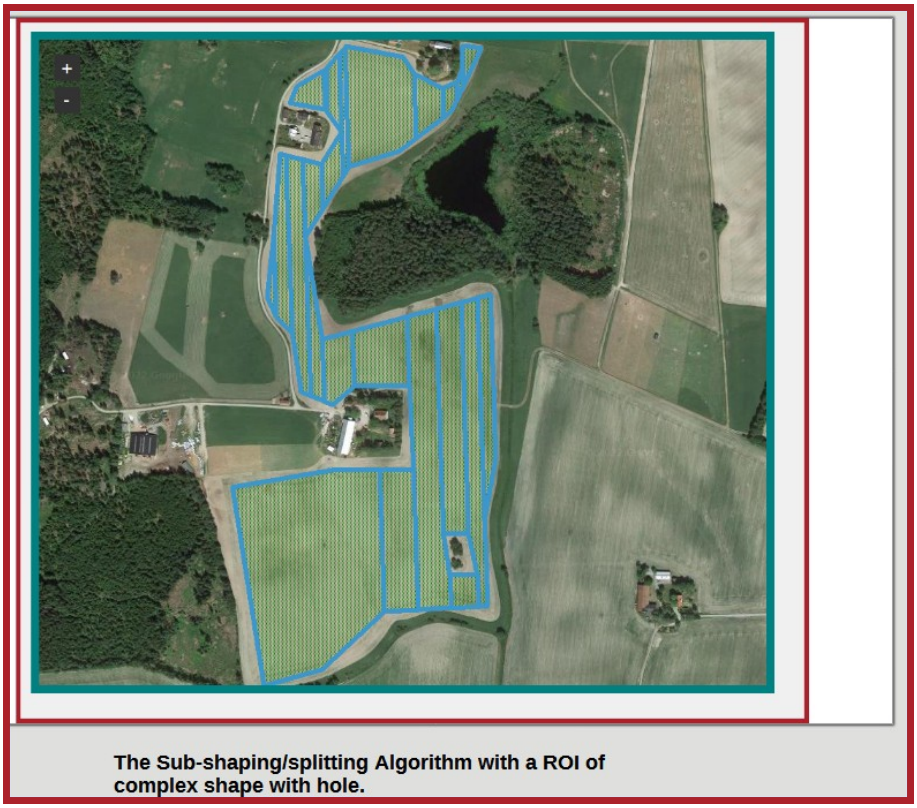

Figure 30: A complex ROI and the results of its sub-shape.

4.1.4 The Applicability of The algorithm For Different Known Shapes

The app can be examined further for other ROI shapes, particularly the ‘L’, ‘S’, ‘C’, ‘T’, and ‘H’ shapes that are frequently mentioned in the literature [1], in relation to CPP. (figures 31, 32, Error: Reference source not found, Error: Reference source not found) showing some ROI examples with different shapes and their CPP results respectively.

