NOVEL ENTROPY FUNCTION BASED MULTI-SENSOR FUSION IN SPACE AND TIME DOMAIN: APPLICATION IN AUTONOMOUS AGRICULTURAL ROBOT

by

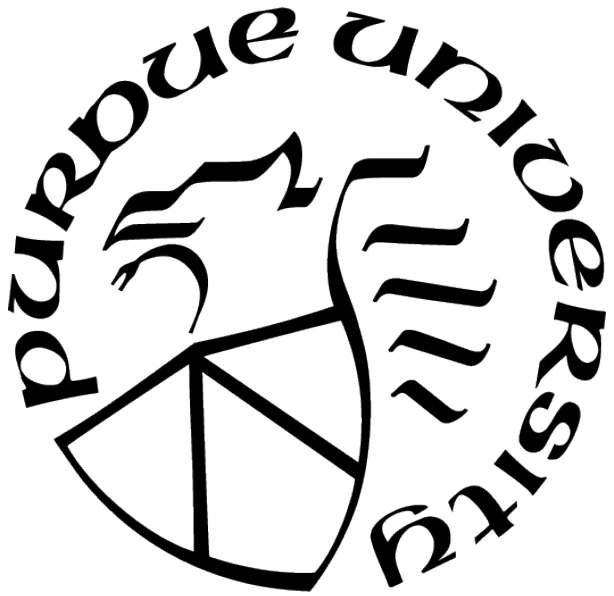
Md Nazmuzzaman Khan

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



School of Mechanical Engineering

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Sohel Anwar (Co-Chair)

School of Mechanical Engineering

Dr. Gregory Shaver (Co-Chair)

School of Mechanical Engineering

Dr. George Chiu

School of Mechanical Engineering

Dr. Mohammad Al Hasan

School of Computer & Information Science

Approved by:

Dr. Nicole L. Key

To the healthcare workers who dedicated their lives during COVID-19 pandemic.

ACKNOWLEDGMENTS

First, I want to express my gratitude to my major co-advisor Dr. Sohel Anwar. Dr. Anwar not only helped me with my research but also guided me through my PhD life. I honestly believe that I am a better person now because of his mentorship. I also want to thank my other major co-advisor Dr. Gregory Shaver. I remember the first day I met him and presented my research ideas. Dr. Shaver told me that the work I am doing is very helpful for the agriculture industry, which boosted my confidence. Dr. Shaver also introduced me to industry personnel and helped me to find a job. I also want to thank Dr. Mohammad Al Hasan for actively helping me with my research. Dr. Hasan helped me with improving our machine learning algorithms and also helped me with my writing. I want to thank Dr. George Chiu for being on my committee and help me with my research direction and goals.

I also want to thank my lab mates and people who worked on AgBot project. I have learned more from them about problem solving under real-life conditions than any courses I have taken. I want to thank the people from mechanical engineering department (Jerry, Linda, Susan and many more) who helped me throughout this whole process with guidance.

I want to thank Dr. Sebastian Raschka (Assistant Professor of Statistics at University of Wisconsin-Madison; author of the book "Python Machine Learning"), François Chollet (author of the book "Deep Learning With Python"), Dr. Adrian Rosebrock (author of the book "Deep Learning for Computer Vision with Python") and Aurelien Geron (author of the book "Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems") for writing such interesting and easily readable books. Also for open sourcing all the codes for easy implementation.

Finally, I want to thank my family (my family back at home and my wife Suchana who is also a PhD student). During the highs and lows of this PhD life they stuck to me and I am thankful for that.

TABLE OF CONTENTS

LIST OF TABLES 9

LIST OF FIGURES 10

ABBREVIATIONS 14

ABSTRACT 15

1 INTRODUCTION 17

1.1 Motivation 17

2 RELATED STUDIES 22

2.1 Crop Row Detection 22

2.2 Computer Vision in Precision Farming 23

2.3 CNN based Plant/Weed Classification 24

2.4 Sensor Fusion 26

2.5 Object Detection and Fuzzy-fusion 29

3 CROP ROW DETECTION 31

3.1 Introduction 31

3.2 Methodology 33

3.3 Pseudocode (CAROLIF) 39

3.4 Methods Used for Comparison 40

3.5 Results 42

3.5.1 Qualitative Comparison 42

3.5.2 Quantitative Comparison 43

3.5.3 Processing Time 47

3.5.4 Performance on Video Input 49

3.5.5 Effectiveness of Projective Transformation 51

3.6 KF based Crop Row Center Tracking 53

3.6.1 State Model 53

3.6.2 Measurement Equations . . . 54
3.6.3 Update Equations . . . 54
3.6.4 Correction Equations . . . 55
3.7 Conclusions . . . 56

4 CNN BASED WEED CLASSIFICATION . . . 58
4.1 Methodologies . . . 58
4.1.1 Weed Image Dataset and Image Processing . . . 59
4.2 CNN and Transfer-learning . . . 60
4.2.1 CNN Models . . . 62
6-layers CNN (Model-1) . . . 63
Transfer-learning with VGG16 (Model-2) . . . 64
Transfer-learning with InceptionResNetV2 (Model-3) . . . 65
4.3 Classification Report . . . 67
4.4 Real-time Classification from Video Input from Single Camera . . . 69
4.4.1 One Pigweed in Video (Fig. 4.8) . . . 69
4.4.2 One Ragweed and One Pigweed in Video, Separately Placed (Fig. 4.9) 69
4.4.3 Two Pigweeds and One Ragweed in Video Placed Together (Fig. 4.10) 70
4.4.4 One Pigweed and One Ragweed in Video Placed Together (Fig. 4.11) 72
4.5 Effect of Noise and Motion Blur on Classification Accuracy (model - 2) . . . 74
4.6 Conclusions . . . 74

5 ROBUST COLOR BASED WEED SEGMENTATION . . . 77
5.1 Color Based Image Segmentation . . . 77
5.2 Histogram Based Image Statistics . . . 80
5.3 Proposed Brightness/Contrast Control Steps . . . 83
5.3.1 How to Quantify ‘Good’ and ‘Bad’ Quality Image . . . 86
5.4 Conclusions . . . 88

6 DECISION LEVEL SENSOR FUSION IN SPACE AND TIME DOMAIN . . . 89
6.1 Dempster-Shafer Evidence-based Combination Rule . . . 90

6.1.1 Frame of Discernment (FOD) . . . 90
6.1.2 Basic Probability Assignment (BPA) / Mass Function . . . 90
6.1.3 Dempster-Shafer Rule of Combination . . . 91
6.1.4 Belief and Plausibility Function . . . 91
6.2 Paradoxes (Source of Conflicts) in DS Combination Rule . . . 92
6.2.1 Completely Conflicting Paradox: . . . 93
6.2.2 "One Ballot Veto" Paradox: . . . 93
6.2.3 "Total Trust" Paradox: . . . 94
6.3 Eliminating the Paradoxes of DS Combination Rule . . . 94
1. Modification of DS Combination Rule . . . 94
2. Revision of Original Evidence before Combination . . . 95
3. Hybrid Technique Combining both Modification of DS Rule and Original Evidence . . . 96
6.4 Entropy in Information Theory under DS Framework . . . 97
6.4.1 Properties of Proposed Entropy Function . . . 98
6.5 Proposed Steps to Eliminate Paradoxes in Space Domain . . . 100
6.6 Proposed Steps for Time-domain Data Fusion . . . 105
6.6.1 Anti-disturbing Ability and Transition Property of Proposed Algorithm 110
6.6.2 Modification of BPA for CNN Based Object Classification under DS Framework . . . 113
6.6.3 Effect of Number of Time-steps (fusion-time) on Fused Output . . . 117
6.7 Application of the Proposed Algorithm . . . 117
6.7.1 Fusion when Faulty Sensor is Present in Sensor-array . . . 118
6.7.2 Fusion when Weed is Partially Occluded . . . 120
6.8 Conclusions . . . 123

7 IMPROVING THE ROBUSTNESS OF OBJECT DETECTION THROUGH A MULTI-CAMERA BASED FUSION ALGORITHM USING FUZZY LOGIC . . . 125
7.1 Objectives . . . 125
7.2 Projective Transformation . . . 126

7.2.1 Homography Matrix Calculation . . . 129
7.3 CNN based Weed Detection from Classification . . . 130
7.3.1 IOU Overlap Calculation . . . 131
7.4 Fuzzy Logic . . . 133
7.4.1 Fuzzy Steps and Rule-set: . . . 137
7.5 Results . . . 139
7.5.1 Scenario 1: High Confidence . . . 140
7.5.2 Scenario 2: OK Confidence . . . 141
7.5.3 Scenario 3: Low confidence . . . 142
7.6 Conclusions . . . 143
8 CONCLUSIONS . . . 146
9 FUTURE WORKS . . . 149
REFERENCES . . . 151
A CODE: CHAPTER 3 - CROP ROW DETECTION . . . 164
B CODE: CHAPTER 5 - STATISTICS PARAMETERS OF IMAGE . . . 169
C CODE: CHAPTER 6 - MULTI-SENSOR TIME DOMAIN FUSION . . . 171
D CODE: CHAPTER 6 - MULTI-SENSOR SPACE DOMAIN FUSION . . . 174
VITA . . . 178

LIST OF TABLES

3.1 Tuning parameters for CAROLIF. 39

3.2 IOU value for different algorithms. 46

3.3 Processing time for different algorithms. 49

3.4 Performance of CAROLIF on real-time video. 52

4.1 Weed image dataset. 60

4.2 Confusion matrix based on test image set (Model-1). 64

4.3 Confusion matrix based on test image set (model-2). 65

4.4 Confusion matrix based on test image set (model-3). 66

4.5 Sample confusion matrix. 67

4.6 Classification report. C = Cocklebur, R = Ragweed and P = Pigweed. Inference time (classifying image to infer a result) tested on a core-i5, 8gb ram machine. . 68

5.1 Mean, variance, skewness, kurtosis values for 'good-quality' images in the three R, G, B spectral channels. 86

5.2 Mean, variance, skewness, kurtosis values for 'bad-quality' images in the three R, G, B spectral channels. 87

6.1 Bel and Pl values for Example 6.1 92

6.2 Bel and Pl values for example 6.6 99

6.3 Bel and Pl values for Example 6.7 102

6.4 Evidence combination results based on different combination methods for Example 6.7 104

6.5 Input data of each time step for Example 6.8 110

6.6 Data combination results based on different combination methods for Example 6.8 111

6.7 Input data of each time step for Example 6.9 113

6.8 Classification report of CNN classifier (Model-2). 115

7.1 Optimization methods and reprojection error. 130

7.2 Fuzzy rules. 142

LIST OF FIGURES

1.1 Retrofitted Yamaha ATV - 2018 Agricultural Robot (AgBot). 18

1.2 Graph showing herbicide resistance increases within selected countries. 19

1.3 Research contributions and relation between them. Different color shows different aspects of contribution. 20

2.1 Sample images from different plant leaf datasets. 26

2.2 Data fusion at three different levels. (**a**) Signal-level fusion, (**b**) feature-level fusion, and (**c**) decision-level fusion. 28

3.1 Steps of the proposed CAROLIF algorithm with images. 33

3.2 (a) Projective transformation is used from Agricultural robot to transform camera 1 view to virtual camera 2 view. (b) For two camera planes, image planes are related by projective transformation when all world points are co-planer. (c) Original image from camera 1 view with boundary points. (d) Cropped image using boundary points. (e) Transformed cropped image from camera 1 view to camera 2 view. 34

3.3 Comparison of Kmeans, MeanShift, Agglomerative and HDBSCAN clustering algorithm on crop row detection. Run-time indicated on the top left corner of each image. Different colors indicate different clusters. Black colors indicate outliers. 37

3.4 Example cases for easy and challenging scenario. 43

3.5 Comparison of crop row detection in challenging scenarios for Hough transform, Sliding window, TMGEM, Cluster-Least square and CAROLIF (proposed) methods with ground truth results.(Row 1) intermittent, very early crop growth, no weed. (Row 2) early crop growth stage, high weed pressure, crop rows are connected, crop rows and weed indistinguishable. (Row 3) early crop growth stage, exceptionally big intermittent weed, curved crop rows. (Row 4) medium crop growth, crop missing from rows, no weed. (Row 5) early crop growth stage - intermittent, concentrated weed growth. 44

3.6 IOU between a ground truth bounding box (in green) and a detected bounding box from algorithm (in red). 45

3.7 Actual bounding boxes. Red lines show ground truth detection and blue lines show algorithm detection of crop rows. Green bounding boxes capture the height and width of ground truth lines. Pink bounding boxes capture the height and width of algorithm detection lines. 45

3.8 Box plot comparison of IOU values for different methods under easy and challenging scenario. 46

3.9 (left side) Ground truth value and boundary boxes from CRBD dataset. (right side) CAROLIF detection and boundary boxes. 47

3.10 Processing time of each step of CAROLIF crop row detection algorithm. ROI size (120 by 80) pixels. 48

3.11 Performance of CAROLIF on real-time video captured from an agricultural vehicle. 50

3.12 Comparison of crop row detection results with and without projective transformation. Four different scenarios are chosen. (T_1) and (T_2): sunny, no weed, no shadow, intermittent crop growth. (T_3): sunny, no weed, shadow present. (T_4): high weed pressure, shadow, intermittent crop growth. 51

3.13 KF based crop row center tracking. Orange colored 'plus' signs show the calculated crop row center from CAROLIF output at time t. 56

4.1 Classification and spray system pipeline of AgBot. 59

4.2 Sample images of the weed dataset. 60

4.3 Simple representation of Transfer-learning. 62

4.4 6-layer end-to-end CNN architecture(Model-1). 64

4.5 Training and validation accuracy (Model-1). 65

4.6 Training and validation accuracy (model-2). 66

4.7 Training and validation accuracy (model-3). 67

4.8 Classification accuracy from video input using transfer-learning VGG16 (model-2). Only one Pigweed plant on video. 70

4.9 Classification accuracy from video input using model-2. One Pigweed plant and one Ragweed plant separately placed. 71

4.10 Classification accuracy from video input using model-2. Two Pigweed plants and one Ragweed plant on video, placed together. 72

4.11 Classification accuracy from video input using model-2. One Pigweed plant and one Ragweed plant on video, placed together. 73

4.12 Effect of Gaussian and Salt-and-Pepper noise on classification accuracy. 75

4.13 Effect of Gaussian and Salt-and-Pepper noise on classification accuracy. 76

5.1 Block diagram of proposed image enhancement process. Blue arrow shows how usually spray decision is made. Red arrow shows the redundant system for bad quality image. 78

5.2 (a)ExG, (b)ExGR, (c)CIVE and (d)GREEN based image segmentation. 80

5.3 ExG, ExGR, CIVE and GREEN based image segmentation performance under different lighting conditions. 81

5.4 A good-quality image with histogram and statistics parameters for R, G and B channels of the image. 83

5.5 A bad-quality image with histogram and statistics parameters for R, G and B channels of the image. 84

5.6 Original bad-quality image, bad-quality image after CLAHE, visible spectral-index based image segmentation (GREEN) on CLAHE applied image, histogram of original image, histogram of CLAHE image. 85

6.1 Comparison of convergence of evidence m(A) for Example 6.7. 106

6.2 Simple representation of sensor-fusion in space and time domain. 107

6.3 Comparison of anti-disturbing ability of several combination rules for Example 6.8. 112

6.4 Transition property of the proposed algorithm for Example 6.9. 114

6.5 Real-time weed classification from video input using CNN classifier (Model-2). Classification % is showing CNN output of video feed at each time step. This CNN output is used as BPA in fusion algorithm. 115

6.6 Effect of considering precision and recall on updating BPA on real-time weed classification. Classification % are showing BPA from equations (6.29) and (6.30) [top figure]. Time-domain fusion of updated BPA for $t_s = 5$. Classification % showing fused results when BPA from equations (6.29) and (6.30) goes through the proposed fusion algorithm [bottom figure]. 116

6.7 Effect of fusion-time (t_s) during time-domain sensor fusion on real-time weed classification from video input. Classification % showing fused results when BPA from Fig. 6.5 goes through the proposed time-domain fusion algorithm (step 1 - step 8). from section 6.6 118

6.8 Placement of multiple cameras on an AgBot. 119

6.9 Reduced unstable classification with time domain sensor fusion. Reduce the effect of faulty sensor evidence with space domain sensor fusion. 120

6.10 Ground truth value for Ragweed and Pigweed at the top. (a) Classification accuracy from left, center and right camera for Pigweed. (b) Classification accuracy from left, center and right camera for Rigweed. (c) Reduced unstable output (smooth curve) with time domain fusion. Each line shows the classification % of a specific weed for a specific camera. (d) Eliminated the effect of faulty sensor (right camera) evidence on final classification output with space domain fusion. 121

6.11 Improved classification for partially occluded weed data with space domain sensor fusion. 122

6.12 Ground truth value for Ragweed and Pigweed at the top. (a) Classification accuracy from left, center and right camera for Pigweed. (b) Classification accuracy from left, center and right camera for Rigweed. (c) Reduced classification error (smooth curve) with time domain fusion. (d) Eliminated the effect of partial occlusion on final classification output with space domain fusion. 122

7.1 (a) For multiple cameras, image planes are related by projective transformation when all world points are co-planer. (b) Actual setup of two cameras with co-planer checkerboard points. 128

7.2 Checkerboard corner points for left camera (camera 2). 131

7.3 Checkerboard corner points for right camera (camera 1). 132

7.4 Reprojection of camera 2 image to camera 1 image using homography matrix. . 133

7.5 Steps to create a object detector from CNN classifier. 134

7.6 How BB overlap changes when weed is placed at different distance from camera. "Too far", "Far", "Close" and "Too close" represents the distance of the weed from camera. "After reprojection" shows the BB overlap when left camera image is reprojected using H. BB size, width: 500 pixels; height: 350 pixels. 135

7.7 IOU overlap value with respect to distance from camera.(a) BB size, width: 500 pixels; height: 350 pixels.(b) BB size, width: 450 pixels; height: 300 pixels. . . . 136

7.8 Fuzzy set for weed bounding box distance in camera view for right camera. [Input]. 137

7.9 Fuzzy set for weed bounding box distance in camera view for left camera. [Input]. 138

7.10 Fuzzy set for bounding box overlap between left and right camera [Input]. . . . 139

7.11 Fuzzy set for fusion confidence [Output]. 140

7.12 Basic configuration of the fuzzy system. 141

7.13 Fuzzy confidence score measurement (High confidence). 143

7.14 Fuzzy confidence score measurement (OK confidence). 144

7.15 Fuzzy confidence score measurement (Low confidence). 145

ABBREVIATIONS

AgBot	Agricultural Robot
HDBSCAN	Hierarchical Density-Based Spatial Clustering of Applications with Noise
RANSAC	Random Sample Consensus
CAROLIF	Clustering Algorithm based RObust LIne Fitting
IoU	Intersect over Union
TMGEM	Template Matching followed by Global Energy Minimization
KF	Kalman Filter
CNN	Convolutional Neural Network
DS	Dempster-Shafer
BB	bounding-box
RGB	Red, Green and Blue channel
LIDAR	Light Detection and Ranging
TOF	Time of Flight
BB	Bounding Box
R-CNN	Regions with CNN features
ROI	Region of Interest
IoU	Intersect over union
CLAHE	Contrast Limited Adaptive Histogram Equalization
HE	Histogram Enhancement
BPA	Basic Probability Assignment
FOD	Frame of Discernmen

ABSTRACT

How can we transform an agricultural vehicle into an autonomous weeding robot? A robot that can run autonomously through a vegetable field, classify multiple types of weeds from real-time video feed and then spray specific herbicides based of previously classified weeds. In this research, we answer some of the theoretical and practical challenges regarding the transformation of an agricultural vehicle into an autonomous weeding robot.

First, we propose a solution for real-time crop row detection from autonomous navigation of agricultural vehicle using domain knowledge and unsupervised machine learning based approach. We implement projective transformation to transform camera image plane to an image plane exactly at the top of the crop rows, so that parallel crop rows remain parallel. Then we use color based segmentation to differentiate crop and weed pixels from background. We implement hierarchical density-based spatial clustering of applications with noise (HDBSCAN) clustering algorithm to differentiate between the crop row clusters and weed clusters. Finally we use Random sample consensus (RANSAC) for robust line fitting through the detected crop row clusters. We test our algorithm against four different well established methods for crop row detection in-terms of processing time and accuracy. Our proposed method, Clustering Algorithm based RObust LIne Fitting (CAROLIF), shows significantly better accuracy compared to three other methods with average intersect over union (IoU) value of 73%. We also test our algorithm on a video taken from an agricultural vehicle at a corn field in Indiana. CAROLIF shows promising results under lighting variation, vibration and unusual crop-weed growth.

Then we propose a robust weed classification system based on convolutional neural network (CNN) and novel decision-level evidence-based multi-sensor fusion algorithm. We create a small dataset of three different weeds (Giant ragweed, Pigweed and Cocklebur) commonly available in corn fields. We train three different CNN architectures on our dataset. Based on classification accuracy and inference time, we choose VGG16 with transfer learning architecture for real-time weed classification.

To create a robust and stable weed classification pipeline, a multi-sensor fusion algorithm based on Dempster-Shafer (DS) evidence theory with a novel entropy function is proposed.

The proposed novel entropy function is inspired from Shannon and Deng entropy but it shows better results at understanding uncertainties in certain scenarios, compared to Shannon and Deng entropy, under DS framework. Our proposed algorithm has two advantages compared to other sensor fusion algorithms. First, it can be applied to both space and time domain to fuse results from multiple sensors and create more robust results. Secondly, it can detect which sensor is faulty in the sensors array and compensate for the faulty sensor by giving it lower weight at real-time. Our proposed algorithm calculates the evidence distance from each sensor and determines if one sensor agrees or disagrees with another. Then it rewards the sensors which agrees with another according to their information quality which is calculated using our novel entropy function. The proposed algorithm can combine highly conflicting evidences from multiple sensors and overcomes the limitation of original DS combination rule. After testing our algorithm with real and simulation data, it shows better convergence rate, anti-disturbing ability and transition property compared to other methods available from open literature.

Finally, we present a fuzzy-logic based approach to measure the confidence of the detected object's bounding-box (BB) position from a CNN detector. The CNN detector gives us the position of BB with percentage accuracy of the object inside the BB on each image plane. But how do we know for sure that the position of the BB is correct? When we are detecting an object using multiple cameras, the position of the BB on the camera image plane may appear in different places based on the detection accuracy and the position of the cameras. But in 3D space, the object is at the exact same position for both cameras. We use this relation between the camera image planes to create a fuzzy-fusion system which will calculate the confidence value of detection. Based on the fuzzy-rules and accuracy of BB position, this system gives us confidence values at three different stages ('Low', 'OK' and 'High'). This proposed system is successful at giving correct confidence score for scenarios where objects are correctly detected, objects are partially detected and objects are incorrectly detected.

1. INTRODUCTION

With the world's population exceeding 7 billion and still growing, there is an enormous challenge to produce enough food. The growing population has growing food demand and United Nation's (UN) Sixty-fourth General Assembly says that we need to double the food production by 2050 to meet the need [1]. With increasing costs of manual labor and chemicals used in the current agricultural practices, an automated agricultural system may not only be economically feasible, but also environmentally friendly.

Precision farming improves crop yields and makes farming environmentally friendly. Precision farming reduces environmental impact and lowers the usage of artificial chemicals. Automation and robotics research in farming domain increased due to precision farming. The overall goal is to increase the efficiency of the farming process and reduce herbicide usage. Increase in herbicide resistance from various species of weeds have been observed over the years [2]. To reduce labor, cost, and herbicide resistance of weeds; interest in agricultural robots capable of autonomous weed detection and elimination has been growing. To integrate innovative technologies to improve observation, intervention, analytic, and data storage in agricultural robotics, an annual agricultural robotic competition dubbed "AgBot Challenge" was initiated in 2016 at Rockville, Indiana (http://www.agbot.ag/). The goal is to do the following:

- Autonomously drive 1000-ft rows and autonomously turn at each end.
- Autonomously observe crop plants growth and fertilize. Identify three common weeds and spray herbicides based on detection.

Such a robot built by retrofitting a Yamaha Wolverine 4x4 vehicle with sensors and actuators in the Mechatronics and Automotive Research Lab is shown in Fig. 1.1.

1.1 Motivation

Many farms are using a zero-tillage approach to weed management where unwanted plants are typically eliminated using a variety of different available herbicides. When applied in a blanket fashion, herbicides are applied indiscriminately to the entire field with no

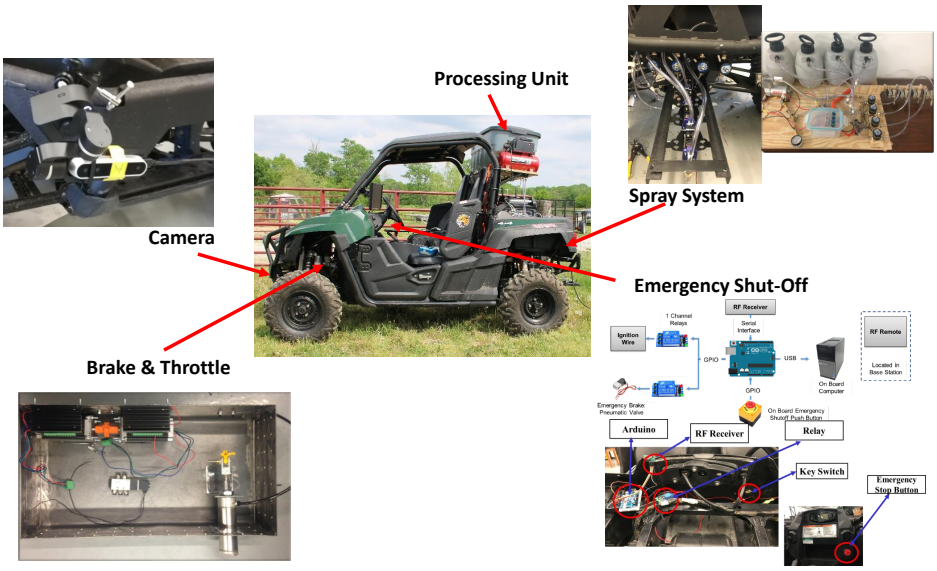


Figure 1.1. Retrofitted Yamaha ATV - 2018 Agricultural Robot (AgBot).

specialization for individual plants. This widespread use of herbicides, however, has led to a steady increase in herbicide resistance from various species of weeds over the years. This can be seen graphically in Fig. 1.2, which shows cases of herbicide resistances increasing over time in selected countries since 1950, where each case as defined by [3] is "A unique pairing of species and site of action (herbicide method of removal)" [4].

