Development of Topological Mappings for Autonomous Agricultural Vehicles

Moein Mehrolhassani

Submitted to the
Institute of Graduate Studies and Research
in partial fulfilment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Eastern Mediterranean University
September 2016
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Mustafa Tümer
Acting Director

I certify that this thesis satisfies the requirements as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Prof. Dr. H. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Doctor of Philosophy in Computer Engineering.

Asst. Prof. Dr. Mehmet Bodur
Supervisor

Examining Committee

1. Prof. Dr. Atilla Elçi
2. Prof. Dr. Kemal Leblebicioğlu
3. Asst. Prof. Dr. Adnan Acan
4. Asst. Prof. Dr. Mehmet Bodur
5. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

Automation system of agricultural crop plantation requires many subsystems such as low level tracking, path planning, obstacle detection, manoeuvres at the path terminations, etc. This study proposes semantic annotation for the information flow between the automation subsystems, filling the gap between the planning and implementation of crop production by developing two missing subunits: determination of obstacles that may threaten agricultural vehicles using the satellite images of target field, and determination of proper path for the agricultural vehicles to process rows of crops. For the attributes of obstacles, semantic annotation on the map of target field is preferred using Resource Description Framework/Extensible Mark-up Language (RDF/XML) in order to be exchangeable and reusable with other stages, systems, devices and applications. Developed Matlab code determines the target field by a GPS coordinate inside the field. An interactive initialization stage provides download of the satellite images from Google Maps API for determination of the field boundaries. The code for detection and positioning of the circular shaped obstacles are using Prewitt, Sobel, Roberts, and Canny edge detection, and Hough transformation algorithms. The developed method is tested on 51 target fields. It provides 45% improvement in detection error rate compared to raw application of the algorithms.

Keywords: Image processing, Obstacle detection, Path planning, Semantic annotation, RDF/XML mapping.

ÖZ

Tarımsal tahıl üretimi otomasyon sistemlerinde alt düzey iz takibi, yol planlaması, engel tayini, yol sonunda manevra gibi birçok alt sistem gerekir. Bu çalışma, tarımsal otomasyonu gerçekleştirmede gereken tarım aracına tehdit olabilecek engelleri uydu görüntüsünden tanıma ve araçların ürün sıralarını işlemesine uygun yol planlama alt sistemleri arasındaki bilgi akışının semantik işaretleme yoluyla çözülmesi önerilmektedir. Engellerin özellikleri hedef tarlanın haritasına RDF/XML kullanarak semantik işaretleme yöntemiyle kaydedilmekte, böylece birimler arasında verimli bilgi akışı sağlanmaktadır. Bilginin başka sistem, araç, ve uygulamalar için dönüştürülebilir ve tekrar kullanılabilirliği için semantik işaretlemede RDF/XML tercih edilmiştir. Geliştirilen Matlab kodu tarlayı içindeki her hangi bir GPS koordinatından belirlemektedir. Google Maps API kullanarak indirilen uydu görüntüsünde tarla sınırları etkileşimli giriş aşamasıyla belirlenmektedir. Çembersi biçimleri bulmak ve yerini belirlemek üzere Prewitt, Sobel, Roberts, ve Canny kenar belirleme ile Hough dönüşümü kullanılmaktadır. Geliştirilen yöntem 51 hedef tarla üzerinde sınanmıştır. Geliştirilen yöntem, kenar bulma ve Hough dönüşümlerinin ham kullanımına göre engel bulma hatasını %45 düşürmüştür.

Anahtar Sözcükler: Görüntü işleme, Engel bulma, Yol planlama, Semantik işaretleme, RDF/XML haritalama.

ACKNOWLEDGMENT

Foremost, I would like to express my sincere gratitude to my supervisor Asst. Prof. Dr. Mehmet Bodur for supporting me during this study. His patience, motivation, passion, and immense knowledge always encouraged me to always keep going. His supervision helped me in all the time of study and writing of this thesis.

I would like to thank my thesis committee members Asst. Prof. Dr. Adnan Acan, and, Asst. Prof. Dr. Ahmet Ünveren for their insightful comments and encouragement, but also for the hard question, which motivated me to widen my research from various perspectives.

My sincere thanks also goes to Prof. Dr. Atilla Elçi who I had the honour to be his student during my master degree. I would also like to thank Prof. Dr. Kemal Leblebicioğlu for his valuable comments in my PhD thesis defence.

Last but not the least important, I owe more than thanks to my family members including my parents, brother, sister and also my wife for their financial and emotional supports and encouragements through my study.

TABLE OF CONTENTS

ABSTRACT iii

ÖZ iv

ACKNOWLEDGMENT v

LIST OF FIGURES ix

LIST OF TABLES xi

LIST OF ABBREVIATIONS xii

1 INTRODUCTION 1

1.1 Automation of Crop Plantation in Agricultural Industry 1

1.2 Agricultural Path Planning and Image Processing 2

1.3 Industrial Automation of Crop Plantation Systems 6

1.4 Path Planning Using Satellite Images 8

1.4 Access to Google Satellite Images 9

1.5 Coordinate Systems of Google Satellite Images 10

1.6 Image Processing Tools for Detection and Locating Obstacles..... 12

1.6.1 Image Filtering and Processing 13

1.7 Annotation of Information into Map Images 14

1.8 Focus of the Thesis, and the Problem Definition 14

1.9 Contents of Further Chapters 17

2 DESCRIPTION OF PROPOSED SYSTEM 18

2.1 System Architecture 18

2.2 Initialization 19

2.3 Segmentation and Extraction of Target Field 19

2.4 Detection of Obstacles on the Field 21

2.4.1 Detection Algorithm 21
2.4.2 Calculation of Positions 23
2.4.3 Semantic Annotation 28
2.4.4 Improved Detection Algorithm 30
2.4.4.1 T-range 31
2.4.4.2 Max-of-All 32
2.4.4.3 Double Layered Check (DLC) 32
2.4.5 Accuracy and Error Evaluation 33
2.5 Trajectory Points 34
2.5.1 Generation of Trajectories 34
2.5.2 Annotation of Trajectory Points 38
2.5.3 Semantic Annotation 40
3 RESULTS 42
3.1 Extraction and Detection 42
3.1.1 Canny 43
3.1.2 Prewitt 44
3.1.3 Roberts 44
3.1.4 Sobel 45
3.1.5 Detection Improvement Results 46
3.1.5.1 T-Range Results 46
3.1.5.2 Max of All Results 47
3.1.5.3 DLC Results 49
3.1.6 FPE and FNE Reduction 50
3.1.7 Locating Accuracy Check 52
3.2 Trajectory Points Generation 54

3.3 Semantic Annotation 58

4 DISCUSSION 63

4.1 Effect of Zoom Level and Noise Reduction Algorithm 63

4.2 Effect of Edge Detection Algorithms 65

4.3 Discussion on Path Planning Methods and Semantic Annotation 71

5 CONCLUSION 74

REFERENCES 77

APPENDICES 86

Appendix A: Segmentation and Extraction Source Code 87

Appendix B: Obstacle Detection Source Code 91

Appendix C: Duplicity Check Source Code 93

Appendix D: Code for Geographical Positioning of Obstacles 94

Appendix E: Code for Semantic Annotation of Obstacles 95

Appendix F: Code for DLC Algorithm 98

Appendix G: Code for Detection and Verification using Grey-Level Intensity Threshold 101

Appendix H: Obstacle and Error Counts by Default Threshold 104

Appendix I: Obstacle and Error Counts for DLC 106

LIST OF FIGURES

Figure 1: Sample of Google-Maps API to access a satellite image 10

Figure 2: Google's method of tiling the Mercator projection of the Earth 12

Figure 3: Inverted outputs of edge detection methods 14

Figure 4: Four stages of the developed system 18

Figure 5: Progress of image processes to extract the field and obstacles. 21

Figure 6: Images while morphological reconstruction to filter noise, ... 22

Figure 7. GPS coordinates of a location. 23

Figure 8. Trimming image size for the target field 28

Figure 9: RDF/XML data graph of detected tree 29

Figure 10: Semantic annotation structure of obstacles in RDF/XML file 30

Figure 11: Detection results using threshold range on Sobel. 31

Figure 12: Production reference and trajectory points, a- overview, b- details 37

Figure 13: An obstacle on a track-line appears in the form of a gap 40

Figure 14: RDF/XML data graph representing a trajectory point 40

Figure 15: Semantic annotation structure of the trajectory points 41

Figure 16: Examples of cropped images with varius size, shape and complexity..... 43

Figure 17: Canny edge and obstacle detection results 43

Figure 18: Canny edge and obstacle detection results 44

Figure 19: Robert edge and obstacle detection results... 45

Figure 20: Samples results of Sobel's edge and obstacle detection 46

Figure 21: T-Range outcomes on Sobel comparing to default threshold 47

Figure 22: Max-of-All method comparing to Sobel with default parameter sets 48

Figure 23: Comparison between T-Range and Max-of-All 48

Figure 24: DLC detection results comparing to Sobel with default paameter-sets ... 49

Figure 25: Classes of correct and incorrect detections 50

Figure 26: FNE improvements using DLC 51

Figure 27: Suppressed FNE using DLC.. 52

Figure 28: Developed system’s obstacle positioning accuracy 53

Figure 29: Detected obstacle dimensions calculated by the developed method compared with google and its actual size (from left to right)................................... 54

Figure 30: An example of parallel path planning in an agricultural field................. 55

Figure 31: Trajectory reference points plotted on the field`s image......................... 55

Figure 32: Demonstrating the longest straight edge of the field............................... 56

Figure 33: Graphical and semantic annotation of obstacles, ... 59

Figure 34:Section of the annotation file in RDF/XML syntax format....................... 62

Figure 35: Correlation between zoom level on the map and the image quality 64

Figure 36: No. of detections using the entire range threshold on Sobel in all fields. 66

Figure 37: Maximum detection of obstacle-candidates for edge detection methods using dynamic threshold 69

Figure 38: Proof of produced trajectory points being in a straight line in pixels 72

Figure 39: Proof of produced trajectory points being in a straight line in GPS coordinates 72

Figure 40: Sample output of a path planning algorithm 73

LIST OF TABLES

Table 1: Trajectory points of each track in pixel and GPS pairs 39

Table 2: Statistics of the trajectory path presented in Figure 28............................... 56

Table 3: Track and segment details of the example in Figure 28 57

Table 4: Computation time and details for various row distances............................. 58

Table 5: Obstacle detection and positioning data included in semantic annotation .. 60

Table 6: Array of trajectory points in pairs of *x*, *y* and *lat, long*............................... 61

Table 7: Initial comparison of edge detection methods 65

Table 8: Detection improvements from Sobel with static threshold value to Sobel with dynamic threshold... 67

Table 9: Correct detections by Sobel with default-threshold and Max-of-All methods 68

Table 10: DLC vs. Sobel with default threshold in detection................................... 70

LIST OF ABBREVIATIONS

ACC	Adaptive Combination of Classifiers
API	Application Program Interface
BBP	Bounding-box’s start point of crop
CCP	Corrected center point
DLC	Double Layered Check
FNE	False negative error, detected obstacle actually does not exist
FPE	False positive error the missed obstacle in the detection
GIS	Geographic information system
GPS	Global Positioning System,
OCP	Original center pixel
SEGON	Dual multiScalE Graylevel mOrphological open and close recoNstructions
SHT	Standard Hough Transform
XML	Extensible Mark-up Language
A_i	ith anchor point
$c_{r/d}$	Constant, $\pi/180$
d(p_1,p_2)	distance from point-1 to point-2
d_{ms}	Crop row spacing in meters; distance between crop tracks in meters.
d_{ps}	Crop row spacing in pixels; distance between crop tracks in pixels.
ED_m	Line segment of the longest edge of the target field. $ED_m = \{p_{EDm,1}=(x_{EDm,1},\ y_{EDm,1}), p_{EDm,2}= (x_{EDm,2}, y_{EDm,2})\}$
G	Gap on reference trajectory representing the area occupied by an obstacle. $G = \{(x_{G,1},y_{G,1}), (x_{G,2},y_{G,2})\}$
L_E	Circumference of Earth along Equator line.

lat_C	The angular latitude in degrees of centre point C
lng_C	The angular longitude in degrees of centre point C
m_{EDm}	Slope of longest edge of the target field.
m_{EDmp}	Slope of the perpendicular line to ED_m.
$N_{G,I}$	Number of gaps on i^{th} track-line
N_R	Number of track-lines, number of anchor points that generates valid reference trajectories.
$N_{S,I}$	Number of reference trajectory segments on i^{th} track-line.
N_S	Total number of reference trajectory segments.
N_x	Width of image in x-direction.
N_z	Number of pixels in x-direction covering circumference of the Earth, N_z=2^{Z+8}.
Ni	pixel colour value (2.3)
p_{OCP}	original centre pixel coordinate $p_{OCP} = (x_{OCP}, y_{OCP})$
p_{BBP}	bounding-box start pixel coordinate $p_{BBP} = (x_{BBP}, y_{BBP})$
p_{CCP}	correction offset in pixels $p_{CCP} = (x_{CCP}, y_{CCP})$
$p_{T,i}$	pixel coordinates of objects $p_{T,i} = (x_{T,i}, y_{T,i})$
p_i	pixel coordinates of point-I $p_i = (x_i, y_i)$.
p_{EDmp}	midpoint of line segment $ED_m, p_{EDmp=} (x_{EDmp}, y_{EDmp})$.
pixel(x,y)	coordinate of location in pixels
Ri	segmented regions (2.3)
$\mathcal{R}\alpha_{min}$	minimum obstacle radius
$\mathcal{R}\alpha_{max}$	maximum obstacle radius.
tile(x,y)	coordinate of tile numbers in tile matrix of a zoom level.
r_E	average radius of Earth sea level at Equator line
$(x_{R,i,k}, y_{R,i,k})$	pixel coordinate of reference trajectory points.

Z	Google-Maps service zoom level.
Δ_C	Earth resolution at centre point C of tile; distance on Earth per pixel on a tile of map.
Δ_E	length on Equator in meters per pixel of Google-Map image along x-direction.
Δ_G	GPS resolution of image; angular degrees of latitude per pixel.
θ_e	constant, 360^o, angular distance of one revolution around Earth.

Chapter 1

INTRODUCTION

1.1 Automation of Crop Plantation in Agricultural Industry

Developments of modern agricultural machinery and genetically improved seed technology opened a new perspective in agricultural production that allows very high efficiency per labour through the mechanization of plantation and harvest of agricultural crop production, which is a part of agricultural industrialization. Parallel to the mechanization of the agricultural processes such as ploughing, seeding, and harvesting the crops, in the recent decades, the rapid developments of the artificial intelligence and robotics provided new kind of automation tools that combines the intelligence of image and sensory signal processing to the mechanized production tools of the agricultural industry to satisfy the demands of market for high volume production at better quality than the traditional manual agriculture. V. Blanco reported that increasing volume and quality together with the production efficiency depends largely on increased accuracy and efficiency of ploughing, seeding, harvesting, fertilization methods [1]. In most of the agricultural production systems, including the traditional plantation methods, planting crops in regular rows at homogenous crop density is critical point to increase the production efficiency. Efficient application of various kinds of agricultural vehicles in agricultural processes such as for ploughing, seeding, fertilizing and harvesting requires tracking regular lines of plantation in the fields.

Many methods for the automation of these agricultural vehicles have been proposed by researchers based on the trajectory tracking techniques, which is a common concept of robotics and automation. The term "control of autonomous agricultural vehicles" describes the systems developed to keep a tractor on a pre-planned desired trajectory with minimum lateral error along the trajectory. The lateral error corresponds to perpendicular distance to the desired trajectory. Lateral error is one of the key optimization variables in control of autonomous agricultural vehicles, and it has been studied by many researches. J. Gomez-Gil proposed inexpensive GPS measurement system together with a Kalman-filter which provides less than 6 mm lateral error [2]. Even at low speeds, the lateral control of the tractor is relatively hard problem compared to the moderate speed highway vehicle because considerable side slip of the tires makes the system unstable, almost comparable to driving on loose or slippery ground. Adaptive and/or predictive control laws that adapts on the parameters which produce the side slip provide sufficiently low lateral error in tracking the linear trajectories [3]. But, adaptive nature of control develops unexpectedly high lateral errors especially at the curvatures of the trajectories. Double Look-Ahead Reference Point (LARP) approach reduces the maximum lateral errors at the curvatures down to 3 or 4 mm [4]. Further automation of the process requires planning of the optimum paths which may be converted to a convenient trajectory during the operation of the agricultural vehicle for most common agricultural processes in the fields.

1.2 Agricultural Path Planning and Image Processing

Path planning for the agricultural machinery has attracted considerable attendance of the researchers. It has its own characteristic constraints that is not exhibited by ordinary off-road path planning algorithms [5]. Timo Oksanen and his co-author proposed top down and bottom up algorithms to split a field systematically into subfields and merge these trapezoidal areas to larger blocks to search optimum or valid routes in straight lines [6]. Zhang proposed a path search algorithm with dynamic model of a tractor to guide the manoeuvre of the tractors at the end of the fields using real time kinematic global positioning system with less than 3 cm positional error [7]. Ant colony optimization methods have been introduced for optimal route planning to accomplish optimization criteria such as B-patterns to minimize the operational time, non-working travelled distance, and fuel consumption at 2011 [8].

Some agricultural forestry lands may contain obstacles on the already specified desired paths, which shall be detected and reconsidered for modification of the desired trajectory. For the unplanned obstacles on the trajectories, probabilistically robust path planning algorithms have been proposed using rapidly exploring random trees which provides efficient identification and execution of paths in real-time [9]. Similar to the agricultural cases, a path-planning algorithm is proposed to generate dynamic paths to cover an area completely, using a neural network to solve the obstacle avoidance problem of a cleaning robot [10]. Any problem related to multiple tractors in the same field may be solved using the algorithms developed for multiple semi-autonomous vehicles in the traffic, where, the optimization of real time path planning was accomplished by a 4-layer path-planning algorithm to ensure the paths without collision [11].

Once the geographical map of the agricultural land is known, various search techniques may be employed to increase the performance of agricultural machines by searching an optimal path of operation. A path planning algorithm has been proposed to minimize a cost function minimization for the complete coverage of a farm field by a trajectory using a greedy search and a heuristic algorithm that determine the minimum of a cost function for the best Hamiltonian solution in the graph [12]. Another study proposed a depth-first search algorithm by dividing the field to several sub regions and covering all sub regions along their length with the trajectories to reduce the number of turns [13]. An algorithm to determine the turning areas has been proposed for complete coverage of a field minimizing the overlap of processed paths [14]. An approach has been proposed to solve path planning problem in a field considering the kinematic constraints of the agricultural machines, and their short and long term dynamic accuracy, which raises problems especially when there are various kinds of obstacles in the fields [1].

Locating the position and orientation of an agricultural machine in the land exerts an important problem in the automation and control of the machinery. There are local methods based on optical sensors with cameras, and infrared or laser beams, which has important drawbacks due to extreme dusty operating conditions of the agricultural machines. Methods using GPS to ensure the lateral stability of a vehicle has been proposed with considerable low lateral measurement errors down to 6 cm in [15], [16]. An algorithm that use Kalman-filtered laser scanner measurements has been developed to detect obstacles on a field to accomplish real time autonomous operation of a tractor, reducing the error of estimated position down to 2.7 cm [17].

Combining several sources of information to obtain a more detailed and accurate environment model is another important topic in agricultural industrialization as well as in many other fields such as land exploratory robots. In a study, a hybrid method has been developed to merge two or more semantic cartographic maps of obstacles in a single annotated map by generating and processing cartographic information layers. This method is promising to develop more detailed semantic map and objects of agricultural fields starting from satellite images [18].

At the image processing and object recognition side, the problem is considered at much larger perspectives rather than processing the image for only agricultural purposes. The global features of the images, which is called "gist of scene", has been included to object detection algorithm to reduce the ambiguity in local features of the image to improve the speed of local detector systems that can be applied to obstacle detection systems [19]. The recognition performance for objects were enhanced by using discriminative image patches instead of the complete image of the object by image patch histograms classification and modelling [20].

An important development in the semantic annotation field was development of dual multi-scale grey level morphological reconstructions, shortly SEGON, which segments the objects in an image to improve image retrieval performance [21]. SEGON can classify patches of large images in semantic concepts by learning to distinguish objects based on the latent Dirichlet allocation model [22]. Filtering a noisy image is necessary in many cases to prepare an image for object recognition tasks. A median filtering method was developed for noisy compound images of text and objects to reduce salt and pepper noise before further processing. Finally, a framework for detecting objects in static images by trained log-linear models of image patches was proposed to recognize given set of objects, using an Adaptive Combination of Classifiers (ACC) [23]

1.3 Industrial Automation of Crop Plantation Systems

An ideal automated agricultural plantation requires a complex system architecture made of several levels of subsystems, such as collection of data about the target piece of land, recognition and planning of agricultural areas, detection of possible obstacles in and at the border of the land, determination of optimum paths and trajectories for the target agricultural applications on plantation area, scheduling, determination of tractor location and direction, and tracking the trajectories along the pre-determined paths, manoeuvre at the path terminations, keeping logbook for records of successful and problematic tasks during the process, etc. Implementation of each of these subsystems requires an application layer that needs considerable information exchange to accomplish a successful and efficient operation of the overall system.

Many subsystems of this ideal architecture have been covered in literature as explained shortly in the previous subsections. Modern differential GPS based location and direction system employs ordinary GPS and a Kalman-filter to determine tractor location with less than 6 cm tolerance while the tractor goes on a linear trajectory at a speed of 5 m/s [2]. A typical lateral deviation less than 5 cm is achieved by predictive and adaptive predictive control methods considering the side-slip motion of the tractors on the linear trajectories [3]. The considerably high lateral error of predictive control methods at the curvatures of the trajectory is reduced to less than 0.5 cm by using LARP method [4]. There are path planning systems available in literature that determines the optimum paths once a geographic map is available.

Industrialized crop plantation is always carried by seeding the crop in rows like the traditional agriculture. The rows of crops were a result of a sequence of traditional agricultural phases of ploughing, seeding, and harvesting. For an efficient farming of crops, several types of implants carry out the necessary tasks on appropriate number of rows per pass. Some of these tasks require higher precision than the others. In general, a precision of about 10 cm is necessary for efficient ploughing and harvesting. Improving the accuracy better down to 2 cm reported to provide higher efficiency for seeding, fertilization, pesticide application, thus improves overall efficiency of agricultural production.

Path planning for the tractors is a typical problem of agricultural robotics, where the vehicles shall operate in the field to process rows of plants. Naturally, the automation of industrialized agriculture was first started in cultivation of crops, maize and potatoes at very large lands, in US, Canada, and Spain, where a human operator sets the tractors to track a pre-specified trajectory which and the lateral controller on the tractor manages to track a pre-specified trajectory to complete the assigned task. In the semi-automated application, some parts of the tasks are fully automatic and an operator is required to be on the board continuously to check the performance of automated process. In practice, the application of semi-automatic tractors is economically feasible to only for processing very large lands which are already refined from obstacles, since an unplanned action introduce considerable interruption of operational time. As a result, the technology of industrialized agriculture is at a stage of change to full-automation of the processes, which means, agricultural machinery once prepared and set for operation in a field shall carry the task uninterruptedly for all expected and unexpected cases without depending on an operator during the operative part of the process.

1.4 Path Planning Using Satellite Images

A real-time agricultural obstacle detection and path planning system for fields of crops obviously requires real-time observations. Such real time observations may be obtained by optical methods such as video image analysis of mobile or stationary cameras, radar or laser beam scanning, along with commercial professional real-time geographical satellite images. However, the status of the obstacles in an agricultural field mostly shows very slow changes even in a couple of years, and drastic changes can be easy to detect before starting the automatic process on the field. Therefore, this thesis was conducted using the freely accessible geographical satellite images rather than using images from a commercial real-time satellite image provider. Real time path planning that considers even the short term changes in the environment is a necessity especially when the agricultural automation targets whole system to run without human labour, relying on the high level automation algorithms that searches the optimum path when the obstacle is stationary, and adjusts its speed when the obstacle is a moving object such as another tractor, a human, or an animal.

In this thesis, the source of observations for agricultural path planning system is the satellite image of the field at the geographical map server of Google-Maps. This data base is developed and managed by a division of US Company Google, which is called Google-Maps. It provides satellite images of land along with street maps of cities. Google-Maps provides many services along with the geographic images, such as, car, train, bus, bike or walking route planning, travel scheduling, describing local public transportation systems, and search of local public transportation options in most of the metropolitan areas.

Google has a group of geographic moderators who approves public labels suggested by users on the map as described on “wikihow” web page. Google-Maps service provides a search tool to start a text search for a public label. It is possible to zoom in and out for 23 zoom levels, and pan on the map to up, right, down or left directions. Each geographic map can be precisely accessed by the zoom level and the geographic coordinates of the map centre.

1.4 Access to Google Satellite Images

Google Company manages two commercial satellite image interface systems, Google-Earth, and Google-Maps, by user friendly graphical interface applications. The update rate of the images is 1 to 3 years. Compared to the other free access web mapping services, the satellite images provided by Google are both better updated, and higher in accuracy. These two factors made Google preferable over the other free satellite image services like Yahoo or Terrafly for many application fields. Images of many areas are as old as five years, but at urban areas, they are refreshed almost monthly. Google declares on their web pages that their applications Google-Maps and Google-Earth use the same satellite data bank. The web page developers.google.com explains how to access to google map service to get a static map by an application program interface through the web on an example for a 600x300 pixel satellite image of New York city at zoom level 13 is shown in Figure 1. The image received in return to this code may be mapped to world geographical coordinate system which links the pixels of the images to the GPS coordinates on the earth accurately by the 3rd version of adjusted API.