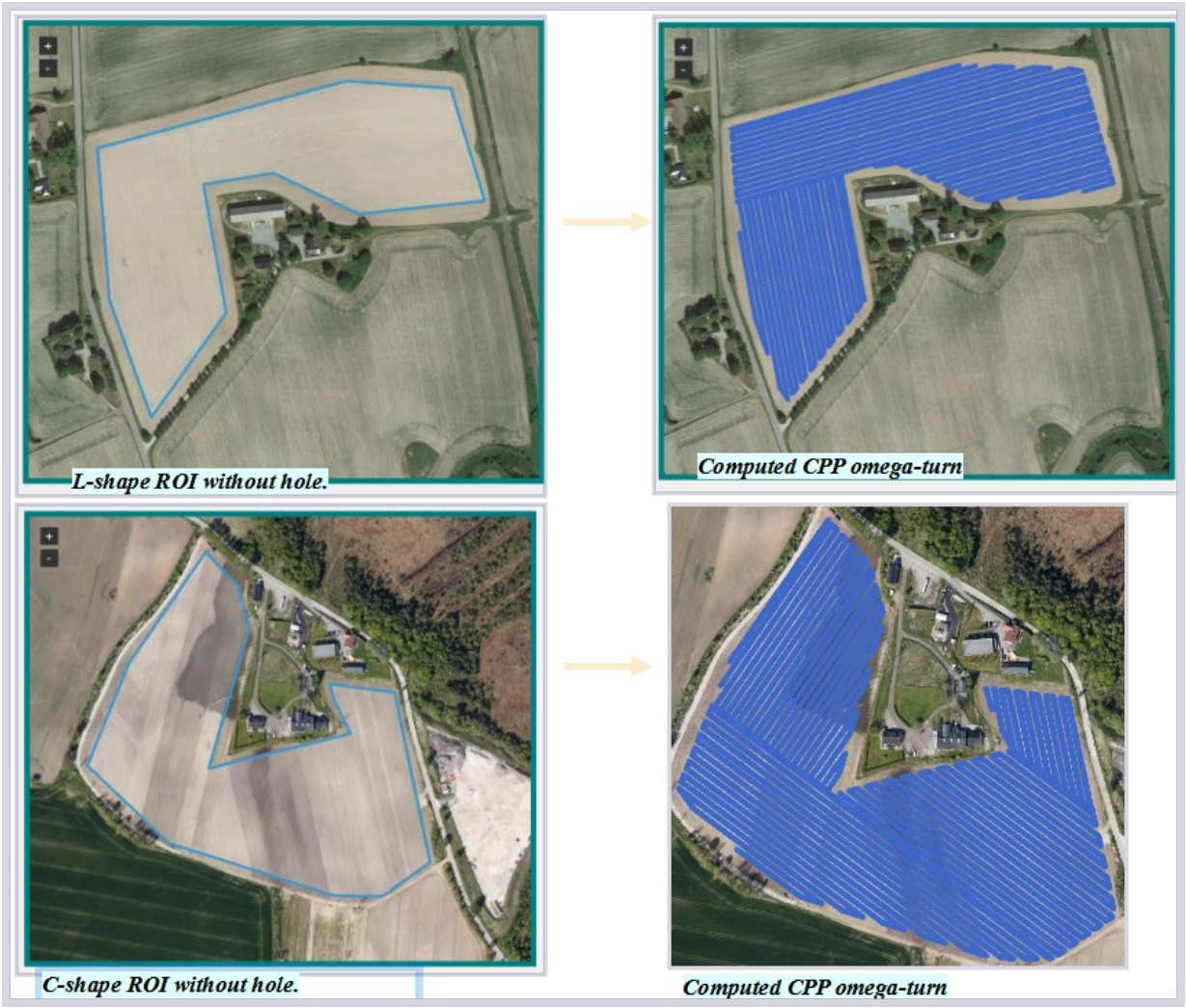

Figure 31: ROIs with 'L' and 'C' shapes and their CPP results.

This app has been successfully tested with many ROIs of different size and shape. The following is a ROI a little bit similar with ‘S’. The following was the output of this ROI after it has gone through the algorithm.

Figure 32: The CPP output after the app applied to a complex "S" shape ROI.

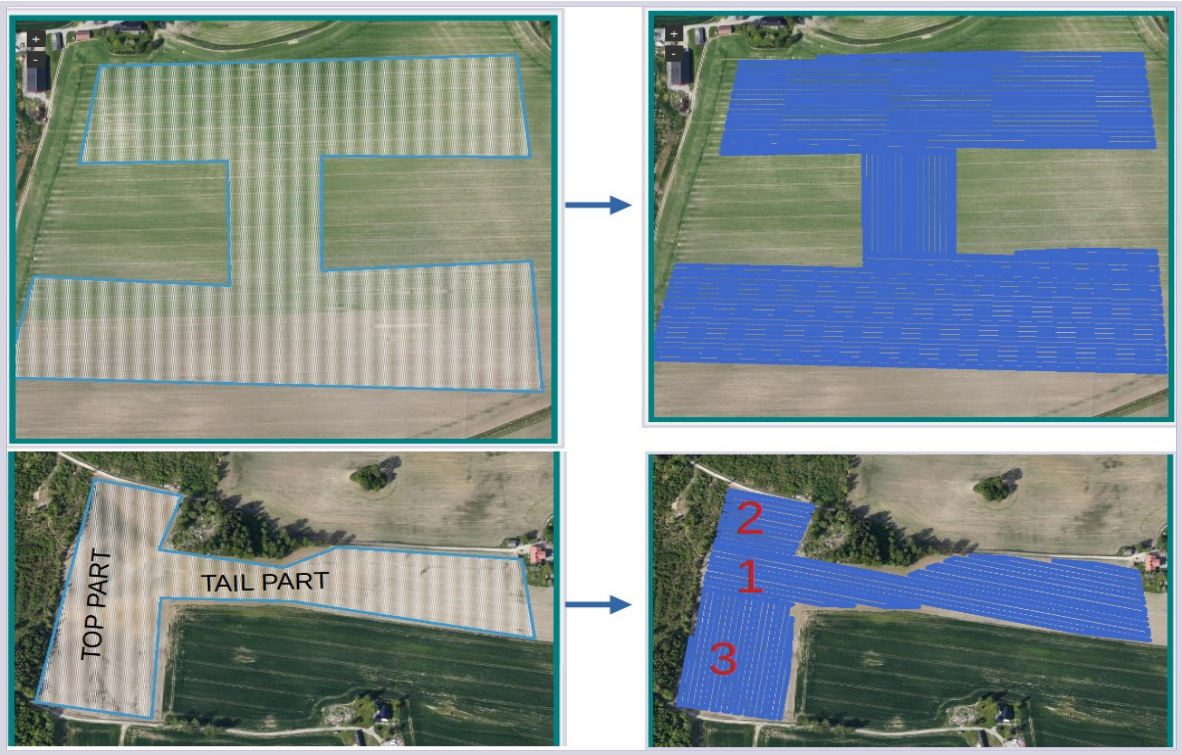

Figure 33: The CPP output after the app applied to an "H" (above) and a "T" shape ROI(below) with longer tail.