How the herbicide is sprayed is a important factor in optimizing herbicide performance [5]. Herbicide spraying is a low precision method and only a small fraction reaches the actual weeds. [6]. Also, pesticide application has been described as the "least effective industrial process on earth" [7]. Herbicide resistance is a natural process and usually happens due to over-application of herbicides. It happens through "repeated herbicide use to a point where the weed population is no longer controlled by the herbicide at the recommended rate under agricultural conditions" [8]. Herbicide resistant weeds can't be eliminated but they can be controlled. By rotating herbicide sites of action, herbicide mixture, herbicide resistant crops and combination of all is used as integrated weed management. As a result, early individual weed detection with automated herbicide spray will limit the use of herbicide use and can be used as an integral part of weed management system.

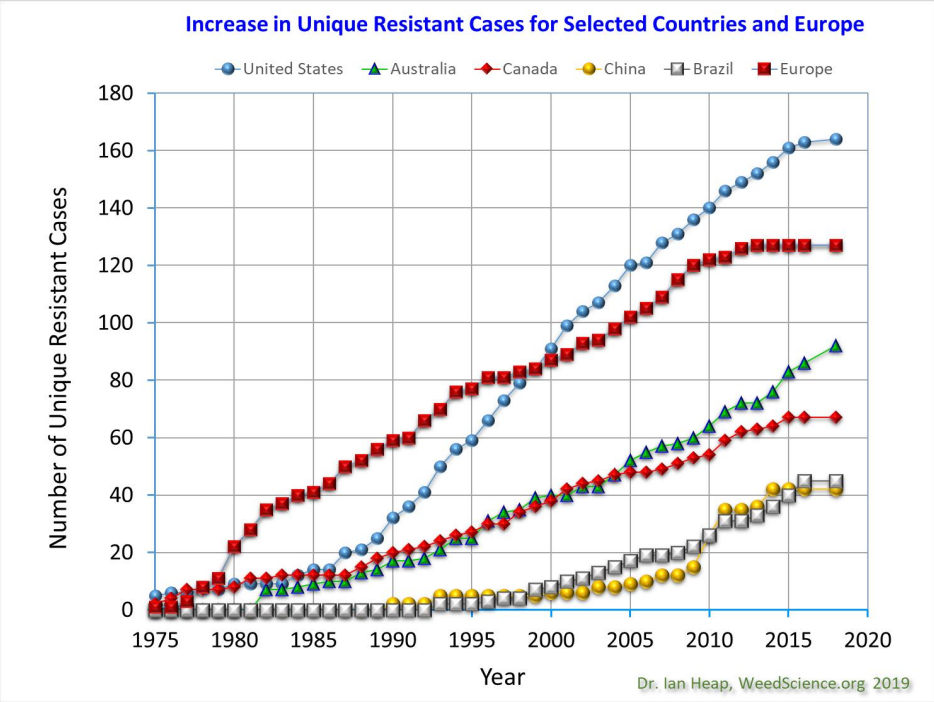


Figure 1.2. Graph showing herbicide resistance increases within selected countries.

With rapidly increasing global population, the demand for higher crop yield is also increasing rapidly. Weeds are one of the major culprits behind lower crop yield. They grow randomly in field and compete with crops for water, nutrients and sunlight. To eliminate weeds and reduce uncontrolled spray of herbicides, real-time detection of weeds with high accuracy using low cost sensors is needed. Herbicide resistance is costly in terms of weed control and yield loss. As a result, research is needed in precision weed management where each species weed is classified and treated individually.

Conventional machinery sprays herbicides uniformly to the total field, which results in high cost on herbicides [9] and catastrophic environmental pollution [10]. But research suggests that weeds grow in patches, not homogeneously in fields. Marshall et al. [11] showed that between 24% and 80% of the sample area was free of grass weeds, when he investigated the presence of three different grass in crop fields. Jhonson et al. [12] investigated 7 maize fields and 5 soybean fields for weeds. They reported that about 30% of the crop field area was free of broad-leaf weeds. They also discovered that about 70% area was free of grass weeds where no herbicide was applied. In [13] and [14], the authors stated that, based on

weed pressure and number of patches, reductions between 42% (soybean and maize) and 84% (maize) in the amount of applied herbicide could be achieved. This suggests that, if individual weeds can be classified from continuous video feed, farmers can fine-tune (or an autonomous system can automatically adjust) the rate of herbicide application. Such an effort will reduce herbicide-resistant weeds and treatment cost significantly.

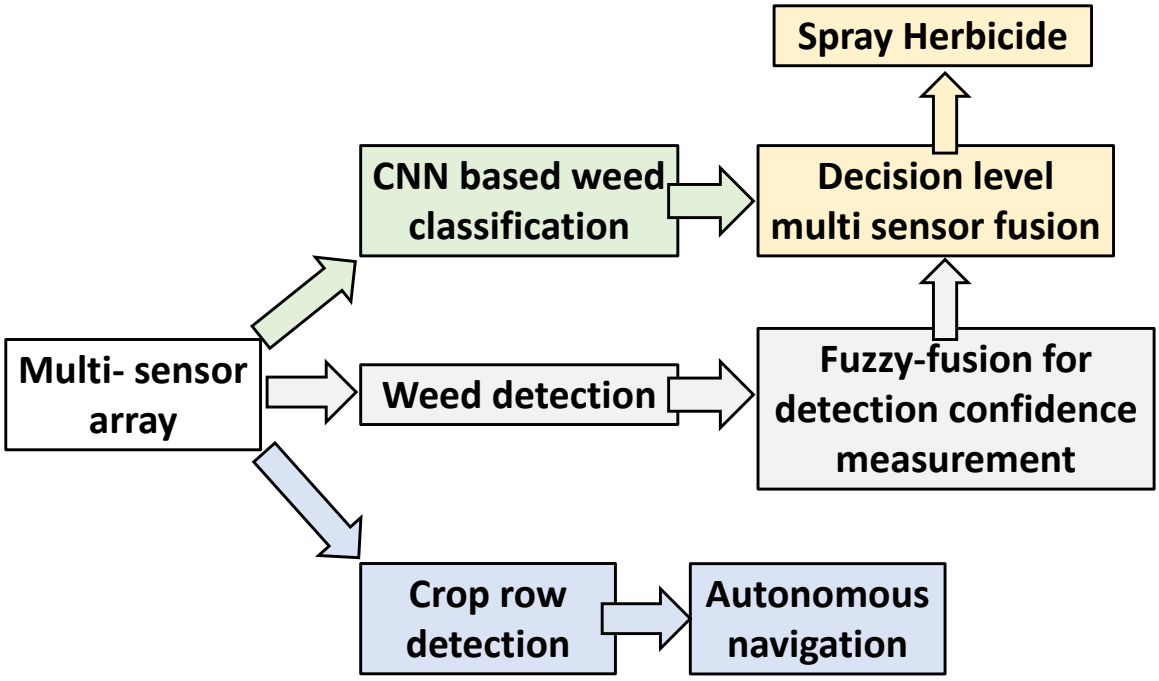


Figure 1.3. Research contributions and relation between them. Different color shows different aspects of contribution.

In this research, we propose a state-of-the-art individual weed classification system with decision level sensor fusion. Individual weed classification is a complex problem under adverse weather condition and few researches have been conducted for real-time application. The proposed pipeline has a fail-safe for weed/non-weed classification for adverse weather condition. Multi-sensor input improves the overall classification accuracy by increasing field of view, adding multiple classifiers into the system and reducing the possibility of occlusion. A real-time decision level sensor fusion algorithm is proposed to effectively combine all the sensor outputs into a valid object classification. We also include a real-time crop row detection system for autonomous navigation of the vehicle. And finally, we add a fuzzy logic based

confidence measurement system for CNN based object detection to address the sub-optimal detection of a CNN detector under noise or vibration. Fig. 1.3 shows the complete pipeline of the proposed system and contribution of our research. Innovations of our work will try to answer the following questions:

1. How to develop a real-time crop row detection system for autonomous navigation of AgBot?

2. How to classify individual weeds of similar nature (color, shape, height) with reasonable accuracy ($> 95\%$) from a small weed dataset?

3. What steps are needed to deploy a weed classification system on an AgBot using reasonable hardware? How to make classification real-time and robust against noise, motion blur and occlusion?

4. How evidence-based decision level sensor fusion can improve the overall classification accuracy beyond state-of-the-art?

5. What are the effects of evidence-based sensor fusion in both space and time domain?

6. How do we know the position of the BB given by a detector is correct? How can we measure a confidence score of the BB position given by the detector?

2. RELATED STUDIES

2.1 Crop Row Detection

Autonomous navigation in agriculture has several advantages, like, reduce operator fatigue, improve profit and efficiency, and enhance operation safety. To support autonomous navigation, a crucial task is to develop the capability of computer vision based detection of crop rows from image/video data. To address this, several methodologies have already been proposed [15]–[18]; however, our analysis on these existing methodologies reveal that their detection quality deteriorates significantly, when the crop row image is complex. Specifically, the complexity arises due to two reasons: first, missing of crops in some segment of the crop row causing discontinuity in a crop row, and the second, significant weed growth in the area between the crop rows. When an image exhibits both the above reasons simultaneously, distinguishing crop rows from the weed becomes very difficult, and mostly all the existing methods fail to achieve a satisfactory performance in solving the crop row detection task.

The desiderata of a practical crop row detection method that works satisfactorily in real-life deployment are below: (1) It is capable of detecting crop rows even with high weed pressure; (2) It is applicable at different types of crop fields; (3) It is capable to detect crop rows at different crop growth stages; (4) It is capable to detect straight and reasonably curved crop rows; and finally, (5) The processing time of detection on an off-the-shelf computer satisfy real-time requirements.

Basso and de Freitas [15] used a filtered Hough transform [16] method and achieved around 30 frame per second(FPS) for 320x240 image resolution on RPi-3B embedded system. Although they showed detailed results about how the algorithm performed for different image sizes and frame rates (max speed used for good detection: 2 m/s), no detailed calculation is showed about the effect of weed pressure and crop missing on row detection accuracy.

Zhang et al. [17] proposed a double thresholding (combining OTSU method with particle swarm optimization) on segmented image with linear regression for line fitting. They said, due to double thresholding their method can separate weed from crop rows. But how this method performs under high weed pressure is not shown in detail. Also, least squares fitting is highly affected by presence of weed in image.

Sainz-Costa et al. [18] analysed video sequences to detect crop rows. They used a five step process to detect crop rows. First, they segmented the image. Then after clearing from noise, they divided the image in horizontal stripes. Then from each strip crop centers are extracted. This worked reasonably well in low weed pressure.

Vidovic et al. [19] applied the vanishing point principle to detect parallel crop rows. Dynamic programming based optimization was applied for straight and curved crop row detection. But for an image with resolution of 640x480 needed around 5 seconds of processing time.

Tracking similar looking objects in a dynamic system increases complexity in computer vision application. To make an overall robust system, tracking algorithm can be incorporated with crop row detection. According to [20], three most common methods for object tracking are: point tracking, kernel tracking and silhouette tracking. Point based tracking is very useful in tracking small objects (like crop row centers) in video sequence. Within point tracking; kalman filter, particle filter and multiple hypothesis tracking are the most common. Kalman filter is a good option where number of objects to track are around 5 [21].

2.2 Computer Vision in Precision Farming

Automated crop yield estimation for corn [22], vineyard [23], pineapple [24] and apple [25] using off the shelf RGB (red, green and blue) or stereo camera can be found in recent literature. But the main constraints are weather conditions, specular reflection and color heterogeneity. Artificial lighting and artificially placed landmarks [25] has been proposed to overcome these constraints. In 3D vision application for plant phenotyping, it can be seen that shadowing devices are commonly used to maintain constant lighting conditions when used outside [26]. No artificial lighting is used for inside (greenhouse) application.

Piron et al. [27] used specific light based coding to detect crop and weed at early stage. They used the difference in height as the main feature for differentiation. Some limiting factors were dynamic range, reflection, occlusion, high weed growth stage and low crop growth. Šeatovic et al. [28] used depth camera to detect broad-leafed weed in real-time. They also integrated a spraying system. To reduce noise, they used cover to take images and used clear soil background. This doesn't reflect real-life scenario. As a result, their detection

decreased when noise was present in background. Spectral information gathered from ground or air can be useful for vegetation detection. Strothmann et al. [29] used three wavelengths (05, 532, and 650 nm) in a triangulation system to create 3D shape reconstruction and get the reflectance information. This method was somewhat successful in detecting objects with high water content at wavelength between 600 - 1000 nm.

Weiss et al. [30] used a 3D LIDAR (pulse modulation-FX6 LIDAR by Nippon Signal) for crop detection in indoor environment with different machine learning classifiers. Logistic regression showed classification precision of nearly 99% for six different plant species. This indicates that the dataset may have been too simple to mimic real-life scenarios. When they used the same 3D LIDAR for plant detection and mapping at outdoor conditions, they achieved accuracy of about 60%. The authors indicated the sensor's low pixel resolution (29 x 59 pixels) as one of the main reasons for low accuracy. The authors considered 3D LIDAR as the most promising sensor technology for agricultural robotics [26] due to the advantages of the sensor (reliability under different light and weather conditions) over camera. Several applications of LIDAR and TOF cameras for crop density measurements [31], inter-plant spacing [32], automatic pruning [33] can be found in literature.

2.3 CNN based Plant/Weed Classification

Recent advancement in machine learning domain can now replace feature based plant catalog and detection [34] which saves cost and time. In recent times, CNN classifier has attracted substantial attention in the weed classification task due to its (CNN) superior performance in large-scale image classification. For instance, Potena et al. [35] used two different CNNs, to process RGB and near infrared (NIR) images for crop and weed detection. A smaller CNN was used for vegetation segmentation, and a deeper CNN was used for crop and weed classification. The classifier yielded the best mean average precision of 98.7%. Yu et al. [36] used VGGNet architecture to detect weeds in bermuda grass. They achieved over 95% F_1 score and outperformed GoogleNet architecture. Suh et al. [37] compared the performance of transfer learning with six different CNN architectures for sugar beet and volunteer potato detection. VGG19 with transfer learning showed the highest performance for this binary classification task with over 98% accuracy. However, Suh et al. stated

that classification is only one step in weed detection, and the real-time performance of a complete pipeline for weed detection may be poorer. Olsen et al. [38] trained InceptionV3 and ResNet50 CNN architectures from scratch with 17509 labeled images containing 8 different weed classes. They achieved over 95% accuracy and the inference-time of their system was reasonable for real-time applications. However, deep CNNs are particularly susceptible to blur and noise [39]. As a result, their classification accuracy can deteriorate rapidly when detecting weeds from video feed in presence of blur, noise, occlusion, vibration and different lighting conditions. To spray herbicides correctly, a steady signal is needed from classifiers which should be robust against these adverse conditions. A way to achieve this goal is to use multiple sensors and a sensor fusion architecture, which combines classification results from each sensor at the decision level.

Transfer learning means reusing a pre-trained machine learning model to complete a similar but new task. Jeon and Rhee [40] used transfer learning with pre-trained GoogleNet architecture to detect plant leaves with Flavia dataset [41]. Different image scales and image colors are used as image augmentation techniques to increase the variability of the dataset. They achieved over 99% accuracy for leaves with no damage and 94% accuracy for leaves with 30% damage. Kaya et al. [42] tested four different transfer learning model with four different publicly available plant leaf datasets. They compared their results with other Deep Neural Nets (DNNs) and traditional machine learning based leaf classification methods. Datasets used in this study are, Flavia (32 species and 1900 images), Swedish Leaf Dataset [43] (15 tree classes, 1125 images), UCI leaf dataset [44] (40 species and 443 images) and PlantVillage dataset [45] (14 species, 38 classes, 54306 images). Their comparison showed that deep learning based models have supiority in classification accuracy compared to traditional feature-based machine learning models. They also showed that CNN models with transfer learning give better results than end-to-end CNN models. Fig. 2.1 shows sample images from these datasets. It's clear from the figure that they are clean images with no luminance variation and they don't usually represent a plant which is seen from an AgBot on a field. As a result, these datasets are quite good for model testing and evaluation but their performance may vary on real plant images taken from an AgBot. As a result,

research is needed on a dataset which represent actual field scenarios and implemented on data similar to on field conditions.

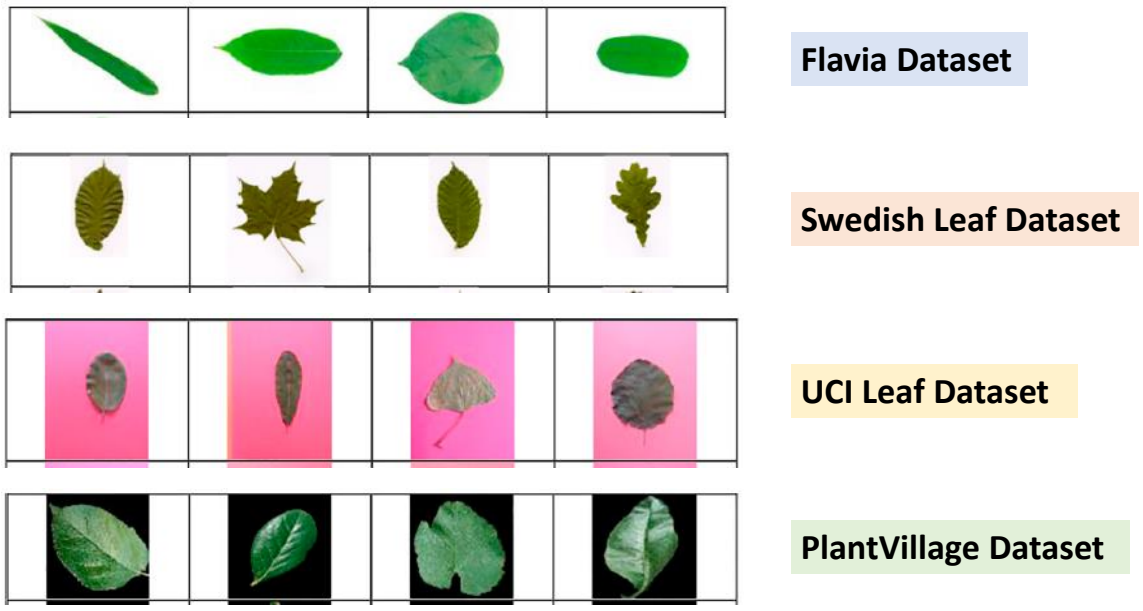


Figure 2.1. Sample images from different plant leaf datasets.

2.4 Sensor Fusion

Multi-sensor fusion overcomes the limitations of single information source by combining information from multiple sensors in a meaningful way. Based on the identified strengths and weaknesses of previous work, a principled definition of information fusion is proposed in [46] "Information fusion is the study of efficient methods for automatically or semi-automatically transforming information from different sources and different points in time into a representation that provides effective support for human or automated decision making." A multi-sensor system has two distinct advantages over a single sensor system when used with proper fusion algorithm:

- A single sensor is not suitable for redundant system. If that sensor is faulty, it will shut down the whole system. A multi-sensor system on the other hand, provides diverse and

redundant information. Also, a proper sensor fusion algorithm can detect the faulty sensor from the system and shut it down.

- Multiple heterogeneous sensors can provide diverse set of information and complement each others limitations. It helps to create a system more robust against noise and interference.

A robust decision level fusion algorithm can fuse data from multiple sensors and achieve better object classification than single sensor. As shown in Fig. 2.2, in terms of where fusion in happening, there are three types of sensor fusion. At signal level, unfiltered signals from sensors (example: image pixels from camera) are fused . At feature level, features are calculated from sensor signals. Moment or area calculation of blobs from image pixel data are one example of feature calculation. Then features are fused to achieve the objective of the system. At decision level, object is already identified from each sensor. Then decision is made by fusing the detected object from multiple sources and creating one detection ID. Final classification accuracy is increased by taking advantage of the best classification result from one of the sensors from the sensor array [47]. At decision level, a crucial issue in multi-source information fusion is, how to represent and determine the imprecise, fuzzy, ambiguous, inconsistent, and even incomplete information [48]. As a tool to manipulate an uncertain environment, Dempster–Shafer evidence theory is an established system for uncertainty management [49], [50].

Dempster-Shafer theory (short for DS theory), also called belief function theory, as introduced and developed by Dempster and Shafer [51], [52], has emerged from their works on statistical inference and uncertain reasoning. As a tool to manipulate an uncertain environment, DS evidence theory established a rounded system for uncertainty management and information fusion [53]–[56]. Information can be fused in both space and time domain. In space domain, information is fused at each time step. Information from multiple sources are gathered at each time step and fused to reduce the effect of faulty sensor. However, in real-time application of multi-sensor systems, time-domain information is also generated. Here information from single sensor and multiple time steps are fused. At a certain time-step noisy, distorted or even wrong results are obtained due to environmental noise or wrong

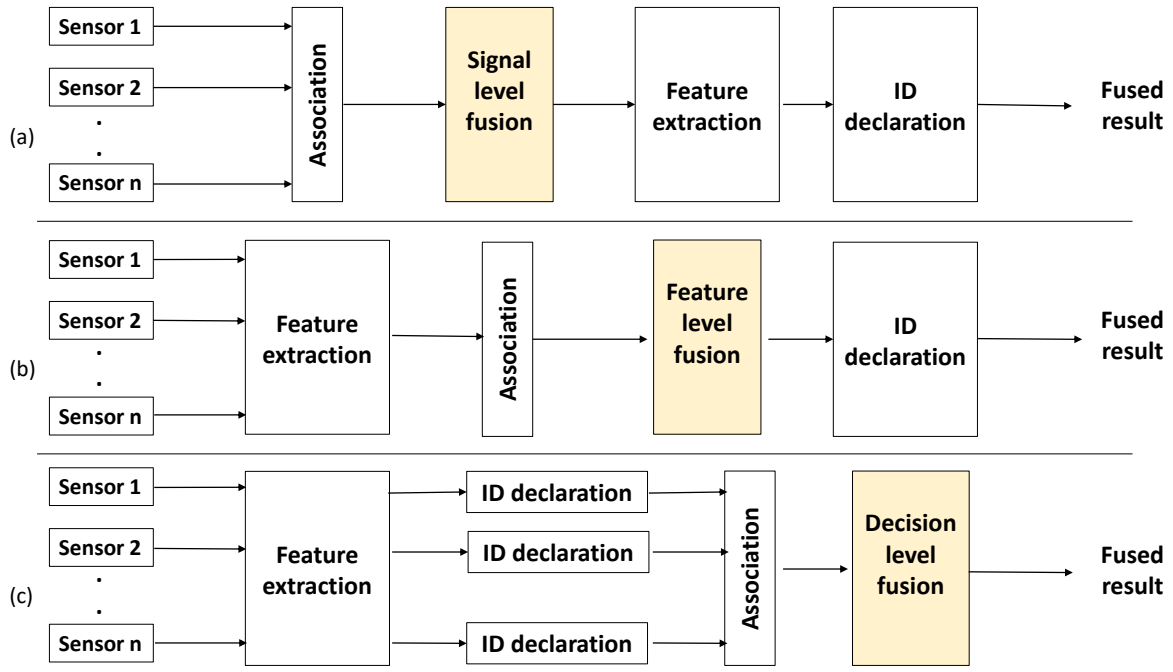


Figure 2.2. Data fusion at three different levels. (**a**) Signal-level fusion, (**b**) feature-level fusion, and (**c**) decision-level fusion.

output from sensors in space-domain. In time domain we by use the information available at previous time steps to capture the dynamic behavior of the system and reduce the disturbance of the final output.

Time factor is considered in research on time-domain evidence combination under DS theory. Hong and Lynch [57] applied the original DS method to time-domain fusion, but didn't mention how the limitations of the original DS method [58] can be reduced. Song et al. [59], [60] assumed that the evidence will decay over time. They proposed a credibility decay model. But his methods showed poor anti-disturbing ability when conflicting (noisy) evidence is present in time-domain. Chengkun et al. [61] improved the credibility decay model by using the exponential smoothing function. The anti-disturbing ability of Chengkun's method was better but the convergence rate was poor. Moreover, no research method in time domain fusion under DS theory is applied on real-time data.

Unfortunately, application of decision level sensor fusion for crop/weed classification in agricultural robotics is very limited. Most of the applications of sensor fusion in agricultural

robotics can be found in mapping and tracking. Both LIDAR sensor and vision systems have been used separately or in fusion mode for crop row tracking in sugar beet [62] and rice rows [63]. LIDAR and a color camera device for detecting tree trunks [64] and autonomous tractor guidance in some crops [65] has been published. The aim of these works is to develop a system which is able to track crop rows accurately in different fields and in diverse working conditions. A recent study in Iowa state university [66] used depth and 2D color data from Microsoft Kinect v2 for Lettuce and Broccoli detection. RANSAC is used for depth curve fitting to detect ground, then green color pixels are detected in 2D images. Connected component clustering method is used to detect plant features. Adaboost (adaptive boosting) method uses set of weighted weak classifiers to generate a strong classifier. For this work, Adaboost based classification model provided the lowest test error (6%), outperforming a 5-layer neural network (17%). However, this method may not be feasible for outdoor real-time usage without modification, as Kinect v2 is an indoor sensor. Also, their model solves a 2-class (crop/non-crop) classification task, which is much simpler than classifying a handful of weed species, often having similar features. Besides, none of the existing works show how a sensor fusion algorithm can help to overcome some of the limitations of CNN based weed classification.

2.5 Object Detection and Fuzzy-fusion

Real-time weed detection is an emerging field for agricultural robotics. DNNs are used for real-time weed detection, crop management and path planning extensively [9], [67]. With emergence of agricultural robotics, accurate identification of weeds and crops with deeper understanding of overall object detection technology becomes more important. In general, object detection systems detect a target object using classical computer vision and deep learning techniques, determine the category of the detected object and create a bounding box (BB) marking the position of the object [68], [69]. But real-time object detection is a complex challenge due to the effect of background, noise, occlusion, resolution and scale affecting the performance of the system [68]. In 2013, R-CNN (Regions with CNN features) [70] showed much improvement in object detection compared to conventional computer vision techniques and started the trend on CNN based object detection.