Two Example HTML Codes to get an image from Google-Maps Satellite Image Database

Centre Point: EMU, CMPE Department, Cyprus, Zoom Level: *Z*=18

`https://www.google.com/maps/@35.1462487,33.9081493,18z/` opens a javascript map which allows you to get satellite, or hybrid (satellite with map overlay) images. Specified GPS coordinate points to EMU, CMPE department building.	

Centre Point: New York City Centre, NY, Zoom Level: *Z*=13

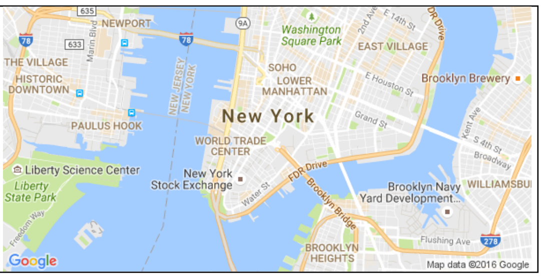
`http://maps.googleapis.com/maps/api/staticmap?center=New+York,NY&zoom=13&size=600x300` opens a controlled web image which allows you to get map or satellite (Earth) images.	 NEWPORT Washington Square Park EAST VILLAGE THE VILLAGE HISTORIC DOWNTOWN SOHO LOWER MANHATTAN Brooklyn Brewery New York PAULUS HOOK WORLD TRADE CENTER Liberty Science Center Liberty State Park New York Stock Exchange FDR Drive Brooklyn Bridge Brooklyn Navy Yard Development WILLIAMSBU BROOKLYN HEIGHTS Flushing Ave Google Map data ©2016 Google

Figure 1: Sample of Google-Maps API to access a satellite image

1.5 Coordinate Systems of Google Satellite Images

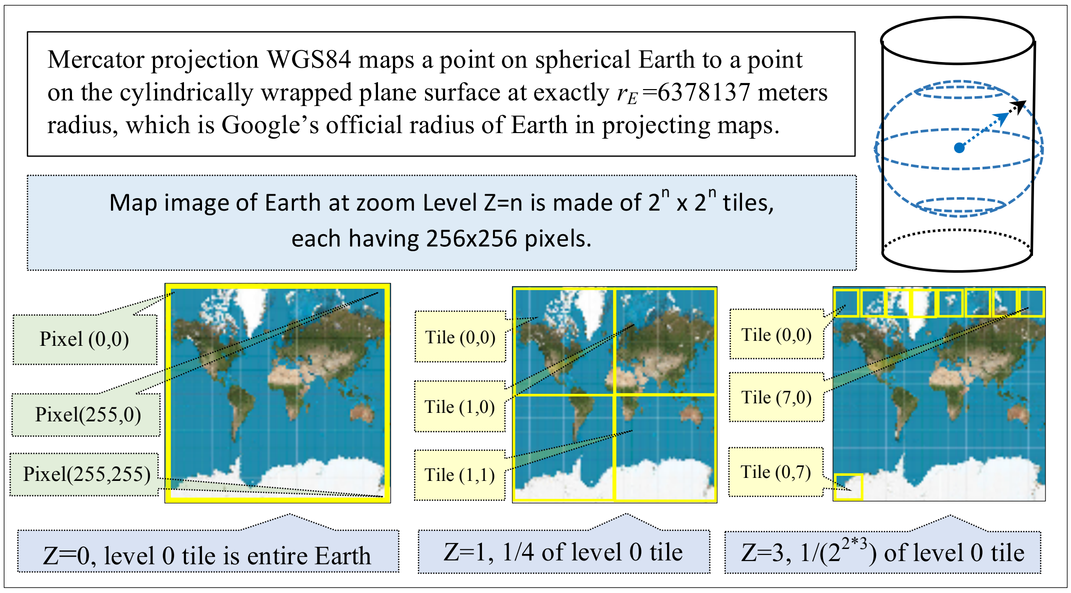
A region on the spherical shaped globe is mapped to Google-Maps satellite images by method of Mercator cylindrical projection standardized at 1984 with the name World Geodetic System 1984, shortly WGS84. Mercator projection is a geodesic projection system to map the points on the earth on a plane, invented by cartographer Gerardus Mercator in 1569. It is basically the surface of sphere projected on a cylinder wrapped to the sphere around the equator. Although it distorts the map more and more while approaching the poles, it is still in common use since most urban land is around eclipses, far away from the poles. And in this region, deformation of shape for small lands such a single country is below detectable limits of human eye. It is also the simplest projection to map the points of GPS coordinates to a planar surface in shortest time with minimum computational effort.

Google map database uses three kinds of coordinate systems because it keeps the details of the map at 23 zoom levels, Z=0, …, 22, in 256 by 256 pixel tiles. Zoom level Z=0 is a single tile that represents Mercator projection of the satellite image of Earth as an 256 by 256 pixel image as demonstrated in Figure 2. Pixels in each tile are addressed by the pixel coordinate, with pixel(0,0), pixel(0,255) and pixel (255,255) indicating the top-left, top-right, and bottom-right corner pixels of the tile respectively. Zoom level Z partitions the first zoom-level tile into 2^Z by 2^Z tiles. Each tile is addressed by a coordinate such as tile(0,0), tile(0, 2^Z– 1), and tile(2^Z –1, 0) are coordinates of the top-left, top-right, and bottom-left corners respectively.

Google-Maps server addresses the points on the Earth using the old Navstar coordinate system, which is now named the Global Positioning System, or shortly GPS coordinate system. It is common coordinate system of standard GPS modules using WGS84 standard coordinate system to address any point on Earth precisely using decimal latitude and longitude angles in degrees. Intersection of Greenwich longitude and Equatorial line is the origin, (0,0) of GPS coordinates. But, WGS84 uses multiple reference datum points to correct errors developed by several reasons such as ellipsoid shape of the Earth.

The main advantage of Mercator projection is in preserving the x- and y-direction scale ratio of distances locally for any location of Earth surface. That means for larger zoom levels, the distance scales of tiles are scaled by cosine of latitude in both x- and y- direction equally. Consequently the deformation in the maps of cities, or small countries is at negligible level, although the Earth distance per pixel of map image changes by cosine of latitude of pixel(0,0) of that tile.

As an example, the GPS coordinates for Eastern Mediterranean University, Computer Engineering Department Parking entrance is (*lat*=35.1462487, *long*=33.9081493) as used in Figure 1, to access the map around CMPE building at zoom level 18.


Mercator projection WGS84 maps a point on spherical Earth to a point on the cylindrically wrapped plane surface at exactly r_E=6378137 meters radius, which is Google's official radius of Earth in projecting maps.
Map image of Earth at zoom Level Z=n is made of 2^n x 2^n tiles, each having 256x256 pixels.
Pixel (0,0)
Pixel(255,0)
Pixel(255,255)
Tile (0,0)
Tile (1,0)
Tile (1,1)
Tile (0,0)
Tile (7,0)
Tile (0,7)
Z=0, level 0 tile is entire Earth
Z=1, 1/4 of level 0 tile
Z=3, 1/(2^{2*3}) of level 0 tile
Figure 2: Google's method of tiling the Mercator projection of the Earth

1.6 Image Processing Tools for Detection and Locating Obstacles.

Detection of obstacles on a satellite image requires a series of image processing methods to be applied on the image to filter noises, and clear out the externals of the target land. Another set of image processing algorithms are necessary to isolate the obstacles to the agricultural vehicles for the determination of the location and size of the obstacles. This thesis targeted to use well defined and standardized image processing methods which are available in Matlab Image Processing Toolbox, mainly to simplify the reproducibility of the results.

1.6.1 Image Filtering and Processing

Although Google-Maps satellite images were pre-processed for best noise and contrast conditions to have best images for human eye, the images are quite noisy for detection of large objects such as a tree on the field. Any small and undesired objects and marks which appear on the image is removed or filtered by median filtering to reduce a false or misdetection. Median filtering is a well-known method which appears in image processing textbooks, where the target pixel value of resulting image is replaced by the median of the values of the pixels in a neighbourhood of the target pixel in the input image. It is mostly applied for a neighbourhood of 3x3 or 5x5 matrix. It removes sharp singular marks, and noisy spots preserving the edges of the large objects [24]. Keeping the edges of the field, and large obstacles are essential to perform obstacle detection and field extraction, therefore this method is preferred to other noise reduction methods in this study.

1.6.2 Edge and Shape Detection Algorithms

Detection of objects in the field is possible by many methods including template matching as well as searching and classifying the objects after edge detection and segmentation techniques. In general, template matching is a method with computation complexity higher than order-2 with respect to the number of pixels in the image, while most of the edge detection algorithms has computational complexity order-1. This fact is one of the significant reasons to prefer edge detection rather than template matching in determination of the attributes and location of the obstacles in this study. As seen in Figure 3, the edge detection methods considered in this study are Roberts, Prewitt, Sobel and Canny, all available in the Image Processing Toolbox of Matlab [25] [26] [27] .

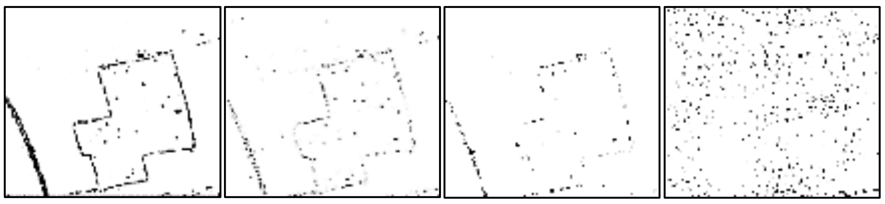

(a) (b) (c) (d)

Figure 3: Inverted outputs of edge detection methods

Once the edges of a region are determined, detection of the shape of that region is possible by Hough transformation. Originally, the Hough transformation is developed and patented by Paul Hough to detect the continuity of two line segments in 1962. Later, the method is adapted for circular and elliptic shapes [28]. Today, the generalized version of this method is available as a standard image processing tool in Textbooks, and in the Image Processing Toolbox of Matlab.

1.7 Annotation of Information into Map Images

The image processing methods provide critical information for the location and size of the obstacles in the field, which is critical information of path planning algorithm to determine the obstacle free trajectories. The critical information obtained from the image processing of the satellite image is transferred to the image using semantic annotation in Resource Description Framework/Extended Markup Language (RDF/XML) [29].

1.8 Focus of the Thesis, and the Problem Definition

The previous subsections described the developments of the agricultural industry for automation of the crop plantation, and states the availability of key technologies to develop fully automated agricultural machineries. Although the proposed methods in the literature provide some solutions for several levels of an agricultural automation system, none of them provided a practically applicable precise global positioning based path planning environment for annotation methods starting with a satellite image of an agricultural field. In the existing literature, most authors considered the obstacles to be processed manually, and therefore they considered each subsystem as an individual system with its own particular geographic map, list of restricted areas, and trajectory to cover the entire field completely. This type of isolation of subsystems from each other reduces the possibilities of information-exchange between the consecutive subsystems.

This gap in the automation of the agricultural machines directed this study to develop a system for planning the motion of the agricultural machines in the fields of crops. The availability of the satellite images with sufficient resolution shaped one bounds of the study, while so many published trajectory tracking studies determined the other bound at the planning of the trajectories, rather than developing low level control to track a trajectory [30].

Consequently, this study focus on detection of the location of typical obstacles for agricultural machines in a field by processing the Google-Maps Satellite image of the field, and annotate their position, type, and properties on the map of the field, which is proposed to support the information-exchange between the subsystems of an automated agricultural plantation system. Even though this thesis is carried on the images supplied by the free services of Google-Maps, the methods proposed in this thesis are expected to be directly applicable on higher-resolution images supplied by other commercial non-free satellite services for obstacle detection and trajectory generation purposes.

The principal contribution of the thesis is to propose a methodology of precise geographical mapping for agricultural lands starting from a satellite image of the target land. It targets to fill the gap of preparation of a suitable geographic map for the path planning steps of the agricultural automation, which requires typical data for types of soil, slope of land, boundaries of the field, locations and types of the obstacles, etc. Along with this principal contribution, this thesis aims to contribute in image processing area to develop methods of obstacle detection, and to determine the best edge detection algorithm among the four well known and widely used methods: Sobel, Prewitt, Roberts, and Canny, together with the Circular Hughes Transformation, for the purpose of recognition of the obstacles.

One of the aims of this study is to generate precise trackable trajectory points for specific tasks to be performed by autonomous agricultural vehicles. At the implementation, the preferred path planning algorithm simply generates the crop lines parallel to the longest edge of the target field with a constant crop-row distance, because this study does not aim to develop a new path planning algorithm. Rather, it targets to introduce a convenient semantic annotation to implant available path planning algorithms which may generate trajectory points to satisfy the concern of agricultural policies. Considering agricultural efficiency, the type of the plantation crop, the soil properties and condition of the field, available tools, and surrounding environment may affect the crop-row direction, distance, and depth. Higher level crop policy algorithms shall be implemented in future to determine the crop-lane directions, crop-row distance, and depth depending on many factors other than the satellite image.

Along with automatic obstacle recognition and crop-row determination, this thesis implemented standard semantic annotation system to describe several properties of the agricultural land, obstacles, and designed crop-row paths for agricultural machinery and tasks. The implemented subsystems provide simple, but robustly efficient examples of using the proposed semantic annotated map to fulfil a frontier role in filling the gap of information flow from a geographical map image to vehicle trajectories with semantic annotation.

1.9 Contents of Further Chapters

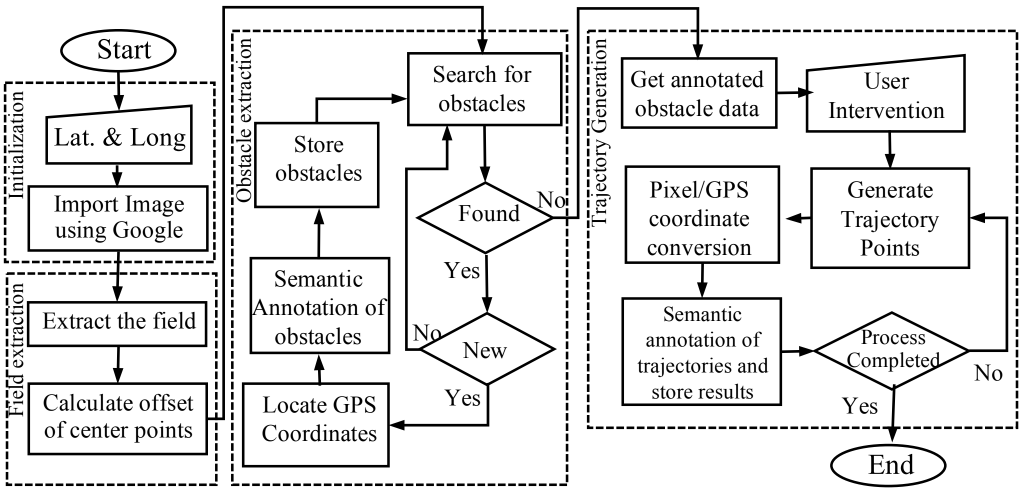
In the remaining chapters of this dissertation the architecture of the proposed system and the preliminaries of this architecture are introduced in Chapter 2, where the detailed information about each stage of the implementation is explained together with the initialization process of the application for a target field, giving examples of operations at each stage on a set of field samples. Experimental results are presented in Chapter 3 for a set of almost 50 target fields, and interpretations of results are discussed in Chapter 4. Finally, Chapter 5 states the conclusion for the overall thesis.

Chapter 2

DESCRIPTION OF PROPOSED SYSTEM

2.1 System Architecture

The proposed system requires completing some stages before presenting adequate semantic annotations. The process starts with capturing the top-view image of the target field. Also, obstacles are to be recognized and trajectory points to be produced before forming the exchangeable data. As demonstrated in Figure 4 the developed system consists of four main stages. The first stage, “Initialization,” is used to locate the field and receive the satellite image. In the second stage, "Field Extraction," the target agricultural area is extracted from the obtained image. “Obstacle Detection,” the third stage, is used to detect, recognize, and annotate obstacles and store the results. The trajectory reference points are generated and annotated in the final stage.



Start
Initialization
Lat. & Long
Import Image using Google
Field extraction
Extract the field
Calculate offset of center points
Obstacle extraction
Search for obstacles
Store obstacles
Found
No
Yes
Semantic Annotation of obstacles
New
No
Yes
Locate GPS Coordinates
Trajectory Generation
Get annotated obstacle data
User Intervention
Pixel/GPS coordinate conversion
Generate Trajectory Points
Semantic annotation of trajectories and store results
Process Completed
No
Yes
End

Figure 4: Four stages of the developed system

2.2 Initialization

At the initialization step, the system requires the GPS coordinate $GPS_F=(lat_F, long_F)$, of any point F in the target field to describe the field location, and get the satellite image of that location. The developed system uses Google-Maps service API for locating and importing the satellite image containing the target field using an API code. As mentioned in the previous chapter, Google-Maps service provides various zoom levels, from one to twenty-two. At zero zoom level, which is the minimum possible zoom, the entire earth fits in the picture, and each higher zoom level doubles the detail of map, doubling pixel per distance both in x and y axis. The boundaries of the desired field are determined using the highest possible zoom level (largest Z) that allows the entire field to fit into the image frame. Depending on the size of the target field, proper value of Z is determined manually before initialization of automatic image processing, because this step relies on the user's information for the boundaries of the field.

During the image processing phase, median filtering is applied to the image to reduce and remove noise (also referred to as salt and pepper noise). The median filtering uses a 3x3 neighbourhood matrix [31].

2.3 Segmentation and Extraction of Target Field

The second stage of the process converts the received satellite image into grayscale for extraction of the target field purpose, by reducing the hue and saturation while maintaining the luminance. This conversion is performed by using weighted sum of Red, Green and Blue components of the coloured image in (0.298 Red + 0.587 Green + 0.114 Blue). Representing the image in the grayscale format by this weighted colour composition improves the opportunity of detecting the regions in addition to detecting the edges more precisely. Additionally, application of 3x3 median filtering on the grey scaled image minimizes the effect of noises in detecting the regions. The target area is extracted from the original satellite image by using the segmentation and sectioning methods developed in [32], [33].

Figure 5b shows the image of the target field in binary format after the noise reduction. In the filtered grey scale image, adjacent pixels of the same colour value (*ni*) are connected to form regions (*Ri*). The regions are numbered and sorted based on the number of connected pixels as illustrated in Figure 5c. The target field is detected as the region with the highest number of connected pixels which contains the centre pixel of the image. Also, the centre pixel of the image is mapped to mark the location of the GPS point supplied by the user at the initialization stage. This mark is necessary to save the pixel coordinate of the GPS point for the further positioning calculations [34].

The bounding box information of the extracted field is used to remove all unnecessary parts of the image by cropping the bounded region tightly. Cropping results in a smaller image, which contains the target field with much less irrelevant areas as shown in Figure 5d and Figure 5e. The effect of cropping the image on reducing the error in obstacle detection is small but positive. Appendix A presents corresponding Matlab codes for segmentation and extraction.

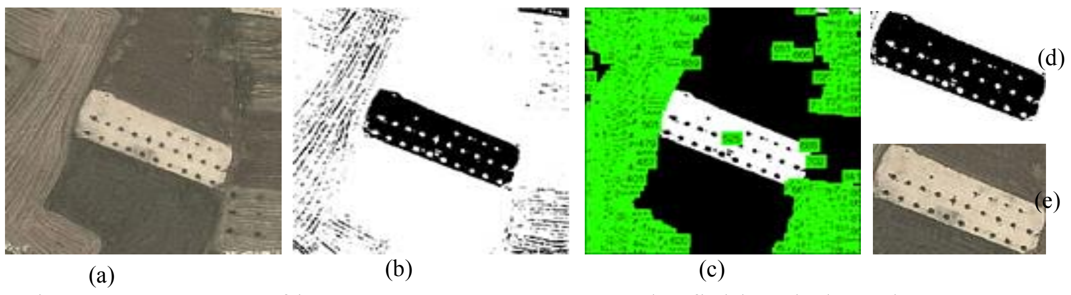


Figure 5: Progress of image processes to extract the field and obstacles. a- Raw image input, b- binary conversion (inverted), c- segmented image, d- cropped picture to fit the field in frame, e- the field in original colours

2.4 Detection of Obstacles on the Field

The next stage of the process is to find obstacles on the obtained satellite image. The following sections describe the three sub-stages related to the obstacle detection stage: detection, positioning and semantic annotation of obstacles.

2.4.1 Detection Algorithm

After the image is tightly cropped to the bounds of the field, the image is processed to detect objects inside the field boundaries (Appendix B). But any spurious noise in the image, some of them outside the boundaries of the field, is expected to appear because of the grey-scale conversion operation. Removal of this noise is necessary even if they appear outside of the boundaries since noise creates problems, and increases error rate in detection of obstacles.

Figure 6a shows some examples of these spurious noises inside and outside of the field boundaries. This noise is eliminated by reconstructing the image from the largest stored region, and filling the outside of this region on the image of target field. Figure 6b shows the reconstructed and filled image that eliminates the noise outside the field boundaries.

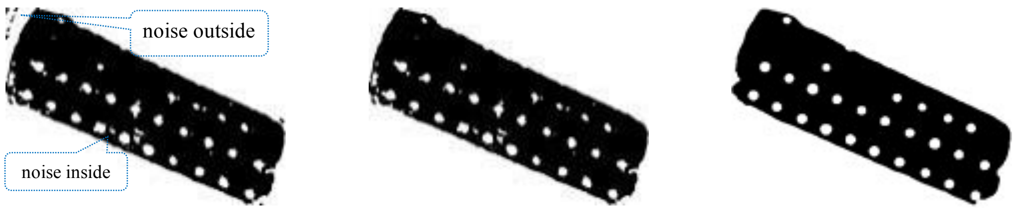


Figure 6: Images while morphological reconstruction to filter noise, a- Undesired segments on a binary image, b- undesired segment are removed from outside, c- desired field free off undesired objects

Noises which are at the regions out of boundaries of the field are removed and filtered by applying morphological image reconstruction [35]. Removal of some objects is necessary because they are too small to be counted as an obstacle such as large thorns. The output of this procedure is a more precise and clear image, and, after the inversion, white regions denote obstacles while dark region represents the field as shown in Figure 6c. The pixel values of inverted binary image are used in obstacle detection phase through segmentation method.

As stated earlier, each zoom level of Google tiles covers different surface areas. Consequently, the same object may have different radius value in pixels when processed at different zoom levels. Based on our observations, at zoom level Z =18, a typical obstacle on the field may have radius in the range of 4 to 18 pixels. Therefore, in the obstacle detection stage zoom level is set to 18 in all experiments to provide consistency in calculating efficiency and comparing the results.

Usually top-view graphical appearance of trees on satellite images is elliptic or circular shaped. This kind of round objects can be easily recognized by *Circular Hough Transform*. In the application, the object polarity for the Hough transform is set to "bright", and the value of edge gradient threshold is set to 0.27, default value of sensitivity factor = 0.85 is preferred, and computation method is set to "Phase Code".

During detection phase, any recognized obstacle is validated by a duplicity check algorithm and is accepted as a new entry if the detected obstacle passes the test as seen in Appendix C. The duplicity check works by comparing the centre pixel value of the detected obstacle with the centre pixel values of validated obstacles in the list, and if no duplicated values are found will pass the test. Upon validation, the obstacle including its centre pixel value and the radius is saved to the list of identified obstacles for later determination of its diameter in the metric system.

2.4.2 Calculation of Positions

Google Maps works on GPS coordinates and Mercator projection, which provides equal scaling of x- and y-direction at any location of the Earth for sufficiently small parts on globe [36]. GPS coordinates of a location *A*, GPS_A, is a vector with two components, the first component is the latitude angle in degrees, and the second component is the angle of longitude of the target location, as shown in Figure 7.

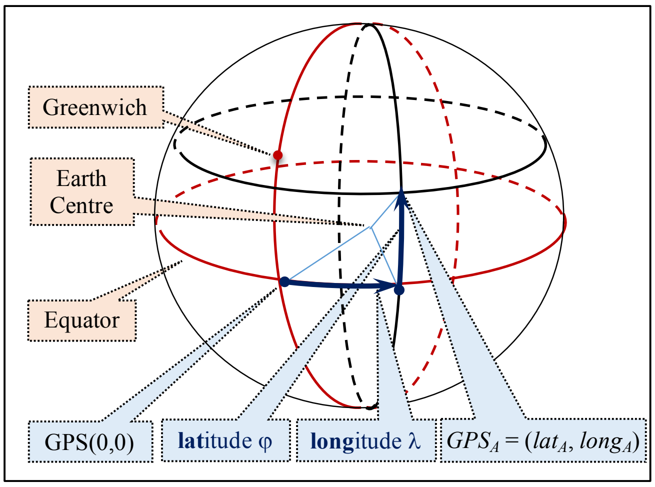


Figure 7. GPS coordinates of a location.

Image resolution of each tile in the map is 256x256 pixels, and first zoom level tile covers the whole Earth exactly in a single tile. Google-Maps service provides the GPS coordinates of any point in the image, but the coordinates are not embedded into the picture. Therefore, while processing the images, any associated GPS coordinate is expected to shift depending on image operations. The well-known method to convert the GPS coordinates to distances is the Haversine formulas. However, this thesis is concentrated on the distances corresponding to the Google-Maps tile pixels, and therefore the following paragraphs develop pixel based distance calculations on Google-Maps images.