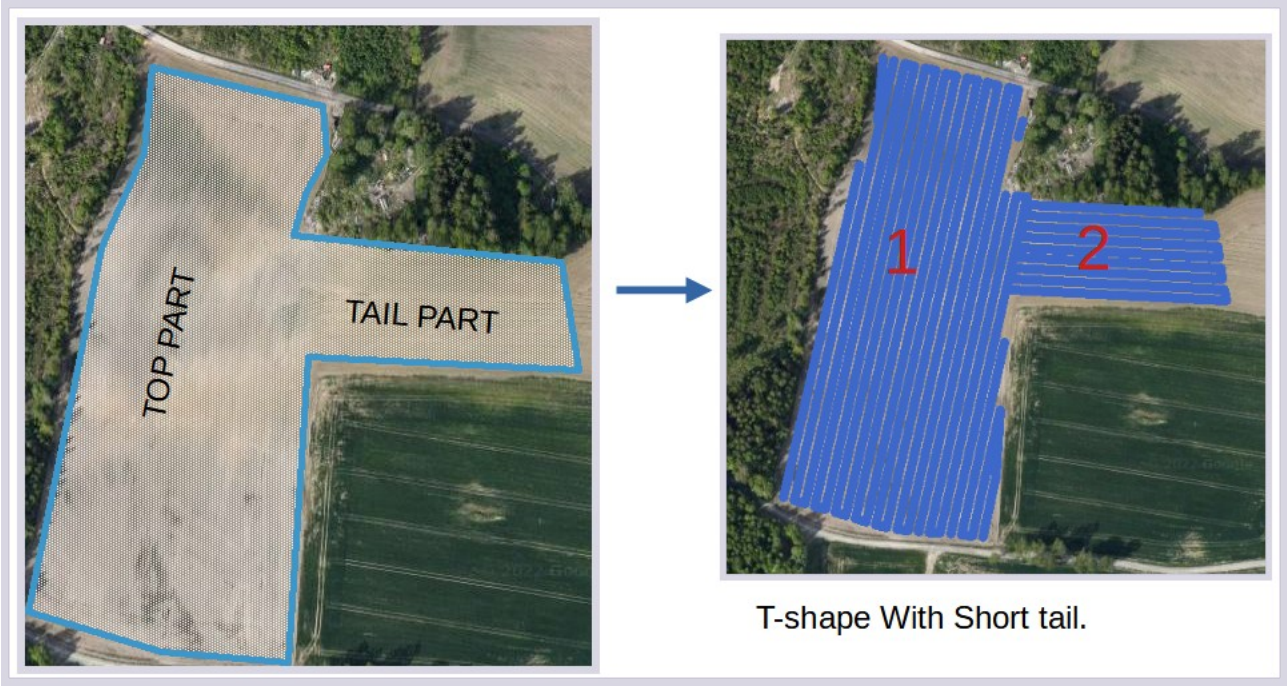

Figure 34: T-shape's ROI with short tail

A comparative analysis is performed to evaluate the result of the proposed CPP algorithm with that of a well-known method. This experiment is carried out for more than1000 different ROIs with different shapes and sizes between areas 15-50 hectare. The outcome of this experiment has been thoroughly discussed in the methodology section. In (fig. 35), one of the examples of ROI is illustrated.