Many state-of-the-art CNN detector models like SPP-net [71], Fast R-CNN [70], YOLO [72], RetinaNet [73] improved detection accuracy on standard image datasets using new CNN architectures. But in most of cases, CNN models are trained with images without noise or degradation. During training, image augmentation is used to introduce noise, to increase the robustness of the model but sometimes they may fail to capture the real scenario. It is not possible to include all probable type of noise during the training of a CNN. In real scenario, noise can affect the quality of image due to sensor quality, lighting, vibration, exposure time etc. It is already shown that introducing carefully selected noise can produce wrong results even though they have no effect on visual recognition [74]. Prasun et al. [75] showed how different image degradations can affect the performance of CNN models. They were unable to come up with a solution which can produce a CNN architecture robust against image degradation when large number of classes are present like ImageNet dataset. Morever, in recent times, it is observed that the accuracy of CNNs reduces significantly when only tested on negative images which shows an inherent bias towards positive training dataset [76]. In this research, we want address this inherent limitation of CNN regarding uncertainty towards correct object detection and BB creation with a multi-camera setup and fuzzy logic.

Fuzzy logic has high potential in understanding complex systems where analytical solution may not exist or the system is not understood properly but can be observed [77]. According to T. Ross, fuzzy systems are useful in two scenarios: "(1) in situations involving highly complex systems whose behaviors are not well understood and (2) in situations where an approximate, but fast, solution is warranted" [77]. In agricultural robotics, fuzzy neural network based sliding mode control is used to build apple picking robot [78]. Romeo et al. used fuzzy clustering with dynamic threshold for greenness identification [79]. Meyer et al. used fuzzy clustering for classifying plant and soil from color images [80]. Fuzzy classifier is used to detect weeds in real-time in sugarcane fields [81]. Fuzzy expert system is used in soil management [82], predict cotton yield [83] and crop disease management with text-to-talk user interface [84]. In general, fuzzy logic is used as a classifier for crop or weed detection and as an expert system for crop, weed and soil management. But no through research studies are found which uses fuzzy logic to improve weed detection accuracy using multi-camera setup.

3. CROP ROW DETECTION

Autonomous navigation of agricultural robot is an essential task in precision agriculture, and success of this task critically depends on accurate detection of crop rows using computer vision methodologies. This is a challenging task due to substantial natural variations in crop row images due to various factors, including, missing crops in parts of a row, high and irregular weed growth between rows, different crop growth stages, different inter-crop spacing, variation in weather condition, and lighting. The processing time of the detection algorithm also needs to be small so that the desired number of image frames from continuous video can be processed in real-time. To cope with all the above mentioned requirements, we propose a crop row detection algorithm consisting of the following three linked stages: (1) projective transformation to transform camera view and color based segmentation for differentiating crop and weed from background, (2) differentiating crop and weed pixels using clustering algorithm and (3) robust line fitting to detect crop rows. We test the proposed algorithm over a wide variety of scenarios and compare its performance against four different types of existing strategies for crop row detection. The proposed algorithm achieves overall IOU of 0.73 and shows robustness under challenging scenarios with high weed growth. Experimental results show that the proposed algorithm performs almost equal or better than the competing algorithms with reasonable accuracy. We also perform additional experiment to test the robustness of the proposed algorithm over different values of the tuning parameters and over different clustering methods, such as, KMeans, MeanShift, Agglomerative, and HDBSCAN. Additionally to increase the robustness of the proposed algorithm under vibration, a detailed theoretical model of kalman filter based crop row center tracking from video feed is also presented.

3.1 Introduction

The desiderata of a practical crop row detection method that works satisfactorily in real-life deployment are below: (1) it is capable of detecting crop rows even with high weed pressure; (2) it is applicable at different types of crop fields; (3) it is capable to detect crop rows at different crop growth stages; (4) it is capable to detect straight and reasonably curved

crop rows; and finally, (5) the processing time of detection on an off-the-shelf computer satisfy real-time requirements. In this work, we propose a Clustering Algorithm based RObust LIne Fitting (CAROLIF) crop row detection method which satisfy all the above requirements. A short summary of proposed crop row detection method is provided below.

The basic idea of our proposed method is to use a robust clustering method so that the green weed pixels are detected as noise points, whereas the green crop row pixels are detected as data clusters. We want each crop row to be detected as one distinct cluster so that we can fit a regression line through that cluster which we can return as detected crop row. At the first step, the input image is cropped, transformed to an image plane vertical to the crop rows, segmented and cleared from small noise or weed segments. At this point the image has two channels, black and white pixels. The black pixels contain background and the white pixels contain crop and weed. Now the goal is to separate weed pixels from crop pixels, so that lines can be fitted to the crop rows. A clustering algorithm is used which can cluster each crop row pixels and separate weed pixels from the clusters. If the crop row clusters contain weed pixels, they can hamper the accuracy of line fitting algorithm. As a result, a robust line fitting algorithm is used to fit a line on each crop row cluster which omits the effects of outlier points (weed or noise). Finally the fitted lines are plotted on the image and returned as detected crop rows. All the steps of the proposed algorithm is visually presented in Fig. 3.1.

For autonomous agricultural robot navigation, CAROLIF must be able to process video feed from moving camera. Moving camera on an agricultural robot introduces issues like vibration and crops covered by moving machinery. These issues may reduce the detection accuracy of CAROLIF, which may result in inaccurate navigation of the robot. To address this issue, a kalman filter (KF) [85] based object tracking methodology is introduced in this paper. KF based tracking system will keep track of the crop row centers even if accuracy of CAROLIF is reduced or information is lost between the frames due to vibration. Moreover kalman gain can be tuned for better crop row centers tracking for curved rows.

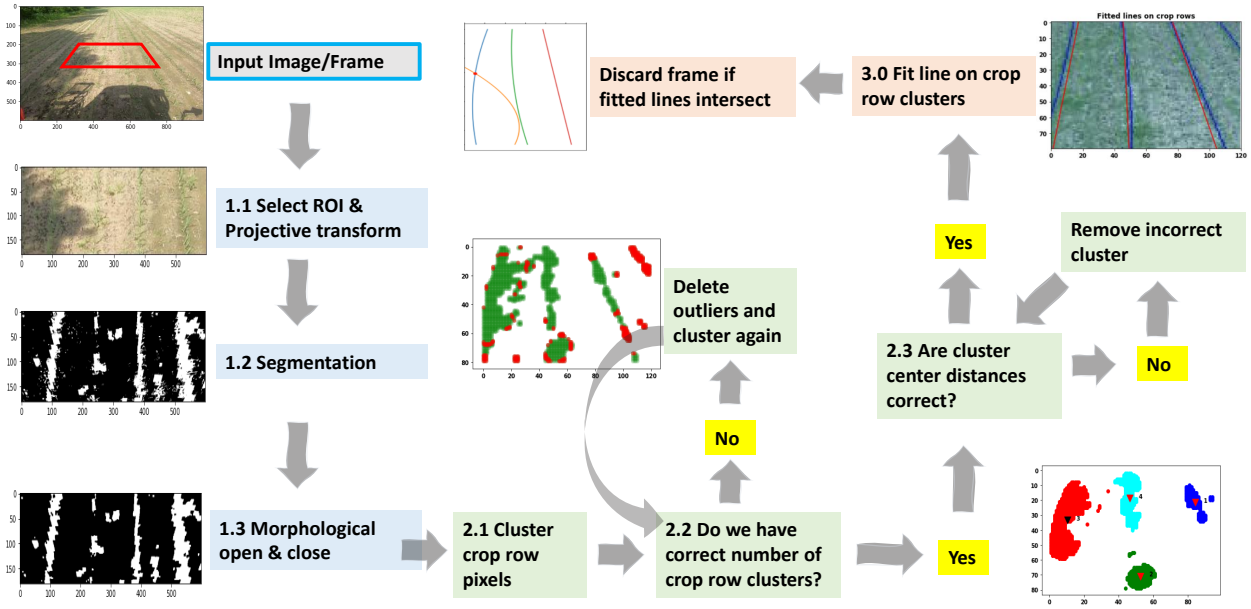


Figure 3.1. Steps of the proposed CAROLIF algorithm with images.

3.2 Methodology

Step 1: Pre-processing (Step 1.1) First a region of interest (ROI) is selected which contains 3 or more crop rows. All the algorithmic procedures are applied within this ROI. Selecting this ROI has three advantages. First, green pixels near horizon are congested and hard to separate. As a result, they increase false detection. Selecting this ROI eliminates the need to process those green pixels. Secondly, this ROI is almost one-fourth the size of the original image. So only operating in this section reduces the computational cost and processing time. And third, curved rows within this small ROI are reasonably straight. We are also using projective transformation to transform the crop rows converging at infinity to actual parallel lines. As a result, fitting straight lines to these crop rows provide reasonable result.

Projective Transformation: The relationship that maps a set of points from one image plane to a set of points to another image plane is called projective transformation. Planar homography is the projective mapping from one plane to another. According to Hartley and Zisserman [86], projective transformation is defined as, "A planar projective transformation

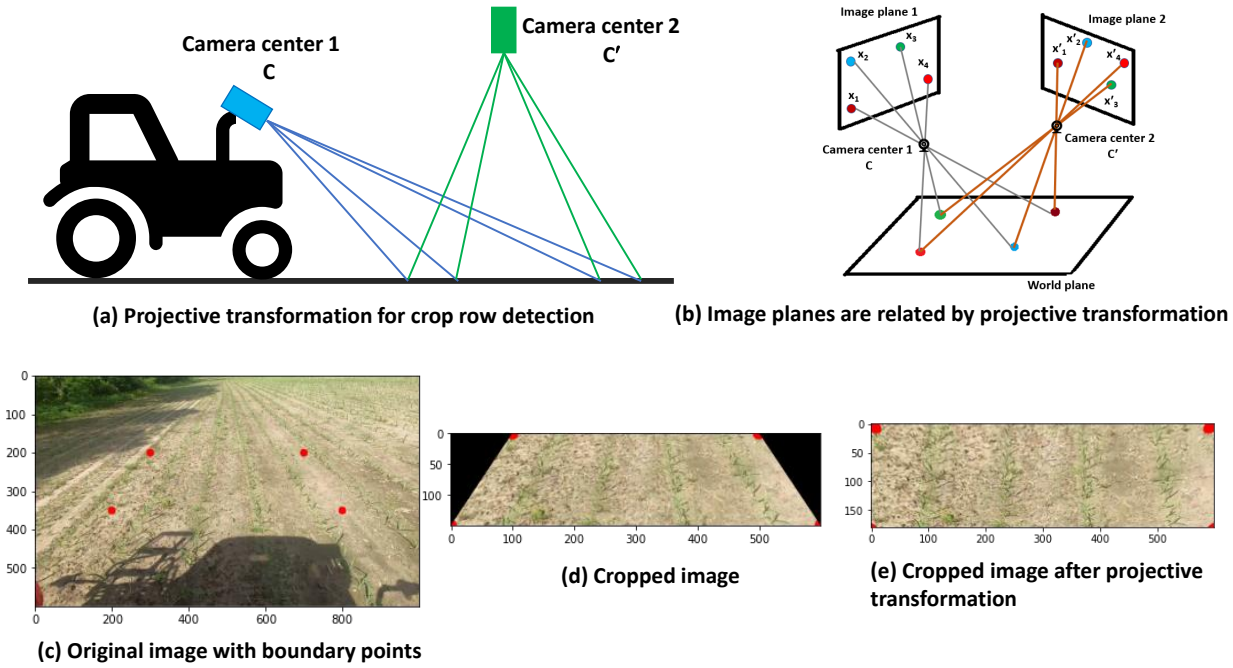


Figure 3.2. (a) Projective transformation is used from Agricultural robot to transform camera 1 view to virtual camera 2 view. (b) For two camera planes, image planes are related by projective transformation when all world points are co-planer. (c) Original image from camera 1 view with boundary points. (d) Cropped image using boundary points. (e) Transformed cropped image from camera 1 view to camera 2 view.

is a linear transformation on homogeneous 3-vectors represented by a non-singular 3-by-3 matrix."

$$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{3.1}$$

Or in short, $x' = H.x$, where H is the homography matrix. This homography matrix relates the position of a point from source image plane (image plane 1 in Fig. 3.2(a) and (b)) to a destination image plane (image plane 2 in Fig. 3.2(a) and (b)). $(x'_1.x'_2, x'_3)^T$ and $(x_1.x_2, x_3)^T$ are coordinates of a single point on two image planes. There are eight independent ratios in H (h_{33} is a scaling factor) which means a projective transformation

has eight degrees of freedom [86]. Here, homogeneous coordinates are used to represent the points.

Let's consider a pair of inhomogenious matching points (x, y) and (x', y') on image plane 1 and 2 respectively. We are considering inhomogenious coordinates because they can be measured directly from image plane (coordinates of points in pixel). From equation (3.1):

$$x' = \frac{x'_1}{x'_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{3.2}$$

$$y' = \frac{x'_2}{x'_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{3.3}$$

After rearranging:

$$x'(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13} \tag{3.4}$$

$$y'(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23} \tag{3.5}$$

where, $x = x_1/x_3$ and $y = x_2/x_3$. Four points on each image plane will create eight linear equations and four points are sufficient to solve for H between two image planes. The only condition is, no three points can be collinear [86]. With scale factor $h_{33} = 1$ and four set of points on each image plane, we get the following:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x'_2x_2 & -x'_2y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y'_2x_2 & -y'_2y_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x'_3x_3 & -x'_3y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y'_3x_3 & -y'_3y_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{21} \\ h_{32} \end{pmatrix} = \begin{pmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{pmatrix} \tag{3.6}$$

Boundary points (red dots on Fig. 3.2 (c)) are used to solve for the H matrix. These points are selected by us so that the lines created by the two red dots along the crop rows are

parallel to the crop rows. The idea is that, after transforming the image from image plane 1 to image plane 2 (top view), the crop rows will be parallel to each other and will not meet at infinity. This will help us to distinguish between the crop rows after clustering and we will be able to impose strict straight line boundary condition to each fitted line. After H matrix is calculated, it is then applied to the whole image plane 1 (Fig. 3.2 (d)) to convert it to image plane 2 (Fig. 3.2 (e)). From Fig. 3.2 (e), it is clear that the crop rows are now parallel to each other and not meeting at infinity. Fig. 3.12 shows the performance of CAROLIF with and without perspective transformation.

(Step 1.2) Given an input image in RGB color space, each channel is separated by following system:
R = red channel of input image
G = green channel of input image
B = blue channel of input image
After splitting the channels, the following normalization scheme is applied, which is common in agronomic image segmentation [87]:

$$\begin{aligned} r &= \frac{R_n}{R_n+G_n+B_n} \\ g &= \frac{G_n}{R_n+G_n+B_n} \\ b &= \frac{B_n}{R_n+G_n+B_n} \end{aligned} \tag{3.7}$$

where R_n, G_n and B_n are the normalized RGB coordinates ranging from 0 to 1 and are obtained as follows:

$$R_n = R/R_{max}, G_n = G/G_{max}, B_n = B/B_{max} \tag{3.8}$$

where R_{max}, $G_{max}\, and\, B_{max}$ are maximum values of R, G and B channels respectively. A small number is added to the denominator of normalization step to avoid division by zero. Green color (vegetation) can be extracted using the following equations [88]:

$$ExG = 2g - r - b \tag{3.9}$$

(Step 1.3) Morphological opening and closing operation on the binary image to reduce noise.

Step 2: Cluster An ideal clustering algorithm for crop row detection should have two specific features. First, the clustering algorithm should be able to differentiate between different crop row clusters without any prior knowledge (we don't know exactly how many crop rows may appear inside ROI). Second, should have robust and intuitive tuning parameters to tune the algorithm for wide variety of scenarios.

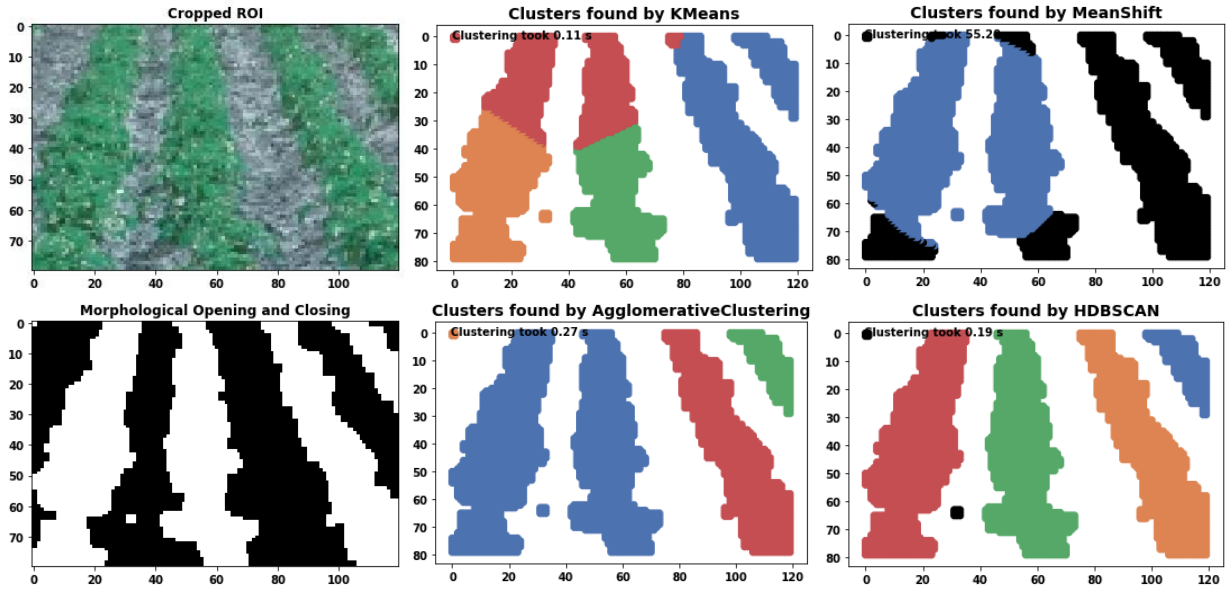


Figure 3.3. Comparison of Kmeans, MeanShift, Agglomerative and HDBSCAN clustering algorithm on crop row detection. Run-time indicated on the top left corner of each image. Different colors indicate different clusters. Black colors indicate outliers.

Four different types of clustering algorithms are tested and a sample output is presented in Fig. 3.3. Kmeans [89] fails because crop rows are anisotropic data which means they are elongated along a specific axis. Because Kmeans treats every data point equally, it fails to distinguish any local variation within a cluster. The advantage of Meanshift [90] over Kmeans is that we don't have to specify the number of clusters. Meanshift algorithm assumes an underlying probability density function of the data and locates centroids at the maxima of

the density function. The default parameter *bandwidth* which dictates the size of the region to search through in scikit-learn [91] shows wrong results. Also it is order of magnitudes slower than other tested methods and not applicable for real-time usage. Agglomerative clustering [91] is a hierarchical method which group data into clusters based on similarity. It starts with treating each data point as a single cluster then merges clusters into a single cluster until some criterion is met. But for this to work, 'number of clusters' has to be known before hand. HDBSCAN [92] is a density based method which extends DBSCAN [93] into a hierarchical clustering algorithm. The only tuning parameter used is *min cluster size* and its effect is explained later. From Fig. 3.3 it is clear that HDBSCAN successfully clustered the crop rows. It also identifies the outliers (black pixels) which is then omitted from output clusters.

Preventive measures to reduce wrong clustering is implemented into the algorithm. If number of cluster inside ROI is lower than three (Fig. 3.1 step 2.2), the algorithm assumes some of the clusters are merged due to high amount of weed. It then iteratively deletes outliers (weeds) to find crop row clusters. Sometimes due to high weed growth or intermittent crop growth, HDBSCAN creates multiple clusters for a single crop row. This can be controlled by changing the *min cluster size* parameter. But for fixed *min cluster size* parameter, *crop row distance* is used as a threshold parameter (Fig. 3.1 step 2.3). If multiple clusters are close by (determined by the distance of crop rows), then the cluster with smaller data points and smaller height will be deleted. In Fig. 3.1, the green colored cluster (step 2.3) will be deleted.

Step 3: Line fitting Random Sample Consensus (RANSAC) [94] is used for line fitting on each cluster due to simple implementation and robustness. Because RANSAC iteratively determines the best data points to fit a line in a cluster, it excludes the outliers (possibly weed data points) and shows robust line fitting when weed is present.

The full algorithm for crop row detection is presented in Algorithm 1. Some important tuning parameters of the algorithm and their effects on crop row detection is discussed in Table 3.1.

Table 3.1. Tuning parameters for CAROLIF.

Parameter	Value	Description
ROI size	(120 by 80), (160 by 80)	This algorithm works better if there are 3-4 crop rows inside the ROI. Based on camera position and camera viewing angle, the ROI size should be fixed.
min cluster size	50, 100, 200	HDBSCAN parameter to affect clustering. Set it to the smallest size grouping that you wish to consider a cluster. For big number (200): at early crop growth stage, combine multiple rows into one cluster. For small number (50): at early intermittent crop growth stage, create multiple clusters for one crop row. As crop grows, shows less effect on clustering. $minclustersize = 50$ shows reasonable results on CRBD dataset for this study. Should be tuned based on growth stage, ROI size, crop type.
crop row distance	15, 25, 30	This is the pixel value of center distance between two consecutive crop rows. Actual distance converted to pixel. This value should be calculated based on actual crop row distance on field and camera parameters.

3.3 Pseudocode (CAROLIF)

This section explains Algorithm 1. The function $ProjectiveTransformation$ takes a full size RGB image, ROI size and boundary points as inputs. Input image is cropped using ROI size (Table 3.1). Boundary points are four corners for the crop image to be transformed. This function returns the projective transformed cropped image where crop rows are parallel to each other. $SegmentAndClean$ section of the code uses the cropped projective transformed image as input. First, ExG and Otsu thresholding is used to create a binary image from the color image where green pixels have value 255 and everything else is zero. Then morphological

OPEN and CLOSE operation cleans the image from noise. $ClusterImage$ section of the algorithm takes the clean segmented image as input. HDBSCAN is applied to the segmented image to find individual clusters. Sometimes due to high weed pressure or unusual crop growth, multiple crop rows are connected together. These clusters have a very high number of data points compared to normal crop row clusters. When this scenario arises, an iterative outliers deletion process starts and runs until connected crop rows are separated. The output of this function is the number of clusters (with location of each cluster's data point) which are believed as crop rows. Then two checks are completed based on geometric knowledge of the crop rows. First, the crop rows are usually a constant distance apart in 3D world. That distance is transformed to pixel value (for the camera image plane) and checks are completed to see if two clusters are within that distance. If yes, then it is assumed that one of them is a weed cluster. We identify and delete the weed cluster based on height and number of data points. After fitting straight lines through each cluster the final check is completed. Because we are using projective transformation, each fitted line should have a slope close to 90°. The slope of the lines changes from 90° when the agricultural vehicle is turning. In this study, we use a slope threshold of $[70, 110]$ degree. Lines outside this slope range are omitted from final output. Finally we show the plotted lines over the detected crop rows.

3.4 Methods Used for Comparison

Four different methods along with the proposed method is compared in this study. The first one, Hough transform [16] is a feature extraction technique in digital image processing. After step 1, Canny edge detection [95] is applied to extract the edges from the binary image. With the coordinate transformation, the colinear points in edges of the binary image converted to concurrent lines in parameter space by voting. Hough transform detects lines by accumulating the votes. The main tuning parameter is *threshold* (The minimum number of intersections to "detect" a line). But this parameter is not intuitive. Also changing the parameter slightly drastically changes the output. Also Hough transform outputs a lot of false positive lines without any filtering. In this paper we have filtered out the lines with low slope and lines which are close by. But selecting the threshold for the slope and threshold

Algorithm 1 CAROLIF

```
INPUT : color image
OUTPUT : fitted lines over crop rows
Function ProjectiveTransformation (color image, ROI, boundary points):
    Use ROI to crop the input image
    Use boundary points for projective transformation
    return transformed image
Function SegmentAndClean (transformed image):
    Binary segment image using ExG and Otsu
    Noise reduction with morphological operations
    return binary segmented clean image
Function ClusterImage (segmented image):
    Cluster segmented image with HDBSCAN
    Check if correct number of clusters present
    If not, delete outliers and check again
    return crop row clusters
Check if crop row cluster's distances are correct
Delete incorrect clusters
fit straight line on crop row clusters with RANSAC
plot straight lines over crop rows and show
```

to determine which lines are close by are not robust. Output changes drastically based on these parameters.

The second method is named Sliding-window. After step 1 (Fig. 3.1), a window of size (20 by 20) slides over the ROI and calculates the center points of the white pixel blobs inside that window. After sliding over the whole ROI, we essentially have the center points of the crop rows. Then least-square straight line is fitted on the crop row center points. Some variant of this method is available in literature [17], [96]. One of the main limitations of this method is to figure out where the crop row starts or the segments where each row resides. As a result, high weed pressure drastically reduces the accuracy for this method.

The third method is Template Matching followed by Global Energy Minimization (TMGEM) [19]. It uses dynamic programming for efficient global energy minimization. This method can work without any prior knowledge of crop row number, reasonably insensitive to weed and works with different crop growth stages. The authors of this paper also created Crop row benchmark dataset (CRBD). This evaluation image set includes 281 images of maize, celery, potato, onion, sunflower and soya bean crops. The images are taken at varying yaw,

pitch and roll angles; different amount of weed pressure and lighting conditions. This dataset is used in this study for comparison and testing.

The fourth method is named Cluster - Least square. Here after step 2 (Fig. 3.1), least square straight line fitting is used. Least square works by making the total square of error as small as possible. As a result, this method is sensitive towards outliers. When outliers (weed) are present then least square pulls the line toward it and decreases the accuracy of crop row detection.

The final method is named CAROLIF and is the proposed method. Other than TMGEM, all the other methods are built and implemented from scratch by us.

3.5 Results

Thirty different cases (images) from the CRBD dataset are used to test the effectiveness of the CAROLIF algorithm. Two different scenarios are defined. "Easy scenario" is defined when crop rows are reasonably separate, none or low weed pressure, crop rows are straight or reasonably curved. "Challenging scenario" is defined when crop growth is intermittent or crop missing from rows, weed pressure is high, crop rows are interconnected due to high weed pressure, crop rows are curved. Fig. 3.4 shows some cases for easy and challenging scenario. Fig. 3.5 qualitatively compares the results of five different methods.