Mercator projection delivers vertical and horizontal direction of the map always equally scaled at any location of the projection, but the scaling factor gets smaller while the location gets closer to the Poles, because scaling factor changes by cosine of longitude. Appendix D presents the developed geographical positioning code.

Google officially assumes the Earth a sphere with radius exactly r_E = 6378137 meters. Accordingly, one complete tour of θ_e=360 degrees along the circumference (i.e., equator line) of the Earth is approximately L_E= 40,075,017 meters. Increasing Google-Map zoom level Z one step increases the details of the map twice in horizontal and twice in vertical direction. At any zoom level Z, the distance L_E, or complete revolution around Earth, θ_e=360°, is covered by 2^Z tiles. Each tile has 256 pixels, giving total $N_z = 256\times2^Z = 2^{Z+8}$ pixels around Equator. Consequently, at zoom level Z the angular displacement corresponding to each pixel is

$$\Delta_G = \frac{\theta_e}{2^{Z+8}} \tag{1}$$

where, Δ_G denotes degrees of latitude per pixel in x-axis direction, or equivalently degrees of longitude per pixel in y-axis direction on a tile if Z is sufficiently large to ignore projection distortions in the tile. For example, at Z =18, one pixel movement in north or east direction corresponds to $\Delta_G = 5.36441803 \times 10^{-6}$ degrees change in GPS longitude or latitude. If the location is on the equatorial line, the Earth resolution at this tile is calculated by using the circumference of the Earth along Equator

$$\Delta_E = L_E / N_Z \quad , \tag{2}$$

where, Δ_E denotes the distance on Earth per pixel on any tile centred at latitude = 0. Equation (2) provides also the distance along Equator in meters per pixel of image. For example according to (2), at zoom level 18, each pixel at the equator corresponds to 0.5971645 meters.

If the pixel on the image is not on Equator line, the distance equivalent of the pixel in x-direction requires a correction for the spherical shape of the Earth. For a point C on the Earth closer to the poles, the round trip distance along a parallel is shorter compared to equatorial round trip L_E. This round trip distance decrements by the cosine of latitude of the GPS coordinate. Consequently the Earth resolution around a centre point C gets smaller

$$\Delta_C = \Delta_E \cos(c_{r/d}\ lat_C) \quad , \tag{3}$$

where, lat_C is the angular latitude of C in degrees; $c_{r/d} = \pi/180$ is the conversion factor from degrees to radian; and Δ_C is the distance on Earth in meters per Google-Maps pixel, which depends on lat_C of the centre point and mostly called as Earth resolution of image at tile centre. The target field is identified by the GPS coordinate of a location inside the field. But, most of the operations are carried on the tile image using pixel coordinates, and the objects are decided using their sizes in meters. Therefore, determination of the distance in meters is of particular interest. The GPS resolution of a zoom level, that is the angular displacement of longitude per pixel of a Google map is constant, Δ_G=360/N_z . But the Earth resolution Δ_C, that is, the distance on Earth in meters per pixel of image, depends on latitude, and requires a correction.

The pixels of an image is conventionally addressed using (0,0) for top-left corner. But, on the satellite image received from the Google service, the specified centre GPS address, GPS_C, is located always at the centre of the original image. Therefore the pixel coordinate of original centre pixel, p_{OCP} = (x_{OCP}, y_{OCP}), is required for further pixel coordinate calculations. The original centre pixel, p_{OCP}, becomes shifted by the coordinate of bounding box start pixel, p_{BBP} = (x_{BBP}, y_{BBP}) while cropping the image at the field extraction stage. Coordinate calculations need offset correction for this shift of centre pixel. An example of cropping is demonstrated in Figure 8, where a bounding box with p_{BBP} = (163.19, 173.11) is applied on original image to get the cropped image with the origin shifted 163 pixel along *x*, and 173 pixel along *y* direction. The offset of pixel(0,0) of the cropped image in the original image is necessary to transfer the pixel coordinates on the cropped and original images back and forth. The correction offset in pixels, p_{CCP} = (x_{CCP}, y_{CCP}) for the cropped image is

$$x_{CCP} = x_{OCP} - x_{BBP}\,; \quad \text{and} \quad y_{CCP} = y_{OCP} - y_{BBP}\,. \tag{4}$$

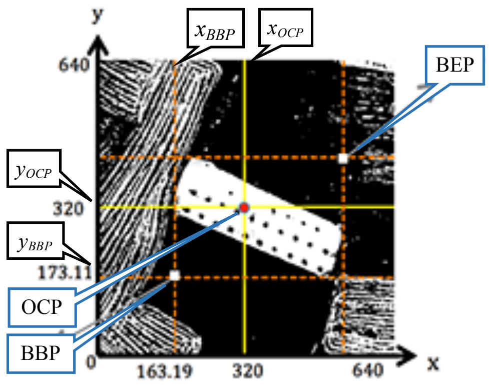
Pivot GPS coordinates of the target field *F* is given by the pairs of latitude, lat_F, and longitude, $long_F$, in degrees with decimal fractions. For example, latitude and longitude of the centre pixel o *C* f the sample field shown in Figure 8.a are at $lat_C = 35.075373$ and $long_C = 33.531142$. The centre pixel is the pivot point for all further coordinate calculations. Its exact pixel location in the image is maintained by *OCP* coordinate and the offset of cropping in pixels is specified by *CCP*.

The corrected centre pixel coordinate in cropped image correspond to pivot GPS coordinate of the target field, (lat_C, $long_C$). Calculation of original pixel coordinate of an object *i* is possible by using CCP and the pixel coordinates of each object $p_{T,i} = (x_{T,i}\ ,\ y_{T,i})$. At the end of determination of obstacles, procedure is completed by calculating GPS coordinate $GPS_{T,i} = (lat_{T,i},\ long_{T,i})$ of the obstacle's centre from its pixel coordinate $(x_{T,i}\ , y_{T,i})$ by equations (5) and (6).

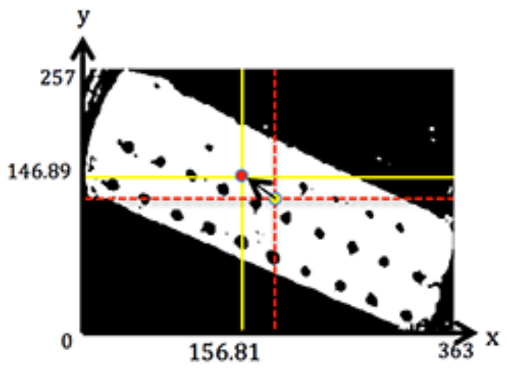
$$lat_{T,i} = lat_C + \Delta_G (x_{CCP} - y_{T,i}) \cos(c_{r/d}\ lat_C) \tag{5}$$

$$lng_{T,i} = lng_C - \Delta_G (y_{CCP} - x_{T,i}) \tag{6}$$

where $\Delta_G = 360^o/N_Z$ is the GPS resolution of the images. After validating that the obstacle is not a duplicate, the properties of the obstacle is stored including its index, position, pixel coordinates, radius, latitude and longitude; and its position is graphically marked by a red circle as described in next section.



a



b

Figure 8. Trimming image size for the target field a- Mapping of field's latitude and longitude coordinated to the center pixel of original image, b- Offset calculation of center point

2.4.3 Semantic Annotation

Identified obstacles are stored and presented in a descriptive format readable by both humans and machines. Detected obstacles are semantically annotated and graphically marked to represent them both in text format, and by marking them on the image. Documenting data with proper standardized titles and formats to remove any ontological ambiguity is accomplished by semantic annotation [37].

In the semantic annotation of the detected obstacles RDF/XML which is defined by W3C [45]. RDF/XML uses triples of Subject, Predicate, and Object to define an entity. The proposed RDF data graph to semantically annotated and represent detected obstacles is shown in Figure 9.

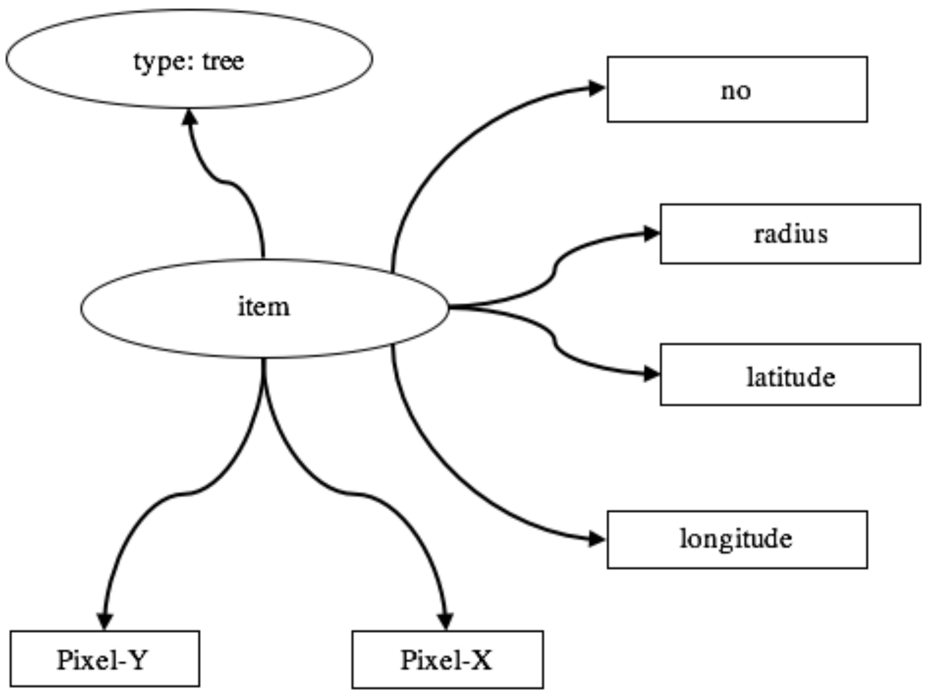


Figure 9: RDF/XML data graph of detected tree

To define the detected obstacle's type which in our case is a tree, we have imported the Plant Onology (PO) [46]. The namespace defined by PO *http://purl. obolibrary.org/obo/PO_0000003/hasNarrowSynonym/Tree* narrows down the whole plant anatomy to a "Tree". Using this namespace, we have semantically represented the detected obstacles "type" as a tree. To have a semantic representation of the tree, its calculated GPS coordinates are expressed by importing W3C Geospatial Ontologies (OGC) [47]. This ontology defines latitude and longitude coordinates with *http://www.w3.org/2003/01/geo/wgs84_pos/lat* and *http://www.w3.org/2003/01 -/geo/wgs84_pos/long* accordingly. In addition, to define the pixel coordinates of the tree Scalable Vector Graphics (SVG) is imported. This ontology defines the tree with its center points x,y pixel coordinates and its radius as "svg:cx", "svg:cy", and "svg:r" attributes on the obtained image. The item number of the tree "obt:no" define as an integer in accordance with XML schema. Figure 10. shows the structure of semantic annotation for a detected and validated obstacle in RDF/XML format.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:obt="http://www.DLC.org/Obstacle#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:gn="http://www.geonames.org/ontology#"
xmlns:wgs84_pos="http://www.w3.org/2003/01/geo/wgs84_pos#"
xmlns:item="http://purl.obolibrary.org/obo/PO_0000003/hasNarrowSynonym
#"
xmlns:svg="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd/circle#">
<rdf:Description rdf:about="http://www.DLC.org/Obstacle#"/>
<item:tree>
        <dc:type rdf:resource="http://purl.org/dc/dcmitype/Image"/>
        <obt:no rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
Number of the tree </obt:no>
        <wgs84_pos:lat> latitude of tree </wgs84_pos:lat>
        <wgs84_pos:long> longitude of tree </wgs84_pos:long>
        <svg:cx> x coordinate of tree in the image </svg:cx>
        <svg:cy> y coordinate of tree in the image </svg:cy>
        <svg:r> radius of the tree</svg:r>
</item:tree>
</rdf:RDF>
```

Figure 10: Semantic annotation structure of obstacles in RDF/XML file

2.4.4 Improved Detection Algorithm

In the preceding stages of this study, out of four nominated edge detection methods (Canny, Prewitt, Sobel, and Roberts), Sobel was selected as the main edge detection method. This selection took place based on Sobel's overall performance which was higher than the others. However, during the study, we realized that Sobel does not provide the best result as expected in some cases. This lack of performance led us to investigate deeper for possible enhancements in the edge detection algorithm. As a result, three methods including "T-range", "Max of All" and "Double Layered Check (DLC)" are introduced and explained in the following sections.

2.4.4.1 T-range

Initially, Sobel was used with its default threshold value (0.27) in Matlab. However, we have decided to consider the complete threshold range for Sobel. Outputs are analysed to find the best possible threshold value in which Sobel reaches its maximum number of detection. Figure 11 shows the results on ten test fields. As presented in this figure, for each field Sobel reaches its detection's peak with a various threshold value. Having such a variety of the threshold values indicates that size, type, and obstacles within the field have a direct impact on choosing the best threshold value. Therefore, Sobel's performance can be improved by selecting the right threshold value for each field.

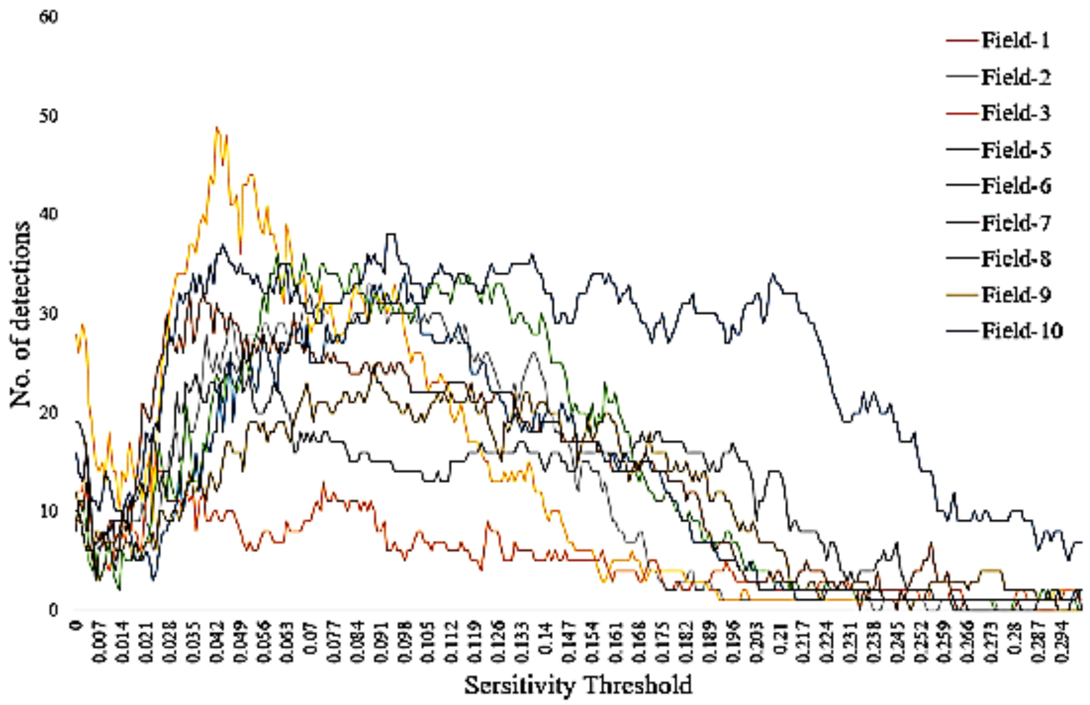


Figure 11: Detection results using threshold range on Sobel.

2.4.4.2 Max-of-All

In this method, outcomes of all four nominated edge detection algorithms on each field are taken into account. As indicated before, in some fields Sobel was not functioning as expected and had much lower performance comparing to the other three methods (Canny, Roberts, and Prewitt). We found higher detection potentials by investigating the results from other edge detection methods (especially Canny) in the fields which Sobel failed. Therefore, a dynamic selection method (Max-of-All) is introduced to determine the best edge detection algorithm on each field individually. The Max-of-All determines which edge detection method finds highest number obstacles (trees) in individual agricultural fields. For example, Canny might find higher number of obstacles in a particular field in comparison to Sobel, Prewitt, and Roberts, therefore, Canny would be the nominated edge detection method for that particular field. However, it does not mean that Canny have reached the highest number of "correct detections". It simply implies that Canny found higher number of obstacle-candidates which includes correct detections, incorrect detection or both. The decision on which of the detections are correct is made after applying gray-level intensity threshold, which classifies the detections into two categories of correct and incorrect detections. Then we can decide which detection is valid and which one is invalid. This evaluation is done in section 2.4.5 (Accuracy and Error Evaluation).

2.4.4.3 Double Layered Check (DLC)

In the previous section "Max-of-All" method improved T-range's results overlay. However, there were still some cases in which T-range performed better than the Max-of-All regarding obstacle detection. The inconsistency in achieving the best result by either of these two methods (T-range and Max-of-All) led use to investigate even further more to achieve the best result. To overcome this issue, a new algorithm called Double Layered Check (DLC) is introduced which merges T-range and Max-of-All as coded in Appendix F. DLC applies both threshold range and the maximum number of obstacle detection on each edge detection algorithm (Canny, Roberts, Sobel, and Prewitt) for each field individually.

2.4.5 Accuracy and Error Evaluation

Two types of error and a location mapping test are used to evaluate the developed system. All the error evaluation tests are applied to the individual edge detection methods before and after improvements to compare and observe the enhancement level. The first error type is FPE (False Positive Error), which refers to the missed obstacle in the detection. If there exists an obstacle within the field and the detection method fails to capture the obstacle, it would be counted as an FPE.

In order to reduce the FPE error the proposed system takes advantage of grey-level intensity threshold to differentiate correct and incorrect candidates as coded in Appendix G. During the evaluation process grey-level intensity value of each detected candidate is compared to a specified threshold value. If the grey-level is below the threshold value, the obstacle-candidate is considered as a correct detection and is processed accordingly. Otherwise the obstacle-candidate is considered as a wrong detection and discarded.

The second type of error represents wrong detections in which an obstacle has been detected and captured; however, such an obstacle does not exist. This kind of error could happen due to the noise and inaccurate parameter sets for example threshold. This type of error is called FNE (False Negative Error).

As discussed before the proposed system works with a single GPS coordinate of the target agricultural. This point is provided during the initialization stage, and all further positioning procedures rely on that single point. A comparison test is performed to evaluate the accuracy of the system in mapping that single GPS coordinate to pixels and locating obstacles. This test inserts the calculated location of an obstacle in Google Maps and checks if it points to the right object.

2.5 Trajectory Points

One of the primary goals of the implemented system is to generate trajectory reference points which autonomous agricultural vehicles could follow. These traceable points will form a sequence of GPS coordinates. The list of path points consists of tracking paths and sectors, and it is produced after computing all necessary information about the field area and the obstacles within. The developed system is capable of integrating the generated trajectory points into any path planning algorithm. The produced trajectory points may need further processing to determine the direction along these trajectories, and the depth of the crop rows to be executed by agricultural vehicles. In the implemented coding, the route planning algorithm creates paths parallel to the longest edge of the field. The path planning unit is purposely coded in simplest form since this thesis aims only to validate the integration of a path planning unit to the obstacle mapping unit, and in practice path planning unit may be replaced by codes of already available better crop-row trajectory planning algorithms.

2.5.1 Generation of Trajectories

In the trajectory point generation phase, we have to make sure that no collision occurs while vehicles are following the reference points. To have a collision free path we have to avoid placing any trajectory point on the detected obstacles. In the binary image, the area with white pixels represents the main field. As mentioned before, the path planning algorithm proposed in this study creates paths parallel to the longest edge of the field. This longest edge is detected using Standard Hough Transform (SHT). The angle between the origin to the closest point (θ_h) is set to $90 \leq \theta_h < 89$ in the algorithm. Also, the distance is set to 0.5 between the line (ρ_h) and the origin correspondingly. The length of each edge in the field can be calculated by Pythagorean Theorem using equation (7), since both the pixel coordinate axes and also Earth coordinate axes are perpendicular. Let the coordinate of the first and second points be denoted by $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$. Pythagorean equation finds the Euclidian distance between these points, $\mathrm{d}(p_1,p_2)$, which is also the length of the line from point p_1 to point p_2.

$$\mathrm{d}(p_1, p_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{7}$$

Among the distances between the consecutive corner points, the highest distance and its corner points are saved as $ED_m=\{p_{EDm,1}=(x_{EDm,1}, y_{EDm,1}), p_{EDm,2}= (x_{EDm,2}, y_{EDm,2})\}$, the longest side of the field. The slope of this edge is

$$m_{EDm} = \frac{y_2-y_1}{x_2-x_1} , \tag{8}$$

The bolded and highlighted blue line shows ED_m in Figure 10a. The crop rows are generated parallel to ED_m, with d_{ms}, the user set crop row spacing. The distance between the rows, d_{ms}, is entered in meters. But for pixel domain calculations, it is converted to pixels and denoted by d_{ps}.

$$d_{ps} = d_{ms}/\Delta_C \tag{9}$$

The crop rows are generated on the image with constant spacing d_{ps} which is entered by the user by setting d_{ms} in meters. As shown in Figure 12.b, anchor dots in red (temporary points) are placed as reference points to mark spacing of each row on the perpendicular line starting from the midpoint of ED_m towards the centre of field C. Drawing lines through these anchor points results in equally spaced rows with a distance d_{ps}. For the anchor points, the midpoint p_{EDmp} and the slope of the anchor line are calculated by

$$p_{EDmp} = \left(\frac{x_{EDm,1}+x_{EDm,2}}{2}, \frac{y_{EDm,1}+y_{EDm,2}}{2}\right) , \tag{10}$$
$$m_{EDmp} = \operatorname{atan}\,(m_{EDm}) ,$$

where m_{EDmp} is the slope of the perpendicular line to the edge ED_m. Crop row anchor points, A_i, are inserted on the image with a constant increment d_{px} and d_{py} to have equal distance from each other using the equations until generated point lies outside the field boundaries using (11) and (12).

$$x_{A,i} = x_{EDmp} + i\, d_{px} \sin\left(m_{EDmp}\right) \tag{11}$$

$$y_{A,i} = y_{EDmp} + i\, d_{py} \cos\left(m_{EDmp}\right) \tag{12}$$

After this step, a sequence of the trajectory reference points is generated in a straight line using $y = m\,x + y_0$ for parametric values of x starting from 1, ending at N_x, the width of image in pixel coordinates. As a result, $p_{R,i,k} = (x_{R,i,k}\ , y_{R,i,k}\ )$ pixel coordinates are generated to form trajectory points for the crop row, or track-line parallel to the longest edge by the expressions

$$\text{for } k = 1 \ldots N_x,\ \{\ x_{R,i,k} = k\ ;\ y_{R,i,k} = m_{EDm}\left(x_{PL,i,k} - x_{A,i}\right) + y_{A,i}\ \}. \quad (14)$$

The generated track line points, $(x_{R,i,k}, y_{R,i,k})$, k=1 … N_x are tested for the collision condition to an object or to the boundaries of the field to prevent any overlap of the produced trajectory points with the detected obstacles. The obstacle avoidance algorithm works by mapping the coordinate of each trajectory point in pixel coordinates (x, y) to the binary image of the target field and obstacles. If the colour of corresponding pixel is 0, the trajectory point is discarded, since colour 0 indicates that the point is in an area of an obstacle, or it is outside of the field. If the colour of the trajectory point pixel is 1, the point is accepted to be on the crop row. If the test is positive, trajectory points are marked on the image of the field to form a sequence of red dots that appears as a straight line as presented in Figure 12.a.

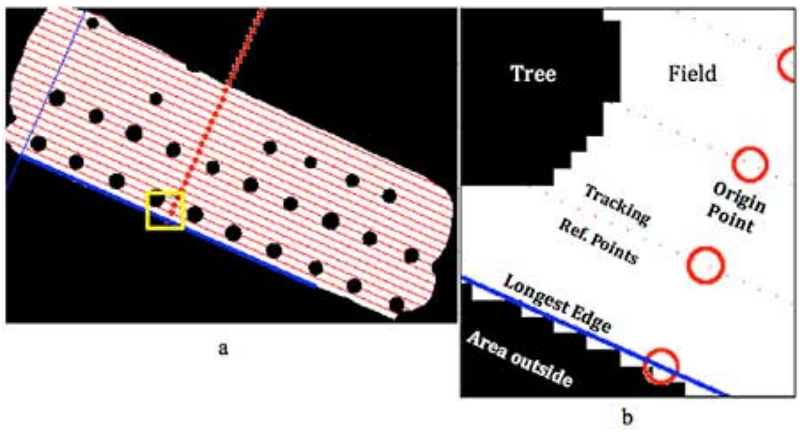


Figure 12: Production reference and trajectory points, a- overview, b- details

2.5.2 Annotation of Trajectory Points

The trajectory points are produced for automation of agricultural vehicles which shall process the field along the crop rows by tracking these trajectories. The agricultural vehicles, which are installed with GPS tracking devices, shall follow reference points in GPS coordinates rather than pixel coordinates. Therefore, it is necessary to re-map the produced trajectory points from pixel format into GPS coordinate format. As mentioned before, the distance on Earth is not proportional to the angular GPS distance. Accordingly, the precise length of a single degree of longitude per pixel (*LDPP*) needs to be calculated to have high accuracy in positioning. The Pixel-to-GPS coordinate mapping operation converts the crop-row points from pixel coordinates to GPS coordinates, and stores them in $m \times 4N_R$ format 2D-array, where the points on each track are represented on a row of four columns, $\{(x_{R,i,k}, y_{R,i,k}), (lat_{R,i,k}, long_{R,i,k})\}$, as presented partially in Table 1, where the first two columns are pixel, and the last two are GPS coordinates of the points on the path. A path may contain several gaps, corresponding to obstacles marked on image. Data in Table *1* shows a part of the first track, and the complete path table contains 4n columns for total N_R reference trajectory paths (or tracks), so that columns 1-4 represents the first track, 5-8 represents the second, and so on. The pixel coordinates of each trajectory point in pixel $R_{i,k} = (x_{R,i,k}\ , y_{R,i,k})$ are presented in the first pair of columns, and the same point's GPS coordinates $(lat_{R,i,k}\ , long_{R,i,k})$ are shown in the second pair of columns.

Table 1: Trajectory points of each track in pixel and GPS pairs

k	$x_{R,1,k}$	$y_{R,1,k}$	$lat_{R,1,k}$	$long_{R,1,k}$
23	63	12.391…	35.07558507	33.53089255
24	64	12.836…	35.07558409	33.53089524
25	65	13.281…	35.07558311	33.53089792
26	66	13.725…	35.07558214	33.53090060
27	**67**	**14.170…**	**35.07558116**	**33.53090328**
28	**77**	**18.619…**	**35.07557139**	**33.53093011**
29	78	19.064…	35.07557042	33.53093279
30	79	19.509…	35.07556944	33.53093547

If a crop row passes through an area occupied by an obstacle, it results in a gap on the crop row, by breaking the line into two line segments and a gap segment which is specified by its terminal points. A gap $G=\{(x_{G,1},y_{G,1}), (x_{G,2},y_{G,2})\}$ on the crop row represents the location of an obstacle. For a solid example, in Table 1 and Figure 13, $x_{R,1,k}$ jumps from 67 to 77 indicating the occupied location by an obstacle, where the gap segment is specified by pixel coordinates G={(67, 14.17), (77, 18.619)}. This gap also indicates the end point of one path segment and the start point of the next one. Total number of tracks, N_R, is determined by counting the anchor points that generates any valid reference path in the field. As a result, the columns of matrix enlarges to $m = 4N_R$. Number of gaps on a track-line, $N_{G,i}$, is counted by testing sequence of the first column for each track, segments of each track, $N_{S,i}$, is obtained by checking first column and counting the gaps on each crop row. The total number of segments in the entire path N_S, are calculated using following equations respectively.

$$N_R = \mathrm{m} / 4 \tag{15}$$

$$N_{S,i} = N_{G,i} + 1 \tag{16}$$

$$N_S = \sum_{i=0}^{N_R} N_{S,i} \tag{17}$$

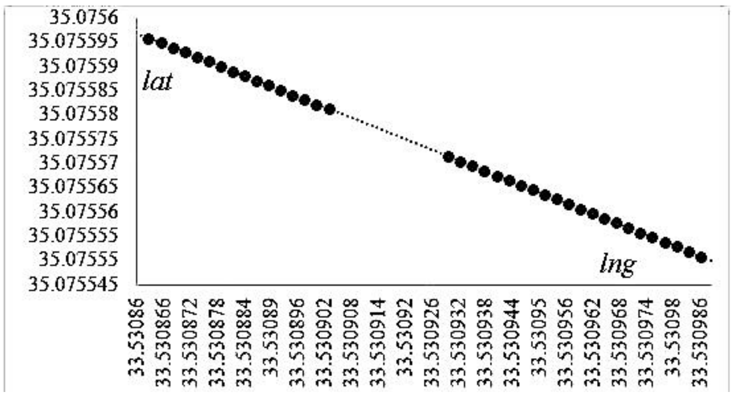


Figure 13: An obstacle on a track-line appears in the form of a gap

2.5.3 Semantic Annotation

Detected obstacles and generated trajectory points are semantically annotated and stored in RDF/XML format. These annotations make the computed data exchangeable between other sub-systems or ontologies that work on arable lands. Figure 14 shows the RDF data graph of a generated trajectory point.

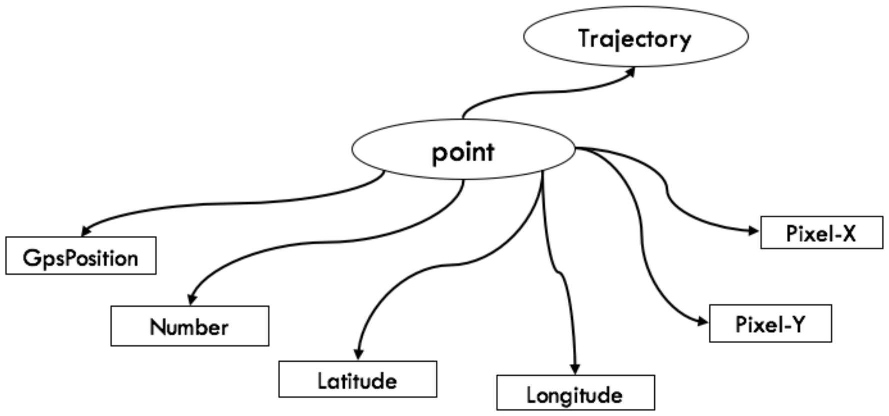


Figure 14: RDF/XML data graph representing a trajectory point

Ontologies are imported to describe a set of terms to represent the generated trajectory points semantically. Each trajectory point is identified as an “item” in the semantic annotation process. Dublin Core metadata ontology is used to describe that the knowledge provided in this study belongs to an image file type [49]. An ontology of Open Geospatial Consortium (OGC) is imported to define the GPS coordinates of the generated trajectory points. OGC provides spatial metadata to be used by other ontologies. The namespace defined to use OGC is *xmlns:item="http://www. opengis.net/gml/*". OGC defines any GPS coordinates by latitude (item:lat) and longitude (item:long). Data type of the trajectory point's number (trj:no) and the track which it belongs to (trj:track) are defined as integer based on SML schema. Scalable Vector Graphics (SVG) ontology is imported to semantically annotate the pixel coordinates of each trajectory point. Each x-pixel coordinate and y-pixel coordinate is presented by "svg:cx" and "svg:cy" respectively [48]. Figure 15 shows the RDF/XML structure to describe and annotate trajectory points.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:trj="http://www.DLC.org/trajectory#"
xmlns:item="http://www.opengis.net/gml/"
xmlns:svg="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd/circle#">
<rdf:Description rdf:about="http://www.DLC.org/Trajectory#"/>
<item:point>
        <dc:type rdf:resource="http://purl.org/dc/dcmitype/Image"/>
        <item:pos> GPS Coordinate </item:pos>
        <item:lat>Point Latitude</item:lat>
        <item:long>Point longitude</item:long>
        <trj:no rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
                                Point Number</trj:no>
        <trj:track rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
                                Track Number</trj:track>
        <svg:cx>Pixel-X Coordinate</svg:cx>
        <svg:cy>Pixel-Y Coordinate </svg:cy>
</item:point>
</rdf:RDF>
```