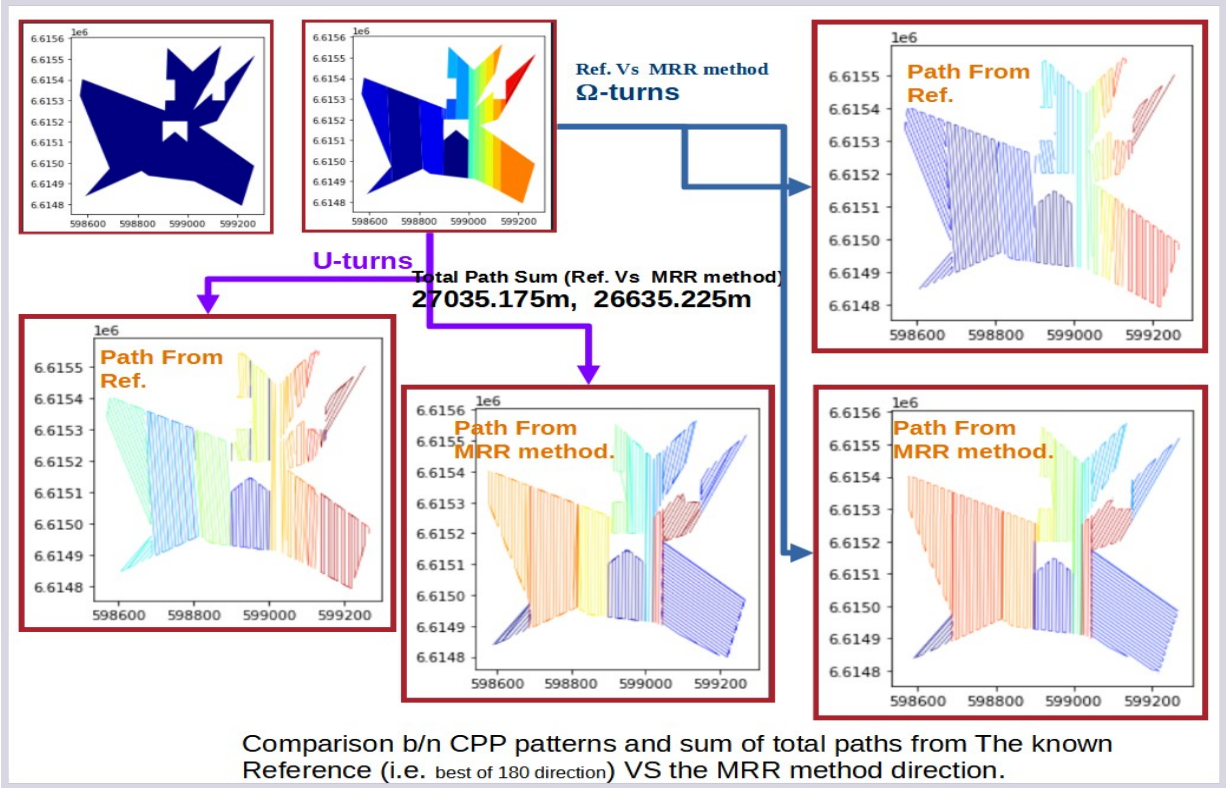

Figure 35: A random ROI with a random non-convex shape/size has been taken, and let the algorithm's sub-shapes have been covered by paths with two methods.

1) The direction of the path has been driven from the best of from 0 to 180-degrees directions(with 1° interval) for every sub-shapes as a Reference (Ref).

2) The direction for each sub-shape taken from the newly suggested method (MRR method). Their resulting path patterns and the total number of paths in the ROI from the CPP outputs of both methods have been computed.

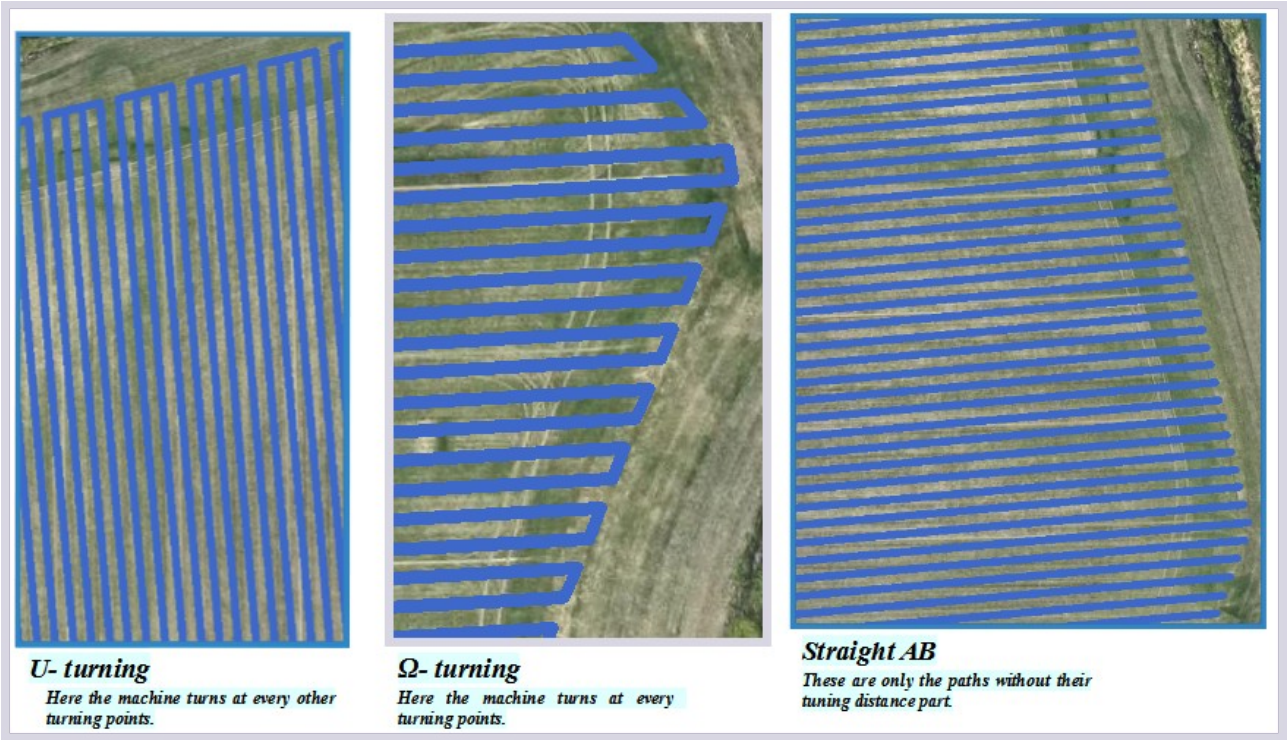

Figure 36: Paths with their turning systems with the proposed app.