3.5.1 Qualitative Comparison

In Fig. 3.5 (Row 1), very early stage of crop growth and crop rows are hard to see with naked eye. Hough transform and Sliding window method fails to correctly detect the third row in the image. TMGEM and the proposed method performs the best. In (Row 2), due to high weed pressure and early crop growth stage, crop row 2 and 3 are connected. This is a hard problem to solve because weeds are at high concentration and any line fitting algorithm fits line through weed pixels. Hough transform and Sliding-window fit this trend. CAROLIF shows the best result in this scenario. In (Row 3), weeds with unusually big size appears. Any usual row detection algorithm will fail and fit lines over weed pixels due to unusually high concentration. This is what happened for all the other methods. CAROLIF overcomes

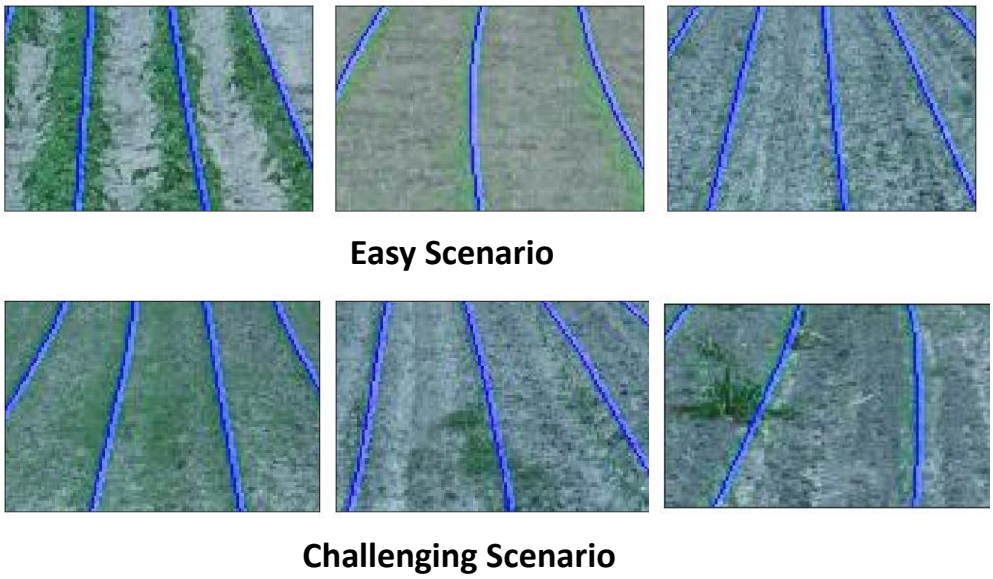


Figure 3.4. Example cases for easy and challenging scenario.

this because RANSAC iteratively finds 2 points with 99% probability to be inliers of a cluster and fit lines over those two points. In (Row 4), top half of crop row is missing for the first row. Also at the top left corner there are some green pixels from another crop row. Cluster - least square fails because least square is not robust against outliers. CAROLIF overcomes this problem. In (Row 5) CARLOLIF shows suboptimal results for the second crop row. This happens because crop row growth is substantially less compared to weed growth. As a result, clustering algorithm deletes the crop row part as outlier. This can be mitigated by tuning the *min cluster size* parameter in HDBSCAN (use a smaller number).

3.5.2 Quantitative Comparison

Intersect over union (IOU) parameter is calculated to quantitatively measure the accuracy of each of the methods. Intersection Over Union (IOU) evaluates the overlap between two bounding boxes. It requires a ground truth bounding box and a predicted bounding box. Then it calculates the ratio of the area where two bounding boxes overlap to the total combined area of the bounding boxes. Fig. 3.6 [97] shows the visual representation of the

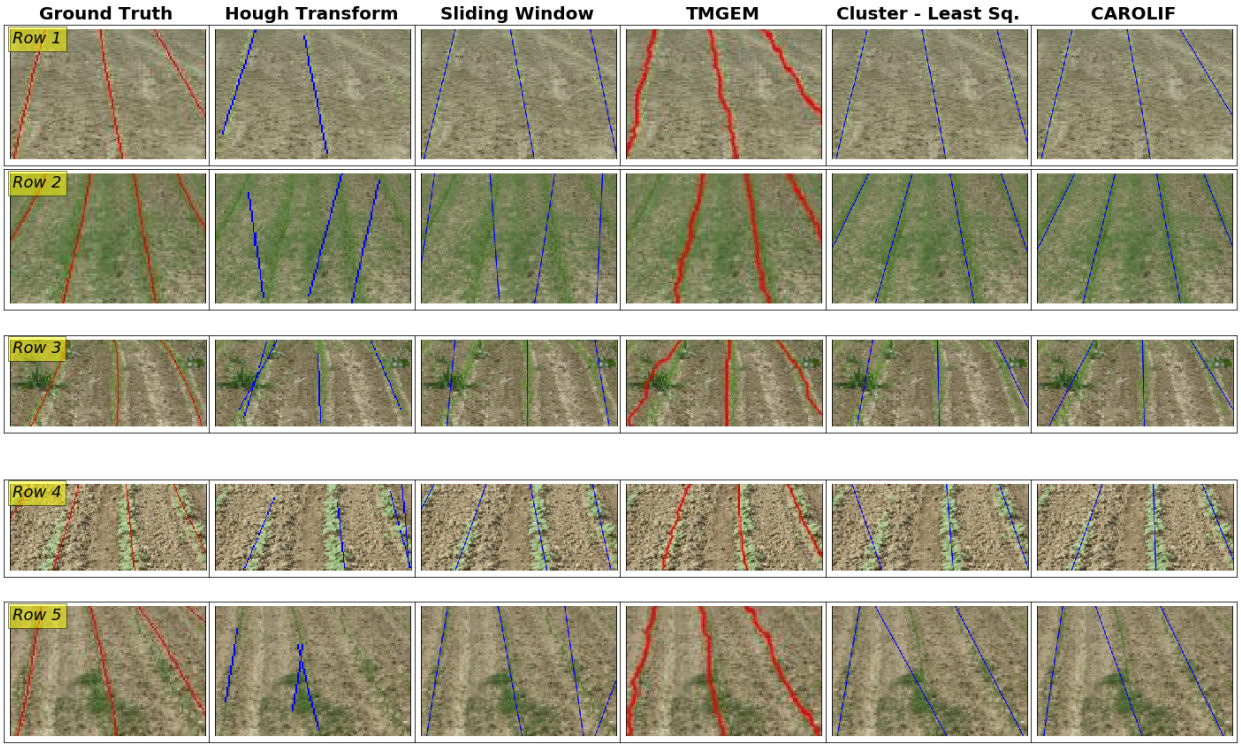


Figure 3.5. Comparison of crop row detection in challenging scenarios for Hough transform, Sliding window, TMGEM, Cluster-Least square and CAROLIF (proposed) methods with ground truth results.(Row 1) intermittent, very early crop growth, no weed. (Row 2) early crop growth stage, high weed pressure, crop rows are connected, crop rows and weed indistinguishable. (Row 3) early crop growth stage, exceptionally big intermittent weed, curved crop rows. (Row 4) medium crop growth, crop missing from rows, no weed. (Row 5) early crop growth stage - intermittent, concentrated weed growth.

IOU metric. Fig. 3.7 shows how the bounding boxes are created for the crop row detection case. IOU is an intuitive parameter. A score of 1.0 means that the predicted bounding box precisely matches the ground truth bounding box. A score of 0.0 means that the predicted and true bounding box do not overlap at all.

Fifteen cases are tested for each scenario (easy and challenging). Only three rows are selected in each case to generate the IOU value to correctly compare the results with TMGEM (the authors of TMGEM has used three detected lines). Fig. 3.8 shows the performance comparison of different algorithms under easy and challenging scenarios. For all the algorithms, the IOU value is higher for easy scenario compared to challenging which implicates that presence of weed deteriorates the accuracy for all the algorithms. TMGEM has the

$$IOU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\quad}{\quad}$$

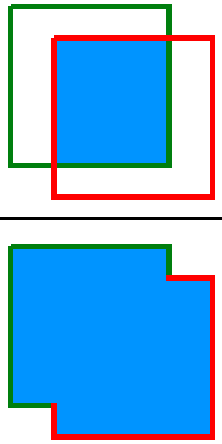


Figure 3.6. IOU between a ground truth bounding box (in green) and a detected bounding box from algorithm (in red).

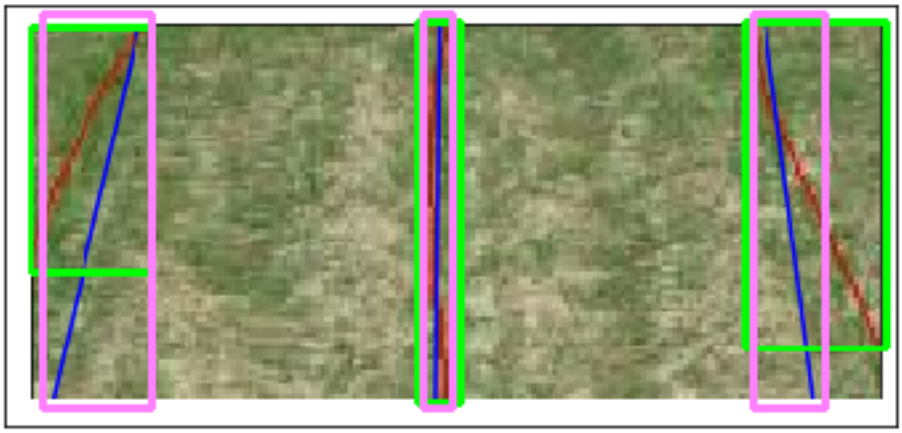


Figure 3.7. Actual bounding boxes. Red lines show ground truth detection and blue lines show algorithm detection of crop rows. Green bounding boxes capture the height and width of ground truth lines. Pink bounding boxes capture the height and width of algorithm detection lines.

highest median IOU for both easy and challenging scenario closely followed by CAROLIF. In easy scenario, TMGEM has the lowest spread of IOU which means this is the most consistent algorithm in terms of crop row detection in easy scenario. In challenging scenario, CAROLIF has the lowest spread and highest minimum IOU value which means under extreme condition CAROLIF performs better and consistently than other algorithms. Overall hough transform and sliding window performs significantly worse compared to other three algorithms. TMGEM, cluster-least sq. and CAROLIF shows comparable performance but

TMGEM performs slightly better than other two algorithms. Table 3.2 shows the mean IOU value for each scenario and average IOU for all the algorithms.

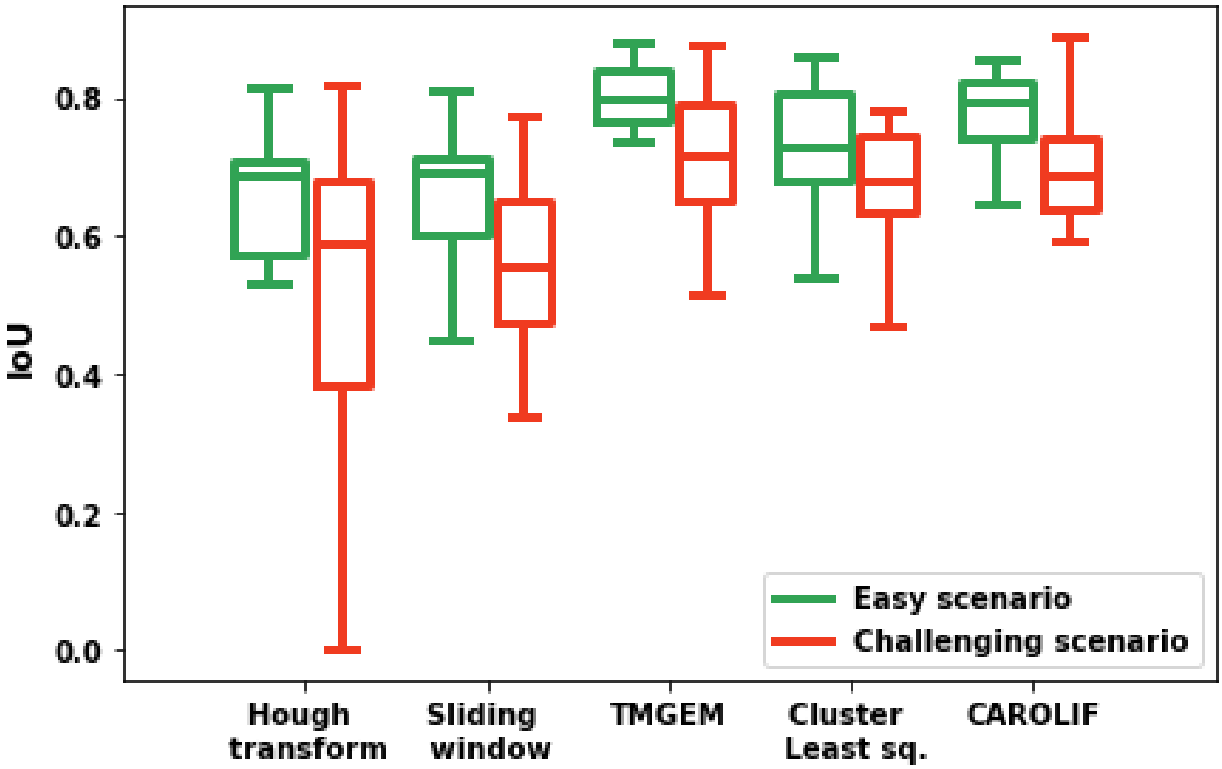


Figure 3.8. Box plot comparison of IOU values for different methods under easy and challenging scenario.

Table 3.2. IOU value for different algorithms.

Algorithm	Mean IOU (Easy)	Mean IOU (Challenging)	Avg. IOU
Hough transform	0.64	0.53	0.58
Sliding window	0.66	0.55	0.61
Cluster - least sq.	0.72	0.67	0.7
CAROLIF	0.76	0.69	0.73
TMGEM	0.79	0.71	0.75

The overall slightly inferior performance of CAROLIF algorithm (or any color based segmentation method) can be explained from the ground truth value of the CRBD dataset. One example is presented in Fig. 3.9. The ground truth detection lines (red lines) for the left and center row are not exactly at the center of the rows. Ground truth value of left row

is biased towards left and for center row biased towards right. Any color based segmentation method separates the green pixels from the background pixels and then plot the best fit lines of the green pixels as detected rows. As a result, the best fit lines go through the center of the detected crop rows. If the ground truth value is not at the center of the crop rows then IOU value will be lower even though the algorithm outputs the best fit line. In Fig. 3.9 we can argue that, for left and center row, the detected lines by CAROLIF are more accurate than the ground truth value. The deviation of ground truth value of CRBD dataset can be one of the main reasons behind the overall inferior IOU value. Although for comparison purpose, this would be true for all the color based segmentation methods and the algorithm with highest IOU value will be the better choice.

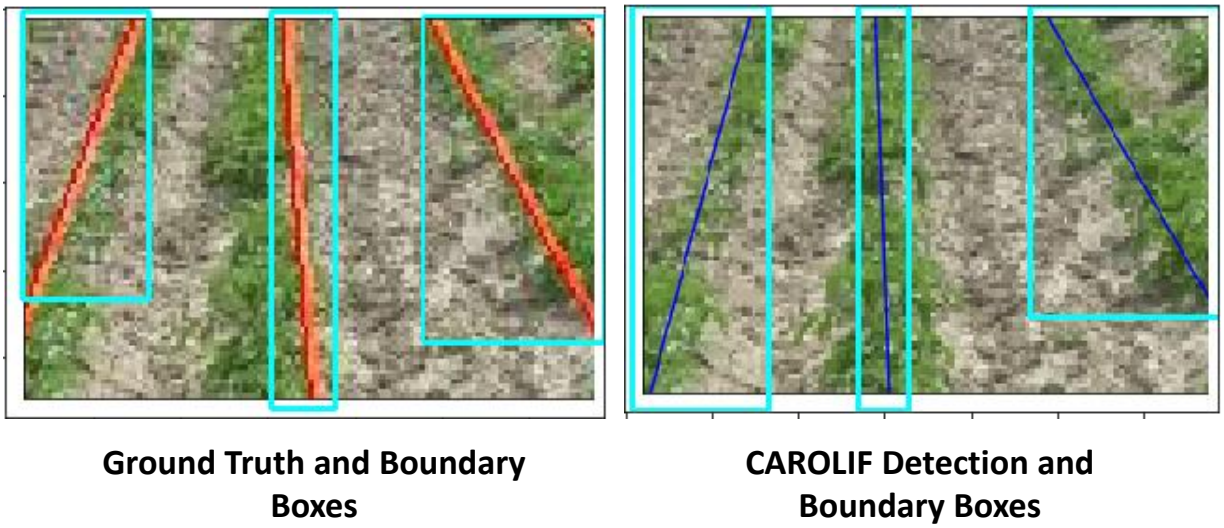


Figure 3.9. (left side) Ground truth value and boundary boxes from CRBD dataset. (right side) CAROLIF detection and boundary boxes.

3.5.3 Processing Time

For a real-time algorithm, processing time is important. the average time spent by each step of the algorithm is presented in Fig. 3.10. It is tested on a personal computer without

any parallel-processing or multi-threading techniques (which may reduce the processing time significantly). Also processing time for image acquisition hardware is not included. The algorithm takes 108 ms (around 10 frame per second (FPS)) to process each image. For a slow moving agricultural vehicle, 10 FPS is a reasonable processing time for real-time application. More than 90% of the time is spent by the clustering step. Step 2.2 (iteration) only happens for challenging scenarios (high weed pressure or crop rows connected). For normal crop growth and low weed pressure scenario, the algorithm bypasses step 2.2. With the development of very powerful embedded hardware like Jetson TX2 which has built-in video processing capabilities, the proposed algorithm is capable of real-time performance even for high speed vehicles.

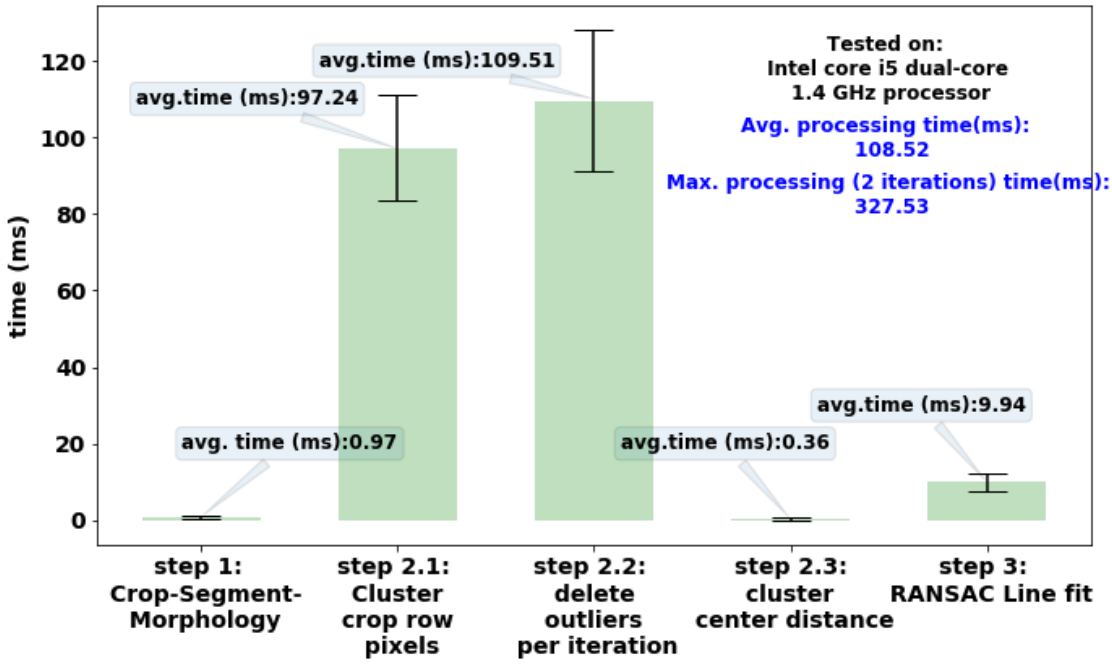


Figure 3.10. Processing time of each step of CAROLIF crop row detection algorithm. ROI size (120 by 80) pixels.

Comparison of processing time is presented in Table 3.3. Again it should be mentioned that no parallel-processing or multi-threading techniques are applied which may reduce the processing time significantly. Clustering methods take order of magnitude higher time compared to Hough transform and Sliding window. TMGEM takes order of magnitude higher

time than clustering methods which makes it unavailable for real-time crop row detection application.

Table 3.3. Processing time for different algorithms.

Algorithm	Processing time (ms)	Avg. FPS
Hough transform*	2.02	495
Sliding window*	2.88	347
Cluster - least sq.*	100.12	9.9
Proposed*	108.52	9.2
TMGEM**	1750	0.5

* core i5 dual-core 1.4GHz processor. (120 by 80) ROI size.
** core i5 quad-core 3.3GHz processor. (320 by 240) ROI size.

3.5.4 Performance on Video Input

Fig. 3.11 shows the crop row detection performance of CAROLIF algorithm on real-time video captured from an agricultural vehicle. A GoPro camera with 4K resolution and 60 FPS video capture capabilities is used to capture the video. The camera was set at the front of a tractor, tilted around 35 degrees and the tractor speed was around 10 miles per hour. It's a two minutes thirty seconds video where the tractor starts with straight crop rows with ideal detection conditions; after about a minute there is a slight left turn and the last thirty seconds covers an area with worst crop row detection conditions with shadows, intermittent crop growth and high weed pressure. This video was captured at a corn field in Indianapolis, Indiana on May 2020. Time step, $t = 5$ means, this frame is taken from video at time 2.5 seconds. $t = 10$ means, this frame is taken from video at time 5 seconds. For $t = 5$ to $t = 20$, the crop rows are well lit by the sun and there is minimal weed growth. CAROLIF performs strongly and detects all the crop rows correctly. At $t = 22$, there is weed growth between the left two rows and also there is presence of shadow. As a result, detection of the left row is not exactly accurate. But at $t = 30$, when we move to the good lighting area, detection of all the crop rows becomes accurate. $t = 35$ and $t = 38$ are also challenging situations due to high weed growth (height of weed is almost equal to height of crop) and shadows. But Crop row detection accuracy is correct for all cases. For $t = 40$ to $t = 60$, there are sparse shadows and light weed growth present. But CAROLIF performs well under these

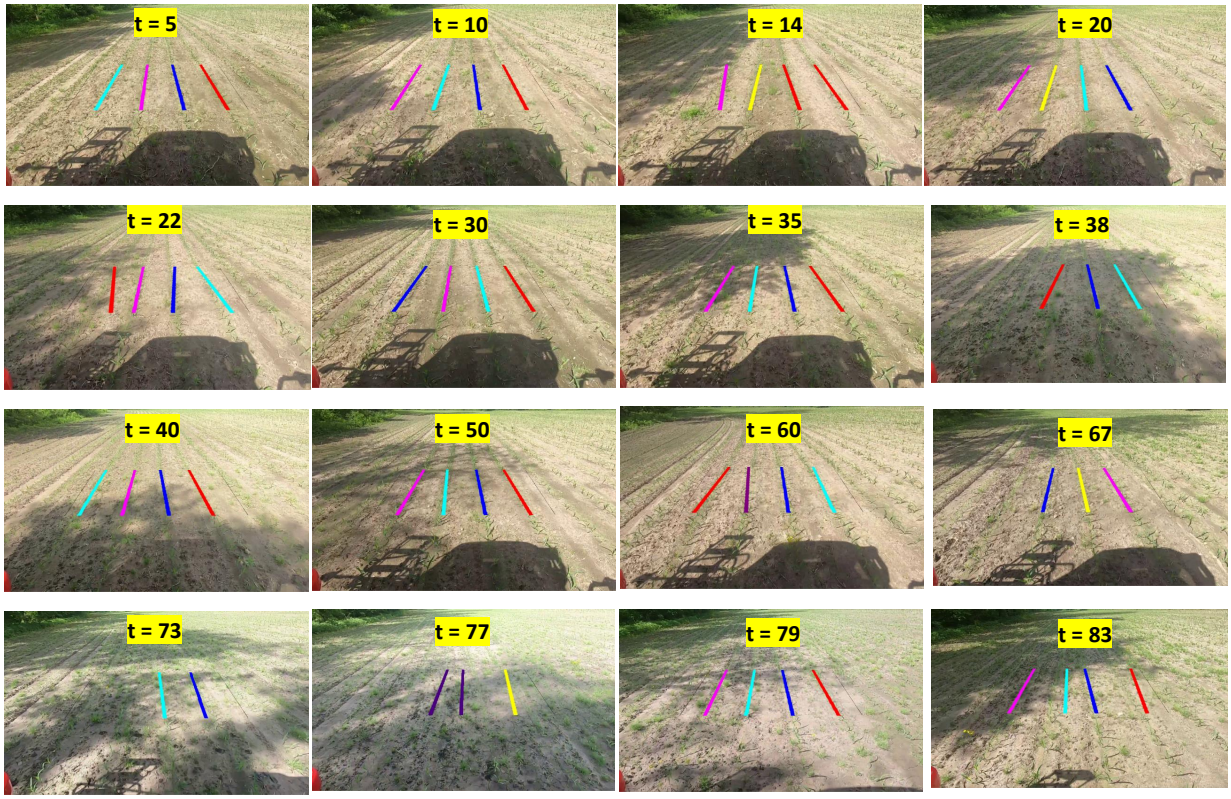


Figure 3.11. Performance of CAROLIF on real-time video captured from an agricultural vehicle.

moderately challenging scenarios and detects all crop rows correctly. At $t = 67$, the tractor is turning left and the left crop row is outside the slope threshold ([70, 100] degree) and omitted from the result. This can be circumvented by modifying the slope threshold. But the other three crop crows are detected correctly. $t = 73$ and $t = 77$ are one of the most challenging scenario a crop row detection system can face. There are shadows and sunlight, weed growth is very high and crop growth is low and sparse. Even for human eyes, it is not possible to detect crop rows correctly in these scenarios. As a result, the performance of CAROLIF deteriorates. But as soon as we move to a slightly better condition at $t = 79$, CAROLIF is able to detect all four crop rows correctly.