Figure 15: Semantic annotation structure of the trajectory points

Chapter 3

RESULTS

3.1 Extraction and Detection

The method proposed in this research is developed and tested using the technical programming language platform developed by MathWorks Inc, called MATLAB. The hardware platform used for experiments is an Apple (MacBook Pro) running on Mac OSX Yosemite (10.10.3), 8 GB of RAM and Intel i7 2.8 GHz CPU. Satellite images used in this study are accessed using an API suggested by Google. These images are received with the maximum resolution of 640 x 640 dpi.

Top-view satellite image of 51 agricultural fields are used in this experiment to evaluate the developed system. These areas have various shapes, sizes, environments and number of obstacles. The zoom level is set to 18 in all the experiments to maintain consistency of the comparisons. The radius range used in this study is in between $\mathcal{R}\alpha_{min}$ = 4 and $\mathcal{R}\alpha_{max}$= 18 corresponding to Δ_z = 18 for obstacle detection purposes. The obtained image is cropped to fit the field's area in the picture to improve accuracy. Figure 16 shows some examples of the cropped images. The total number of obstacles in these 51 test fields is equal to 1602. Appendix H lists counts of initial obstacle detection and their error counts for default sensitivity-threshold value.

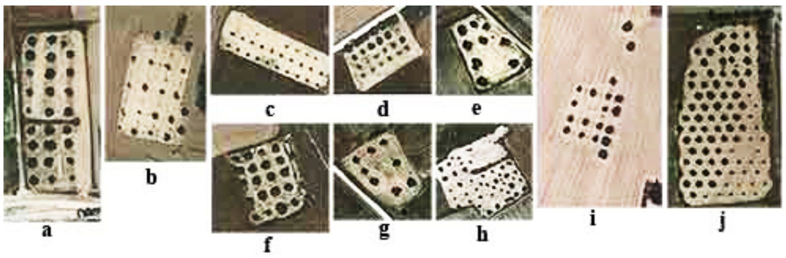


Figure 16: Examples of cropped images with various size, shape and complexity

3.1.1 Canny

Initial results of the experiment show that Canny with default parameter sets detected 1565 obstacles out of 1602. 1282 obstacles identified correctly (80.02%), and 323 obstacles missed which represents an FPE (false positive error) of 25.16% inside the fields. The FNE (false negative error) result for Canny is approximately 18.07% which indicates 283 wrong obstacle detections on the fields. Ten sample fields are presented in Figure 17 to demonstrate the correct detections, FPE and FNE of the Canny. Figure 17-a.1 to 14-j.1 shows binary images of Canny's edge detection results and Figure 17-a.2 to 14-j.2 shows final detection's graphical annotations on the actual image of the field.

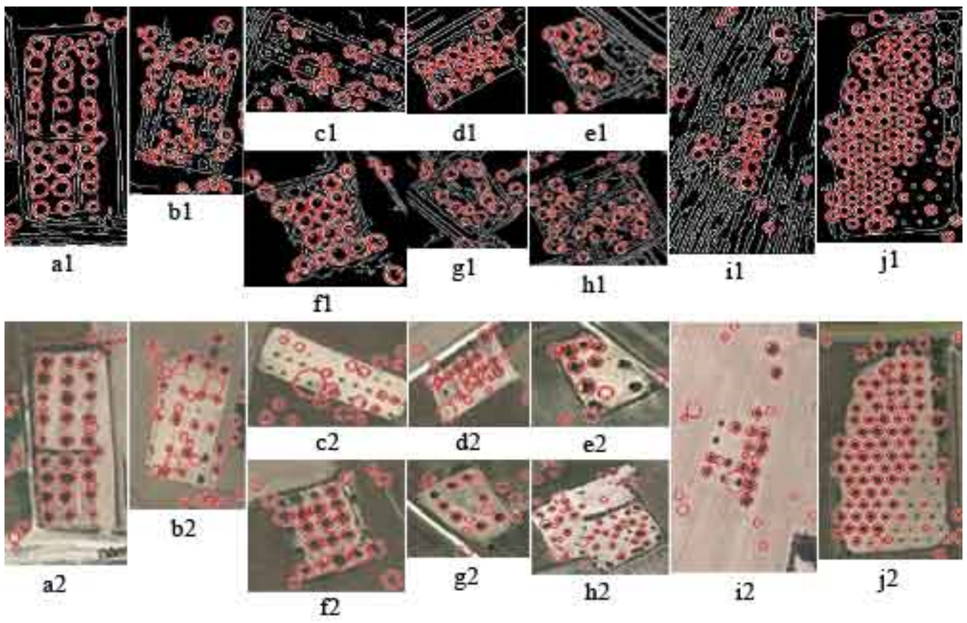


Figure 17: Canny edge and obstacle detection results

3.1.2 Prewitt

Prewitt identified 1616 obstacles in total with the default parameter sets. Out of 1602 obstacles in all the fields, 1333 obstacles (83.2%) were detected correctly. FNE result of Prewitt is 16.8% which denotes 283 missed obstacles inside the fields. The number of wrong detections is equal to 155 which represent an FPE of 9.67%. Figure 18-a1 to 15j1 shows the detected edges of the obstacles on the binary images of the fields. Figure 18-a2 to 15-j2 shows final detection's graphical annotations on the actual image of the fields.

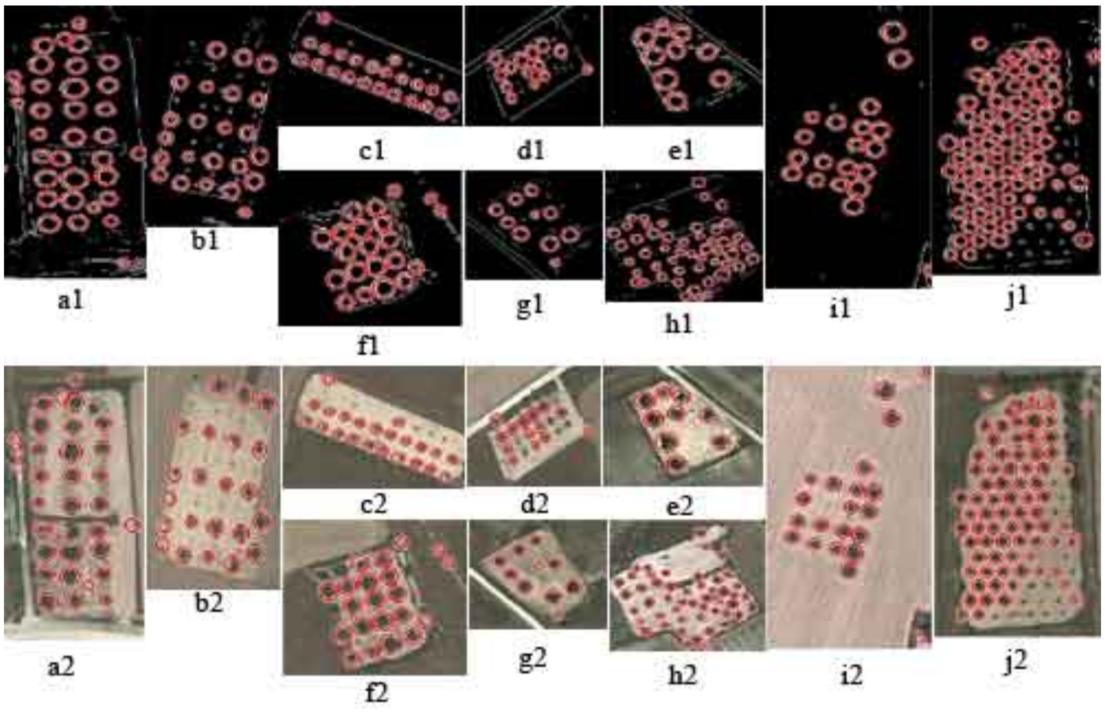


Figure 18: Canny edge and obstacle detection results

3.1.3 Roberts

Results show that Roberts with default parameter set identified 1044 correct obstacles out of 1602, which is equal to 65.16% accuracy in detection. The FPE result for Roberts is 34.76% which indicates 557 missed obstacles. With the total of 105 wrong obstacle detections, Roberts FNE is 9.13%. Figure 19-a1 to 16-j1 shows binary images of Robert's edge detection results and Figure 19-a2 to 16-j2 shows final graphical annotations on the image of the field.

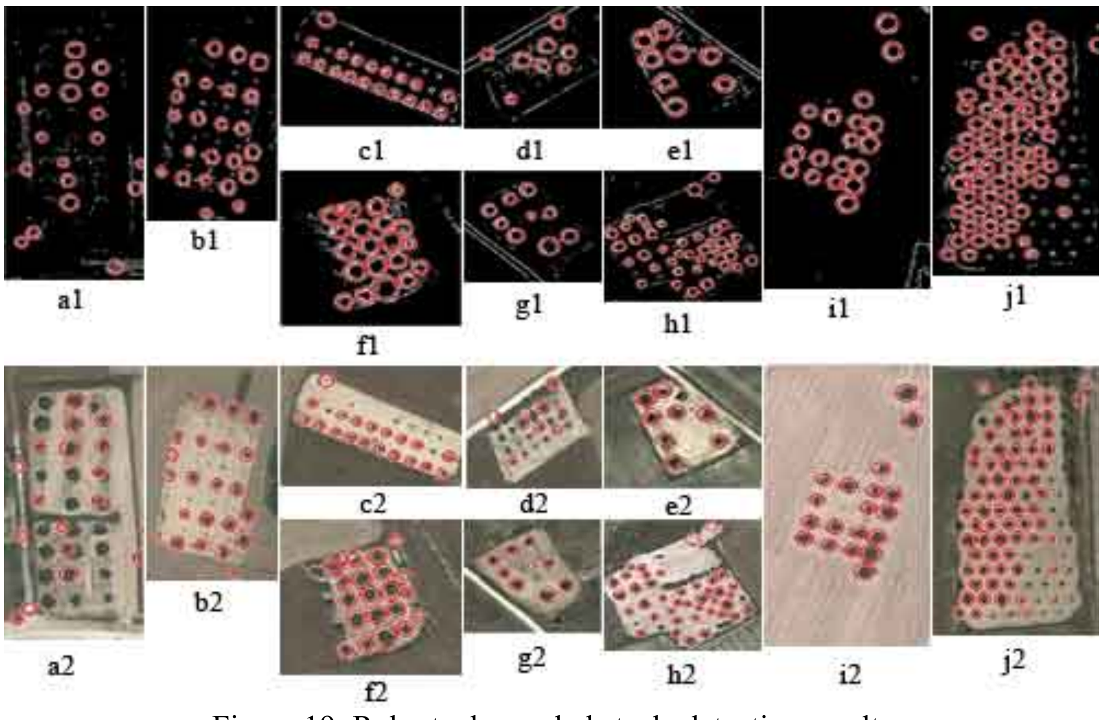


Figure 19: Robert edge and obstacle detection results

3.1.4 Sobel

Sobel with default parameter sets identified 1350 correct obstacles in total, which is equal to 84.26% accuracy in detection. Sobel's PFE result is 15.73% which denotes 252 missed obstacles on the fields. 106 obstacles incorrectly detected by Sobel that results in an FNE of 7.88%. Figure 20-a1 to 17j1 represents the detected edges and obstacles on the binary images of the fields. Figure 20-a2 to 17-j2 shows the final graphical annotation of obstacles on the actual image of the field.

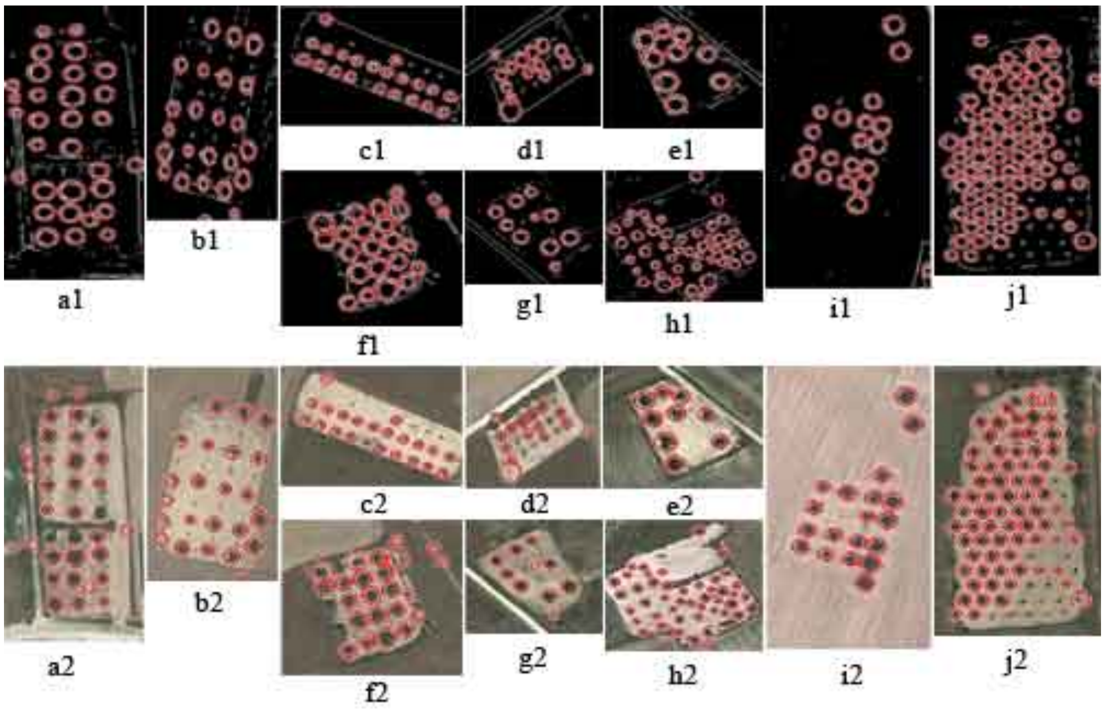


Figure 20: Samples results of Sobel's edge and obstacle detection

3.1.5 Detection Improvement Results

In the second phase of the obstacle detection process, a great potential was discovered to improve the obstacle detection. This discovery led us to develop three experimental methods. Up to this point, Sobel was the nominated edge detection method due to its higher performance in detection comparing to Canny, Roberts, and Prewitt. This decision was made using the default parameter sets of the edge detection techniques, however, with customized parameters better results achieved as demonstrated in the following sections.

3.1.5.1 T-Range Results

As stated before, Sobel could perform better in detecting the obstacles after tuning threshold finely. After employing the T-Range algorithm, we achieved an improvement of 21.43% in detection rates by Sobel. Figure 21 illustrates the detection improvements using Sobel with T-Ranged technique. The chart clearly shows enhanced detections (green line) on most of the fields comparing to the initial detection results (blue line). However, in some cases like field 4 or 32, the primary method still performs better which led us to examine for further improvements.

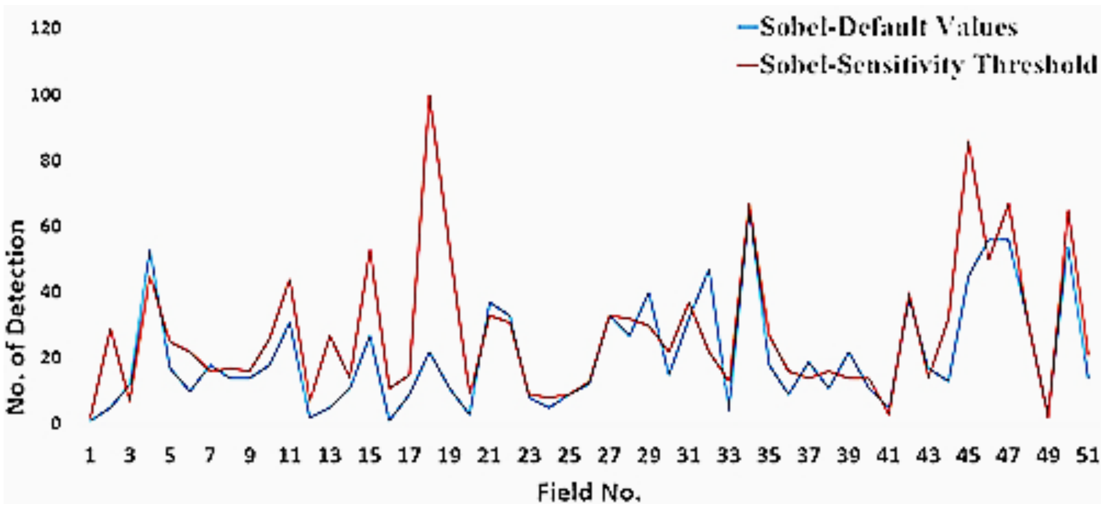


Figure 21: T-Range outcomes on Sobel comparing to default threshold

3.1.5.2 Max of All Results

The second method proposed to improve the obstacle detection is Max-of-All. Using this method, we achieved 16.67% increase in the obstacle detection in overall. The x-axis in Figure 22 indicates the preferred method to Sobel in the case of improvement. These methods which include Canny, Prewitt, Roberts, and Sobel, are abbreviated to C, P, R and S accordingly. As expected in many cases Canny and Prewitt are preferred to Sobel. The red line in Figure 22 represents the Max-of-All method's results in comparison the initial detections of Sobel with default parameter sets. Although less overall improvement achieved (16.67%) comparing to the results of T-Range (21.43%), but we have noticed that in some cases like in field 29 and 39 (Figure 20) Max-of-All method still performs better. This inconsistency in achieving the highest detection result in all fields led us to develop the DLC approach.