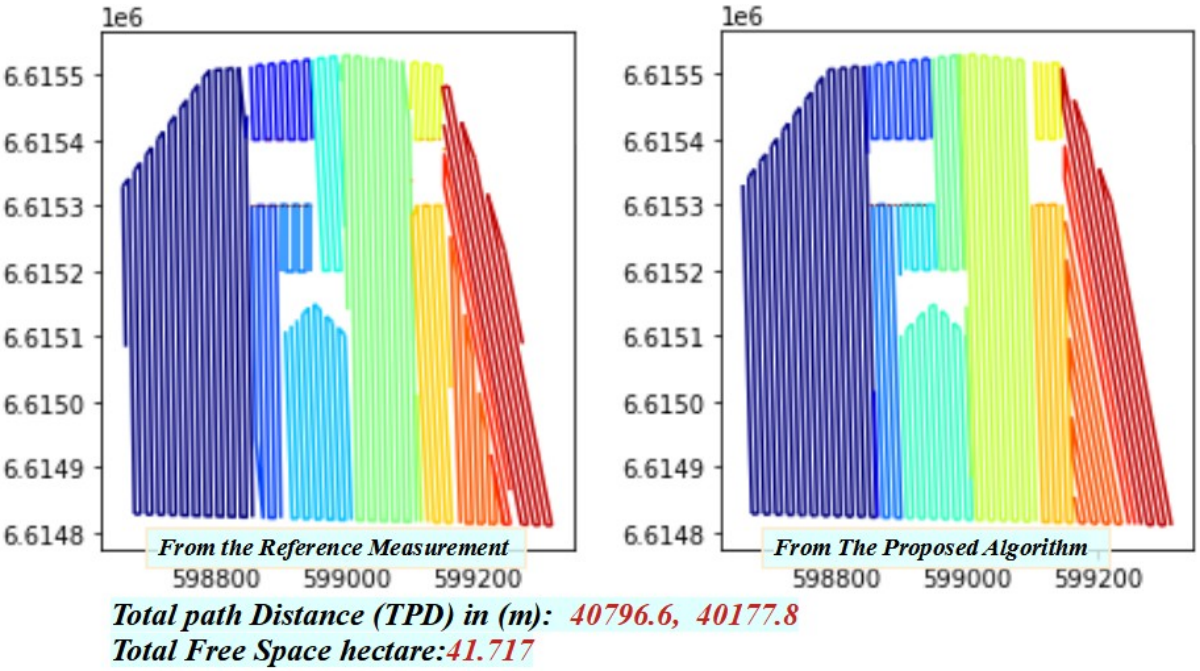

Figure 37: A random ROI with a random convex shape/size has been taken, and let the algorithm's sub-shapes have been covered by paths with two methods. And the result TPD was as shown in the figure.

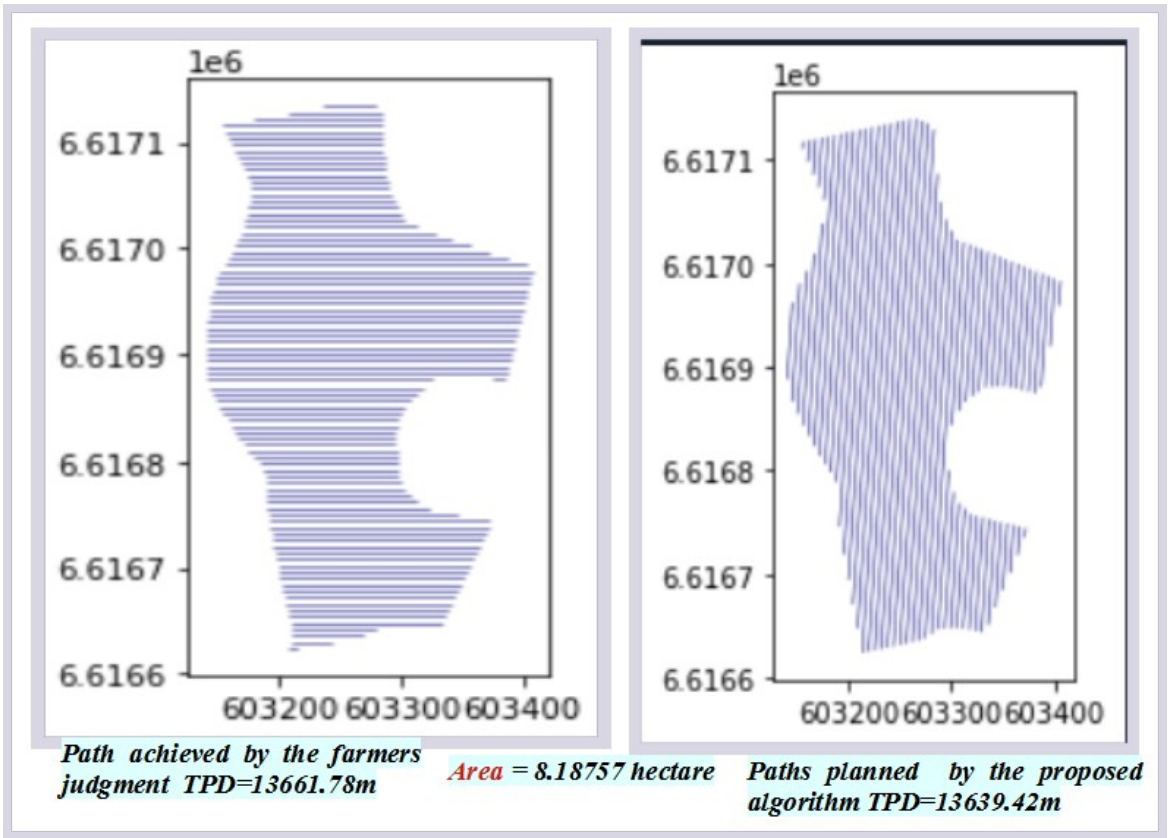

Figure 38: A random farmer's tilled ROI tilled by the farmer(left) this path is compared with the proposed path of the app (right)