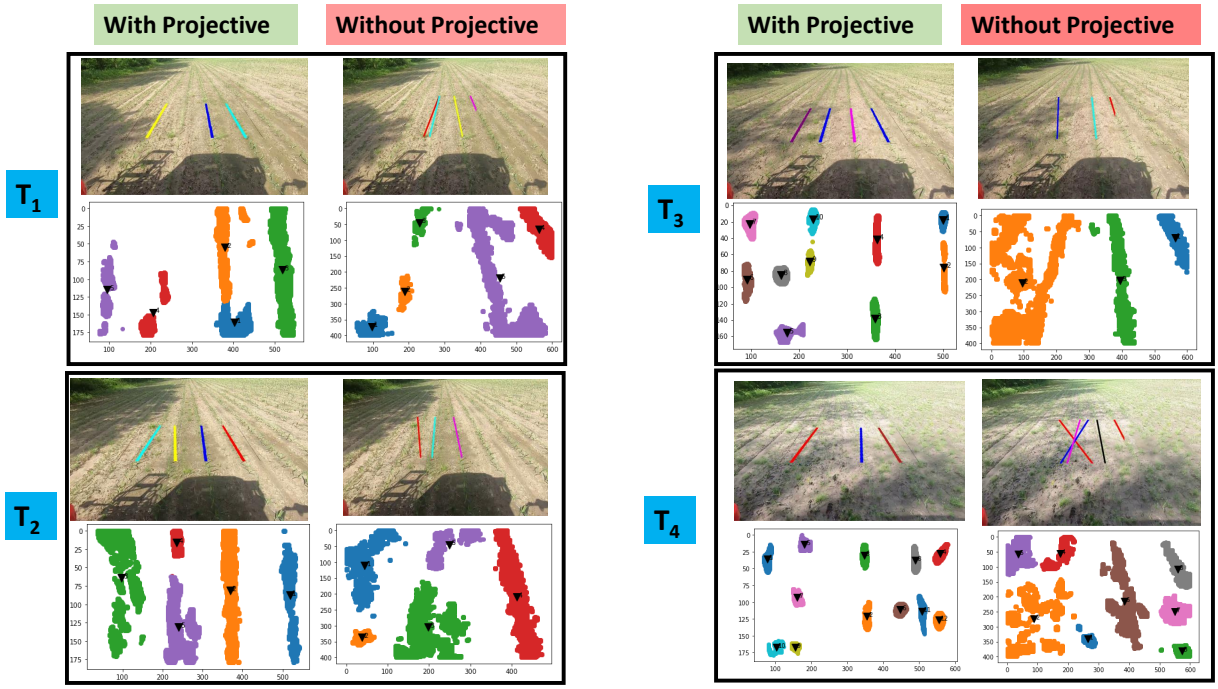


Figure 3.12. Comparison of crop row detection results with and without projective transformation. Four different scenarios are chosen. (T_1) and (T_2): sunny, no weed, no shadow, intermittent crop growth. (T_3): sunny, no weed, shadow present. (T_4): high weed pressure, shadow, intermittent crop growth.

3.5.5 Effectiveness of Projective Transformation

Fig. 3.12 shows how projective transformation reduces false positive crop row detection in complex situations. Four different frames with four different conditions are selected for comparison. At T_1 and T_2, weather conditions are ideal with intermittent crop growth. With projective transformation, clusters are parallel to each other and it is easier to distinguish them. On the other hand, without projective transformation, clusters are haphazardly distributed and it is hard to distinguish them using crop row distance and strict straight line threshold. As a result, without projective transformation, CAROLIF fits lines to wrong clusters. T_3 and T_4 are more complex situations with shadow, high weed pressure and intermittent crop growth. From the cluster output, we can see that after deleting the outliers, there are a substantial number of cluster cores present. Using projective transformation, with strict thresholds, incorrect cluster cores are eliminated and only correct lines are fitted. Specially at T_4, it is hard for naked human eyes to detect correct crop rows. But CARO-

LIF is successful at detecting three out of four crop rows. Moreover, only shows the lines which fits correctly and doesn't show the lines which may not be a good fit, hence reducing false positive detection. We measure the performance of CAROLIF on the video with and without projective transformation. Without projective transformation, several frames fail to produce any meaningful detection and we discard those frames from calculation. We measure the number of false positive, false negative and true positive crop row detection from the video and calculate accuracy, precision and recall. Table 3.4 shows this. CAROLIF has over 90% accuracy on a real-time video with very complex scenarios including shadow, intermittent crop growth and high weed pressure. It shows that performance of CAROLIF is acceptable for real-time application. It also shows how projective transformation improves the performance of CAROLIF in every aspect.

Table 3.4. Performance of CAROLIF on real-time video.

	With Projective Transformation	Without Projective Transformation
Accuracy	90.5%	77.1%
Precision	96.6%	84.5%
Recall	93.3%	89.8%

It is clear that CAROLIF is able to detect crop rows in real-time with very high accuracy under good and moderately challenging scenarios. In situations, when even naked eyes may fail to detect crop rows correctly, detection accuracy deteriorates. But due to distance threshold and slope threshold, false positive detection of crop rows is low even in very challenging scenarios. For real-time application, we can think of two ways to improve performance, when crop row detection accuracy deteriorates. First, we can fall back to GPS signal only until the tractor reaches a point where detection is correct for all four rows. Secondly, we can add a tracking algorithm which can track the crop row centers when sensor output or detection accuracy deteriorates due to very challenging scenarios. The next section explains the theoretical background of how we can add a kalman filter based tracking to CAROLIF to improve the overall robustness of the system.

3.6 KF based Crop Row Center Tracking

A successful tracking algorithm requires instantaneously predicting the state (location, velocity etc.) of an object from sequence of frames (images). There is no single best model and the success depends on the type of data and model being used. In this research we propose a detection -to-track model. The goal is to find a method for estimating the position of crop row centers (state vector) at time t, then update the estimation as new data arrives with next frame with minimum estimation error and be practically feasible. We want to predict the behaviour of a state vector of a linear stochastic system in a dynamic environment based on its previous behavior by minimizing the mean-squared estimation error. Here the description follows the approach taken by [98], [99] to describe the algorithm:

3.6.1 State Model

The crop row center tracking from video can be modeled using Newton's dynamic equation:

$$x_t = x_{t-1} + v_x(\Delta t) + \frac{1}{2}a(\Delta t)^2 \tag{3.10}$$

$$y_t = y_{t-1} + v_y(\Delta t) + \frac{1}{2}a(\Delta t)^2 \tag{3.11}$$

here x_t and y_t are position of crop row center at time t in x and y direction respectively. As shown in Fig. 3.13, (x_t, y_t) can be calculated from the crop rows detected by CAROLIF. Δt is the discrete time interval (frame interval in video sequence in this model). v and a are velocity and acceleration of the vehicle respectively. Equation (4) and (5) can be re-written in vector form:

$$\begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix} + \frac{1}{2}a(\Delta t)^2 \tag{3.12}$$

here $\Delta t = 1$ (consecutive frame) and we can assume constant velocity model ($a = 0$). Then we can re-write Equation (6).

$$x_t = Ax_{t-1} + w_{t-1} \tag{3.13}$$

where x_t is a state vector at time t, w_{t-1} is the Gaussian process noise with zero mean and covariance Q, that needs to be determined. A is state transition matrix with value:

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

3.6.2 Measurement Equations

Measurement of the system:

$$z_t = Hx_t + \nu_t \tag{3.15}$$

where H is the measurement matrix and z_t is the measurement observed at time t. ν_t is the Gaussian measurement noise with zero mean and covariance R. H is given as:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.16}$$

Choosing the optimal R_{2x4} and Q_{4x4} matrices are an important factor in KF design. They can be calculated empirically as part of system performance evaluation.

3.6.3 Update Equations

Measurement information from z_t is used to predict the state x_t. A *priori* estimate of state $\hat{x}_t^-$ and covarience error P_t^- is computed:

$$\hat{x}_t^- = A\hat{x}_{t-1} + w_t \tag{3.17}$$

$$P_t^- = AP_{t-1}A^T + Q \tag{3.18}$$

3.6.4 Correction Equations

It consists of three steps:

- 1) Compute Kalman Gain

$$K = P_t^- H^T (H P_t^- H^T + R)^{-1} \tag{3.19}$$

- 2) Update estimate with measurement z_t

$$\hat{x}_t = \hat{x}_t^- + K(z_t - H\hat{x}_t^-) \tag{3.20}$$

- 3) Update covariance error

$$P_t = (I - KH)P_t^- \tag{3.21}$$

where P is the prediction error covariance, R is the measurement error covariance and K is the Kalman gain. Kalman gain depends on the accuracy of the system. If accuracy of the measurement is high, the Kalman gain is high otherwise it's relatively low.

The above mentioned method can track the crop row centers over consecutive frame sequences. For a single crop row center, the above mentioned method is sufficient. For multiple crop row centers tracking, a data association algorithm may be needed to correctly associate each crop row center with a crop row center track. Hungarian algorithm [100] is such an algorithm which can be used to match detected crop row centers with tracked crop row centers, and determine which tracks have gone missing and which ones to be assigned new track.

Now the question is, how the above mentioned KF based tracking method improve CAROLIF based autonomous navigation? Let's assume the left front wheel of the vehicle is placed at the middle of the first two detected crop rows in Fig. 3.13. What we want is to keep the wheel at the middle of the two crop rows and move forward. Essentially what we want is to track the crop row center in Fig. 3.13. Then the position of the crop row center will be used as an input for autonomous vehicle navigation. But the true location of the crop row center is very difficult to measure accurately from continuous video feed. There are two

major problems: (1) It is hard to track an object if it moves beyond a searched region (this can happen due to sudden vibration); (2) lighting and occlusion can affect the accuracy of CAROLIF and thus making correct calculation of crop row center difficult. The first problem can be solved by predicting the location of crop row center. But to do that, the prediction method needs to be robust enough to handle the source of error. The above mentioned KF based tracking addresses these problems.

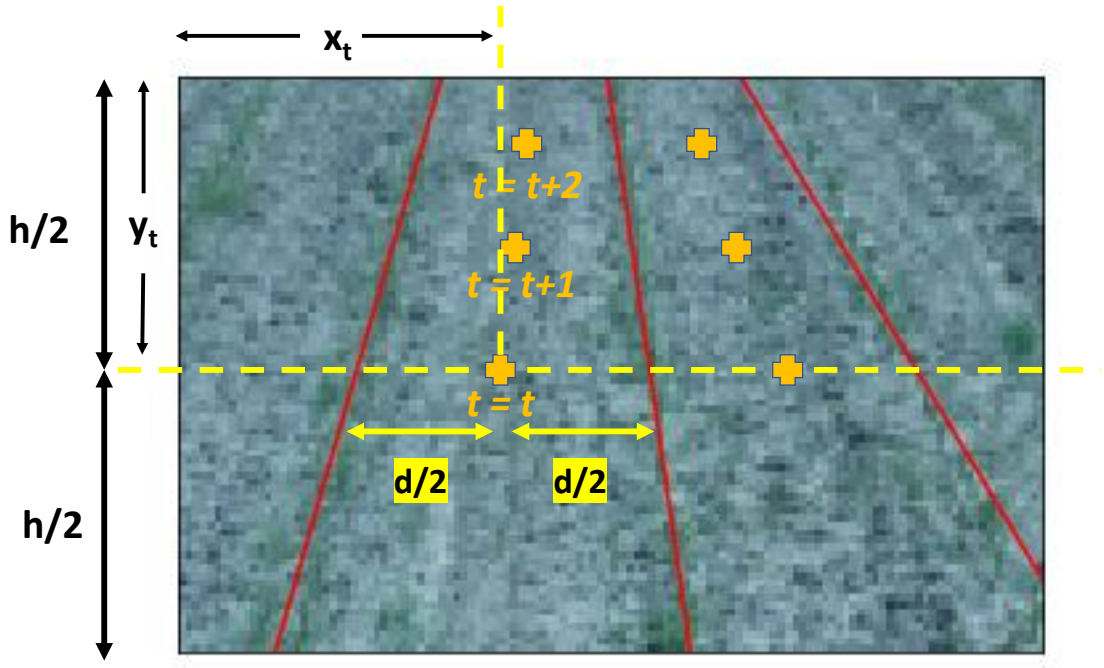


Figure 3.13. KF based crop row center tracking. Orange colored 'plus' signs show the calculated crop row center from CAROLIF output at time t.

3.7 Conclusions

In this research, a new clustering based crop row detection algorithm is presented and tested on static and video data. The proposed algorithm uses clustering and prior knowledge of geometric structure of crop rows to differentiate weeds from crop rows. With the use of a smaller ROI processing time is curtailed and straight lines are accurately fitted to reasonably curved crop rows. The algorithm also uses RANSAC (a robust line fitting technique) to further mitigate the effects of weeds on crop row detection. The proposed algorithm is

applicable for real-time usage on low spec hardware (without GPU) and shows good crop row detection accuracy at very challenging scenarios. Compared to Hough transform (IOU 0.58), sliding window (IOU 0.61), cluster-least sq. (IOU 0.7), CAROLIF shows superior performance (IOU 0.73). Moreover, for all the scenarios (easy and challenging) CAROLIF successfully detects all the crop rows with reasonable accuracy, whereas the other methods completely fails in some scenarios. Although CAROLIF shows slightly reduced performance compared to TMGEM (IOU 0.75), that can be due to incorrect ground truth value of CRBD dataset. Test on video data from an agricultural vehicle shows promising result for real-time application. We achieve 90.5% accuracy, 96.6% precision and 93.3% recall on a video containing very complex scenarios with shadow, intermittent crop growth and high weed pressure. To make this algorithm more robust when performance deteriorates, a detailed mathematical model of KF based crop row center tracking is presented. When correctly tuned using video data from agricultural vehicle, this tracking method can also increase detection accuracy of the whole system. The final goal of this work is to implement this algorithm on an off the shelf single-board computer (Jetson TX2) for real-time detection of crop rows from video input.

4. CNN BASED WEED CLASSIFICATION

In this chapter, we discuss the design and implementation of a convolution neural network (CNN) and transfer-learning based weed classification system on an autonomous AgBot. The United States leads in corn production and consumption in the world with an estimated 50 billion dollar in value per year. But young corn plants are vulnerable to weeds growing in the field. Novel and efficient methods are needed to improve the detection and elimination of weeds that is environmentally friendly, efficient, and more cost effective. In this work, we develop and test three different CNN based classifiers. A dataset of weed images are generated using artificially grown weeds in a greenhouse and naturally grown weeds in corn fields. Each classifier is trained to classify three different types of weeds: Common Cocklebur (*Xanthium strumarium*), Redroot Pigweed (*Amaranthus retroflexus*), and Giant Ragweed (*Ambrosia trifida*). Pretrained classifiers are deployed on a Yamaha Wolverine 4-by-4 which is retrofitted with autonomous navigation, weed detection and control capabilities in our laboratory. The test results show that the VGG16 with transfer-learning based classifier has the highest accuracy (99% training, 97% validation, 94% testing accuracy) for the same epoch whereas the InceptionResNetV2 with transfer-learning based classifier shows lowest overfitting but needs higher processing time to classify weeds from real-time video input.

4.1 Methodologies

We propose a weed classification system that contains two processes: weed classification and spray control. The output from the weed classification process is used as an input to the control spray mechanism. Classification and spray system pipeline of the proposed AgBot is showed in Fig. 4.1. An offline classification system uses supervised learning paradigm to train a model. Once the model is trained, the model is deployed on AgBot, which can classify a weed on the fly. After classification, a signal is sent to the spray system, which sprays appropriate herbicide in the accurate amount on the exact location.

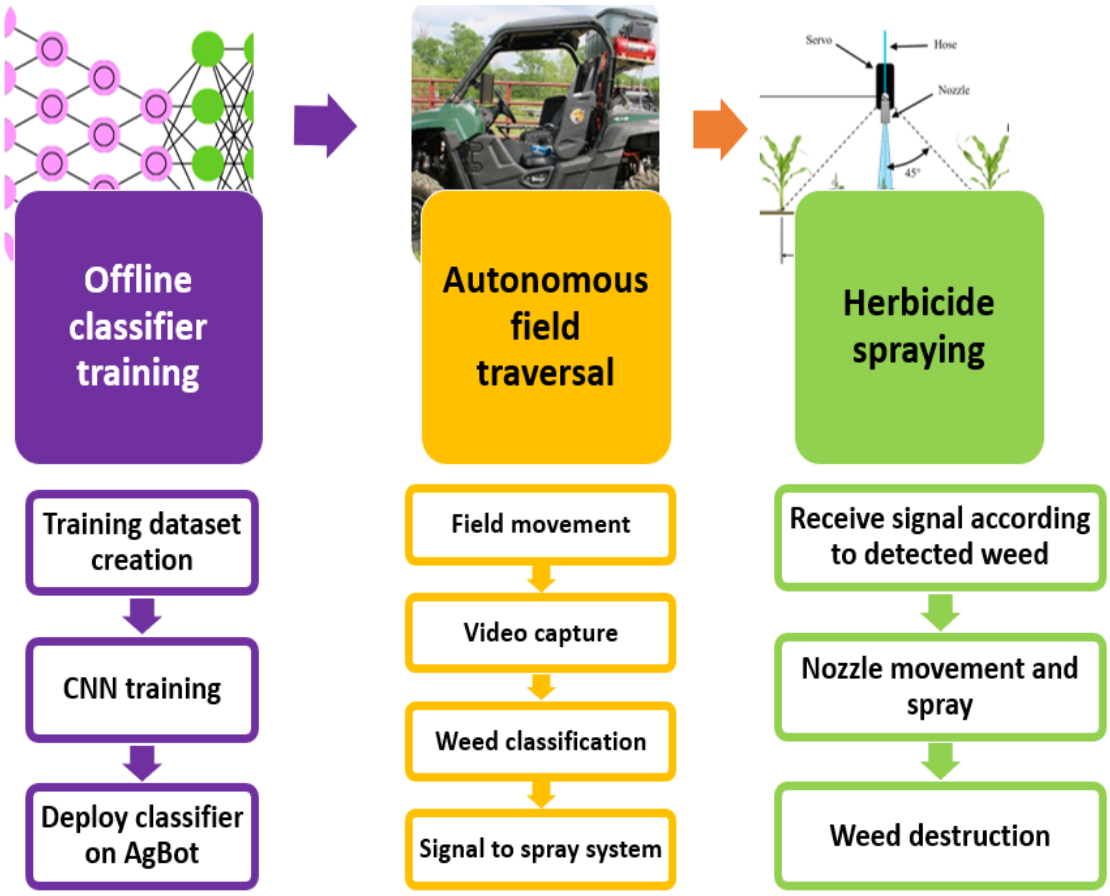


Figure 4.1. Classification and spray system pipeline of AgBot.

4.1.1 Weed Image Dataset and Image Processing

To build the dataset, a commonly available 16-megapixel digital camera is used to capture images. Common Cocklebur (Xanthium strumarium), Redroot Pigweed (Amaranthus retroflexus) and Giant Ragweed (Ambrosia trifida), these three types of commonly found corn weeds are grown in IUPUI Greenhouse to collect images. Additionally, authors have visited actual corn fields during summer time to capture weed images. Sample images of the dataset are presented in Fig. 4.2. Maximum input image size in this study is 299-by-299 pixels, for which the usage of a 16-megapixel camera is sufficient.

The total image data is divided into three partitions: train dataset, validation dataset and test dataset. The train-validation-test data split is roughly 80%-10%-10%, and the instances in these datasets are disjoint to preserve the principle of supervised learning paradigm.

Table 4.1. Weed image dataset.

Dataset	Cocklebur	Pigweed	Ragweed
Train image set (used for training the classifiers)	544	505	552
Validation image set (used for tuning hyper parameters)	65	62	69
Test image set (used for classification report)	65	62	69

Image instances in the train dataset is used to train the CNN based classifiers; instances in validation dataset is used to tune the hyper-parameters of classifier model, and finally, the instances in the test dataset is used to report the performance of the model. The number of images in each dataset is presented in Table 4.1.

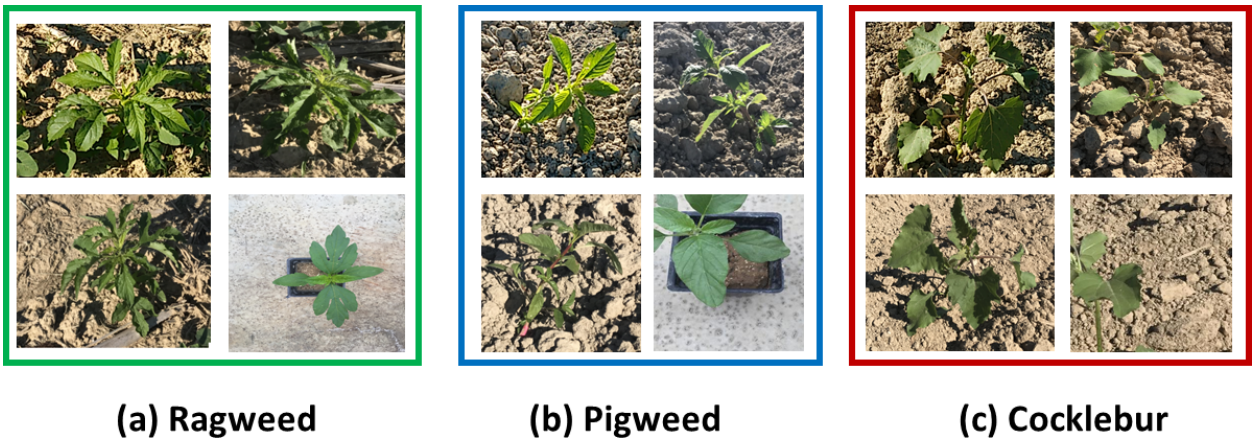


Figure 4.2. Sample images of the weed dataset.

4.2 CNN and Transfer-learning

CNN is a specific kind of deep neural network based classification model, which builds image features at different level of abstractions at different layers of the network. At each layer, a collection of kernels (or filters), which are in the form of 3×3 or 5×5 matrix, are used to detect specific shapes (or objects) within the image representation from the previous layer, by sliding the kernels over the 3D input feature map. The sliding process extracts a feature representing the 3D local patch of an image through element-wise multiplication

of image data with the kernel matrices (a convolution operation, from which the name originated) followed by a sum [101]. Layers of convolution enable the model to learn local image features at increasingly higher level of abstraction.

For supervised classification setup, CNN consists of convolution base (which contains several convolution layers) and fully-connected layers at the end. Convolution base creates a feature representation vector of the image, and the fully-connected layer converts that vector to a class label. This convolution characteristic gives CNN two interesting properties which sets them apart from traditional neural networks [102]:

- CNN learns translation invariant properties. It means, features learned by CNN can be applied to anywhere in an image for detection. As an example, if a CNN learns a feature at upper right corner of an image, it can detect the same feature at lower left corner of a new image.

- CNN learns spatial hierarchies of patterns. Convolution layers at the beginning will learn rudimentary patters like edges and colors. Second layer of convolution will learn patterns comprised of patterns from layer 1 (edges and colors), possibly colored edges and so on. Higher up layers will learn more complex features. If the training images are for cats, higher up layers will learn patterns like eyes and ears. This characteristic of CNN enables it to utilize the benefit of transfer-learning, which we will discuss next.

Transfer-learning in the context of CNN means taking the convolution base of a previously trained network, and use new training data to re-train final layers of CNN and create a new classifier on top of it. The main motivation of transfer learning in CNN comes from the fact that, at the early layers of the network, the model learns rudimentary patterns (edges, color), which are seen ubiquitously in a diverse class of images; so the weights of those layers can be frozen, and a new classifier can be built by re-training only the final few layers of the CNN, enabling the CNN to learn higher level features from the input images by utilizing pretrained rudimentary features from the early (base) layers. Transfer learning has several benefits: it saves substantial computation time by using pretrained network, which has substantially optimized weights at the early layers. As a result, good training performance can be obtained. And most importantly for our task, only a small number of

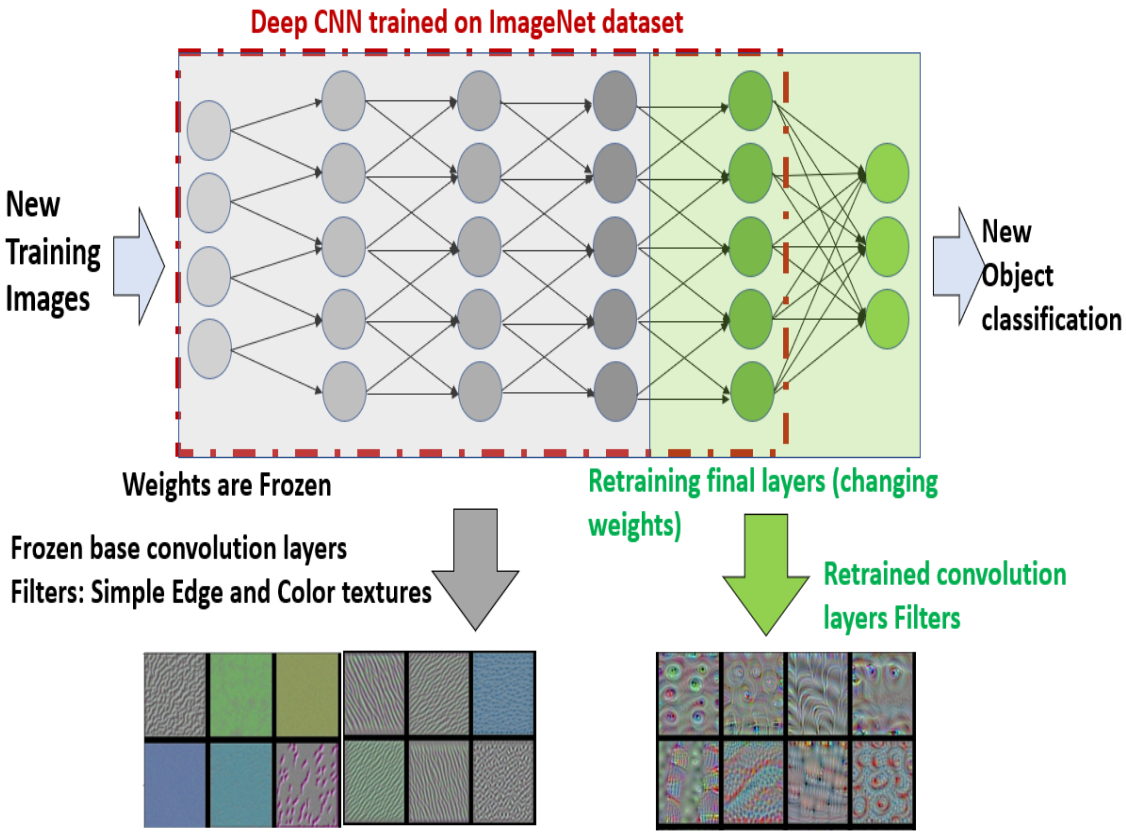


Figure 4.3. Simple representation of Transfer-learning.

training instances are needed for a good performance. The last fact holds due to the fact that for transfer learning, the number of weights that are being learned is small, so a small number of training instances suffices for achieving good performance.