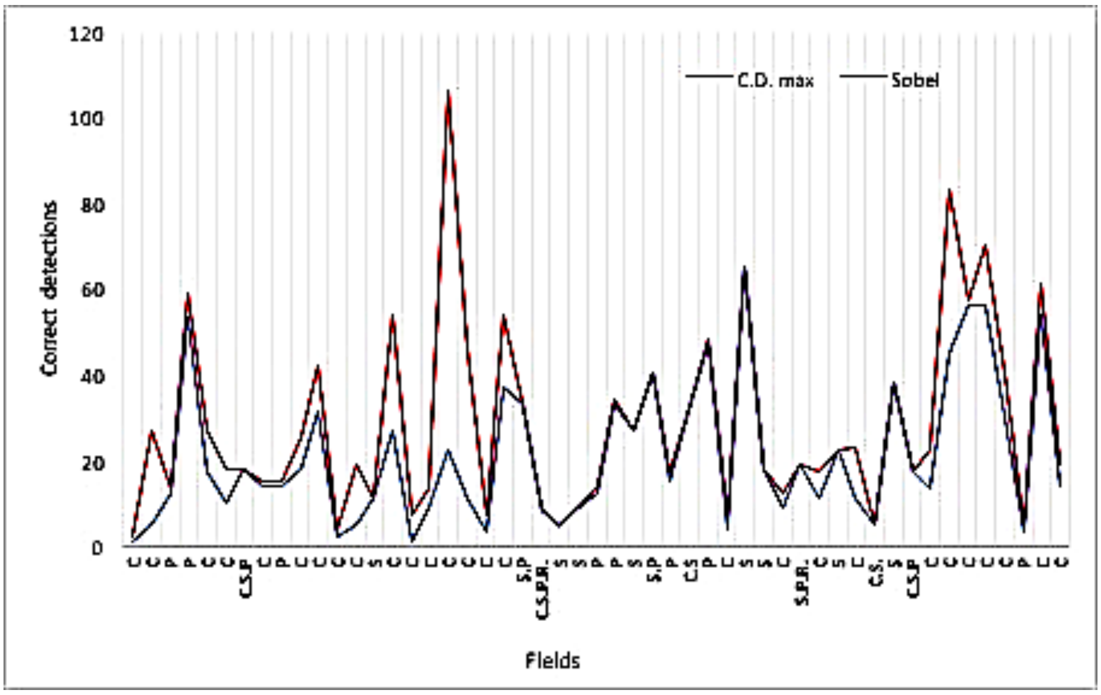


Figure 22: Max-of-All method comparing to Sobel with default parameter sets

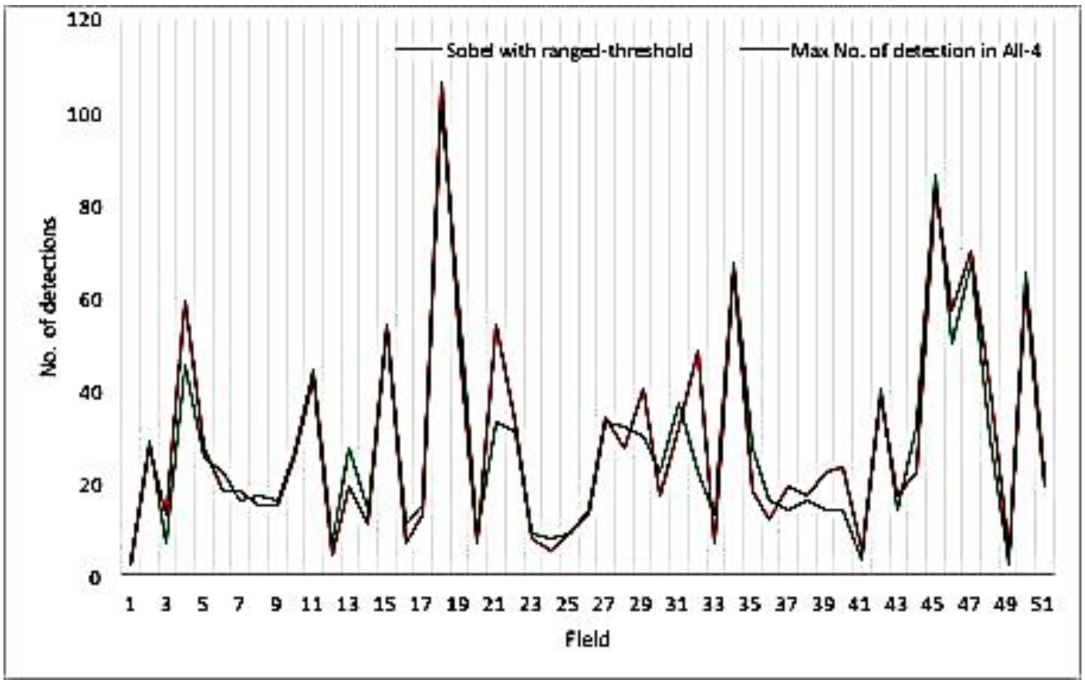


Figure 23: Comparison between T-Range and Max-of-All

3.1.5.3 DLC Results

Double Layered Check (DLC) approach works with merging T-Range and Max-of-All methods. DLC applies both methods at the same time on each field, compares the results and chooses the best among them. By using DLC, either of the T-Range or Max-of-All methods can act as a complementary method to cover each other. Consequently, the obstacle detection rate improved up to 45.5% in overall comparing to the Sobel detection with default parameter sets. Results in Figure 24 indicate that the detection has either improved or remained unchanged in most of the fields. However, in the previous methods, we had many cases with reduced percentage in detection. Appendix I lists counts of detected obstacles and error counts after using DLC method.

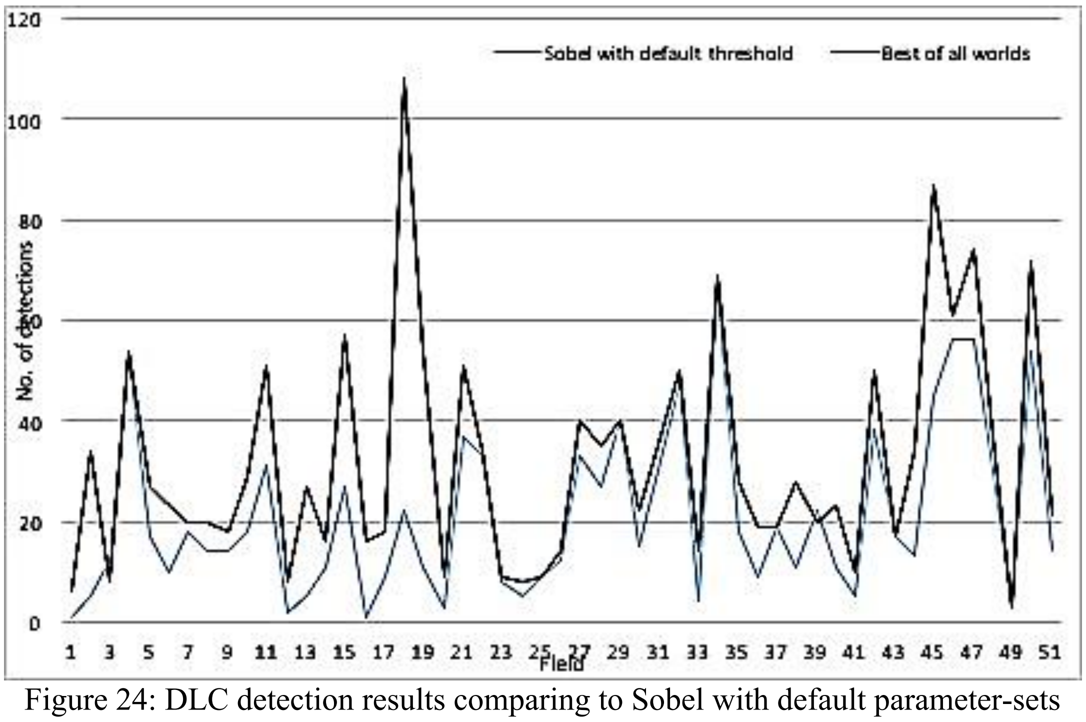


Figure 24: DLC detection results comparing to Sobel with default parameter-sets

3.1.6 FPE and FNE Reduction

DLC increased the number of obstacle detections which could include the actual obstacles and the wrong detection at the same time. Using DLC we have maximized the number of the hits on the field to detect obstacles. Having more hits on the field's area increases the chance of detecting any real obstacle. This would result in less FPE percentage comparing to the other methods. However, by increasing the number of detections, we potentially increase FNE or wrong detections. To increase the detection precision, FNE must be minimized, same as FPE. To do so, we have developed a classification algorithm to differentiate correct and incorrect detection using the grey-level intensity threshold value of 127.37. In Figure 25 correct detections are presented with yellow dots and wrong detections are shown with red dots.

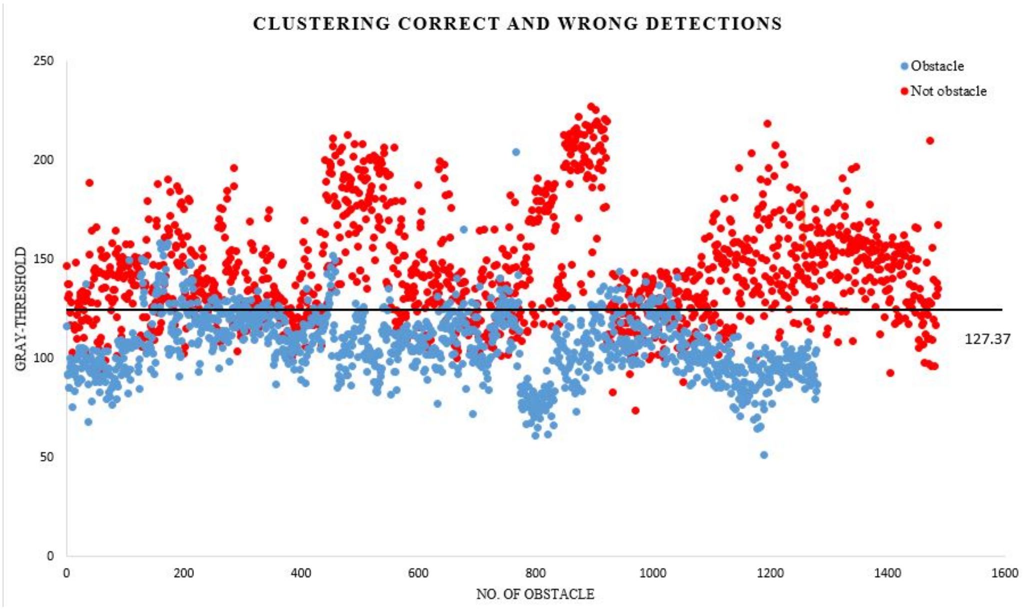


Figure 25: Classes of correct and incorrect detections

As the chart in Figure 26 illustrates, DLC improved the FNE by 80% on overall. This improvement resulted in a higher accuracy in the detection by eliminating the majority of the wrong detections.

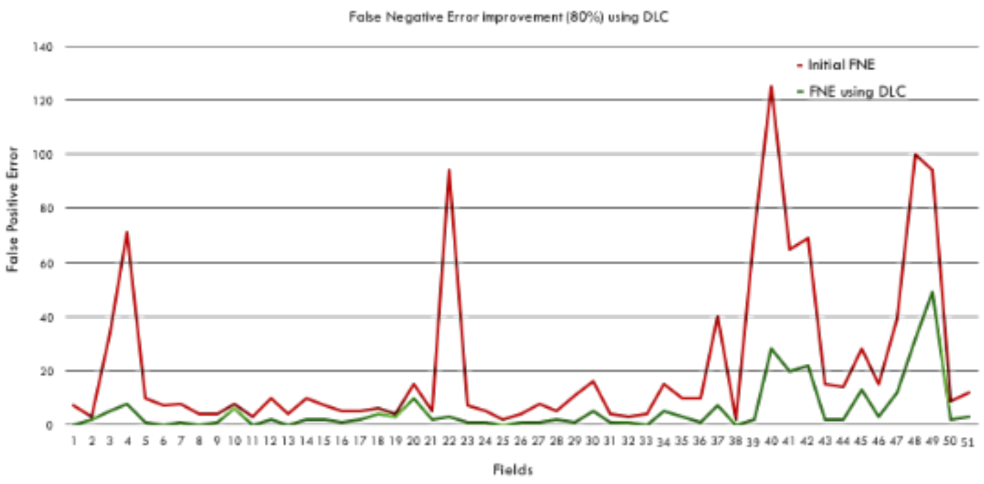


Figure 26: FNE improvements using DLC

Figure 27 shows some sample results of the FNE improvements. The blue circles on these images represent correct detections, and the red circles show suppressed FNEs. Red circles were counted as correct detections before implementation of the DLC. However, the wrong detections (red circles) eliminated by using DLC, which results in improvements on FNE.

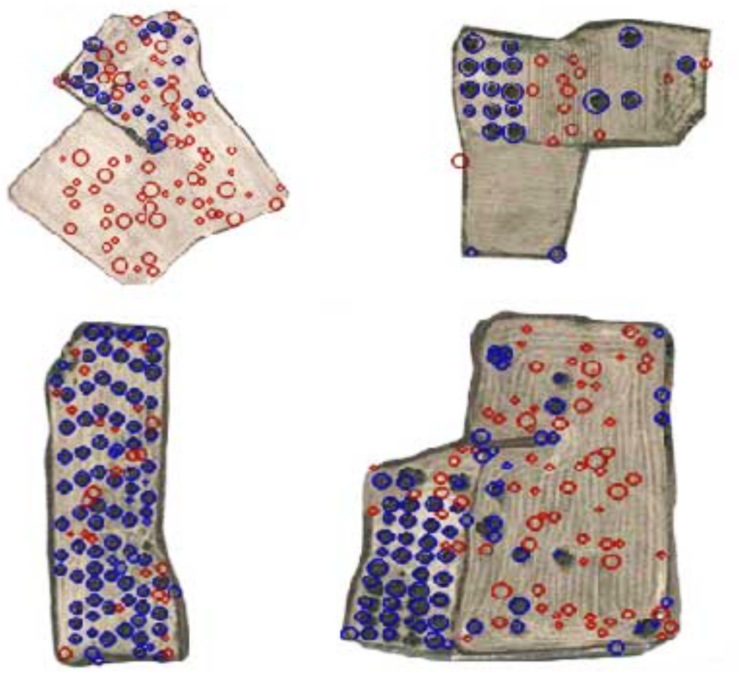


Figure 27: Suppressed FNE using DLC

3.1.7 Locating Accuracy Check

The proposed method provides significant precision for detection and locating of obstacle and producing trajectory points within agricultural fields. This precision, however, highly depends on Google Maps' accuracy in mapping GPS coordinates to pixels in the images. To evaluate the accuracy of positioning obstacles, the location of a detected obstacle is compared with Google Map’s search result as shown in Figure 28. Following detection of an obstacle, the centre pixel's coordinates of the obstacle maps to GPS coordinates (right image in Figure 28). Then the calculated GPS coordinates are searched by Google Maps, and the result is shown in the left image of Figure 28. The search result indicates that Google’s pin (on the left picture) points exactly to the centre of the obstacle detected by the proposed system.

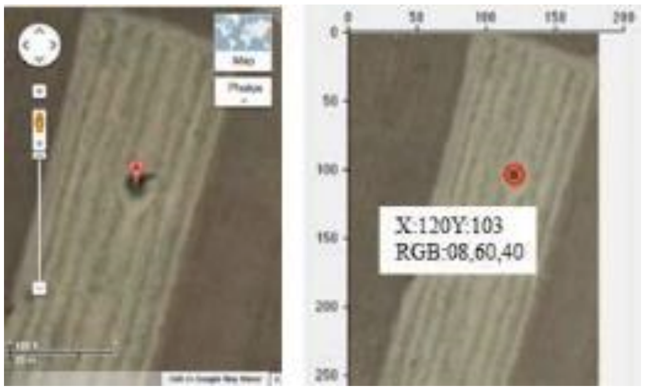


Figure 28: Developed system's obstacle positioning accuracy checked against Google Maps at Lat: 35.218108 and Long: 33.572233

Another advantage of the proposed method is the ability to calculate each detected obstacle's dimensions with significant accuracy. This is important as obstacles appear in different sizes on the field and their dimensions are required to avoid any collision. To demonstrate this, we have considered a sample tree and had its diameter measured in three different ways.

First using the proposed method, and the result is 10.14 meter which is equal to 20.74 pixels on the image as shown in Figure 29a. Second is using Google Maps' scale bar which is almost equal to 10 meters as shown in Figure 29b. The last one was by measuring at the spot which was 11.1 meters (Figure 29c).

Comparing the proposed method's result with Google Maps' scale bar we can see that the difference is as small as 14 cm which denotes a very low percentage of error. However, there is a significant difference between our result and the measured diameter in the spot. The reason is that images provided by Google are usually two to three years old hence not being up to date and the tree's growth not being captured.

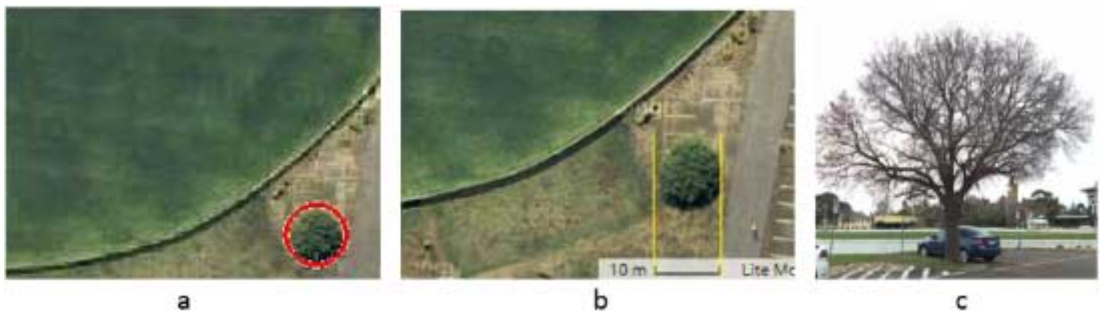


Figure 29: Detected obstacle dimensions calculated by the developed method compared with google and its actual size (from left to right)

3.2 Trajectory Points Generation

As stated before the intention of this study is to produce GPS trajectory points to be followed by GPS-enabled agricultural vehicles. To demonstrate the functionality and compatibility of the proposed system with any path planning algorithm, we have integrated the system into a path planning algorithm. A simple path planning algorithm which produces parallel paths in agricultural fields is selected, and an example of this kind is presented in Figure 30.

Unlike most of the path planning algorithms which draw lines on the image, the developed system produces sequences of points. These points are initially generated based on the pixels in the picture. These points must not overlap any obstacles and must be in the field area.

This results in a sequence of points with some gaps in between as shown in Figure 31. These gaps divide each track into segments and are either due to an obstacle or un-straight borders of the field. The selected path planning algorithm produces tracking paths parallel to the longest straight edge of the field.

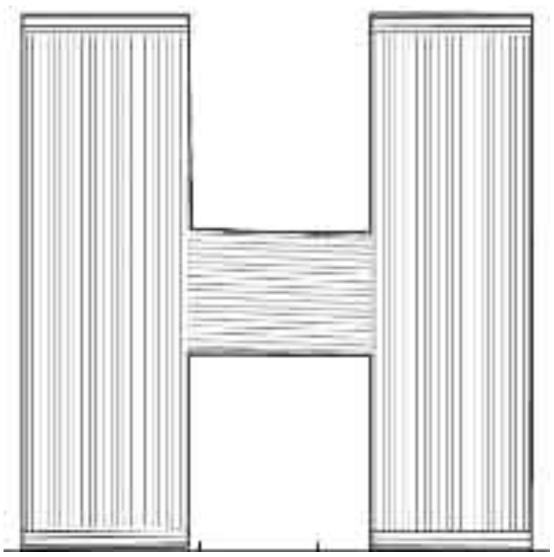


Figure 30: An example of parallel path planning in an agricultural field

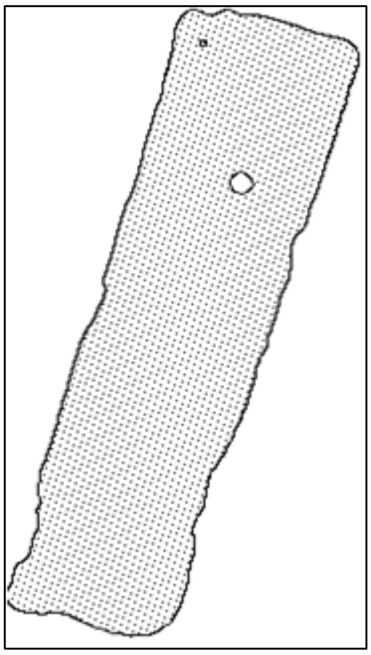


Figure 31: Trajectory reference points plotted on the field`s image

This edge is detected using Standards Hough Transform on the binary image and presented with a bold line in Figure 32. The rests of the tracks on the field are parallel to this line. Generated sequence of trajectory points in pixel coordinates are later mapped into GPS coordinates using the Pixel-to-GPS conversion algorithm.

The produced path which includes tracks, segments and points saved to be used by tracking devices.

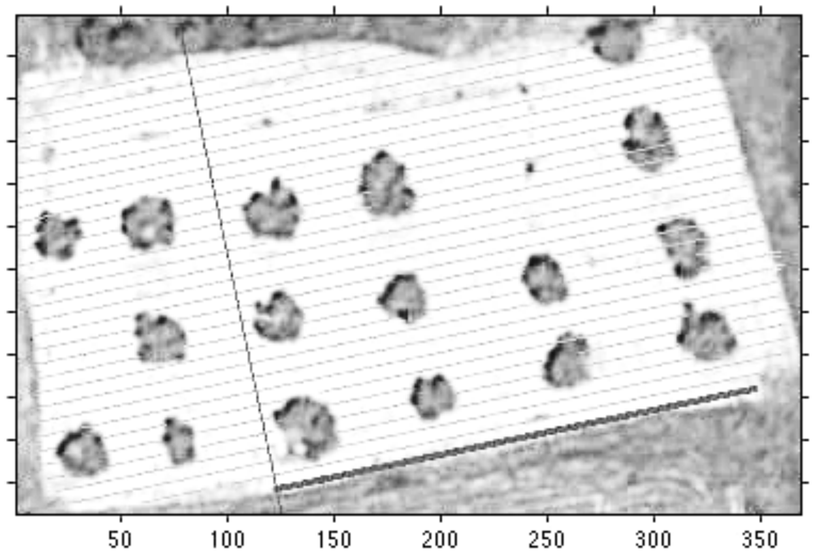


Figure 32: Demonstrating the longest straight edge of the field

Table 2 represent some statistic details of the path in Figure 31. As we can see from the table, the generated path contains 2428 points which are all mapped into GPS points. It also shows that this particular path contains 28 tracks which are formed of 35 segments. The total number of the obstacle detected in the field is 2, and the length of the path is 5698.46 meters excluding required turns.

Table 2: Statistics of the trajectory path presented in Figure 28

Total GPS Points	2428
Total Tracks	28
Total Segments	35
No of Obstacle	1
Total Path Distance (m)	5698.465393

Table 3 presents details of each track. As shown the table track one is divided into two segments. This track is the leftmost track on Figure 31 and split into two due to an un-straight edge of the field. Moreover, the total number of points in this track is 35. Tracks 16, 17 and 18 also divided into two segments due to the same obstacle in the field.

Table 3: Track and segment details of the example in Figure 28

Track	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	
Segments	2	2	1	1	1	1	1	1	1	1	1	1	1	1	
Points	35	84	93	92	93	93	94	94	93	93	94	94	94	94	
Track	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**Total**
Segments	1	2	2	2	1	1	1	1	1	1	1	2	1	2	35
Points	94	92	90	91	93	94	94	94	93	93	92	78	61	29	2428

Depending on the agricultural product on a field, farming might require different row spacing. For example, one type product may require 1m distance between rows and another one may require 3m distance in between each row. Row spacing (d_{ms}) is an input parameter of the proposed system in generating trajectory points. Demonstration of the trajectory generation algorithm is tested for the effect of row spacing with three different row distances, (d_{ms} = {2, 4, 6} meters), applied on the same set of fields. The results presented in Table 4 shows that the computational time is in the range of 2.04 to 25.95 seconds for the row distance of 2m, 1.35 to 14.47 seconds for the row distance of 4m and 1.15 to 10.7 seconds for 6m row distance.

Table 4: Computation time and details for various row distances

Field	Points per path											
	2m				4m				6m			
	Time	Number of Track.	Number of Segments	Total Ref. points	Time	Number of Track.	Number of Segments	Total Ref. points	Time	Number of Track.	Number of Segments	Total Ref. points
1	**2.04**	28	35	2428	**1.35**	14	16	1213	**1.15**	10	13	808
2	**2.22**	64	72	2601	**1.58**	32	37	1325	**1.34**	22	26	877
3	**23.04**	106	128	37553	**13.54**	53	40	18776	**10.3**	36	29	12480
4	**25.95**	107	130	46027	**14.47**	54	69	23085	**10.7**	36	47	15436
5	**4.13**	20	62	5576	**2.96**	10	39	2684	**2.29**	7	15	1930
6	**5.51**	33	93	8422	**3.55**	17	49	4226	**2.86**	11	35	2807
7	**12.19**	64	531	21140	**7.29**	32	286	9648	**5.66**	22	193	6438

3.3 Semantic Annotation

Following the detection and positioning of the obstacles, results are semantically annotated. The annotation is done in two different ways, graphical and textual. After computing location on the image and the radius for any detected obstacle, it is graphically annotated on the picture with circles.Figure 33 shows an example of the graphical annotation. Also, all the detections are semantically annotated and captured in RDF/XML format. This figure also represents a portion of an RDF/XML file generated during the semantic annotation phase of the field. This annotation refers to the obstacle pointed with a white arrow in the figure. As shown, it is the first detected obstacle on the field with X, Y coordinates of 72.441 and 16.951 on the image. Its radius is 2.52 meters and geographically located on a point with the latitude of 35.0755749 and the longitude of 33.5309169 degrees.

```xml
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:obt="http://www.DLC.org/Obstacle#"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:gn="http://www.geonames.org/ontology#"
xmlns:wgs84_pos="http://www.w3.org/2003/01/geo/wgs84_pos#"
xmlns:item="http://purl.obolibrary.org/obo/PO_0000003/hasNarrowSynonym#"
xmlns:svg="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd/circle#">
<rdf:Description rdf:about="http://www.DLC.org/Obstacle#"/>
<item:tree>
        <dc:type rdf:resource="http://purl.org/dc/dcmitype/Image"/>
        <obt:no rdf:datatype="http://www.w3.org/2001/XMLSchema#int"> 1 </obt:no>
        <wgs84_pos:lat>35.07557</wgs84_pos:lat>
        <wgs84_pos:long>33.53091</wgs84_pos:long>
        <svg:cx>72.4411204</svg:cx>
        <svg:cy>16.9516487</svg:cy>
        <svg:r>2.5275816</svg:r>
</item:tree>
</rdf:RDF>
```

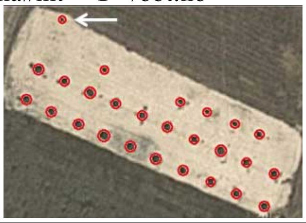


Figure 33: Graphical and semantic annotation of obstacles,

Table 5 represents the details of the first detected obstacle on the target field which is shown in Figure 33. The first column in the table represents the number of the obstacle. This numbering is based on the order of the detection, for example, number 1 means the first detected obstacle. Columns two and three shows pixel coordinates of the obstacle’s center. Columns four and five shows obstacle`s radius in pixels on the image and meters on the ground. The last two columns represent the mapping of the obstacle`s pixel location to latitude and longitude coordinates. Presented data are used to annotate obstacles semantically and generate the RDF/XML file.