4.1.5 Speed of The App

In the results section it has been used many of the challenging ROI shapes some of them are complex in their shapes and having some holes, and other obstacle types inside the ROI. The ROIs used in the result section having sizes between 5 to 33 hectares. Whatever their shapes and sizes and what ever number of holes, and other obstacle types included inside them their over-all processing speed to complete the computation did not take more than 20 sec. It has been conducted another independent experiment on this proposed app to see how wider area and how many holes, and other obstacle types can be the max for the app. In this independent experiment it has been used a random complex shape of over all area about 300,000 hectare of a ROI having about 7 different shape and size holes inside it, the swath width of the machine was assumed to be 3m, and the output would be the coverage path with omega turn type, the ROI and holes have been directly delineated on the app's google satellite base-map as background. In this experiment the app took a bit more than 2minutes to finish all the computation process.

4.2 Discussion

If all of the required inputs are entered, the app can generate results for any (see fig. 24 and 25) shape/size of ROI with many holes and other obstacle types. As shown in fig 26 and 34, the CPP results of the app for ROIs with holes and other obstacle types and curved boundaries are presented. However, because curved sides require many points to be represented with, the algorithm treats all of these points as vertices. As a result, if those points are NCAP-type points, each of them will receive reference lines to form a stripe (sub-shape) see fig. 26. In this case, the number of sub-shapes produced equals the number of NCAP points. A function has been defined to deal with this problem by ignoring the small stripes caused by this case and re-designing the coverage using the general figure. However, this function occasionally resulted in some gaped stripes (see fig. 34). The two options for resolving this issue are as follows. 1) keep the results of CPP for the ROIs with curved sides and let the users return to the field and cover pathless gaps. 2) Regenerate the ROI with as many straight sides as possible to reduce the number of gaps in the results as shown in fig. 34. It should also be noted that the difficult/impossible for the farmer is to detect the optimal direction which is already been defined. Therefore this is a minor issue.

The algorithm was tested for different shapes and size. (I) The 'C' and 'L' shapes were the first to be introduced in this regard. These shapes are depicted in (fig. 31). The proposed algorithm optimally designs the paths for these shapes. II) The other more difficult shape to evaluate was the larger 'S' shape with an area of 199891m^2 (approx. 200 daa) and a hole of 1128 m^2 (fig. 32). The optimized striped sub-shapes illustrated in (fig. 30). Each of the sub-shapes seen in the figure is optimal to produce long paths within each strip. (III) Figure 33 depicts the two difficult shapes 'T' and 'H.' The path patterns in both cases clearly show that they are all aligned to a direction that is convenient for designing paths with the longest possible length in the given ROI. The 'H' shape has been subdivided into logical sub-shapes (left and right bays and the middle part), and the coverage paths have been aligned to a direction in which they can have their optimal length. The 'T' shape has two cases. i) when the tail part is longer (see fig. 33) in the proposed algorithm this ROI has been sub-shaped into three pars (fig. 32) the tail extended until it divides the top part. Dividing the top part increases the turning distance in the top part into double, which is a negative impact. This is not

happened when the tail part is shorter than the top part (see fig. 34) here the ROI is only divided into two part the top part is not divided here. Therefore according to this work if the tail part of the 'T' shape is longer than the top part (fig. 33), then it will be preferable to consider each of the tail and the top part of the 'T' as separate ROIs.

Seeing the paths form both methods in (fig. 35), one can see some alignment differences. According to the sum of total path distance (TPD) result shown in the figure, the TPD from the proposed app is less in amount than TPD from the reference (well-known) method. Which says the coverage path from the proposed method is more optimized than the coverage path from the reference(well-known) result.

According to this result from the independent experiment mentioned in the result's sub section of 'Speed of the App', this app has a speed exceeding all of the CPP based algorithms that the author has encountered in the literature so far. This might be because of the new approach used in this effort, and taking advantages of combining all the best methods and works in GIS, Agriculture and Python libraries in every steps of the coding process.

Designing the path according to the known turning systems (Ω-turn, U-turn, or Straight AB line) can be useful to the user as it is shown in (fig.).

In (fig. 38) a case study about a farmland in Ås. The owner has tilled this piece of land selecting the best direction of his judgment. It can be a good case to demonstrate how helpful this app can be for a small scale farmers as this. The difference the TPD may not be that much impressive in this case but the usefulness of the app increases as the size and complexity of the ROI is getting higher. Using this app the farmer should not wary about the direction, especially to focused about the driving direction continuously will be very tiresome, which signifies such an automatic guidance to the farmer.

4.3 The Needed Future Works

The surplus works and their subsequent results from science and technology these days gives the ease to combine them to create solutions for problems arising in our daily life. In this work this basic idea have been attempted to find alternative solution in CPP. The works in GIS, Agriculture and Python libraries has been used as sources to find some contents from each one of them to be combined as an app which can solve planning problems in coverage path planning (CPP). This has to be pushed more until the required solution in this area be mate.