For a detailed discussion of CNN and transfer learning, we refer the user to the F. Chollet's book [102]. Fig. 4.3 shows a simple representation of transfer-learning architecture.

4.2.1 CNN Models

For this study we have considered 3 different CNN models. Model-1 is a 6-layer deep CNN built from scratch (end-to-end model). Model-2 and model-3 are retrained (using transfer learning) VGG16 [103] and InceptionResNetV2 [104] respectively. These two models are re-trained using the small weed dataset and final layers are rearranged to classify three different weeds. Transfer learning retrains the final layer of the VGG16 and InceptionResNetV2 model

to classify a new dataset by exploiting the large amount of visual knowledge already learned from the Imagenet database. Previous research [45], [105], [106] has shown that transfer-learning has much lower computational requirements than learning from scratch and can be applied to various types of classification. All the training simulations were run on an intel core-i7, 8gb ram, Nvidia GTX 1060 6gb workstation. Models are built with Keras [107] with TensorFlow backend in Python 3.5.

We have selected epochs, learning rate, train batch size, validation batch size, optimizer and loss-function as hyperparameters for tuning. An iteration over all training data is called an epoch. Batch size determines how many times CNN has to update it's internal weights within each epoch. After each epoch, loss-function provides a loss value comparing true targets and predicted targets. Optimizer uses this loss value to update the network's weights. The goal of the network is to minimize the loss value during training. In this study, our goal is to achieve over 95% training and validation accuracy. An excellent explanation on how hyper-parameters should be tuned can be found on chapter 11 of Goodfellow et al.'s book [108]. Note that, random data augmentation is used during the training process of all three CNN architectures. Random images (using rescaling, rotation, width and height shift, zoom and horizontal flip) are generated during each training step to synthetically increase the number of training images.

6-layers CNN (Model-1)

A small 6-layers CNN model is built from scratch to compare results with other two models. Model-1 architecture is presented in Fig. 4.4. This model can be seen as a shorter VGG16 model to fit the small dataset. The structure of the model is same as VGG16 but number of parameters (weights) are significantly lower. The model parameters implemented in this study are learning rate (1e-04), optimizer (rmsprop), loss (sparse categorical cross-entropy). Training and validation accuracy with confusion matrix is presented in Fig. 4.5 and Table 4.2. respectively.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d_1 (MaxPooling2	(None, 74, 74, 32)	0
conv2d_2 (Conv2D)	(None, 72, 72, 64)	18496
max_pooling2d_2 (MaxPooling2	(None, 36, 36, 64)	0
conv2d_3 (Conv2D)	(None, 34, 34, 128)	73856
max_pooling2d_3 (MaxPooling2	(None, 17, 17, 128)	0
conv2d_4 (Conv2D)	(None, 15, 15, 128)	147584
max_pooling2d_4 (MaxPooling2	(None, 7, 7, 128)	0
flatten_1 (Flatten)	(None, 6272)	0
dense_1 (Dense)	(None, 512)	3211776
dense_2 (Dense)	(None, 3)	1539

Total params: 3,454,147
Trainable params: 3,454,147
Non-trainable params: 0

Figure 4.4. 6-layer end-to-end CNN architecture(Model-1).

Table 4.2. Confusion matrix based on test image set (Model-1).

True Label	Cocklebur	65	0	0
	Ragweed	15	42	5
	Pigweed	2	16	51
		Cocklebur	Ragweed	Pigweed
		Predicted label		

Transfer-learning with VGG16 (Model-2)

VGG16 (developed by visual geometry group, 16-layers architecture) is a CNN first used multiple small kernel filters instead of single large kernel filters. VGG16 is trained from the ImageNet Large Visual Recognition Challenge using the data from 2012, where it was tasked with classifying images into 1,000 classes. The top-5 error rate of VGG16 was 7.4% [103]. VGG16 is a bit older model but the structure is similar to model-1. This will give us an idea how much the accuracy and processing time depends on model depth and Imagenet weights. The model parameters implemented in this study included learning rate (1e-05), optimizer

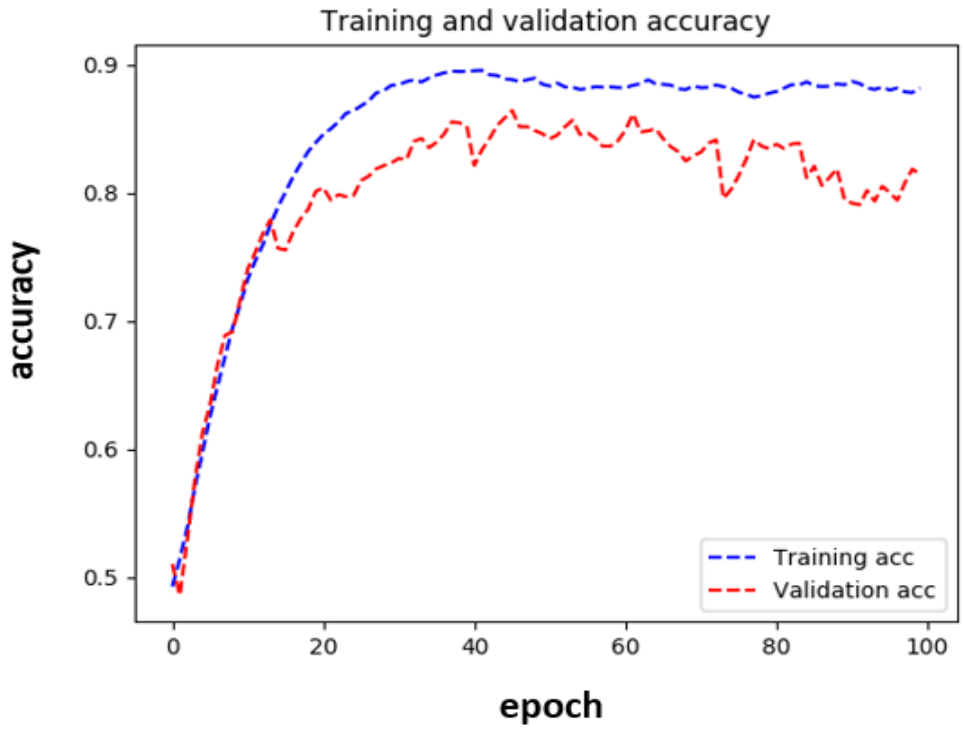


Figure 4.5. Training and validation accuracy (Model-1).

(Adam), loss (sparse categorical cross-entropy). Accuracy plot and confusion matrix of model-2 is presented in Fig. 4.6 and Table 4.3 respectively.

Table 4.3. Confusion matrix based on test image set (model-2).

True Label	Cocklebur	65	0	0
	Ragweed	4	55	3
	Pigweed	0	4	65
		Cocklebur	Ragweed	Pigweed
		Predicted label		

Transfer-learning with InceptionResNetV2 (Model-3)

InceptionResNetV2 is a CNN model developed by Google which combines their previous Inception CNN architecture with Microsoft's ResNet architecture. It is 164 layers deep, trained from the ImageNet Large Visual Recognition Challenge using the data from 2012, where it was tasked with classifying images into 1,000 classes. The top-5 error rate of In-

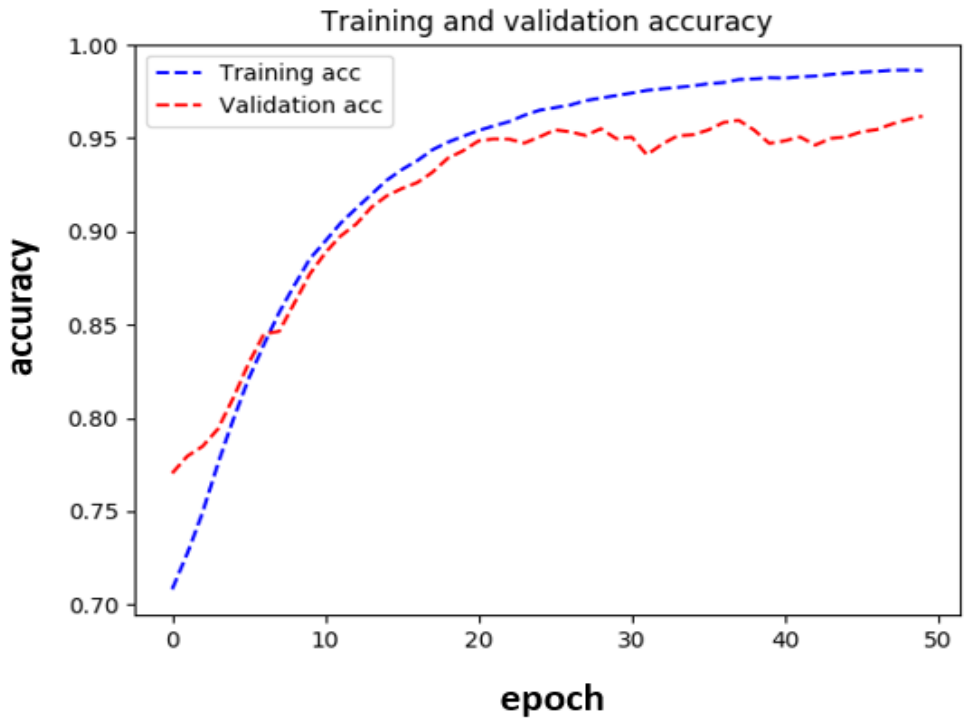


Figure 4.6. Training and validation accuracy (model-2).

ceptionResnetV2 was 3.08%[104]. InceptionResnetV2 is a newer model and the structure is completely different than model-1. The model parameters implemented in this study included learning rate (2e-05), optimizer (rmsprop), loss (sparse categorical cross-entropy). Model-3 accuracy plot and confusion matrix is presented in Fig. 4.7 and Table 4.4 respectively.

Table 4.4. Confusion matrix based on test image set (model-3).

True Label	Cocklebur	65	0	0
	Ragweed	3	44	15
	Pigweed	0	4	65
		Cocklebur	Ragweed	Pigweed
		Predicted label		

From the confusion matrix it is clear that Ragweed is the hardest one to classify. Model-2 misclassifies Ragweed as Cocklebur/Pigweed in 7 instances. Model-3 performs a little worse

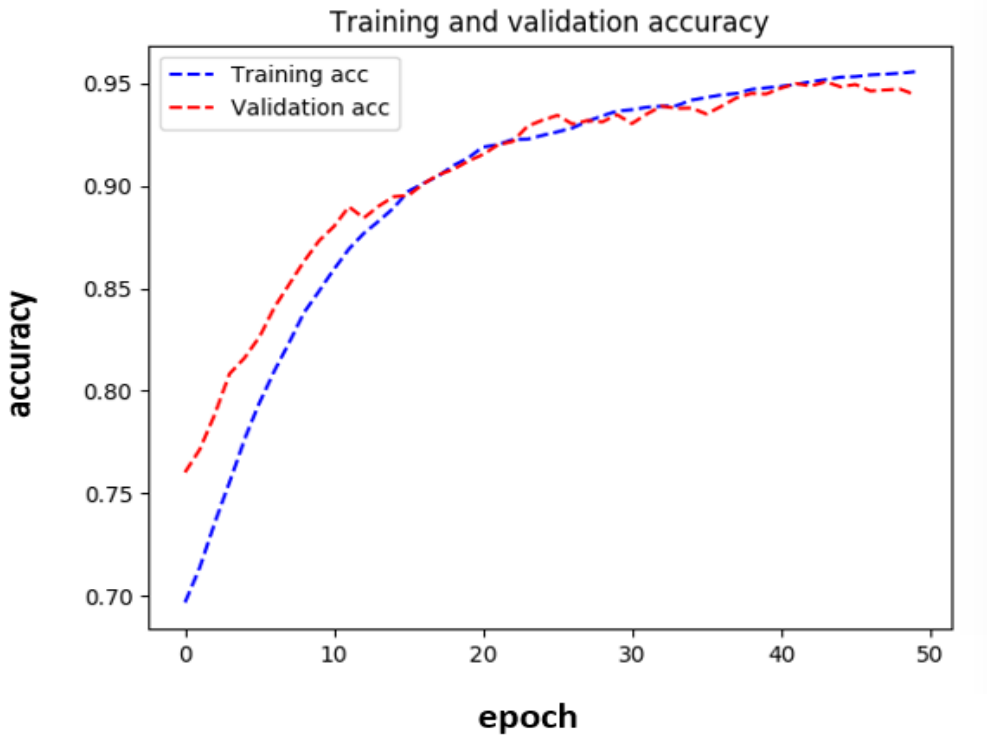


Figure 4.7. Training and validation accuracy (model-3).

(after 50 epoch) and misclassifies Ragweed as Cocklebur/Pigweed 18 times. Model-1 also performs the worst for Ragweed and misclassifies it as Pigweed/Cocklebur 20 times.

4.3 Classification Report

Table 4.5. Sample confusion matrix.

		Predicted class		
		A_1	$...A_j...$	A_n
Actual class	A_1	X_{11}	$...X_{1j}...$	X_{1n}
	$...A_i...$	X_{i1}	$...X_{ij}...$	X_{in}
	A_n	X_{n1}	$...X_{nj}...$	X_{nn}

From confusion matrix (Table 4.5) following parameters can be calculated:

$$Accuracy = \frac{\sum_{i=1}^{n} X_{ii}}{\sum_{\substack{i=1 \\ j=1}}^{n} X_{ij}} \tag{4.1}$$

$$Precision = \frac{X_{\text{ii}}}{\sum_{k=1}^{n} X_{k\text{i}}} \tag{4.2}$$

$$Recall = \frac{X_{\text{ii}}}{\sum_{k=1}^{n} X_{\text{i}k}} \tag{4.3}$$

$$F_1 score_{\text{i}} = \frac{2 * Precision_{\text{i}} * Recall_{\text{i}}}{Precision_{\text{i}} + Recall_{\text{i}}} \tag{4.4}$$

Confusion matrix is a simple metric which shows the actual and predicted labels of a classifier. Accuracy is the most intuitive performance measure which is simply the ratio of correctly predicted observations over all observations. Accuracy is a good performance indicator for symmetric data set. Precision looks at the ratio of correct positive observations. Low precision means false positives are high. Recall is a measure of the ability of a classifier to select instances of a certain class from a data set. Low recall means false negatives are high. The F_1 score is the weighted average of Precision and Recall (harmonic mean of precision and recall). Therefore, this score takes both false positives and false negatives into account. Usually more useful for uneven class distribution. Classification report with inference time is showed in Table 4.6.

Table 4.6. Classification report. C = Cocklebur, R = Ragweed and P = Pigweed. Inference time (classifying image to infer a result) tested on a core-i5, 8gb ram machine.

	model-1			model-2			model-3		
	C	P	R	C	P	R	C	P	R
Precision	0.79	0.74	0.89	0.94	0.94	0.96	0.96	0.92	0.79
Recall	1.0	0.68	0.74	1.0	0.89	0.94	1.0	0.7	0.94
F1 score	0.88	0.71	0.81	0.96	0.94	0.95	0.98	0.79	0.86
Training accuracy	0.85			0.99			0.97		
Validation Accuracy	0.8			0.97			0.93		
Testing Accuracy	0.81			0.94			0.88		
Image size	150x150x3			150x150x3			299x299x3		
Inference time	0.064 s			0.266 s			3.68 s		

For all the models, Cocklebur shows perfect recall. Pigweed is the hardest one to classify. Model-1 mostly false classifies Pigweed as Cocklebur and Model-2 mostly false classifies Pig-

weed as Ragweed. Pigweed is the hardest to classify as model-1 and model-3 show only 0.68 and 0.7 recall respectively. Model-2 consistently performs better than the other two models. But model-1 and medel-2 show overfitting (training accuracy higher than validation accuracy). But model-3 shows almost no overfitting, means training this model for longer time would have resulted better performance than model-2. But model-3 has 13 times higher inference time compared to model-2, which made it unfit to use for real-time classification with our hardware. Model-2 is chosen for real-time deployment which met all our requirements.

4.4 Real-time Classification from Video Input from Single Camera

Four scenarios are chosen to test the performance of model-2 from video feed. For 150 x 150 x 3 video input, frame per second (FPS) is around 5 for a core i5, 8gb ram machine and around 30 FPS for a core i7 8gb ram and GTX 1060 6gb machine. Videos are taken by a 12-megapixel digital camera placed about 2 feet above the plant at 45-degree angle.

4.4.1 One Pigweed in Video (Fig. 4.8)

For 30 seconds of video input, a full Pigweed plant is present in video frame for 26 seconds. The classifier is successful at the Pigweed classification with about 100% accuracy for the whole time. After 26 seconds, Pigweed plant is out of video frame and erroneous classification starts. This classifier is successful at classification from real-time video input when only one type of weed plant is present.

4.4.2 One Ragweed and One Pigweed in Video, Separately Placed (Fig. 4.9)

For 50 seconds of video input Ragweed is in frame for first 25 seconds and it gradually goes out of the frame and Pigweed comes in. For Ragweed, we see variation in classification accuracy between time 12-18 seconds. But overall, the model successfully classifies them with close to 100% accuracy. The model is successful at transitioning from one type of plant to another at real-time. These two scenarios are most commonly found in corn fields.

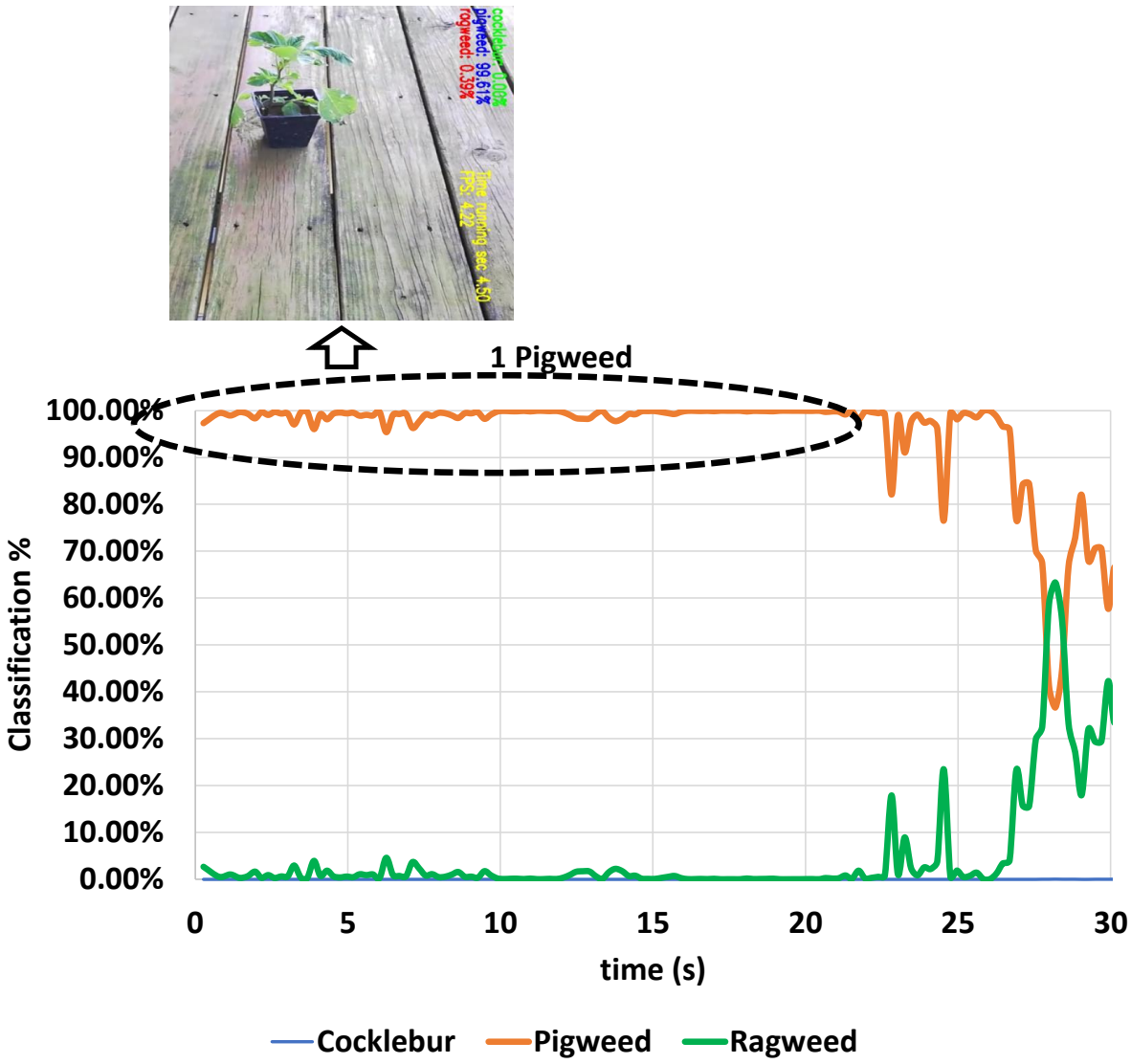


Figure 4.8. Classification accuracy from video input using transfer-learning VGG16 (model-2). Only one Pigweed plant on video.

4.4.3 Two Pigweeds and One Ragweed in Video Placed Together (Fig. 4.10)

Now we introduce a more complicated scenario. Here, Pigweed has higher quantity than Ragweed in the video. As a result, Pigweed should be classified with higher accuracy. Other than 15 seconds and 30 seconds mark in video, the model classifies Pigweed consistently. Ragweed leaves have a very specific 3-pronged feature. And around those specific times, the camera is focused on the Ragweed leaves and picked up this feature. That's when Ragweed

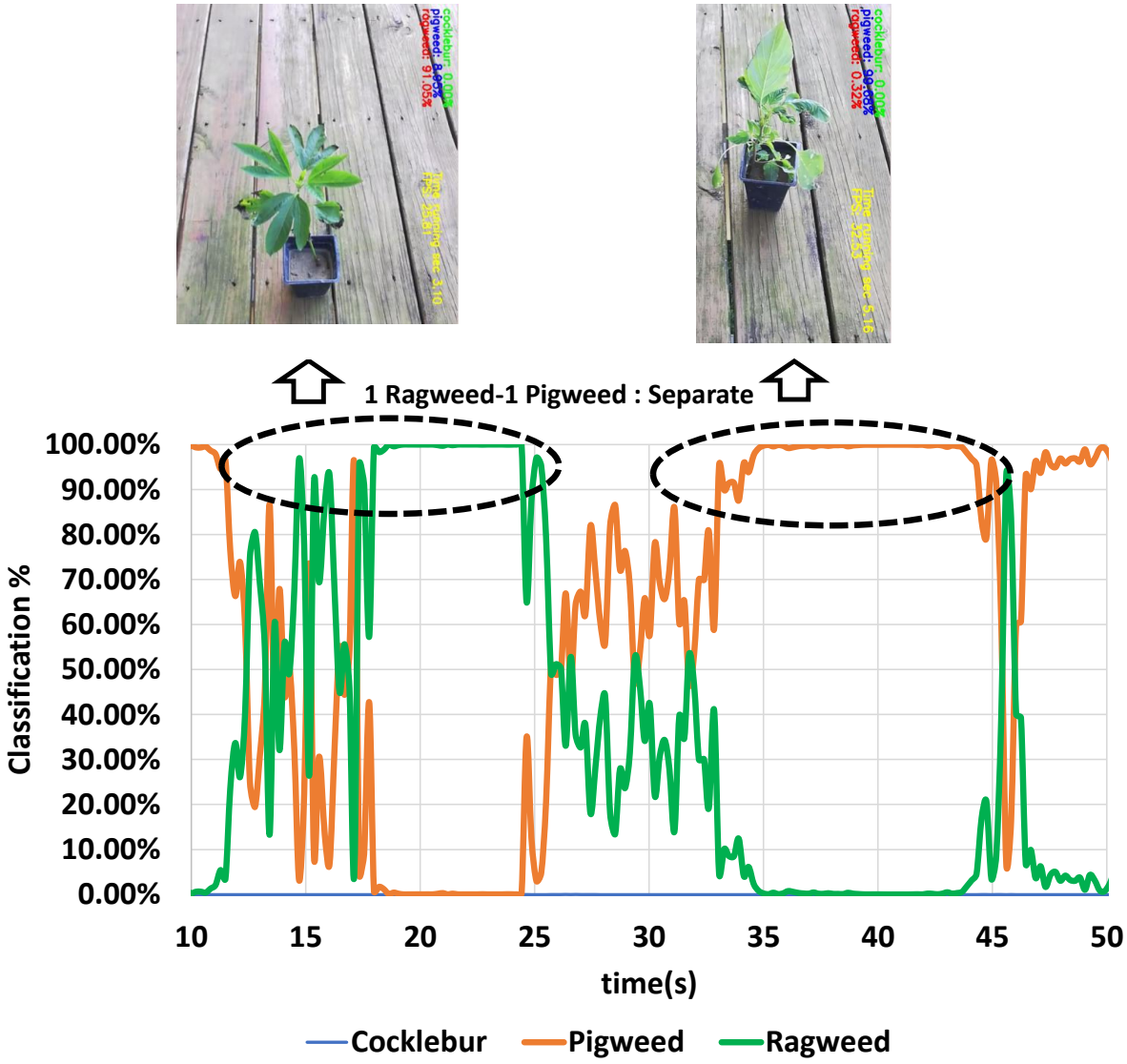


Figure 4.9. Classification accuracy from video input using model-2. One Pigweed plant and one Ragweed plant separately placed.

classification accuracy increases. This shows, how presence of a foreign object can decrease the classification accuracy of a CNN classifier. It also shows that specific structures of leaves are important for correct weed classification. If these structures are occluded due to wind or camera position, CNN classification accuracy can change rapidly.