Table 5: Obstacle detection and positioning data included in semantic annotation

Obstacle No.	X coord. (pix)	Y coord. (pix)	Radius (pix)	Radius (meters)	Lat (degrees)	Long (degrees)
1	72.44112	16.951649	4.23264	2.52758	35.075575	33.530917
2	137.15192	118.71556	5.56901	3.32561	35.075351	33.531091
3	105.87311	105.21672	6.53522	3.9026	35.075382	33.531008
4	43.966544	77.133502	6.13732	3.66499	35.075443	33.530842
5	75.269715	91.760577	5.4431	3.25042	35.07541	33.530925
6	59.345833	128.49931	5.6869	3.39602	35.075331	33.530882
7	29.058078	113.67778	5.90732	3.52764	35.075362	33.530801
8	123.36508	77.755344	4.55023	2.71724	35.075441	33.531053
9	214.81828	116.68807	4.99847	2.98491	35.075355	33.5313
10	91.965574	143.67425	5.96041	3.55935	35.075296	33.53097
11	154.61818	170.45598	6.11502	3.65167	35.075239	33.531139
12	123.00124	157.28777	6.76373	4.03906	35.075268	33.531053
13	169.02055	132.87294	5.24449	3.13182	35.07532	33.531177
14	185.39867	185.43027	6.17136	3.68532	35.075206	33.53122
15	231.4479	161.99766	5.33967	3.18866	35.075257	33.531343
16	199.4047	146.87946	5.58356	3.3343	35.07529	33.531257
17	263.72562	175.68854	6.53168	3.90048	35.075226	33.531432
18	247.3087	128.95209	4.53204	2.70637	35.075329	33.531386
19	294.86756	189.76737	5.3989	3.22403	35.075195	33.531515
20	310.64187	155.64186	5.26233	3.14248	35.07527	33.531558
21	280.90302	143.26558	4.89839	2.92514	35.075298	33.531477
22	218.10521	199.43954	5.79499	3.46056	35.075175	33.531308
23	251.76515	213.31047	5.34811	3.1937	35.075145	33.531399
24	282.55477	227.76111	5.57833	3.33118	35.075112	33.531483
25	328.28792	203.63418	5.69524	3.40099	35.075164	33.531603
26	316.49374	242.74324	5.81192	3.47067	35.075079	33.531571

In addition to the annotation of obstacles, generated trajectory points are semantically annotated and captured in RDF/XML format. During the generation and positioning of the trajectory reference points, the results are saved into an array. Rows in the array indicate the sequence number of the points in each track and columns represents the location of the points. These points are presented with two pairs of values. The first pair is the pixel location of the point on the image and the second pair represents latitude and longitude of the point on the ground. Part of the results array is shown in Table 6, and zero values in the rows indicate the end of the track.

Table 6: Array of trajectory points in pairs of *x*, *y* and *lat, long*.

Trajectory point	Track 1				Track 2			
	x	y	*lat*	*long*	*x*	*y*	*lat*	*long*
1	24	2.0414	35.16743	33.51162	16	11.555	35.16739	33.51158
2	25	2.1113	35.16743	33.51163	17	11.624	35.16739	33.51158
3	26	2.1812	35.16743	33.51163	124	19.099	35.16735	33.51216
4	27	2.251	35.16743	33.51164	125	19.169	35.16735	33.51216
5	28	2.3209	35.16743	33.51164	126	19.238	35.16735	33.51217
6	29	2.3907	35.16743	33.51165	127	19.308	35.16735	33.51217
7	30	2.4606	35.16743	33.51165	128	19.378	35.16735	33.51218
8	31	2.5304	35.16743	33.51166	139	20.147	35.16735	33.51224
9	32	2.6003	35.16743	33.51166	140	20.216	35.16735	33.51224
10	33	2.6701	35.16743	33.51167	141	20.286	35.16735	33.51225
11	34	2.74	35.16742	33.51167	142	20.356	35.16735	33.51225
12	0	0	0	0	143	20.426	35.16735	33.51226
13	0	0	0	0	144	20.496	35.16735	33.51226
14	0	0	0	0	156	21.334	35.16734	33.51233
15	0	0	0	0	157	21.404	35.16734	33.51233
16	0	0	0	0	158	21.474	35.16734	33.51234

The data presented in Table 6 are used to annotate trajectory points using the data structure explained earlier. The "Path" is marked up as the root, and the "Track" as a child node. Each child node consists of two elements, "Number" and “gpsPoint”, and each would have their attributes. A section of the generated RDF/XML file shown in Figure 34 includes the Path and Track number 1. The first trajectory point of the track is located at latitude 35.075595 and longitude 33.530909 coordinates on the ground. The same point on the image is located on pixel coordinates of X=69 and Y=7.7293.

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:trj="http://www.DLC.org/trajectory#"
xmlns:item="http://www.opengis.net/gml/"
xmlns:svg="http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd/circle#">
<rdf:Description rdf:about="http://www.DLC.org/Trajectory#"/>
<item:point>
        <dc:type rdf:resource="http://purl.org/dc/dcmitype/Image"/>
        <item:pos> 35.075595, 33.530909</item:pos>
        <item:lat>35.075595</item:lat>
        <item:long>33.530909</item:long>
        <trj:no rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
                        1</trj:no>
        <trj:track rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
                        1</trj:track>
        <svg:cx>69</svg:cx>
        <svg:cy>7.7293</svg:cy>
</item:point>
<item:point>
        <dc:type rdf:resource="http://purl.org/dc/dcmitype/Image"/>
        <item:pos> 35.075593,33.530914</item:pos>
        <item:lat>35.075593</item:lat>
        <item:long>33.530914</item:long>
        <trj:no rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
                        1</trj:no>
        <trj:track rdf:datatype="http://www.w3.org/2001/XMLSchema#int">
                        1</trj:track>
        <svg:cx>71</svg:cx>
        <svg:cy>8.61913</svg:cy>
</item:point>
</rdf:RDF>
```
Figure 34:Section of the annotation file in RDF/XML syntax format

Chapter 4

DISCUSSION

The proposed method requires only a GPS point inside the target field for initialization, and provides the least complexity for the end user. The developed system is capable of detecting obstacles and generating trajectory points. Also, this system can be used by any path planning algorithm. The output is a path consisting of points sequences in the form of GPS points, making the path traceable with any GPS-enabled autonomous vehicle. Google Map's free API used in this study which provides low-quality images. However, experimental results show that the developed system provides significant precision in detecting and positioning of obstacles and in generating trajectory points.

4.1 Effect of Zoom Level and Noise Reduction Algorithm

Zoom level plays a major role in the detection phase since higher accuracy could achieve at higher zoom levels. The free API provides images with dimensions of 640 by 640 pixels. These dimensions limit us to choose the highest zoom level (Z = 18) to fit the entire field. Consequently, low-quality images are captured. However, using the Business API provided by Google the image quality can improve significantly.

The Business API provides images with dimensions of 2048x2048 pixels which can fit the same fields with higher zoom level (Z= 20). This improves the picture quality by four which would result in considerable improvements in the detection of obstacles. As shown in Figure 35 the same field with Z= 19 has twice visibility and quality comparing to the same field with Z = 18 and this visibility and quality increases to four times with Z = 20.

The developed system shows sufficient performance to detect and extract the target field, however, under some circumstances; it struggles to find the target field and fails to detect boundaries because the edges of the field are not clearly visible for the detection methods which are used to identify the field. The issue arises when the target field is in the same contrast and brightness of the surrounding areas.

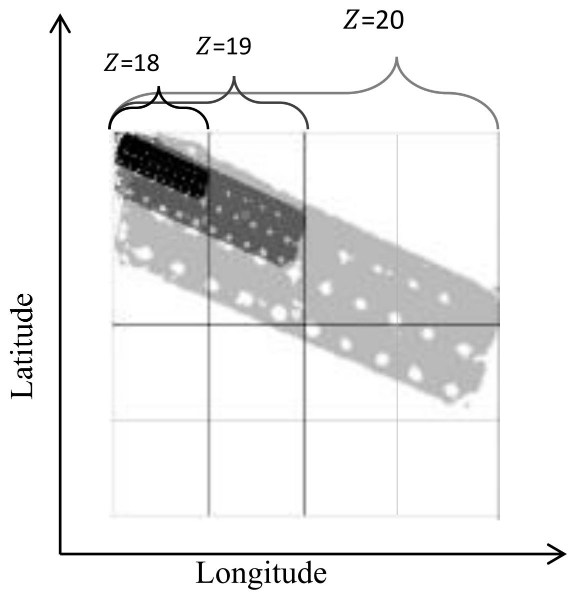


Figure 35: Correlation between zoom level on the map and the image quality

Using noise reduction algorithms removes some existing obstacles in the field like rocks, bushes, old boughs on the ground and etc. which may cause dangerous and harmful operation of vehicles. Although noise reduction algorithms remove these objects from obstacle detection image, they are still visible on the annotation image to the human eyes. Therefore, it is possible to process them during a user intervention step before the path planning, and rely on user intervention to clear them from the list of obstacles.

4.2 Effect of Edge Detection Algorithms

Initially, Sobel was selected as the primary edge detection method for obstacle detection due to its higher performance. Table 7 represents the comparison results between Sobel, Canny, Prewitt and Roberts. This comparison was made using the default parameter sets in the beginning. Table 7 shows that Sobel had better results in correct detection and false negative error (wrong detection).

In regards to the false positive error, although Roberts scored better result comparing to Sobel, the difference is not significant enough to consider. All initial experiments with Sobel and other edge detection methods were conducted using default threshold value of 0.27. However, referring to Figure 11, our observations indicated that Sobel behaves differently with various threshold values while detecting obstacles.

Table 7: Initial comparison of edge detection methods

	Detection			
Method	Correct	FPE	FNE	%
Canny	1282	283	323	80.02
Prewitt	1044	155	283	65.17
Roberts	1333	105	557	83.21
Sobel	1350	106	253	84.27

Figure 36 shows results of applying the entire threshold range from zero to one on Sobel. This figure only shows a portion of the results as the number of detections seeks to zero while reaching large threshold values. The chart indicates that Sobel reaches the maximum number of detections with a threshold value of 0.059 on average.

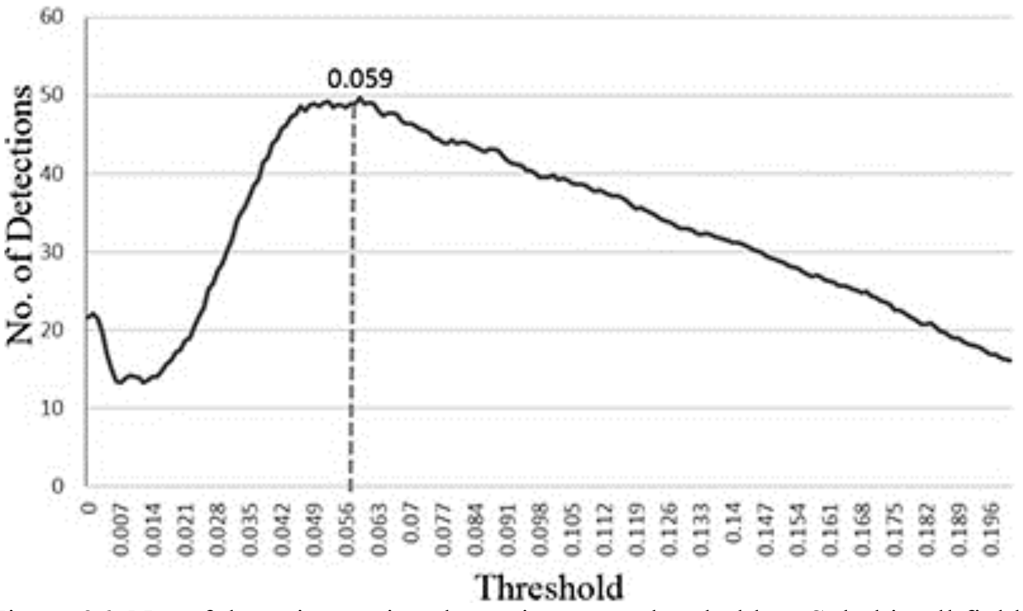


Figure 36: No. of detections using the entire range threshold on Sobel in all fields

Our experiments results show that T-range method improved correct detections up to 21.43% in overall. The improvement of detections is due to the fact that by applying the complete sensitivity-threshold range on Sobel we have increased the possibility of detecting the correct obstacle. The reason is that using T-range and finding the best threshold value will result in the algorithm to find more edges which are like obstacles.

However, these detections may include correct and incorrect obstacles. The verification of correct and incorrect detections is done using another algorithm. This algorithm uses gray-level intensity threshold to differentiate correct detections from the incorrect ones and discard wrong detections. Also, false positive error improved by 20.83% which means fewer obstacles missed during the detection phase.

Table 8: Detection improvements from Sobel with static threshold value to Sobel with dynamic threshold

Field	Sobel (Default Thr.)	Sobel (dynamic Thr.)	**Field**	Sobel (Default Thr.)	Sobel (Dynamic Thr.)
1	1	**2**	27	33	**33**
2	5	**29**	28	27	**32**
3	12	7	29	40	30
4	53	45	30	15	**22**
5	17	**25**	31	32	**37**
6	10	**22**	32	47	22
7	18	16	33	4	**13**
8	14	**17**	34	65	**67**
9	14	**16**	35	18	**27**
10	18	**26**	36	9	**16**
11	31	**44**	37	19	14
12	2	**7**	38	11	**16**
13	5	**27**	39	22	14
14	11	**14**	40	11	**14**
15	27	**53**	41	5	3
16	1	**11**	42	38	**40**
17	9	**15**	43	17	14
18	22	**100**	44	13	**32**
19	11	**54**	45	45	**86**
20	3	**9**	46	56	50
21	37	33	47	56	**67**
22	33	31	48	30	**30**
23	8	**9**	49	3	2
24	5	**8**	50	54	**65**
25	9	**9**	51	14	**21**
26	12	**13**			

After considering Sobel with the dynamic threshold (T-range), we have applied the Max-of-All method on all fields and compared the counts of obstacle-candidate detections in Table 9. Values in bold on each row shows improvements made by the Max-of-All method. By comparing the results in Table 8 and Table 9, we can see that the Max-of-All made less enhancement (values marked by ^{a}) (16.67%) compared to the T-range (21.43%), however, its enhancement is independent of Max-of-All since it provides enhancement on a different set of images as seen in Table 9.

Table 9: Correct detections by Sobel with default-threshold and Max-of-All methods

Field	Default Thr.	Max-of-All	**Field**	Default Thr.	Max-of All
1	1	**2**	27	33	**34**
2	5	**27**	28	27	27
3	12	**13** [a]	29	40	40
4	53	**59** [a]	30	15	**17**
5	17	**27**	31	32	32
6	10	**18**	32	47	**48** [a]
7	18	18	33	4	**7**
8	14	**15**	34	65	65
9	14	**15**	35	18	18
10	18	**25**	36	9	**12**
11	31	**42**	37	19	19
12	2	**4**	38	11	**17**
13	5	**19**	39	22	22
14	11	11	40	11	**23**
15	27	**54**	41	5	5
16	1	**7**	42	38	38
17	9	**13**	43	17	17
18	22	**106**	44	13	**22**
19	11	**46**	45	45	**83**
20	3	**7**	46	56	**57** [a]
21	37	**54** [a]	47	56	**70**
22	33	33	48	30	**40**
23	8	8	49	3	**5** [a]
24	5	5	50	54	**61**
25	9	9	51	14	**19**
26	12	**14**			

The Max-of-All method was successful in improving the detection process in some of the fields which T-range failed to improve. These fields are marked by [a] in Table 9. The T-range method applies over complete threshold range only on Sobel. However, DLC applies the entire threshold range on all four edge detection methods. Instead of using T-range, application of DLC combines the partial improvements together with other edge detection algorithms. Only the meaningful parts of the results are presented in Figure 34 because from the threshold value 0.14 onward the number of detections seeks to zero. From the chart, we can immediately notice that

Roberts and Prewitt reach their highest number of detections with almost the same threshold value of Sobel. However, Canny reaches its maximum number of detections with a higher threshold value.

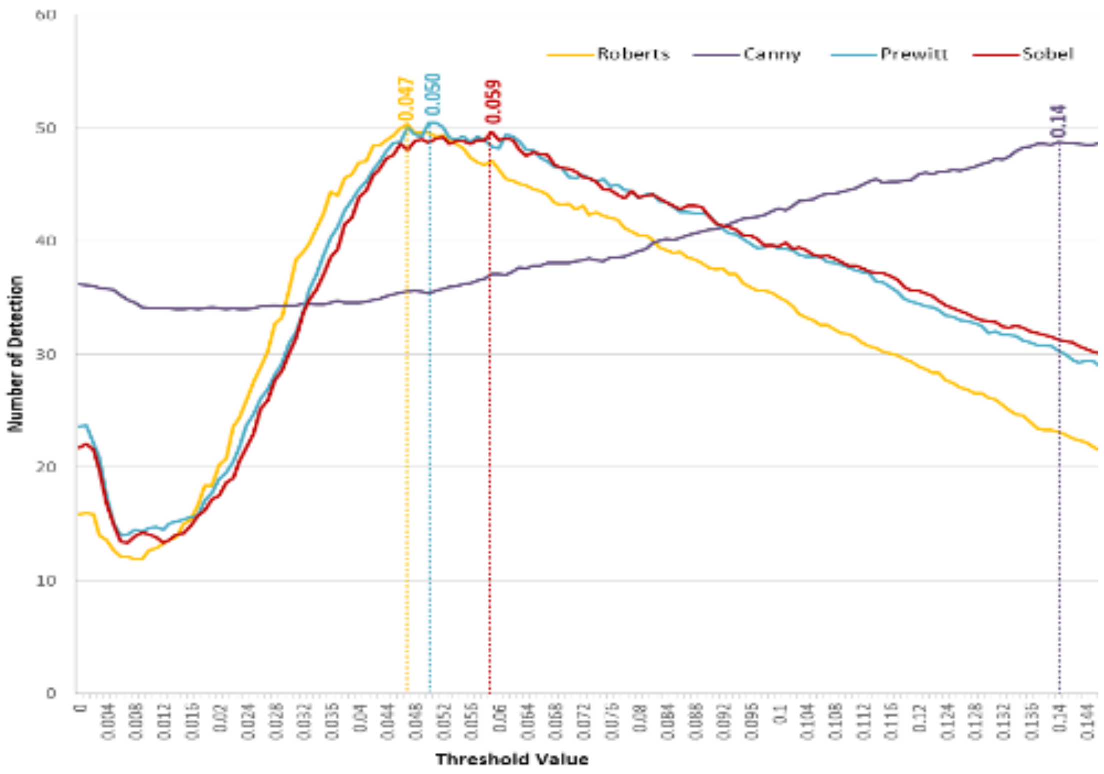


Figure 37: Maximum detection of obstacle-candidates for edge detection methods using dynamic threshold

After applying the entire threshold range on all four edge detection methods, the one with the best performance is selected for each field. DLC results are presented in Table 10 which indicates detection improvements on all the fields but two (marked by [a]). Correct detections are improved up to 45% in overall using DLC. The false positive error improved by 33.3% and the false negative error minimized to almost one-fifth comparing to the initial results gained using Sobel with default parameter sets.

Table 10: DLC vs. Sobel with default threshold in detection

Field	Sobel (Default Thr.)	DLC	Field	Sobel (Default Thr.)	DLC
1	1	**6**	27	33	**40**
2	5	**34**	28	27	**35**
3	12	**8** [a]	29	40	**40**
4	53	**54**	30	15	**22**
5	17	**27**	31	32	**37**
6	10	**23**	32	47	**50**
7	18	**20**	33	4	**14**
8	14	**20**	34	65	**69**
9	14	**18**	35	18	**28**
10	18	**29**	36	9	**19**
11	31	**51**	37	19	**19**
12	2	**8**	38	11	**28**
13	5	**27**	39	22	**20** [a]
14	11	**16**	40	11	**23**
15	27	**57**	41	5	**10**
16	1	**16**	42	38	**50**
17	9	**18**	43	17	**17**
18	22	**108**	44	13	**34**
19	11	**54**	45	45	**87**
20	3	**9**	46	56	**61**
21	37	**51**	47	56	**74**
22	33	**34**	48	30	**37**
23	8	**9**	49	3	**3**
24	5	**8**	50	54	**72**
25	9	**9**	51	14	**21**
26	12	**14**			

Grey-level intensity threshold has a significant impact on minimizing wrong detections by classifying correct and incorrect detections. As shown in Figure 25, grey-level intensity threshold value 127.37 was used to classify results and remove the wrong detections. Although this assisted the system to minimize wrong detections, yet, there exist cases which could be improved. Results in Figure 25 indicate that there are overlapping correct and incorrect detections in the areas close to the preferred grey-level intensity threshold value. The overlapped values introduce additional complexity to the elimination of wrong detections.

4.3 Discussion on Path Planning Methods and Semantic Annotation

The policy to generate trajectory points depends on the selected path planning algorithm. A path planning algorithm might more suitable depending on the decision of the type of the crop, properties of the soil, geometry and slope of the field. Some crops may require deeper crop rows with more distance in between and some may require shallow rows with less distance from each other. The crop selection policy may be influenced by the cultural conventions, regional climate, geographical location of the field, slope and shape of the field and some many other factors, which are beyond the scope of this thesis.

The developed system was tested using a simple path planning algorithm to demonstrate its capability of integration with any path planning algorithm, and to focus on development of obstacle recognition and semantic annotation. Having parallel and straight lines are the main features of the selected path planning algorithm. Crop-lanes are planned parallel to the longest edge of the field, with a constant crop-row distance which is entered at the start of the application. A line equation according to the baseline with fixed increments and various Y-intercepts used to make sure these paths are correctly created. A sample of the generated trajectory points in pixel coordinates and their conversion to GPS coordinates are presented in Figure 38 and Figure 39 accordingly. Results of the linear equation with no exponents higher that one on both figures guarantees the straightness of both tracks.

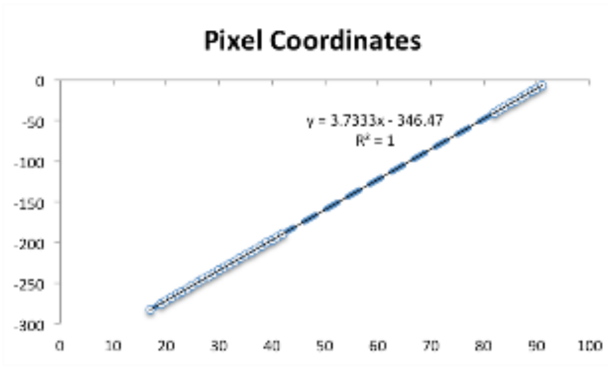


Figure 38: Proof of produced trajectory points being in a straight line in pixels

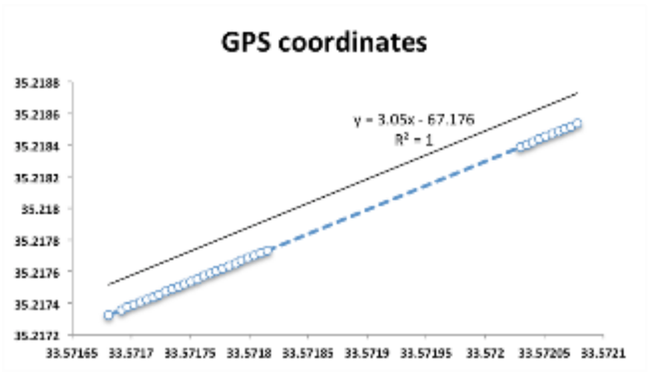


Figure 39: Proof of produced trajectory points being in a straight line in GPS coordinates

Figure 40 provides an example for further explanation the developed system and its integration into other path planning algorithms. As illustrated in the image, this particular path planning algorithm has different rules in creating the path, especially when it comes to the turns at the end of each track. Although the tracks are parallel to each other, it does not mean that the tractor should follow one after the other. In this path, the tractor might need to go from track 1 to 3 due its limitations in manoeuvre at the end of some tracks. Current capability of the GPS installed automatic vehicles safely provides sufficient tracking accuracy to process all crop-rows with a precision less than two cm. Now, when the proposed system integrates into this particular path planning algorithm, we would get a different sort of trajectory points regarding the sequence. For example, while the path is in a straight line we will have the same output structure including the track, the segments, and the points. However, when it comes to the turns, we might have different sort of point sequence representing a semi-circle path to cover the turns. Now, all the autonomous vehicle must do is to follow the generated trajectory points to have a successful turn and back on the next track. Consequently, the annotation of the track would differ too. For example, when it comes to the turns, additional children, sub-children, and elements would be generated during the annotation and creation of the RDF/XML file. However, the annotation of obstacles would not alter by different planning algorithms as obstacles are entirely different entities than the trajectory points.

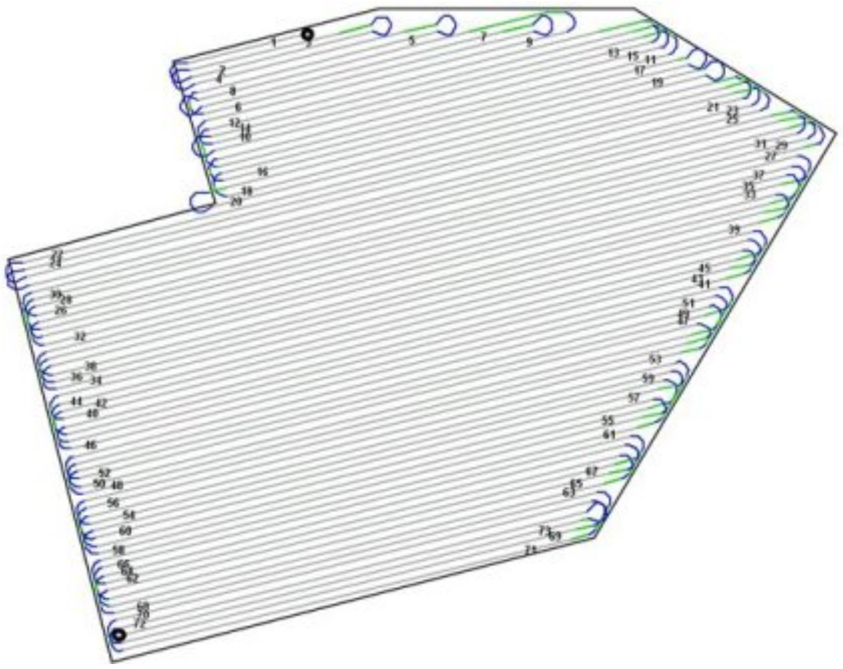


Figure 40: Sample output of a path planning algorithm

Chapter 5

CONCLUSION

In this study, we have proposed a state-of-the-art method to fill the information flow gap between planning and automation in agricultural environment. The system introduced a method to semantically annotate and generate traceable trajectory points in agricultural realm. The trajectory points are produced in the form of GPS coordinates for autonomous agricultural vehicles to trace. Also, we have proposed a new method to recognize, locate and annotate obstacles within the field. This study focused on detecting trees which are considered obstacles in many path planning strategies. In the interest of simplicity and ease of use, a single GPS coordinate of the field's area is required to initiate the process. This GPS point is crucial to the system as all the global positioning computations depend on it. These computations include extracting the target field, locating the obstacles and converting pixel coordinates to GPS coordinates. Furthermore, detected obstacles and generated trajectory points are semantically annotated for data exchange purposes between different sub-systems and ontologies of the agricultural domain.

Various edge detection methods including Canny, Prewitt, Roberts and Sobel are used for detection purposes on top-view satellite images. These images are obtained using the free Google Map API, which provides images with maximum resolution of 640 by 640 dpi. On all the experiments the zoom level is set to Δ_z=18 for consistency and comparison purposes. The success rates of the edge detection methods are compared to select the most feasible method. On the early stages of the study, Sobel was chosen as the primary method with default parameter sets. However, by fine-tuning the sensitivity threshold and introducing T-range, Max-of-All, and DLC algorithms, we have achieved higher precision in the detection phase. The Max-of-All algorithm provided fewer detection improvements (16.6%) comparing to the T-range (21.43%), however, managed to enhance the detection on some of the fields which T-range failed. The DLC algorithm improved the detections up to 45.4% by merging the Max-of-All and the T-range. Also, using the DLC method, false positive error (FPE) reduced by 33% and by using the grey-level-intensity threshold false negative errors (FNE) reduced by 80% on average.

With the assumption of accurate mappings between the satellite images and GPS coordinates by Google Map, the proposed method provides significant precision in locating and identifying obstacles. Minimum initialization requirements (one GPS point) and considerable low processing time (1.15–25.95 seconds) are some advantages of the system. Also, generating reliable and traceable trajectory points in the form of GPS coordinates makes this method feasible for further consideration and utilization for any path planning algorithm in agricultural automation.

The main intention of this study was to produce traceable trajectory points for autonomous agricultural vehicles rather than developing a new path planning algorithm. The developed system integrated into a path planning algorithm for evaluation purposes. Results indicate that this method is capable of generating trajectory points with significant accuracy while merged into a path planning algorithm. Detected obstacles and generated trajectories graphically annotated on the image and also semantically annotated and captured in RDF/XML format. The annotation makes the outputs of the system exchangeable between related applications, ontologies or subsystems of agricultural automation. And, an emerging publication is expected on this issue as a fruit of this dissertation.

In the field extraction process morphological reconstruction was used to remove noises on the obtained image. However, in the future studies, it may be beneficial to use nonlinear diffusion filter rather than morphological reconstruction to reduce missed obstacle recognition rate. Nonlinear diffusion can remove both high and low frequency noises selectively while morphological reconstruction works very effectively for the removal of high frequency noises. Another future study may consider other detection methods to overcome the brightness and contrast issue to improve the field extraction phase. Consequently, more variety of fields will be identified even if the field is at the same grey-intensity level as the surrounding area. Furthermore, in addition to the threshold value, other parameters like object-polarity, computation method, and sensitivity factor could be considered to enhance the object detection phase even more. Instantaneous adaptability is another feature we would like to study on to integrated the proposed method with real-time systems which requires having access to live satellite images. Extending the semantic annotation section is another future possibility to support Web Ontology Language (OWL), which can describe the semantics of classes and properties used beyond the semantics of RDF schema in a formal methodology. Another future improvement effort shall be considered for user intervention to validate detections, reduce errors and to reduce harmful possibilities.

REFERENCES

[1] V. Blanco, L. Carpente, Y. Hinojosa and J. Puerto. (2010). Planning for Agricultural Forage Harvesters and Trucks: Model, Heuristics, and Case Study. *Networks and Spatial Economics,* vol. 10, no. 3, pp. 321-343.

[2] J. Gomez-Gil, R. Ruiz-Gonzalez, S. Alonso-Garcia and F. J. Gomez-Gil. (2013). A Kalman Filter Implementation for Precision Improvement in Low-Cost GPS Positioning of Tractors. *Sensors (Basel),* vol. 13, no. 1424-8220, 8 Nov.

[3] R. Lenain, B. Thuilot, C. Cariou and P. Martinet. (2006). High Accuracy Path Tracking for Vehicles in Presence of Sliding: Application to Farm Vehicle Automatic Guidance for Agricultural Tasks. *Autonomous Robots ,* vol. 21, no. 1, pp. 79-97, 08 2006.

[4] M. Bodur, E. Kiani and H. Hacisevki. (2012). Double Look-Ahead Reference Point Control for Autonomous Agricultural Vehicles. *Biosystems Engineering,* vol. 113, no. 2, pp. 173-186.

[5] B. Xu, D. J. Stilwell and A. J. Kurdila. (2013). Fast Path Re-Planning Based on Fast Marching and Level Sets. *Journal of Intelligent & Robotic Systems,* vol. 71, no. 3, pp. 303-317.

[6] T. Oksanen and A. Visala, (2007). Path Planning Algorithms for Agricultural Machines. *Commission of Agricultural Engineering (CIGR, Commission Internationale du Genie Rural),* vol. 9.

[7] Q. Zhang and H. Qui. (2004). A Dynamic Path Search Algorithm for Tractor Automatic Navigation. *Transactions of the ASAE,* vol. 47, no. 2, pp. 639-646.

[8] A. A. Bakhtiari, J. Mehri, D. Bochtis and H. Navid. (2011). Optimal Route Planning of Agricultural Field Operations using Ant Colony Optimization. *Agricultural Engineering International: CIGR Journal,* vol. 13, no. 2.

[9] M. Kothari and I. Postlethwaite. (2013). A Probabilistically Robust Path Planning Algorithm for UAVs using Rapidly-Exploring Random Trees. *Journal of Intelligent & Robotic Systems,* vol. 71, no. 2, pp. 231-253.

[10] Y. S. X. and L. C. (2004). A Neural Network Approach to Complete Coverage Path Planning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics),* vol. 34, no. 1, pp. 718 - 724, Feb.

[11] R. Kala and K. Warwick. (2013). Multi-Level Planning for Semi-Autonomous Vehicles in Traffic Scenarios based on Separation Maximization. *Journal of Intelligent & Robotic Systems,* vol. 72, no. 3-4, pp. 559-590, 12.

[12] S. Fabre, P. Soukrest, M. Tai'xt and L. Cordesses. (2001). Framework Path Planning for Field Coverage with Minimum Overlapping. *International*

Conference on Emerging Technologies and Factory Automation, Proceedings.

[13] G. Zuo, P. Zhang and J. Qiao. (2010). Path Planning Algorithm Based on Sub-Region for Agricultural Robot. *Informatics in Control, Automation and Robotics (CAR), 2nd International Asia Conference*, Wuhan.

[14] M. Taïx, P. Souères, H. Frayssinet and L. Cordesses. (2006). Path Planning for Complete Coverage with Agricultural Machines. *Field and Service Robotics,* vol. 24, pp. 549-558.

[15] R. Daily and D. M. Bevly. (2004). The use of GPS for Vehicle Stability Control Systems. *IEEE Transactions on Industrial Electronics,* vol. 51, no. 2, pp. 270-277, April.

[16] R. Lenain, B. Thuilot, C. Cariou and P. Martinet. (2007). Adaptive and Predictive Path Tracking Control for Off-Road Mobile Robots. *European Journal of Control ,* vol. 13, no. 4, pp. 419-439, 07.

[17] M. Kise, N. Noguchi, K. Ishii and H. Terao. (2005). Laser Scanner-based Obstacle Detection System for Autonomous Tractor: Movement and Shape Detection Targeting at Agricultural Vehicle. *Journal of the Japanese Society of Agricultural Machinery,* vol. 66, no. 2, pp. 97-104.

[18] D. Bratasanu, I. Nedelcu and M. Datcu. (2011). Bridging the Semantic Gap for Satellite Image Annotation and Automatic Mapping Applications. *IEEE*

Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 4, no. 1, pp. 193-204, March.

[19] K. Murphy, A. Torralba, D. Eaton and W. Freeman. (2006). Object Detection and Localization using Local and Global Features. *Toward Category-Level Object Recognition*, vol. 4170, Springer Berlin Heidelberg, pp. 382-400.

[20] T. Deselaers, D. Keysers and H. Ney. (2005). Discriminative Training for Object Recognition using Image Patches. *IEEE Computer Society Conference: Computer Vision and Pattern Recognition,* vol. 2.

[21] J.-J. Chen, C.-R. Su, W. E. L. Grimson, J.-L. Liu and D.-H. Shiue. (2012). Object Segmentation of Database Images by Dual Multiscale Morphological Reconstructions and Retrieval Applications. *IEEE Transactions on Image Processing,* vol. 21, no. 2, pp. 828-843, 1.

[22] M. Lienou, H. Maitre and M. Datcu. (2010). Semantic Annotation of Satellite Images using Latent Dirichlet Allocation. *IEEE Geoscience and Remote Sensing Letters,* vol. 7, no. 1, pp. 28-32, 1.

[23] A. Mohan, C. Papageorgiou and T. Poggio. (2001). Example-Based Object Detection in Images by Components. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 23, no. 4, pp. 349-361, 4.

[24] Google, "Google Map API," (23 Dec 2015). Retrieved from https://developers.

google.com/maps/documentation.

[25] M. Juneja and P. S. Sandhu. (2009). "Performance Evaluation of Edge Detection Techniques for Images in Spatial Domain," *International Journal of Computer Theory and Engineering,* vol. 1, no. 2, pp. 614-621.

[26] G. Shrivakshan and C. Chandrasekar. (2010). A Comparison of various Edge Detection Techniques used in Image Processing. *IJCSI International Journal of Computer Science ,* vol. 9, no. 5, pp. 269-276, Sep.

[27] M. Bodur, M. Mehrolhassani, and Anas Q. Mahdi. (2014). Comparison of the Edge Detection Methods in Detecting Agricultural Obstacles. *in ICAFS-2014, Eleventh International Conference on Application of Fuzzy Systems and Soft Computing, Proceedings*, pp.93-103, September 2-3, (Paris, France)

[28] M. Bodur, and M. Mehrolhassani. (2013). Textual Annotation, Detection and Positioning of Obstacles in Agricultural Fields using Satellite Images. *ICSCCW-2013 Seventh International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control* , September 2-3. Izmir, Turkey.

[29] M.Bodur, and M. Mehrolhassani. (2015). Satellite Images-based Obstacle Recognition and Trajectory Generation for Agricultural Vehicles. *International Journal of Advanced Robotic Systems,* ISSN 1729-8806, DOI:10.5772/62069,

December 22.

[30] M. Smereka and I. Dulęba. (2008). Circular Object Detection using a Modified Hough Transform. *International Journal of Applied Mathematics and Computer Science - Applied Image Processing,* vol. 18, no. 1, pp. 85-91.

[31] D. Maheswari and V. Radha. (2010). Noise Removal in Compound Image using Median Filter. *International Journal of Advanced Trends in Computer Science and Engineering,* vol. 2, no. 4, pp. 1359-1362.

[32] M. R. Khokher, A. Ghafoor and A. M. Siddiqui. (2013). Image Segmentation using Multilevel Graph Cuts and Graph Development using Fuzzy Rule-Based System. *IET Image Processing,* pp. 201-211, June.

[33] H.-C. Wang, T.-H. Huang and C.-T. Liu. (2011). Automatic Summarization based on Automatically Induced Ontology. *Intelligent Automation and Soft Computing,* vol. 17, no. 4, pp. 447-463.

[34] Y.-W. Tai, J. Jia and C.-K. Tang. (2007). Soft Color Segmentation and its Applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 29, no. 9, pp. 1520 - 1537, Sep.

[35] V. Luc. (2007). Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient Algorithms. *IEEE Transactions on Image*

Processing , vol. 2, no. 2, pp. 176-201.

[36] M. Gavrica, M. Martinovb, S. Bojicb, D. Djatkovb and M. Pavlovic. (2011). Short-and Long-Term Dynamic Accuracies Determination of Satellite-Based Positioning Devices using a Specially Designed Testing Facility. *Computers and Electronics in Agriculture,* vol. 76, no. 2, pp. 297-305.

[37] K. Petridis, S. Bloehdorn, C. Saathoff, N. Simou, S. Dasiopoulou, V. Tzouvaras, S. Handschuh, Y. Avrithis, Y. Kompatsiaris and S. Staab. (2006). Knowledge Representation and Semantic Annotation of Multimedia Content. *IEE Proceedings - Vision, Image and Signal Processing,* vol. 153, no. 3, pp. 255 - 262, June.

[38] M. Nørremark, G. H.W, N. J. and S. H.T. (2008). The Development and Assessment of the Accuracy of an Autonomous GPS-Based System for Intra-Row Mechanical Weed Control in Row Crops. *Biosystems Engineering,* vol. 101, no. 4, pp. 396-410.

[39] A. Sgorbissa and R. Zaccaria. (2013). Integrated Obstacle Avoidance and Path Following through a Feedback Control Law. *Journal of Intelligent & Robotic Systems,* vol. 72, no. 3-4, pp. 409-428, 12.

[40] E. Cuevas, N. Ortega-Sánchez, D. Zaldiva and M. Pérez-Cisneros. (2012). Circle Detection by Harmony Search Optimization. *Journal of Intelligent &*

Robotic Systems, vol. 66, no. 3, pp. 356-376, 5.

[41] J. Kopecký, T. Vitvar, C. Bournez and J. Farrell. (2007). SAWSDL: Semantic Annotations for WSDL and XML Schema. *IEEE Internet Computing,* vol. 11, no. 6, pp. 60-67, Dec.

[42] Waing and N. Aye. (2013). On the Automatic Detection System of Stop Line Violation for Myanmar Vehicles (Car). *International Journal of Computer & Communication Engineering Research (IJCCER),* vol. 1, no. 4.

[43] C. Wanga, T. Ruan Wanb and I. J. Palmerb. (2012). Automatic Reconstruction of 3D Environment using Real Terrain Data and Satellite Images. *Intelligent Automation & Soft Computing,* vol. 18, no. 1, pp. 49-63.

[44] K. Mahalingama and M. N. Huhnsa. (2000). Ontology Tools for Semantic Reconciliation in Distributed Heterogeneous Information Environments. Intelligent Automation & Soft Computing, vol. 6, no. 3, pp. 185-192.

[45] World Wide Web (W3C) Consortium, "W3C Standards," (07 Sep 2016). Retrieved from http://www.w3.org.

[46] Plant Ontology Database Project, "Plant Ontology (PO)," (07 Sep 2016). Retrieved from http://www.plantontology.org.

[47] OGC Working Group, "Open Geospatial Consortium (OGC)," (07 Sep 2016).

Retrieved from http://www.w3.org/2003/01/geo.

[48] SVG Working Group, "Scalable Vector Graphics (SVG)," (07 Sep 2016). Retrieved from http://www.w3.org/TR/SVG/Overview.html.

[49] Dublin Core Metadata Initiative (DCMI), "The Dublin Core Ontology (DC)," (07 Sep 2016). Retrieved from http://www.dublincore.org.

APPENDICES

Appendix A: Segmentation and Extraction Source Code