This proposed app can do only basic solution, because the main aim of this work is not to make full-fledged type of it but to start the pavement. During the application this app has some issues to be improved for example (1) This app can produce only straight paths but frequently there is a demand to include the possibilities to use the curved paths as well. ii) This app does not consider landscapes of the ROIs but machines has to be driven strategically in ROIs of difficult landscapes this can be an interesting feature to be added to this app. iii) In the application of this app to generating the optimized sub-shapes the ROIs/holes and other obstacle types with the curved sides has not been

giving a satisfying result as it is shown in the result and discussion section, it shows that there should be a more inclusive method in splitting/path generating for the ROI by the current algorithm or just an inclusion of such features which can support the proposed app can be improved. iv) Simplifying the usability of the app for instance drawing polygons or lines can also be another interesting input for the betterment of the app. v) The headland turning area depends on many factors (size of machinery, turning facilities, turning type, size/shape of ROI) a feature enabling to design optimized headland area based on knowledge from literature would save much area can also be added. Many more issues can be included under this tittle but these are the main directions to continue to add a block for the app.

CHAPTER 5. Conclusion

Based on the experiments and their results in the result's section, we could see that the expected outputs has been found for every input and in the comparisons the proposed app has done relatively better than its contestants. All of these affirms that this proposed app is working as it should be. And this concludes that this work was successfully completed based on its main objective.

The exceptional conditions of outputs from ROIs, holes, and other obstacle types having curve sides (that is producing pathless gaps in the output of the planned coverage path) and the other condition in the application of the app for long tailed 'T' shape ROIs. Some suggestions has been forwarded.

In the case of the long tailed 'T' shape ROIs, as it is proposed in the discussion section, the problem could be temporarily countered by treating the tail and the top part of 'T' shape ROI as separate ROIs. While the case of finding solution for the problem of coverage path output of the pathless gaps and other needed works for the fulfillment of the app which are well mentioned in the future work section, would expected to be fulfilled step by step in the coming future by other researchers interested in this line. The success of this work implies that the approach of integrating ideas and methodologies specially from GIS, Geomatics, Python libraries, and Agriculture is promising to be referred in problem solving in the CPP related or other problem solving attempts.

On the other hand this app have convenient environment also to be used in GIS and geomatics related works by including any tools needed.

References

[1] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of field robotics*, vol. 26, no. 8, pp. 651–668, 2009.

[2] J. Bruinsma, *World agriculture: towards 2015/2030: an FAO perspective*. Routledge, 2017.

[3] S. Kalaivanan and R. Kalpana, "Coverage path planning for an autonomous robot specific to agricultural operations," in *2017 International Conference on Intelligent Computing and Control (I2C2)*, 2017, pp. 1–5.

[4] E. Rodias, R. Berruto, P. Busato, D. Bochtis, C. G. Sørensen, and K. Zhou, "Energy savings from optimised in-field route planning for agricultural machinery," *Sustainability*, vol. 9, no. 11, p. 1956, 2017.

[5] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[6] J. T. Bonnen, "A century of science in agriculture: Lessons for science policy," in *Policy for agricultural research*, CRC Press, 2019, pp. 105–137.

[7] R. J. MacRae, S. B. Hill, J. Henning, and G. R. Mehuys, "Agricultural science and sustainable agriculture: a review of the existing scientific barriers to sustainable food production and potential solutions," *Biological Agriculture & Horticulture*, vol. 6, no. 3, pp. 173–219, 1989.

[8] D. T. Pham, "Introduction to AI Robotics, by RR Murphy, Bradford Book, MIT Press, Cambridge, MA, 466 pp., 2001, ISBN 0-262-13383-0, 2001 (Hardback,\pounds 34.50)," *Robotica*, vol. 20, no. 5, pp. 569–569, 2002.

[9] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of mathematics and artificial intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[10] J. Z. Barbosa, S. A. Prior, G. Q. Pedreira, A. C. V. Motta, G. C. Poggere, and G. D. Goularte, "Global trends in apps for agriculture," *Multi-Science Journal*, vol. 3, no. 1, pp. 16–20, 2020.

[11] S. de Bruin, P. Lerink, I. J. La Riviere, and B. Vanmeulebrouk, "Systematic planning and cultivation of agricultural fields using a geo-spatial arable field optimization service: Opportunities and obstacles," *Biosystems Engineering*, vol. 120, pp. 15–24, 2014.

[12] G. Mier, J. Valente, and S. de Bruin, "Fields2Cover: An open-source coverage path planning library for unmanned agricultural vehicles," *arXiv preprint arXiv:2210.07838*, 2022.

[13] M. Spekken and S. Bruin, "Optimizing routes on agricultural fields minimizing manoeuvring and servicing time," *Precision Agriculture. Prague, Czech centre for Science and Society*, pp. 411–426, 2011.

[14] S. Pongnumkul, P. Chaovalit, and N. Surasvadi, "Applications of smartphone-based sensors in agriculture: a systematic review of research," *Journal of Sensors*, vol. 2015, 2015.

[15] C. Costopoulou, M. Ntaliani, and S. Karetsos, "Studying mobile apps for agriculture," *IOSR J. Mob. Comput. Appl*, vol. 3, no. 6, pp. 44–49, 2016.