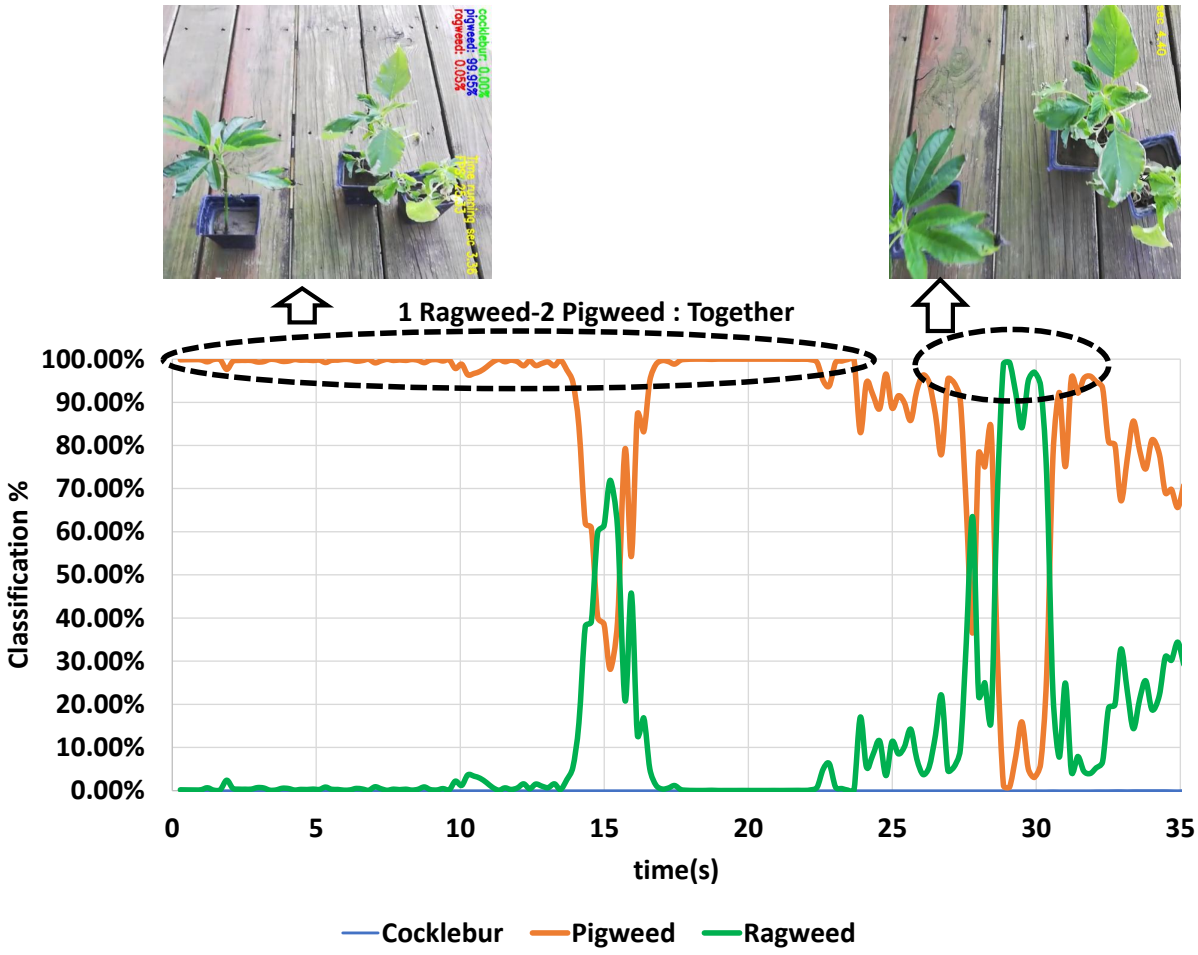


Figure 4.10. Classification accuracy from video input using model-2. Two Pigweed plants and one Ragweed plant on video, placed together.

4.4.4 One Pigweed and One Ragweed in Video Placed Together (Fig. 4.11)

This is the most complex scenario because both the weed plants have same quantity and equal probability of classification. We can't predict how the classifier is going to act. From Fig. 4.11, it is seen that the model classifies the weed depending on which features it is looking at. Between 5 to 12 seconds, the classification changes a lot between Ragweed and Pigweed. Between 12 to 20 seconds, it classifies Pigweed with higher accuracy. At the end, the classifier picks up the Ragweed leaves and classifies it with higher accuracy than Pigweed.

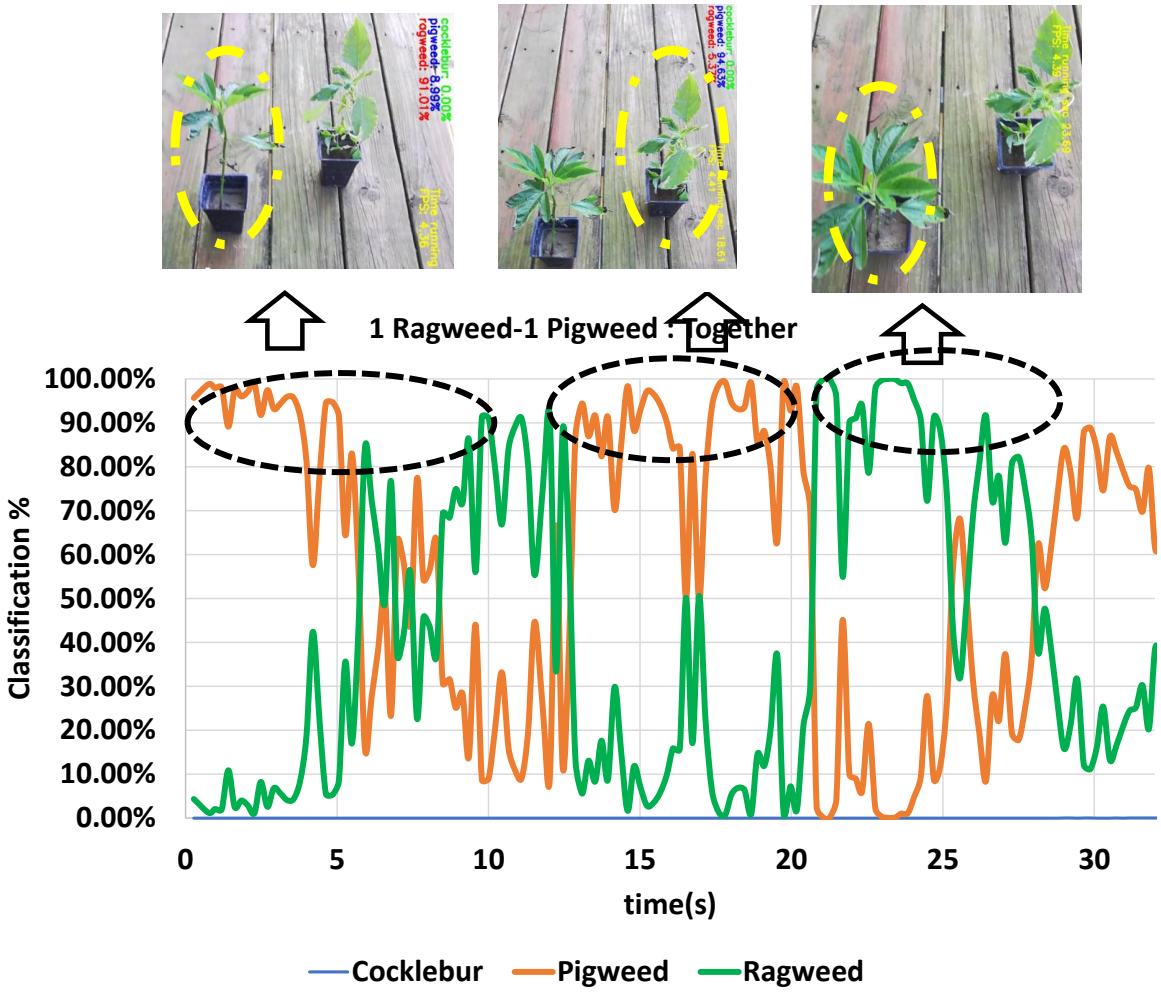


Figure 4.11. Classification accuracy from video input using model-2. One Pigweed plant and one Ragweed plant on video, placed together.

It is evident that, classification accuracy varies based on different scenarios. Also output from CNN classifier is not always stable and does have wrong classification output. But a stable and correct weed classification output is needed for correct herbicide spray system. In Fig. 4.9, if we look into time-steps 8-18, it is clear that the output from the CNN classifier is not clear to send a signal to the herbicide spray system. A decision level multi-sensor fusion algorithm is proposed in the next chapter to overcome this limitation and improve the overall system robustness.

4.5 Effect of Noise and Motion Blur on Classification Accuracy (model - 2)

Model-2's robustness is tested with artificially induced noise in images (gaussian and salt & paper noise). Field condition and low-quality sensor can cause noise in images. Noise can affect the accuracy of a classifier. In this case, gaussian and salt-and-pepper noise is artificially introduced in images to evaluate classifier performance. As seen from Fig. 4.12, noise doesn't affect the classifier accuracy. Varying the intensities of noises within a certain degree also doesn't affect the classification. This could be due to the deeper structure of the VGG16 network, which allows the networks more room to learn features that are not affected by noise [39]. The earlier layers are more affected by the high frequency noise. Responses of the last layers are less affected by noises than first layers. Using transfer learning, we only retrained the final layers of the VGG16 network. This could be another reason for this classifier to be more robust against noise. It should be mentioned that, at the noise level used in this paper, a human observer should be able to correctly classify the weeds. But, more detailed and systematic experiment is needed before drawing any concrete conclusions.

Image capturing from a vehicle on a corn field may produce blurriness, which will affect the classification accuracy. 3 stages of linear motion blurriness (low (10%), medium (25%), high (40%)) is artificially induced on 2 sets (near and far field) of images (Fig. 4.13). For low blur, both near and far field image is correctly classified. For near-field, incorrect classification occurs at medium and high blur. For far-field, incorrect classification occurs at only high-blur. Motion blur has higher impact on near-field image classification inaccuracy than far-field image. This leads to the conclusion that, camera should be set at a distance which correctly captures the features of the weed plants from as far as possible. Also, traditional masks (Unsharp Mask and Gaussian Mask) fails to correct the motion blurriness at high blur. Which indicates, when information is lost due to motion blur, it is hard to recapture.

4.6 Conclusions

This work presents an improved weed classification system using visual data for enabling autonomous precision weeding with transfer learning from the CNN, assuming prior weed

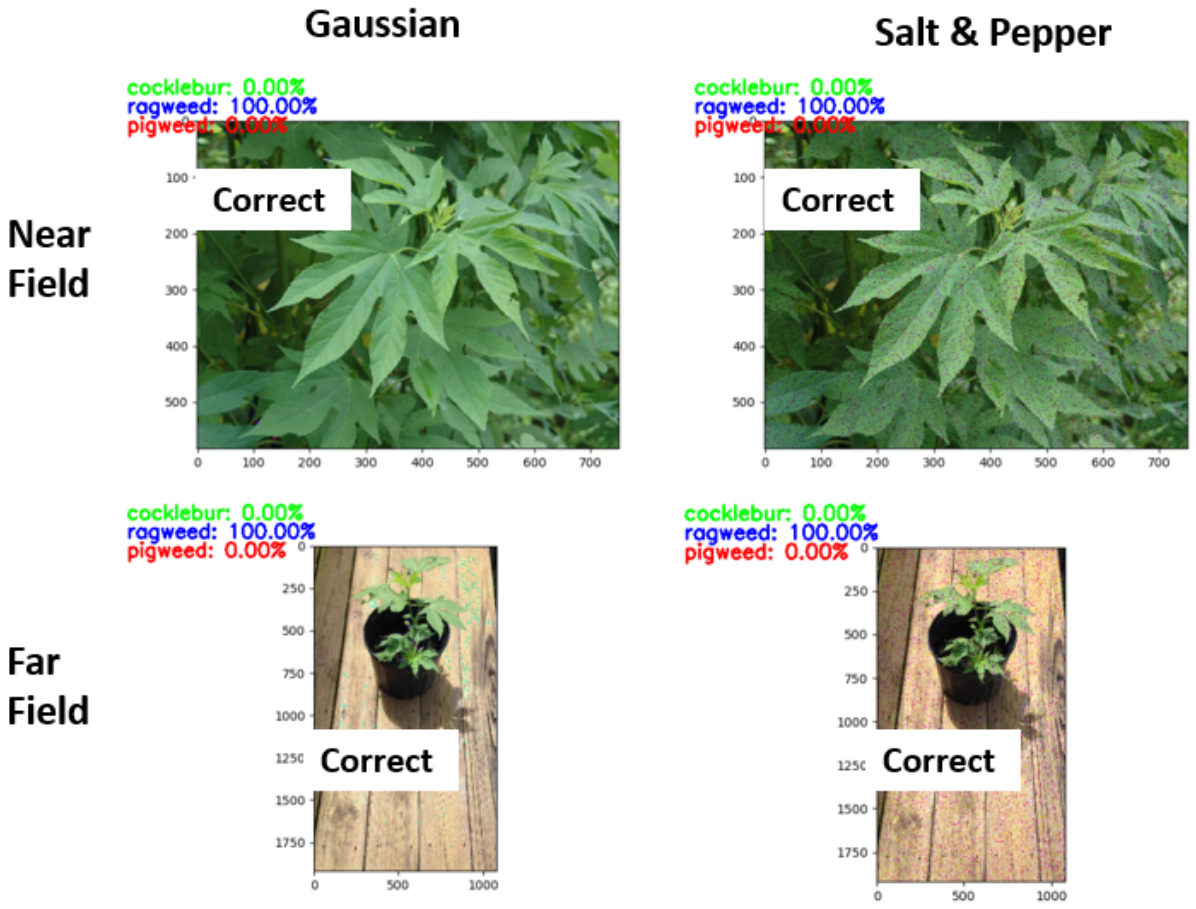


Figure 4.12. Effect of Gaussian and Salt-and-Pepper noise on classification accuracy.

species knowledge. The dataset is important because this dataset contains weed images which captures real life scenarios, which is not always the case for published work in weed classification. Transfer learning avoids the complex and labor-intensive step of hard-coded feature extraction from images and provides higher accuracy compared to end-to-end CNN model. We introduce a practical three-stage pipeline consisting of offline classifier training, autonomous field traversal, and herbicide spraying. The details for offline classifier training and real-time video input steps are summarized in this section. We have tested three different CNN structures, among which VGG16 based transfer learning shows highest testing accuracy (94%) and moderate classification time (0.266 sec on only CPU and 0.032 secs on GPU based system). Real-time classification from video input shows that the model is very accurate in classifying weed plants if only single type of weed is present in the frame, which is expected.

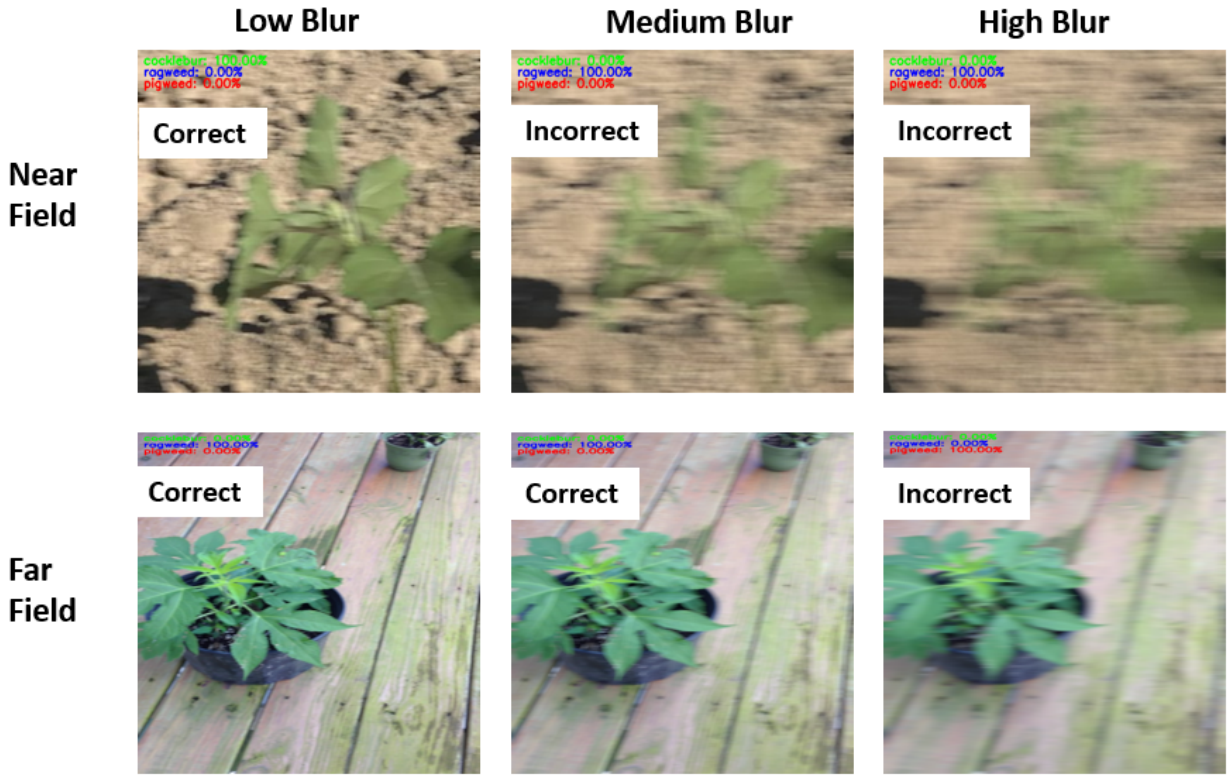


Figure 4.13. Effect of Gaussian and Salt-and-Pepper noise on classification accuracy.

When multiple types of weeds are present on a single frame, the model classifies the weed based on the dominating features of the plants that the camera pick up. A CNN detector instead of classifier can detect multiple objects in a single frame and create a bounding box to mark the object's position. We also show the importance of sensor placement and need of decision level fusion algorithm for accurate weed classification. It is also observed that CNN based classifier performs better for far-field images because the near-field images lost more information for the same amount of blur. Classification accuracy for this method is unaffected by Gaussian and salt-and-pepper noises but more testing is needed before drawing concrete conclusions. This study essentially shows that transfer learning is a viable method to successfully classify harmful weeds in real-time with a relatively small dataset from an autonomous robot.

5. ROBUST COLOR BASED WEED SEGMENTATION

Current image classification techniques for weed detection (classic vision techniques and deep-neural net) provide encouraging results under controlled environment. But accuracy can vary based on different real-world scenarios. Different lighting conditions and shadows directly impact vegetation color. Varying outdoor lighting conditions create different colors, noise levels, contrast and brightness. High component of illumination causes sensor (industrial camera) saturation. As a result, threshold-based classification algorithms may fail. To overcome this shortfall, we used visible spectral-index based segmentation to segment the weeds from background. Mean, variance, kurtosis, and skewness are calculated for each input image and image quality (good or bad) is determined. Bad quality image is converted to good-quality image using contrast limited adaptive histogram equalization (CLAHE) before segmentation. The main objective of this chapter's work is to construct a redundant system which can segment weed and crop (green pixels) under varying outdoor condition.

A version of this chapter was previously published by:
[ASME] [109] [https://doi.org/10.1115/IMECE2019-11077]

5.1 Color Based Image Segmentation

In this study, we have proposed a histogram-based image statistics system to identify image quality. If image is low-quality, it passes through CLAHE step for histogram equalization. Eventually good-quality image is passed to visible spectral based segmentation step for weed segmentation from background. Same good-quality image is passed through transfer-learning based CNN classifier for weed classification. Both the segmented weed image and classified weed image is passed to a decision-making system (future implementation). Based on illumination, noise and motion blur, CNN based weed classification accuracy can vary. Visible spectral based weed segmentation is a fail-safe to the CNN classifier, so that spraying herbicide on weed is not missed due to bad image quality. Fig. 5.1 shows the block diagram of the process.

Given an input image in RGB color space, each channel is separated by following system:
R = red channel of input image

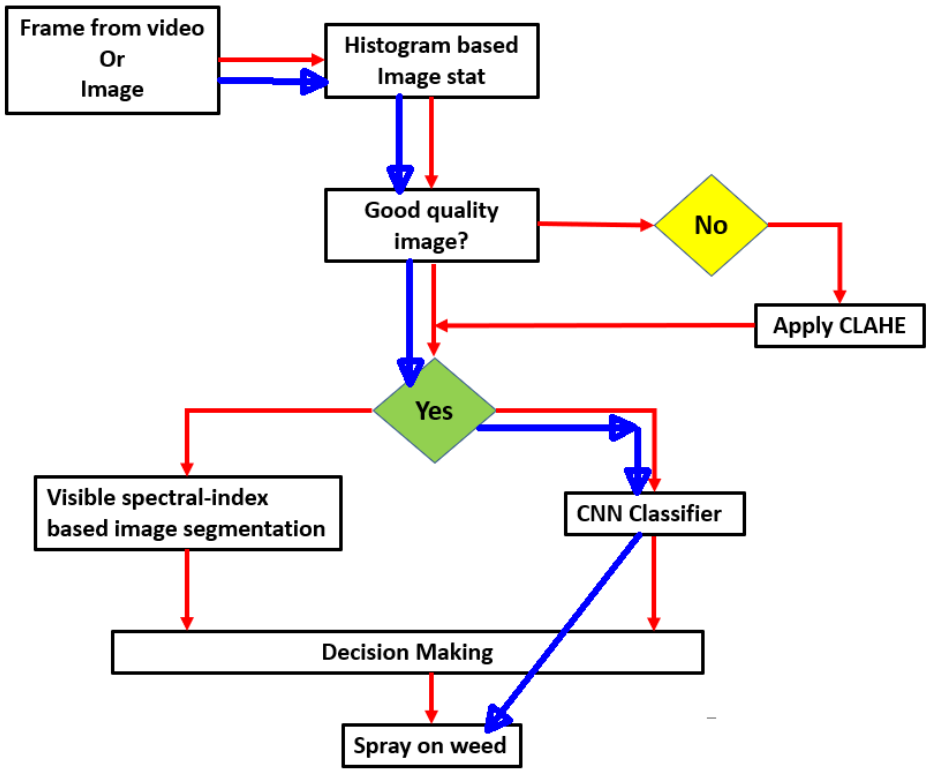


Figure 5.1. Block diagram of proposed image enhancement process. Blue arrow shows how usually spray decision is made. Red arrow shows the redundant system for bad quality image.

G = green channel of input image

B = blue channel of input image

After splitting the channels, the following normalization scheme is applied, which is common in agronomic image segmentation [87]:

$$\begin{aligned} r &= \frac{R_n}{R_n+G_n+B_n} \\ g &= \frac{G_n}{R_n+G_n+B_n} \\ b &= \frac{B_n}{R_n+G_n+B_n} \end{aligned} \tag{5.1}$$

where R_n, G_n and B_n are the normalized RGB values ranging from 0 to 1 and are obtained as follows:

$$R_n = R/R_{max},\ G_n = G/G_{max},\ B_n = B/B_{max} \tag{5.2}$$

where R_{max}, G_{max} *and* B_{max} are maximum values of R, G and B channels respectively. A small number is added to the denominator of normalization step to avoid division by zero. Green color (vegetation) can be extracted using the following equations [88]:

$$Excess\ Green\ , ExG = 2g - r - b \tag{5.3}$$

Excess green minus excess red [110]:

$$ExGR = ExG - 1.4r - g \tag{5.4}$$

Color index of vegetation extraction [111]:

$$CIVE = 0.441r - 0.811g + 0.385b + 18.78745 \tag{5.5}$$

The above three methods are combined to get the resulting green color segmentation:

$$GREEN = w_{ExG} * ExG + w_{ExGR} * ExGR + w_{CIVE} * CIVE \tag{5.6}$$

where w_{ExG}, w_{ExGR} *and* w_{CIVE} are weights for each color extraction method. The following weight values are used in this study: $w_{ExG} = 0.28$, $w_{ExGR} = 0.34$, $w_{CIVE} = 0.38$. The resulting combined image (GREEN), is linearly mapped to range in [0, 255], after which, it is thresholded by applying the Otsu's [112] method, obtaining a binary (single channel) image. Otsu's method assumes bi-modal distribution of histogram and calculates the value between two histogram peaks. In this study, we are segmenting weed from background (foreground-background segmentation) which is bi-modal. So, using Otsu's method is justifiable. The binary image then converted back to RGB (3 channel) image. Here green pixels identify plants and white pixels identify the background. This method is designed to cope with the variability of natural daylight illumination. Fig 5.2 shows how image is segmented using 4 different methods.

Fig. 5.3 shows performance of different segmentation methods under dark and bright lighting conditions. Under dark condition, ExGR performs better and is able to segment

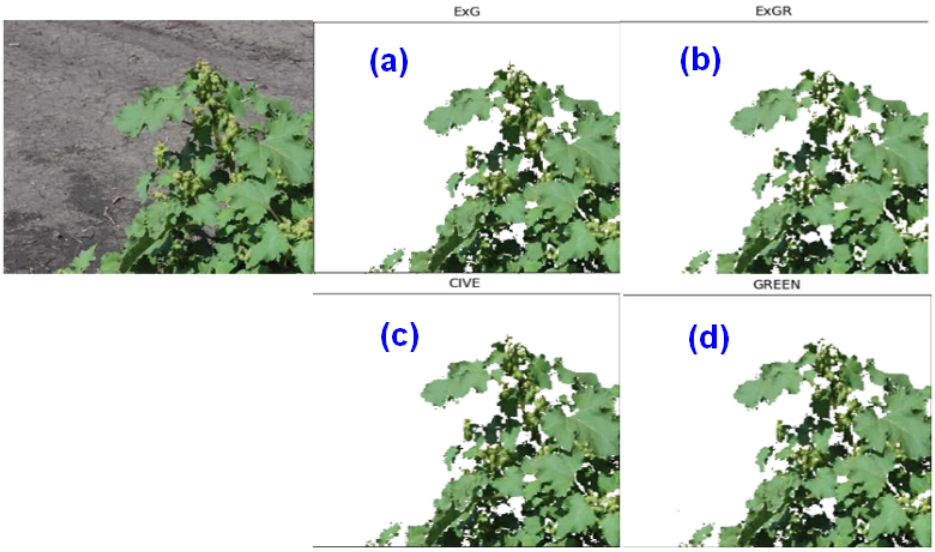


Figure 5.2. (a)ExG, (b)ExGR, (c)CIVE and (d)GREEN based image segmentation.

the weed properly. GREEN is able to capture most of the plant. Under bright light, ExGR fails completely to segment the weed from background. ExG and CIVE is able to segment most of the weed from background. This implicates two things. Firstly, although visible spectral-index based image segmentation is designed to cope with the variability of natural daylight illumination, they may fail under extreme condition. As a result, combination of different methods will be more robust and will be better at segmentation under different condition. Secondly, to prevent segmentation from failing, we have to understand when it is happening. We have to quantitatively measure the difference between 'good-quality' and 'bad-quality' image and convert the 'bad-quality' image into 'good-quality' image. That's where the histogram-based image statistics comes in.