```
% Segmentation and Extraction of Target Field (Automatic)
if strcmp(Choice2,'a')
        [imWidth,imHeight]=size(BW);
        msk=[0 0 0 0 0;
           0 1 1 1 0;
           0 1 1 1 0;
           0 1 1 1 0;
           0 0 0 0 0;];
        % Smoothing image to reduce the number of connected components
        B=conv2(double(BW),double(msk));
        L = bwlabeln(B,8);% Calculating connected components
        mx=max(max(L));
        ConnectedC = bwconncomp(B); % Find connected components in binary image
        % %%%%% this section will label all the objects based on 'bwconncomp'
        % Calculate centroids for connected components in the image using regionprops
        segSel = regionprops (ConnectedC, 'Centroid');
        bwh1 = figure, imshow(BW); % show image
         hold on
                for k = 1:numel(s)
                c = segSel (k).Centroid;
                text(c(1), c(2), sprintf('%d', k), ...
                   'HorizontalAlignment', 'center', ...
                   'VerticalAlignment', 'middle',...
                   'BackgroundColor','green',...
                   'FontSize',16);
                end
        hold off
        saveas (gca,'bwl2.png'); % save image
        figure, imshow(BW); % show image
        AxesL = gca;   % Not the GCF
        figLabeled = getframe(AxesL);
        imwrite(figLabeled.cdata, 'bwLabeled.png');
        imshow('bwLabeled.png', 'Parent', handles.small_axes2);
        maxObj = CC.NumObjects; % Find total number of objects
         % %%%this part will select the biggest section and displays
        numPixels = cellfun(@numel,CC.PixelIdxList); % counts number of pixels in each object
```

```
[biggest,idx] = max(numPixels); % finds the biggest object based on pixel numbers
        for i=1:maxObj % set all pixels in the image to 0 to remove all objects
        BW(CC.PixelIdxList{i}) = 0;
        end
BW(CC.PixelIdxList{idx}) = 1; % set pixels in idx to 1 to show only the biggest object
% display the largest field with label
BW1 = BW;
figure, imshow(BW),title('biggest part');

%%%%%%%%%% this section will label the largest object
cenSeg = segSel (idx).Centroid;
text(cenSeg (1), cenSeg (2), sprintf('%d', idx), ...
'HorizontalAlignment', 'center', ...
'VerticalAlignment', 'top',...
'BackgroundColor','green',...
'FontSize',16);
figure, imshow(BW),title('before crop');
saveas(gca,'bwl3.png')
AxesL1 = gca;   % Not the GCF
figLabeled1 = getframe(AxesL1);
imwrite(figLabeled1.cdata, 'bwLabeled1.png');
imshow('bwLabeled1.png', 'Parent', handles.small_axes3);

%% measuring bounding box of the image
 selReg = regionprops(BW, 'BoundingBox');
 I6 = imcrop(BW1, selReg.BoundingBox);
imshow(I6),title('after crop');
AxesL2 = gca;   % Not the GCF
Labeled2 = getframe(AxesL2);
imwrite(I6, 'Labeled2.png');
hold on
I3 = imread('temp.jpg');
I2 = imcrop(I3, selReg.BoundingBox); %cropping the image
I4 =  im2bw(I2,graythresh(I2));
imshow(I2, 'Parent', handles.main_axes);%display cropped image into main axes GUI
imwrite(I2,'ask_out.png'); % write the result in an image file
imwrite(I4,'ask_out1.png'); % write the result in an image file


% save bounding box info to pass to other procedures
```

```
        handles.BoundingBox1 = selReg.BoundingBox(1);
        handles.BoundingBox2 = selReg.BoundingBox(2);
        guidata(hObject, handles)
elseif strcmp(Choice2,'m') % manual segment selection
        [imWidth,imHeight]=size(BW);
        msk=[0 0 0 0 0;
          0 1 1 1 0;
          0 1 1 1 0;
          0 1 1 1 0;
          0 0 0 0 0;];
        % Smoothing image to reduce the number of connected components
        B=conv2(double(BW),double(msk));
        L = bwlabeln(B,8);% Calculating connected components
        mx=max(max(L));
         %%%%%%%%% this section will label all the objects based on 'bwconncomp'
        segSel = regionprops(L, 'Centroid');
        bwh1 = figure, imshow(BW); % show image
        BW1 = BW;
        figure, imshow(BW),title('biggest part');
        hold on
                for k = 1:numel(s)
                   c = segSel (k).Centroid;
                   text(segSel (1), segSel (2), sprintf('%d', k), ...
                      'HorizontalAlignment', 'center', ...
                      'VerticalAlignment', 'middle',...
                      'BackgroundColor','green',...
                      'FontSize',16);
                end
        hold off
        % save current fig and display in axes
        saveas(gca,'bwl2.png')
        AxesL = gca;   % Not the GCF
        figLabeled = getframe(AxesL);
        imwrite(figLabeled.cdata, 'bwLabeled.png');
        imshow('bwLabeled.png', 'Parent', handles.small_axes2);
        figure, BW2 = bwselect(L),title('All segments-Select desired one'); % allows you to select segment from the screen
        axes(handles.main_axes);
        selReg = regionprops(BW2, 'BoundingBox'); % find bounding box of result
        I3 = imread('temp.jpg');
        I2 = imcrop(I3, selReg.BoundingBox); % crop the image according to the bounding box
        imshow(I2, 'Parent', handles.main_axes); %display croped image into
        main axes GUI
        I6 = imcrop(BW1, selReg.BoundingBox);
        imshow(I6),title('after crop');
        AxesL2 = gca;   % Not the GCF
        Labeled2 = getframe(AxesL2);
        imwrite(I6, 'Labeled2.png');
        imwrite(I2,'ask_out.png'); % write the result in an image file

        % save bounding box info to pass to other procedures
        handles.BoundingBox1 = selReg.BoundingBox(1);
        handles.BoundingBox2 = selReg.BoundingBox(2) ;
        guidata(hObject, handles)
end


```

Appendix B: Obstacle Detection Source Code

```
switch objType
case 'tree'
% (Z=17,Rmin=2) (Z=18,Rmin=4) Min radius suitable for different zooms in
google to find %"tree" or circular shape obstacles
Rmin = 4;
% Max radius suitable for zoom=18 in google to find "tree" or circular shape
obstacles
Rmax = 18;
% finding dark circles
[centersDark, radiiDark] = imfindcircles(outputImage,[Rmin
Rmax],'ObjectPolarity','dark');
save('centersDark.mat', 'centersDark')
save('radiiDark.mat', 'radiiDark')
regionNumber=handles.region;
regionChoice=str2double(regionNumber);
% choosing region to search for obstacles
ext_SegmentSelection % Call "ext_SegmentSelection.m" to select desired
region
k=1;
numOfCircles=size(centersDark,1);
        if length(centersDark)==0 % search for existing obstacles
                disp(' no tree');
        end
% loop into all detected obstacles check and if the detected obstacle is in the
right area
for i=1:numOfCircles
        if (centersDark(i,1)>=xMin && centersDark(i,1)<=xMax) &&
        (centersDark(i,2)>=yMin && centersDark(i,2)<=yMax) ...
        && (outputBW1(round(centersDark(i,2)),round(centersDark(i,1))) == 0)
        centerPoint(k,1)=centersDark(i,1);
        centerPoint(k,2)=centersDark(i,2);
        radiiPoint(k,1)=radiiDark(i,1);
        k=k+1;
        viscircles(centerPoint, radiiPoint,'LineStyle','-'); % labelling circles
        %Save found and inbound obstacle into variables
        load('centerP1.mat', 'centerP1')
        load('radiiP1.mat', 'radiiP1')
        centerP1 = [centerP1; centerPoint]
        radiiP1 = [radiiP1; radiiPoint]
        save('centerP1.mat', 'centerP1')
        save('radiiP1.mat', 'radiiP1')

        ex_SetTreeGpsCoord % get GPS coordinates of the found tree in the
        cropped image
        %% Semantic annotation preparation for detected obstacles
        treeRaius = radiiDark(i);
        centersDark(i,1);
        centersDark(i,2);
        findX = num2str(centersDark(i,1));
        findY = num2str(centersDark(i,2));
        findtTree = '';
        xDoc = xmlread('output.xml');
        allListitems = xDoc.getElementsByTagName('PixelCoordinates');

        if (allListitems.getLength) == 0 % check if XML file is empty
        %######## INSERT into XML file for tree for first time #######
        ext_XmlWriteTree; %Call xmlWriteTree.m to add a tree to XML file
        else
        %   Duplicity check
        end
    end
```

Appendix C: Duplicity Check Source Code