[16] S. E. Eichler Inwood and V. H. Dale, "State of apps targeting management for sustainability of agricultural landscapes. A review," *Agronomy for sustainable development*, vol. 39, no. 1, pp. 1–15, 2019.

[17] M. S. Hossain, M. Mahmud, M. M. Rahman, S. A. Simul, and M. M. Billah, "Analysis of farmers' digital applications (apps) for availing agriculture-related information services," *International Journal of Civil Service Reform and Practice*, vol. 4, no. 2, 2019.

[18] B. Chazelle, "Convex partitions of polyhedra: a lower bound and worst-case optimal algorithm," *SIAM Journal on Computing*, vol. 13, no. 3, pp. 488–507, 1984.

[19] J. C. Latombe, "ìRobot Motion Planning, î Kluwer," *Boston, MA*, 1991.

[20] H. Choset, “Coverage of known spaces: The boustrophedon cellular decomposition,” *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.
[21] E. Galceran and M. Carreras, “A survey on coverage path planning for robotics,” *Robotics and Autonomous systems*, vol. 61, no. 12, pp. 1258–1276, 2013.
[22] H. Choset and P. Pignon, “Cover path planning: The boustrophedron decomposition,” in *Proceedings International Conference on Field and Service Robotics, Canberra, Australia*, 1997.
[23] H. Choset, E. Acar, A. A. Rizzi, and J. Luntz, “Exact cellular decompositions in terms of critical points of morse functions,” in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, 2000, vol. 3, pp. 2270–2277.
[24] P. Rigaux, M. Scholl, and A. Voisard, *Spatial databases: with application to GIS*. Morgan Kaufmann, 2002.
[25] N. Tryfona, D. Pfoser, and T. Hadzilacos, “Modeling behavior of geographic objects: An experience with the object modeling technique,” in *International Conference on Advanced Information Systems Engineering*, 1997, pp. 347–359.
[26] T. Dray and C. A. Manogue, “Journal of Online Mathematics and Its Applications The Geometry of the Dot and Cross Products,” 2006.
[27] E. Lansey, “The dot and cross products.” 2009.
[28] C. C. S. Initiative, “Common core state standards for mathematics,” *Retrieved from Common Core Standards Initiative website: http://corestandards. org/assets/CCSSI_Math% 20Standards. pdf*, 2010.
[29] N.-D. Lane, “Geometric Transformations, by IM Yaglom. Translation by Allen Shields. Random House, 1962. 133 pages. $1.95.,” *Canadian Mathematical Bulletin*, vol. 7, no. 3, pp. 479–480, 1964.
[30] G. Venema, *Foundations of geometry*. Pearson Higher Ed, 2011.
[31] E. W. Weisstein, “Coordinate System.” https://mathworld.wolfram.com/ (accessed Nov. 01, 2022).
[32] T. H. Meyer, *Introduction to geometrical and physical geodesy: Foundations of geomatics*. Esri Press, 2018.
[33] P. R. Wolf, B. A. Dewitt, and B. E. Wilkinson, *Elements of Photogrammetry with Applications in GIS*. McGraw-Hill Education, 2014.
[34] “KOORDINATBASERTE REFERANSESYSTEMER - Kartverket - Google ᖴᐱᒧ.” https://www.google.com/search?q=KOORDINATBASERTE+REFERANSESYSTEMER+-+Kartverket&sxsrf=ALiCzsaDmrXcPWXk_XOKhDoWQ3SfxNARSg%3A1667128026970&source=hp&ei=2lpeY_qjOfC-xc8P79a_cA&iflsig=AJiK0e8AAAAAY15o6kEPlbyG4vAmSlyLjHkzXVPzGKIe&ved=0ahUKEwj6-Nzu54f7AhVwX_EDHW_rDw4Q4dUDCAk&uact=5&oq=KOORDINATBASERTE+REFERANSESYSTEMER+-+Kartverket&gs_lcp=Cgdnd3Mtd2l6EAMyBQghEKABUABYAGCyC2gAcAB4AIAB2QaIAdkGkgEDNi0xmAEAoAECoAEB&sclient=gws-wiz (accessed Oct. 30, 2022).
[35] Y. N. Moschovakis, “What is an algorithm?,” in *Mathematics unlimited—2001 and beyond*, Springer, 2001, pp. 919–936.
[36] E. M. Arkin and R. Hassin, “Approximation algorithms for the geometric covering salesman problem,” *Discrete Applied Mathematics*, vol. 55, no. 3, pp. 197–218, 1994.
[37] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, “Approximation algorithms for lawn mowing and milling,” *Computational Geometry*, vol. 17, no. 1–2, pp. 25–50, 2000.
[38] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[39] B. Donald, K. Lynch, and D. Rus, *Algorithmic and Computational Robotics: New Directions 2000 WAFR*. CRC Press, 2001.
[40] T. C. Shermer, “Recent results in art galleries (geometry),” *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1384–1399, 1992.
[41] F. Li and R. Klette, “An approximate algorithm for solving the watchman route problem,” in *International Workshop on Robot Vision*, 2008, pp. 189–206.
[42] H. Choset *et al.*, “Principles of Robot Motion: Theory, Algorithms, and Implementation ERRATA!!!!,” 2007.

Norwegian University
of Life Sciences