```
%%%   Duplicity check
for L = 0:allListitems.getLength-1 % check if tree exists in XML file
thisListitem = allListitems.item(L);
% Get the label element. In this file, each
% list item contains only one label.
xList = thisListitem.getElementsByTagName('X');
xElement = xList.item(0);
xmlX=str2double(xElement.getFirstChild.getData);
xmlX_string=num2str(xmlX);
yList = thisListitem.getElementsByTagName('Y');
yElement = yList.item(0);
xmlY=str2double(yElement.getFirstChild.getData);
xmlY_string=num2str(xmlY);
% Check whether this is the label you want.
% The text is in the first child node.
        if strcmp(xmlX_string, findX) && strcmp(xmlY_string, findY)
                treeFound = 1; % tree found will set to 1 if the tree exists in XML
                file
                break;
        else
                treeFound = 0;
        end
end
        if treeFound == 0
        %######## update XML file for new tree if "treeFound=0" #######
        ext_XmlWriteTree %Call xmlWriteTree.m to add a tree to XML file
        end
end
end
```

Appendix D: Code for Geographical Positioning of Obstacles

```
%% calculate GPS coordinates of detected obstacles
hold on
% calculating target field's reference point in pixel coordinate
        OrgCenPixelX = imWidth/2;
        OrgCenPixelY = imHeight/2;
% mapping target field's reference point in GPS coordinate
        OrgCenPixelLong = str2double(longCoord);
        OrgCenPixelLat = str2double(latCoord);
        disp(OrgCenPixelX);
        disp(OrgCenPixelY);
        disp(OrgCenPixelLong);
        disp(OrgCenPixelLat);
% calculating reference point's offset in pixel coordinate after crop using
bounding box info.
        CropCenPixelX = round(OrgCenPixelX-(handles.BoundingBox1));
        CropCenPixelY = round(OrgCenPixelY-(handles.BoundingBox2));
% mapping the reference GPS coordinate to its new pixel after crop
        CropCenPixelLong = str2double(longCoord);
        CropCenPixelLat = str2double(latCoord);
        disp(CropCenPixelX);
        disp(CropCenPixelY);
        disp(CropCenPixelLong);
        disp(CropCenPixelLat);
%set zoom level
        disp(zoom);
% Gps points per pixel
        gpsPP = 360/(((2^(str2double(zoom)))*256));
        disp(gpsPP);
% calculating gps point per latitude and longitude degree
        gpsPpLat = (((cos(CropCenPixelLat*(pi/180)) *
        111.321)/111.321))*(gpsPP);
        gpsPpLong = 1*(gpsPP);
        disp(gpsPpLat);
        disp(gpsPpLong);
% calculating obstacles GPS coordinate
        treeGpsLat =(CropCenPixelLat+((CropCenPixelY-
        round(centersDark(i,2)))*gpsPpLat));
        treeGpsLong =(CropCenPixelLong-((CropCenPixelX-
        round(centersDark(i,1)))*gpsPpLong));
        disp(treeGpsLat);
        disp(treeGpsLong);
```

Appendix E: Code for Semantic Annotation of Obstacles

```
if strcmp(handles.procNo,'0')
    %creating the root element "Obstacles"
    docNode = com.mathworks.xml.XMLUtils.createDocument...
    ('Obstacles');
    handles.docNode = docNode;
    obstacleRootNode = docNode.getDocumentElement;
    % creating child element "treeElement" for parent element "Obstacles"
    treeElement = docNode.createElement('Tree');
    obstacleRootNode.appendChild(treeElement);
    handles.treeElement = treeElement;
    xmlwrite('output.xml',docNode);
    guidata(hObject, handles);
end
hold on
% fetching detected obstacles information
treeRaius = radiiDark(i);
tPX = centersDark(i,1);
tPY = centersDark(i,2);
zoom = handles.zoom; % setting zoom level
% Calculating obstacle’s dimensions
NGPPP = 40075017/(((2^(str2double(zoom)))*256));
treeDiameter_meter = (2*treeRaius)*NGPPP
% Add tagged tree to variable
Tree_Tagged_temp = [centersDark(i,1),
centersDark(i,2),treeRaius,treeDiameter_meter,treeGpsLat,treeGpsLong]
Tree_Tagged = [Tree_Tagged ; Tree_Tagged_temp]
    % creating child element "Item" for parent element "Tree"
    treeItem = handles.docNode.createElement('Item');
    handles.treeElement.appendChild(treeItem);

        % creating child element "Number" for parent element "Item"
        treeNumber = handles.docNode.createElement('Number');
        treeItem.appendChild(treeNumber);
            % setting property of the child element "Number"
            treeNumber.appendChild(handles.docNode.createTextNo
            de(sprintf('%i',i)));
            treeItem.appendChild(treeNumber);
        % creating child element "Position" for parent element "Item"
        treePosition = handles.docNode.createElement('Position');
        treeItem.appendChild(treePosition);
```

```
                % setting property of the child element "Number"
                treePosition.appendChild(handles.docNode.createTextNo
                de(sprintf('%s',char(regionNumber))));
                treeItem.appendChild(treePosition);
        % creating child element " PixelCoordinates" for parent element
        "Item"
        treePixelCoordinates =
        handles.docNode.createElement('PixelCoordinates');
        treeItem.appendChild(treePixelCoordinates);
                % creating child element "X" for parent element "
                PixelCoordinates "
                xCoord = handles.docNode.createElement('X');
                treePixelCoordinates.appendChild(xCoord);
                        % setting property of the child element "X"
                        xCoord.appendChild(handles.docNode.createText
                        Node(sprintf('%f',tPX)));
                        treePixelCoordinates.appendChild(xCoord);
                 % creating child element "X" for parent element "
                PixelCoordinates "
                yCoord = handles.docNode.createElement('Y');
                treePixelCoordinates.appendChild(yCoord);
                        % setting property of the child element "Y"
                        yCoord.appendChild(handles.docNode.createText
                        Node(sprintf('%f',tPY)));
                        treePixelCoordinates.appendChild(yCoord);
        % creating child element "'Radius" for parent element "Item"
        treeRadius = handles.docNode.createElement('Radius');
        treeItem.appendChild(treeRadius);
                % setting property of the child element "'Radius"
                treeRadius.appendChild(handles.docNode.createTextNod
                e(sprintf('%f',treeRaius)));
                treeItem.appendChild(treeRadius);
        % creating child element "' Latitude" for parent element "Item"
        treeLatitude = handles.docNode.createElement('Latitude');
        treeItem.appendChild(treeLatitude);
                % setting property of the child element "Latitude"
                treeLatitude.appendChild(handles.docNode.createTextNo
                de(sprintf('%f',treeGpsLat)));
                treeItem.appendChild(treeLatitude);
        % creating child element "' Longitude" for parent element "Item"
        treeLongitude = handles.docNode.createElement('Longitude');
        treeItem.appendChild(treeLongitude);
                % setting property of the child element "Longitude "
                treeLongitude.appendChild(handles.docNode.createTextN
                    ode(sprintf('%f',treeGpsLong)));
                    treeItem.appendChild(treeLongitude);
            % creating child element " GpsPositions " for parent element
            "Item"
            treeGpsPositions =
            handles.docNode.createElement('GpsPositions');
            treeItem.appendChild(treeGpsPositions);
                    % setting property of the child element " GpsPositions "
                    treeGpsPositions.appendChild(handles.docNode.createTe
                    xtNode(sprintf('%f,%f',treeGpsLat,treeGpsLong)));
                    treeItem.appendChild(treeGpsPositions);
xmlwrite('output.xml',docNode); % write XML file for detected obstacles
```

Appendix F: Code for DLC Algorithm

```
%%%%% DLC algorithm (using ranged threshold) for each field
%%%%% including Canny, Sobel, Prewitt and Roberts with/without noise reduction

clc;
clear all;
warning('off','all');
%% Load image
im=imread('img1/croped/2.jpg');
im1=rgb2gray(im); %Convert colour image to grayscale
radius_min = 4;
radius_max = 18;
%%  ###### LOOP TO FIND BEST Threshold value for Sobel ######
cnt=1;
Sobel_Best_Result = 0;
 for Sen_thresh = 0.0:0.001:0.3; %Sensitivity threshold range
    tic % start timer
    close all
    disp (Sen_thresh); %display the current threshold
    %% ########### Sobel threshold range without noise reduction and smoothing
    [BW,thresh] = edge(im1,'sobel',Sen_thresh); %finding edges
    %%%% finds circles in the image which are trees with radios range from 4 to 18 pixels which is best for zoom=18 for google images
    [centers, radii, metric] = imfindcircles(BW,[radius_min radius_max]);
    disp ('Detection'); % display 'Detection' note on screen
    toc % stop timer for detection only and display on screen
     Sobel_total_detections = size(centers,1);
    Sobel_Best_Result(cnt,1) = Sobel_total_detections;
    Sobel_Best_Result(cnt,2) = Sen_thresh
    %%%Display Images of Sobel without noise reduction for each threshold value
    %%%figure,imshow(BW),title('Sobel Without Smoothing and Noise Reduction');
    centersStrong5 = centers(1:size(centers,1),:);
    radiiStrong5 = radii(1:size(centers,1));
    metricStrong5 = metric(1:size(centers,1));
    %%%viscircles(centersStrong5, radiiStrong5,'EdgeColor','b');
    figure,imshow(im),title('Sobel Without Smoothing and Noise Reduction');
    viscircles(centersStrong5, radiiStrong5,'EdgeColor','b');
 %%% show total number of detections
total_detections = size(centers,1);
disp ('Tolal detections:');
disp (total_detections);
disp (thresh);
Best_Result_Nr(cnt,1) = total_detections;
Best_Result_Nr(cnt,2) = Sen_thresh;
% ############ Sobel threshold range with noise reduction and smoothing
[BW_SmNoise,thresh] = edge(im1,'sobel',Sen_thresh); %finding edges
[imx,imy]=size(BW_SmNoise); % preparing for noise reduction
msk=[0 0 0 0 0; % apply mask for noise reduction using the following matrix
    0 1 1 1 0;
    0 1 1 1 0;
    0 1 1 1 0;
    0 0 0 0 0;];
B_SmNoise = conv2(double(BW_SmNoise),double(msk));
%%Smoothing  image to reduce the number of connected components
  [centers_SmNoise, radii_SmNoise, metric_SmNoise] =
imfindcircles(B_SmNoise,[4 16]);
disp ('Detection');
toc % stop timer
%%%%%% Display Images of Sobel with noise reduction
%%%figure,imshow(B_SmNoise),title('Sobel With Smoothing and Noise
Reduction');
centersStrong5_SmNoise = centers_SmNoise(1:size(centers_SmNoise,1),:);
radiiStrong5_SmNoise = radii_SmNoise(1:size(centers_SmNoise,1));
metricStrong5_SmNoise = metric_SmNoise(1:size(centers_SmNoise,1));
%%%viscircles(centersStrong5_SmNoise,
radiiStrong5_SmNoise,'EdgeColor','b');
disp ('Tolal');
toc % stop timer
 figure,imshow(im),title('Sobel With Smoothing and Noise Reduction');
viscircles(centersStrong5_SmNoise, radiiStrong5_SmNoise,'EdgeColor','b');
 %%%%% show total number of detections
total_detections = size(centers_SmNoise,1);
disp ('Tolal detections:');
disp (total_detections);
disp (thresh);
Best_Result_Nr(cnt,1) = total_detections;
Best_Result_Nr(cnt,2) = Sen_thresh;
cnt = cnt+1;
 pause (0.25); %pause matlab for 1(second) to display result
end
save('Sobel_Best_Result.mat', 'Sobel_Best_Result') % Save pixel coordinates of
```

```
trees
% save('radii_Sobel_SNr.mat', 'radii') % save radius of trees
```

Appendix G: Code for Detection and Verification using Grey-Level Intensity Threshold

```
%%% Minimizing FPE using grey-level intensity threshold
clc;
clear all;
drawnow
warning('off','all');
%% Load image
im=imread('../img1/croped/2.jpg');
im1=rgb2gray(im); %Convert colour image to grayscale
im1Vector = im1(:); %converting im1 to vector
%% Canny technique Without Smoothing and Noise Reduction
tic % start timer
[BW,thresh] = edge(im1,'sobel',0.054); %finding edges
[centers, radii, metric] = imfindcircles(BW,[4 18]);
disp ('Detection');
toc % stop timer
disp (thresh);
%subplot(2,2,1)
imshow(im1),title('Canny Without Smoothing');
centersStrong5 = centers(1:size(centers,1),:);
radiiStrong5 = radii(1:size(centers,1));
metricStrong5 = metric(1:size(centers,1));
viscircles(centersStrong5, radiiStrong5,'EdgeColor','r','LineWidth',2)
%subplot(2,2,2)
imshow(im),title('All detections');
viscircles(centersStrong5, radiiStrong5,'EdgeColor','b','LineWidth',2);
total_detections = size(centers,1);
disp (['Tolal detections:',num2str(total_detections)])
save('centers_Canny.mat', 'centers') % Save pixel coordinates of trees
save('radii_Canny.mat', 'radii') % save radius of trees
% Create a logical image of a circle with specified
% diameter, center, and image size.
% First create the image.
[imageSizeY,imageSizeX] = size(im1);
[columnsInImage,rowsInImage] = meshgrid(1:imageSizeX, 1:imageSizeY);
grayThresh = 127.37; %set Gray brightness level to distinguish between tree
and non-tree objects, Smaller=Darker
IsTreeCount = 1;
NotTreeCount = 1;
```

```
% plotting PEs (possible eleminations)
%subplot(2,2,3)
imshow(im),title(['PE (possible eliminations),
','grayLevel=',num2str(grayThresh)]);
viscircles(centersStrong5, radiiStrong5,'EdgeColor','b','LineWidth',1);
indexGray = 0;
for Circle = 1:total_detections % choose the circle variable number
     indexGray = 0;
    % Next create the circle in the image.
    centerX = centers(Circle,1);
    centerY = centers(Circle,2);
    radius = radii(Circle,1);
    circlePixels = (rowsInImage - centerY).^2 ...
        + (columnsInImage - centerX).^2 < radius.^2;
    % circlePixels is a 2D "logical" array.
    circlePixelsVector = circlePixels(:);
    cnt = 0;
    for i = 1:size(circlePixelsVector)
         if circlePixelsVector(i) == 1
            cnt = cnt+1;
            indexGray(cnt,1) = i;
            indexGray(cnt,2) = im1Vector(i);
        end
     end
    % calculate gray mean of a circle
    centers(Circle,3) = mean(im1Vector(indexGray(:,1)));
    centers(Circle,4) = radii(Circle,1); % insert radius of the tree in pixels in the
fourth
    column of the "centers" variable
    viscircles([centerX,centerY], radius,'EdgeColor','y','LineWidth',2);
    drawnow % update the image when an wrong detection found
    pause(0.6);
    %checks if gray mean is > gray-threshold then prints a green circle
    %indicating not tree
    if centers(Circle,3) > grayThresh
        viscircles([centerX,centerY], radius,'EdgeColor','r','LineWidth',2);
        drawnow % update the image when an wrong detection found
        centers(Circle,3) % current obstacle's gray-threshold value
       % k = waitforbuttonpress ; %wait for keyboard or mouse click to continue
    else
        viscircles([centerX,centerY], radius,'EdgeColor','g','LineWidth',2);
        drawnow % update the image when an wrong detectio found
        centers(Circle,3) % current obstacle's gray-threshold value
    end
```

```
      end
toc % stop timer
saveas(gcf,'F51.png') % Save current figure to png
```

Appendix H: Obstacle and Error Counts by Default Threshold

		Canny				Sobel				Prewitt				Roberts			
Field	Total Num. of Obstacles	Correct Detection	FPE	FNE	Total Num. of Detections	Correct Detection	FPE	FNE	Total Num. of Detections	Correct Detection	FPE	FNE	Total Num. of Detections	Correct Detection	FPE	FNE	Total Num. of Detections
1	39	**2**	37	12	14	**1**	38	2	3	**0**	39	4	4	**0**	39	0	0
2	36	**27**	9	7	34	**5**	31	1	6	**2**	34	2	4	**0**	36	2	2
3	17	**5**	12	15	20	**12**	5	2	14	**13**	4	2	15	**1**	16	4	5
4	74	**51**	23	41	92	**53**	21	26	79	**59**	15	12	71	**56**	18	12	68
5	27	**27**	0	9	36	**17**	10	1	18	**16**	11	2	18	**1**	26	3	4
6	24	**18**	6	9	27	**10**	14	4	14	**9**	15	3	12	**1**	23	2	3
7	22	**18**	4	9	27	**18**	4	3	21	**18**	4	2	20	**13**	9	3	16
8	26	**15**	11	5	20	**14**	12	6	20	**13**	13	5	18	**9**	17	3	12
9	20	**13**	7	6	19	**14**	6	6	20	**15**	5	5	20	**2**	18	3	5
10	35	**25**	10	8	33	**18**	17	7	25	**18**	17	8	26	**15**	20	2	17
11	64	**42**	22	10	52	**31**	33	6	37	**34**	30	4	38	**4**	60	2	6
12	12	**4**	8	10	14	**2**	10	4	6	**2**	10	3	5	**0**	12	1	1
13	35	**19**	16	5	24	**5**	30	1	6	**5**	30	1	6	**0**	35	2	2
14	18	**6**	12	10	16	**11**	7	3	14	**10**	8	5	15	**6**	12	3	9
15	75	**54**	21	10	64	**27**	48	6	33	**26**	49	6	32	**0**	75	3	3
16	106	**7**	99	5	12	**1**	105	5	6	**2**	104	6	8	**0**	106	3	3
17	18	**13**	5	8	21	**9**	9	2	11	**7**	11	3	10	**0**	18	3	3
18	128	**106**	22	8	114	**22**	106	4	26	**15**	113	2	17	**0**	128	0	0
19	66	**46**	20	10	56	**11**	55	6	17	**11**	55	6	17	**0**	66	5	5
20	90	**7**	83	8	15	**3**	87	10	13	**2**	88	11	13	**1**	89	7	8
21	56	**54**	2	2	56	**37**	19	4	41	**36**	20	3	39	**12**	44	5	17
22	45	**25**	20	78	103	**33**	12	15	48	**33**	12	19	52	**29**	16	8	37
23	9	**8**	1	9	17	**8**	1	6	14	**8**	1	5	13	**8**	1	3	11
24	67	**1**	66	4	5	**5**	62	4	9	**4**	63	4	8	**1**	66	2	3
25	9	**6**	3	7	13	**9**	0	2	11	**8**	1	0	8	**6**	3	5	11
26	16	**12**	4	3	15	**12**	4	1	13	**14**	2	1	15	**9**	7	5	14
27	68	**30**	38	8	38	**33**	35	6	39	**34**	34	6	40	**18**	50	9	27
28	42	**21**	21	10	31	**27**	15	1	28	**26**	16	1	27	**19**	23	1	20
29	45	**23**	22	16	39	**40**	5	7	47	**40**	5	11	51	**37**	8	6	43
30	36	**6**	30	15	21	**15**	21	4	19	**17**	19	4	21	**11**	25	4	15

Table continues on the next page.

Table continued from the previous page.

		Canny				Sobel				Prewitt				Roberts			
Field	Total Num. of Obstacles	Correct Detection	FPE	FNE	Total Num. of Detections	Correct Detection	FPE	FNE	Total Num. of Detections	Correct Detection	FPE	FNE	Total Num. of Detections	Correct Detection	FPE	FNE	Total Num. of Detections
31	56	**32**	24	4	36	**32**	24	2	34	**29**	27	2	31	**8**	48	0	8
32	52	**33**	19	24	57	**47**	5	11	58	**48**	4	6	54	**28**	24	7	35
33	39	**7**	32	8	15	**4**	35	4	8	**2**	37	4	6	**1**	38	5	6
34	145	**44**	101	19	63	**65**	80	2	67	**53**	92	7	60	**28**	117	9	37
35	76	**15**	61	13	28	**18**	58	8	26	**16**	60	7	23	**7**	69	7	14
36	26	**12**	14	18	30	**9**	17	6	15	**9**	17	7	16	**1**	25	2	3
37	19	**14**	5	12	26	**19**	0	1	20	**19**	0	2	21	**19**	0	1	20
38	40	**17**	23	8	25	**11**	29	6	17	**12**	28	4	16	**7**	33	3	10
39	24	**16**	8	17	33	**22**	2	11	33	**21**	3	13	34	**21**	3	10	31
40	34	**23**	11	59	82	**11**	23	27	38	**14**	20	25	39	**1**	33	9	10
41	23	**5**	18	24	29	**5**	18	5	10	**3**	20	6	9	**0**	23	3	3
42	79	**26**	53	58	84	**38**	41	6	44	**33**	46	9	42	**1**	78	7	8
43	17	**17**	0	11	28	**17**	0	4	21	**17**	0	5	22	**13**	4	5	18
44	36	**22**	14	19	41	**13**	23	7	20	**13**	23	6	19	**2**	34	4	6
45	96	**83**	13	31	114	**45**	51	26	71	**48**	48	28	76	**22**	74	19	41
46	65	**57**	8	22	79	**56**	9	6	62	**56**	9	5	61	**28**	37	7	35
47	76	**70**	6	36	106	**56**	20	4	60	**57**	19	4	61	**10**	66	5	15
48	48	**40**	8	53	93	**30**	18	24	54	**30**	18	25	55	**21**	27	19	40
49	26	**4**	22	41	45	**3**	23	18	21	**5**	21	17	22	**0**	26	6	6
50	85	**61**	24	9	70	**54**	31	1	55	**55**	30	2	57	**8**	77	3	11
51	24	**19**	5	12	31	**14**	10	6	20	**15**	9	4	19	**0**	24	5	5

Appendix I: Obstacle and Error Counts for DLC

	After applying Threshold and Max method (DLC)								
Field Number	Edge Detection Method	Correct Detections	FPE	FNE	Total Num. of Detections	Best Threshold	Wrong Detections Eliminated After Using Threshold	FPE after Threshold	FNE after Threshold
1	Roberts	6	33	7	13	0.031	7	2	0
2	Roberts	34	2	3	37	0.06	1	1	2
3	Roberts	8	9	34	42	0.043	29	1	5
4	Canny	54	20	71	125	0.14	63	3	8
5	Canny	27	0	10	37	0.176	9	0	1
6	Prewitt	23	1	7	30	0.094	7	0	0
7	Roberts	20	2	8	28	0.047	7	0	1
8	Roberts	20	6	4	24	0.046	4	1	0
9	Canny	18	2	4	22	0.516	3	1	1
10	Prewitt	29	6	8	37	0.087	2	2	6
11	Roberts	51	13	3	54	0.071	3	1	0
12	Canny	8	4	10	18	0.138	8	1	2
13	Sobel	27	8	4	31	0.169	4	0	0
14	Canny	16	2	10	26	0.328	8	1	2
15	Canny	57	18	7	64	0.285	5	1	2
16	Canny	16	90	5	21	0.528	4	1	1
17	Prewitt	18	0	5	23	0.075	3	0	2
18	Canny	108	20	6	114	0.325	2	1	4
19	Sobel	54	12	4	58	0.073	1	3	3
20	Sobel	9	79	15	24	0.175	5	2	10
21	Canny	51	4	5	55	0.417	3	0	2
22	Canny	34	11	94	128	0.15	91	2	3
23	Sobel	9	0	7	16	0.106	6	1	1
24	Sobel	8	59	5	13	0.203	4	1	1
25	Sobel	9	0	2	11	0.152	2	0	0
26	Prewitt	14	2	4	18	0.1	3	2	1
27	Roberts	40	28	8	48	0.099	7	2	1
28	Prewitt	35	7	5	40	0.121	3	3	2
29	Roberts	40	5	11	51	0.089	10	3	1
30	Roberts	22	14	16	38	0.074	11	2	5

Table continues on the next page.

Table continued from the previous page.

	After applying Threshold and Max method (DLC)								
Field Number	Edge Detection Method	Correct Detections	FPE	FNE	Total Num. of Detections	Best Threshold	Wrong Detections Eliminated after Using Threshold	FPE after Threshold	FNE after Threshold
31	Roberts	37	19	4	41	0.117	3	1	1
32	Canny	50	2	3	53	0.32	2	1	1
33	Canny	14	25	4	18	0.497	4	1	0
34	Canny	69	76	15	84	0.462	10	5	5
35	Canny	28	48	10	38	0.496	7	3	3
36	Canny	19	7	10	29	0.185	9	3	1
37	Canny	19	0	40	59	0.122	33	7	7
38	Canny	28	12	2	30	0.756	2	3	0
39	Canny	20	4	71	91	0.139	69	1	2
40	Canny	23	11	125	148	0.197	97	11	28
41	Canny	10	13	65	74	0.244	45	5	20
42	Canny	50	29	69	119	0.281	47	7	22
43	Prewitt	17	0	15	32	0.088	13	2	2
44	Prewitt	34	2	14	46	0.099	12	3	2
45	Canny	87	9	28	115	0.414	15	5	13
46	Roberts	61	4	15	76	0.057	12	4	3
47	Canny	74	2	39	113	0.238	27	6	12
48	Canny	37	11	100	137	0.163	68	11	32
49	Canny	3	23	94	97	0.192	45	21	49
50	Roberts	72	13	9	81	0.031	7	3	2
51	Canny	21	3	12	32	0.197	9	1	3