5.2 Histogram Based Image Statistics

A histogram represents the distribution of pixel intensities (color or gray scale) in an image. When plotting the histogram, the X-axis serves as pixel values. For RGB color space, each channel will have pixel values in the range of 0 to 255 (for 8-bit image). The Y-axis indicates number of pixels. When we construct a histogram with 0-255 pixel values (bins), we are effectively counting the number of times each pixel value occurs at a certain

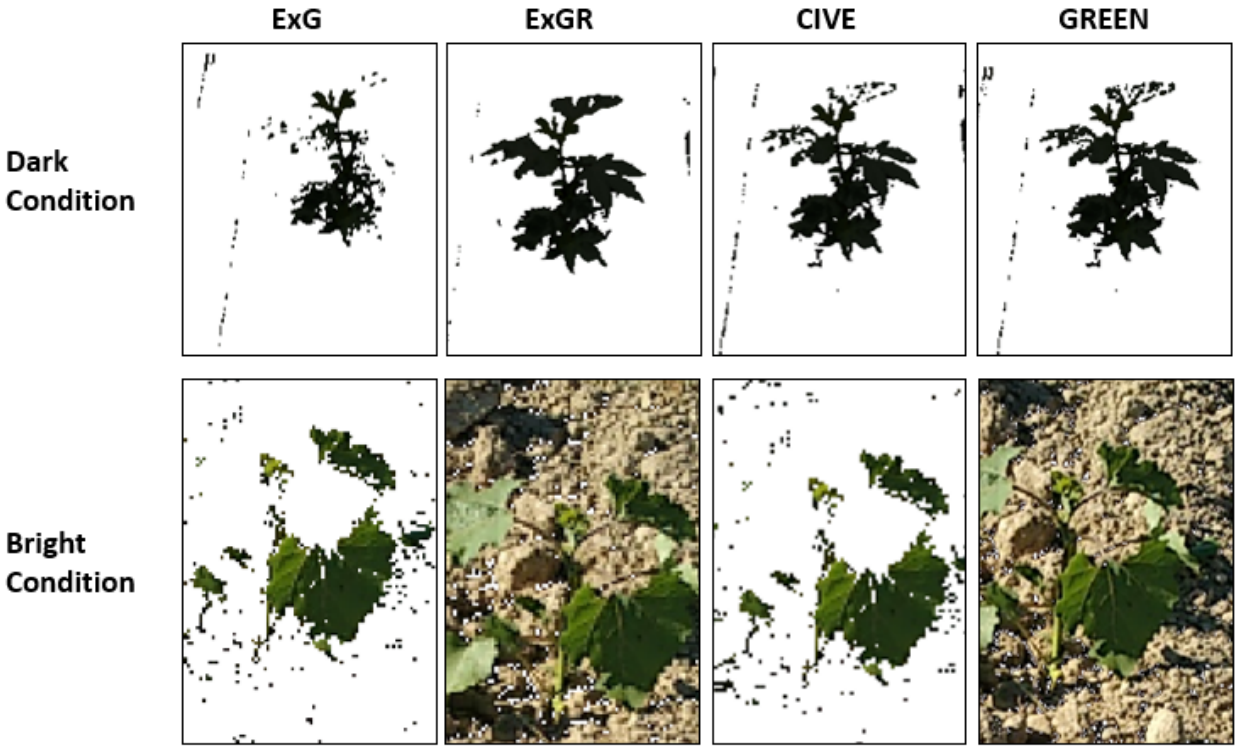


Figure 5.3. ExG, ExGR, CIVE and GREEN based image segmentation performance under different lighting conditions.

bin. If number of pixels (Y-axis) is high at bin (X-axis) value 0, the image is generally darker. If number of pixels (Y-axis) is high at bin (X-axis) value 255, the image is generally brighter. Examining the histogram of an image, a general idea of the contrast, brightness and intensity distribution can be achieved.

For gray image (single channel), histogram is defined by discrete function $h(a) = n_a$, where 'a' represents gray level (X-axis) and n_a represents number of pixels (Y-axis) for each value of 'a'. Probability of each 'a', $p(a) = h(a)/(M * N)$, where M and N are rows and columns of the image respectively. Let 'a' be a random variable denoting gray levels, the nth moment of 'a' about the mean is defined as [113]:

$$\mu_n = \sum_a (n(a) - m)^n . p(a) \tag{5.7}$$

Where m is mean value of sum of n(a)

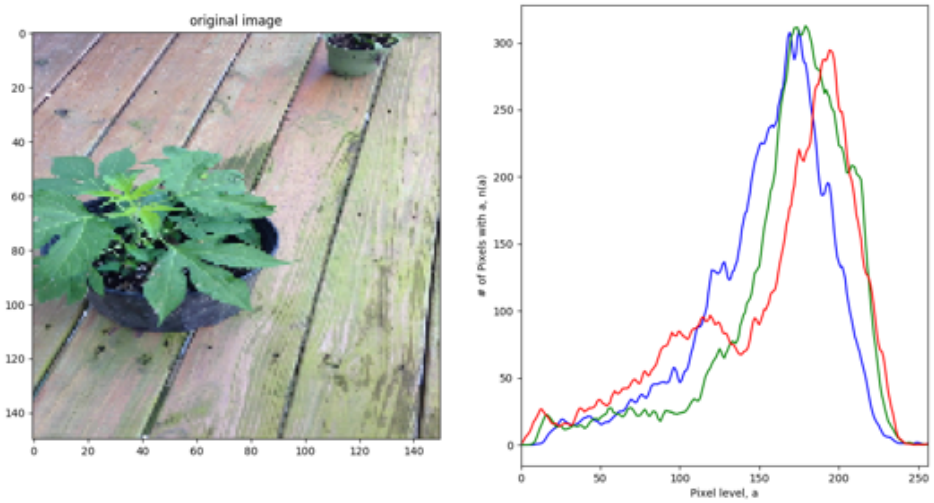
$$m = \sum_{a} n(a).p(a) \tag{5.8}$$

Following additional statistical parameters are calculated:

$$\begin{aligned} variance &= \mu_2(a) \\ skewness &= \frac{\mu_3(a)}{\mu_2^{1.5}(a)} \\ kurtosis &= \frac{\mu_4(a)}{\mu_2^2(a)} \end{aligned} \tag{5.9}$$

Following information can be summarized from the statistical parameters [79]. The mean determines the average level of brightness. Variance is a measure of gray-level contrast, where high values indicate dispersion of values around the mean and low values are indicative of a high concentration of values around the mean. The skewness measures the asymmetry in the distribution. A positive skewness is presented when the histogram has low values around high brightness (around X-axis pixel value 255) and high values in the part of low brightness values (around X-axis pixel value 0). In the opposite case the skewness is negative. The kurtosis provides information about how the distribution behaves around the peak. Low kurtosis indicates flat top parts in the histogram around the mean but high values are indicative of peaks around the mean with high slopes. Skewness and kurtosis are both zero for Gaussian distributions.

According to Romeo et al. [79], images highly contrasted are considered as images with sufficient quality and vice versa. An image with sufficient contrast should be identified by mean values in the central part of histogram, high variance, low skewness (positive or negative) and high kurtosis. On the contrary, an image with insufficient contrast is identified by mean values either low or high, high skewness (positive or negative) and low kurtosis. The next step is to determine the ranges of variability for the above parameters. The goal is to use these parameters to convert low quality images into high quality images.

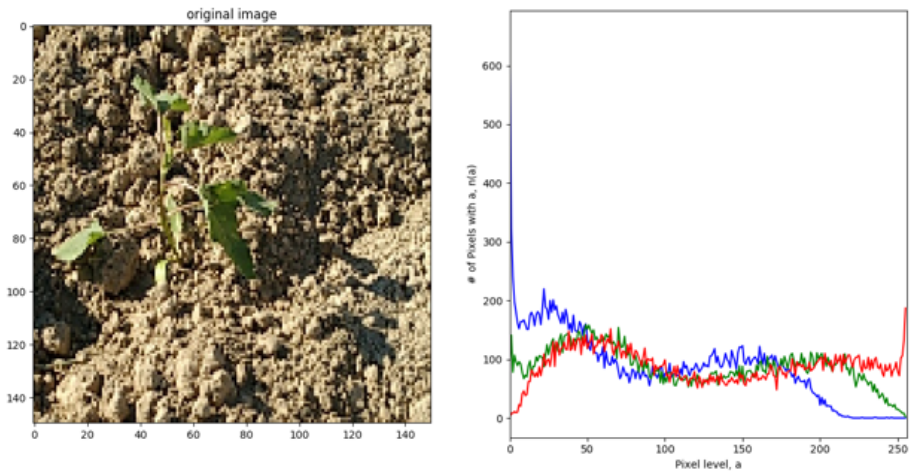


	R	G	B
Mean	157.5	200.63	183.5
Variance	7472.2	9181.6	7581.7
Skew	0.124	0.62	0.34
Kurtosis	1.59	2.07	1.98

Figure 5.4. A good-quality image with histogram and statistics parameters for R, G and B channels of the image.

5.3 Proposed Brightness/Contrast Control Steps

The shortcoming of histogram enhancement (HE) is over-enhancement in images with large smooth area. For images captured under low light condition, HE causes over-enhancement after contrast enhancement and increases the noise. Contrast limited adaptive histogram equalization (CLAHE) overcomes the over-enhancement problem of HE by minimizing noise-like artifacts in homogeneous regions [114]. As we have seen already, visible spectral-index based image segmentation may fail for bad-quality image. Applying CLAHE to bad-quality image before segmentation will solve this problem. In this study we propose an automatic method to define image quality based on image histogram statistics so that low-quality images can be converted to high quality images. We have tested the effects of CLAHE's 'clip-



	R	G	B
Mean	96.44	99.18	140.43
Variance	646.96	841.81	10778.7
Skew	0.35	0.102	3.70
Kurtosis	3.45	2.43	18.80

Figure 5.5. A bad-quality image with histogram and statistics parameters for R, G and B channels of the image.

limit' and 'step-size' variables on histogram enhancement and clip-limit = 3.0 and grid-size = (8x8) is fixed for this application. Steps are as follows:

1. Determine the ranges of good quality image parameters (mean, variance, skewness and kurtosis) using a set of training images. See equation (5.10) for parameters.

2. Calculate mean, variance, skewness and kurtosis of test image for R, G and B channels. In this case only mean and kurtosis according to equation (5.10). Determine if good or bad quality image.

3. For bad-quality image, apply CLAHE to convert it to good-quality image.

4. Apply visible spectral-index based image segmentation (GREEN) on good-quality image.

5. Apply morphological operation (opening and closing) on segmented image to fill-up void and reduce noise.

Fig. 5.4 and 5.5 shows example of good-quality and bad-quality images respectively. Good-quality image has mean at the center (relatively), higher variance and lower skew and lower kurtosis compared to bad quality image. Fig. 5.6 shows how CLAHE affects the histogram. GREEN segmentation fails to process original image because blue-channel is left tilted (low-quality image). CLAHE equalized the histogram and GREEN can then successfully segment the image. CLAHE step needs 0.212 seconds to process a 150x150x3 image.

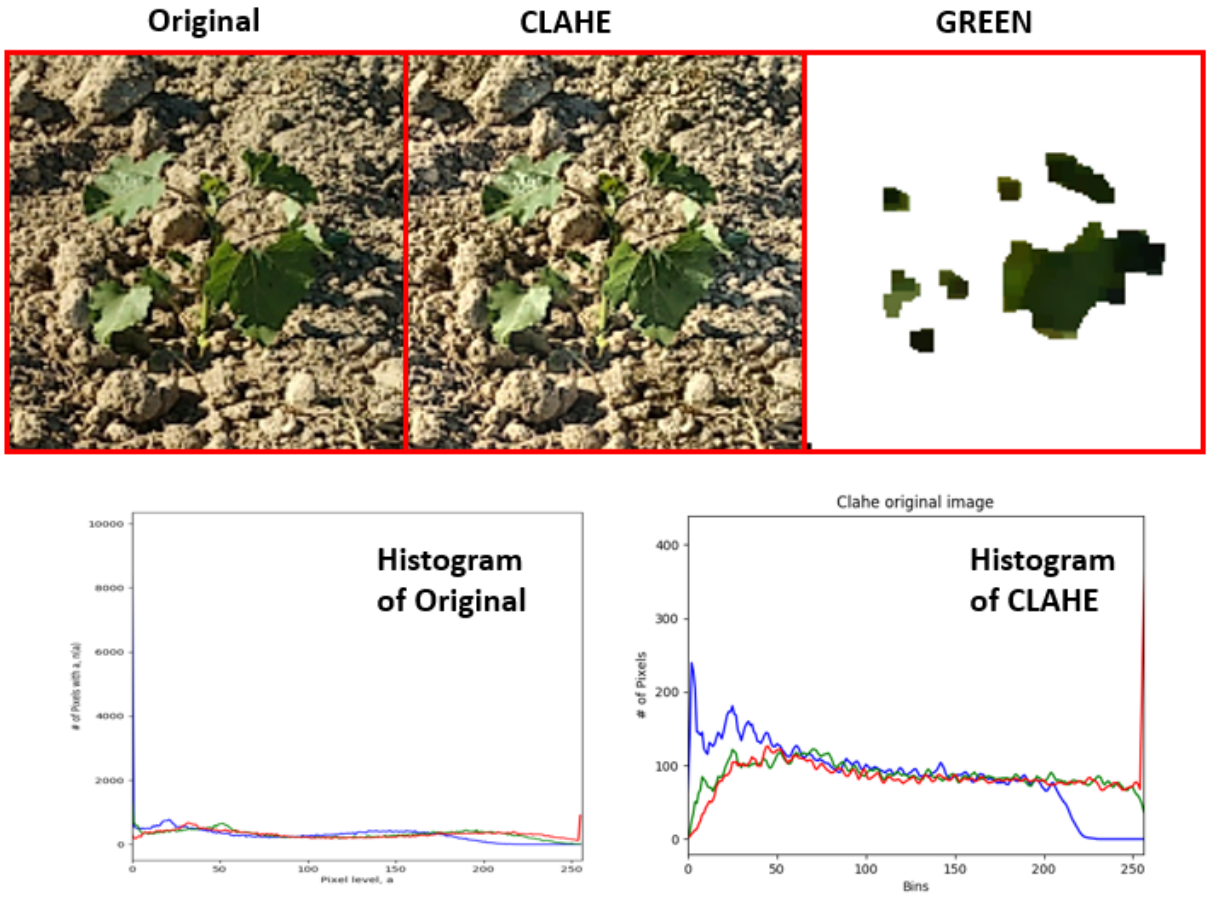


Figure 5.6. Original bad-quality image, bad-quality image after CLAHE, visible spectral-index based image segmentation (GREEN) on CLAHE applied image, histogram of original image, histogram of CLAHE image.

5.3.1 How to Quantify 'Good' and 'Bad' Quality Image

Now that we have an idea how good-quality and bad-quality images affect the histogram-based statistics parameters, we need to determine the ranges of these parameters. When proper ranges of good-quality image parameters are determined, CLAHE can be applied on bad-quality images to convert them into good-quality images. Table 5.1 and 5.2 shows mean, variance, skew and kurtosis (statistics parameters) values in R, G, B channels for good-quality and bad-quality images respectively. 15 images are selected for both (good and bad) cases and their minimum, maximum and average values are shown here.

Table 5.1. Mean, variance, skewness, kurtosis values for 'good-quality' images in the three R, G, B spectral channels.

		R	G	B
	Max	663.5	411.1	365.5
Mean	Min	157.5	170.8	153.4
	Avg	288.8	253.6	243.2
	Max	246758.1	96448.5	68238.1
Variance	Min	7472.2	7577.2	5685.2
	Avg	54515.4	30349.6	24210.3
	Max	0.37	0.32	0.34
Skew	Min	0.009	0.021	0.013
	Avg	0.19	0.29	0.16
	Max	1.73	2.07	1.98
Kurtosis	Min	1.47	1.5	1.52
	Avg	1.64	1.77	1.69

From Table 5.1 and 5.2, several observations can be made. In general, good quality image (Table 5.1) has mean at the center (relatively), higher variance and lower skew and kurtosis compared to bad quality (Table 5.2) image. For all three channels, parameter values overlap for mean, variance and skew; for good and bad quality images. Bad quality maximum mean value is higher than good quality minimum mean value. Same is also true for variance and skew. For kurtosis, overlapping occurs for G and B channels. This indicates that the training weed dataset available to the authors are not diverse enough to clearly distinguish between good quality and bad quality images. But from Table 5.1, statistics features for good quality images are clearly identified. Skew and kurtosis parameters are chosen because

they are reasonably distinguishable between good and bad quality images. This step needs an average 0.0023 seconds to process a 150x150x3 image.

Table 5.2. Mean, variance, skewness, kurtosis values for 'bad-quality' images in the three R, G, B spectral channels.

		R	G	B
	Max	391.1	368.5	436.6
Mean	Min	96.4	99.2	140.4
	Avg	198.4	192.4	235.4
	Max	33391.4	35723.3	53021.5
Variance	Min	646.9	841.8	9955.4
	Avg	11090.1	11355.6	22818.8
	Max	1.15	0.87	3.72
Skew	Min	0.35	0.10	0.32
	Avg	0.76	0.54	2.06
	Max	5.13	4.6	18.8
Kurtosis	Min	2.06	1.88	1.87
	Avg	3.27	2.78	10.13

The following criteria are chosen to distinguish the bad quality image from good quality image. Good quality image criteria are:

$$\begin{aligned} &R\ channel\ kurtosis < 1.73 \\ &G\ channel\ kurtosis < 2.07 \\ &B\ channel\ kurtosis < 1.98 \\ &R\ channel\ skew < 0.37 \\ &G\ channel\ skew < 0.62 \\ &B\ channel\ skew < 0.34 \end{aligned} \tag{5.10}$$

In conclusion, for every image kurtosis and skew will be calculated. If skew and kurtosis are within range of Eq. (5.10), it is good quality image and GREEN is applied for segmentation. If bad quality, CLAHE will be applied to make the image good quality. Then segmentation will be applied.

5.4 Conclusions

The proposed algorithm using histogram-based image statistics for image quality determination and brightness control, significantly improves the inherent limitation of color-based segmentation (bad performance under extreme lighting) under various test conditions. The real-time processing of the histogram based statistical algorithm is only 0.0023 seconds which is significantly shorter than that of classification (0.266 seconds) and CLAHE (0.212 seconds) methods. Such a fast processing time of the proposed histogram-based algorithm is suitable for real-time control applications.

6. DECISION LEVEL SENSOR FUSION IN SPACE AND TIME DOMAIN

Multi-sensor data fusion technology in an important tool in building real-time decision making applications. Modified Dempster-Shafer (DS) evidence theory can handle conflicting sensor inputs and can be applied without any prior information. As a result, DS based information fusion is very popular for decision making application. But original DS theory produces counter-intuitive results when combining highly conflicting evidences from multiple sensors. An algorithm which is successful in fusing highly conflicting information in spatial and time domain is not easy to find. In this chapter, we tackle these complications of original Dempster-Shafer (DS) framework. An eight-step algorithm is proposed which can eliminate the inherent paradoxes of classical DS theory. A novel entropy function is proposed based on Shannon entropy which is better at capturing uncertainties compared to Shannon and Deng entropy. Multiple examples are presented to show that the proposed method is effective in handling conflicting information in spatial domain. Simulation results show that, proposed algorithm has the best convergence rate in space domain fusion and accuracy compared to other "revision of original evidence before combination" methods from open literature. Proposed eight-step algorithm is slightly modified and then applied to time domain to achieve the sequential combination of time-domain evidence. Simulation results show that this method is successful in capturing the changes (dynamic behavior) in time-domain object classification. This method also shows better convergence rate, anti-disturbing ability and transition property compared to other methods available from literature.

A portion of this chapter was previously published in:

[Sensors] [115] [https://doi.org/10.3390/s19235187]

[Sensors] [116] [https://doi.org/10.3390/s19214810]

6.1 Dempster-Shafer Evidence-based Combination Rule

6.1.1 Frame of Discernment (FOD)

The frame of discernment contains M mutually exclusive and exhaustive events (also represented by X in this research).

$$X = \Theta = \{\theta_1, \theta_2, ..., \theta_M\} \tag{6.1}$$

The representation of uncertainties in the DS theory is similar to that in conventional probability theory and involves assigning probabilities to the space Θ. However, the DS theory has one significant new feature: it allows the probability to be assigned to subsets of Θ as well as the individual element θ_{i}. Accordingly, we can derive the power set 2^{Θ} of DS theory:

$$2^{\Theta} = \{\phi, \{\theta_1\}, \{\theta_2\}, ..., \{\theta_1, \theta_M\}, ..., \Theta\} \tag{6.2}$$

where ϕ is empty set. It is clearly seen in (6.2) that the power set 2^{Θ} has 2^M propositions. Any subset except singleton of possible values means their union. For example, $\{\theta_1, \theta_2, \theta_3\} \equiv \{\theta_1 \cup \theta_2 \cup \theta_3\}$. Complete probability assignment to power set is called basic probability assignment (BPA).

6.1.2 Basic Probability Assignment (BPA) / Mass Function

Evidences in DS theory are acquired by multi-sensor information. Mass function (mass) is a function, $m : 2^{\Theta} \rightarrow [0, 1]$ that satisfies (6.3) and (6.4):

$$m(\phi) = 0 \tag{6.3}$$

$$\sum\{m(\theta) \forall \theta \in 2^{\Theta}\} \tag{6.4}$$

m is called basic probability assignment. Elements of power set having $m(\theta) > 0$ is called focal elements. This can be explained with the help of a simple example. Let the three objects to be detected be, $\Theta = \{a, b, c\}$. Powerset, $2^{\Theta} = 2^3 = \{\phi, a, b, c, \{a, b\}, \{a, c\}, \{b, c\}, \Theta\}$.

From a sensor or by an expert following mass values are assigned, $m(a) = 0.2,\ m(b) = 0.3,\ m(a,b) = 0.4,\ m(a,b,c) = 0.1$.The four subsets are called focal elements.

6.1.3 Dempster-Shafer Rule of Combination

The purpose of data fusion is to summarize and simplify information rationally, obtained from independent and multiple sources. DS combination rule emphasizes on the agreement between multiple sources and ignores all the conflicting evidences through normalization. Any two mass functions B and C over same FOD with atleast one focal element in common can be combined into a new mass function using DS combination rule. The combination of two mass functions can also be said as taking the orthogonal sum, $\oplus$. The combination of two mass functions the DS combination rule for combining two evidences m_1 and m_2 is defined:

$$m_{12}(A) = \frac{\sum_{B \cap C}\{m_1(B).m_2(C)\}}{1 - K} \tag{6.5}$$

When $A \neq \phi$ and $m(\phi) = 0$.

$$K = \sum_{B \cap C = \phi} \{m_1(B).m_2(C)\} \tag{6.6}$$

where K is the degree of conflict in two sources of evidences. The denominator $(1 - K)$ is a normalization factor, which helps aggregation by completely ignoring the conflicting evidence and is calculated by adding up the products of BPA's of all sets where intersection is null. DS combination rule in (6.5) and (6.6) conforms to both commutative law and associate law.
$m_1 \oplus m_2 = m_2 \oplus m_1$
$(m_1 \oplus m_2) \oplus m_3 = m_1 \oplus (m_2 \oplus m_3)$

6.1.4 Belief and Plausibility Function

Given a basic assignment m we can define a belief function: Bel : $m : 2^{\Theta} \rightarrow [0, 1]$, such that for any $A \subset \Theta$:

$$Bel(A) = \sum_{B \subseteq A} \{m(B)\} \tag{6.7}$$

$Bel(A$ measures the belief that the element is member of A. $m(A)$ measures the amount of belief that one commits exactly to A alone, $Bel(A)$ measures the total belief that the special element is in A. Based on the same premise,

$$Pl(A) = 1 - Bel(\bar{A}) \tag{6.8}$$

$Pl(A)$ measures the degree to which one fails to doubt A. $Pl(A)$ measures the total belief mass that can move into A, whereas $Bel(A)$ measures the total belief mass that is constrained to A.

Example 6.1: Given, m(A) = 0.48, m(B) = 0.24, m(C) = 0.08, $m(\Theta) = 0.2$
Bel (A,B) = m(A) + m(B) + m(A,B) = 0.48 + 0.24 = 0.72
Pl (A,B) = m(A) + m(B) + m(A,B) + m(A,C) + m(B,C) + m(A,B,C) = 0.48 + 0.24 +0.2 = 0.92

Similarly, Bel and Pl values can be calculated for all the BPA which is shown in Table 6.1.

Table 6.1. Bel and Pl values for Example 6.1

	A	**B**	**C**	**A,B**	**A,C**	**B,C**	**A,B,C**
Bel(.)	0.48	0.24	0.08	0.72	0.56	0.32	1.0
Pl(.)	0.68	0.44	0.28	0.92	0.76	0.52	1.0

6.2 Paradoxes (Source of Conflicts) in DS Combination Rule

DS theory introduced and developed by Dempster and Shafer [52], [117], [118], has many merits by contrast to Bayesian probability theory [119]. But to use DS sensor fusion algorithm for robust application, we have to overcome the fusion paradoxes. Based on the application in a multi-sensor system, this theory also has its shortcomings [120]. The different levels of performance of sensors, cluster, and interference of a complex environment may lead to conflicts among evidences. When evidences are highly conflicting, the fusing results

obtained by the DS combination method are normally contrary to common sense. When the conflicting factor K is close to 1, this rule cannot obtain reasonable fusing results as the denominator is approximate to 0. These counter-intuitive phenomena of the DS theory are called paradoxes. According to [58], there are mainly three types of paradoxes.

6.2.1 Completely Conflicting Paradox:

In this situation, there are two sensors and one sensor output completely contradicts the other sensor output. Following example depicts the situation:

Example 6.2: In the multi-sensor system, assume that there are two evidences in the frame, $\Theta = \{A, B, C\}$ and proposition A is true.
Sensor 1 : $m_1(\text{A}) = 0.7$, $m_1(\text{B}) = 0.2$, $m_1(\text{C}) = 0.1$,
Sensor 2 : $m_2(\text{A}) = 0.0$, $m_2(\text{B}) = 1$, $m_2(\text{C}) = 0$,

Here two sensors are completely conflicting each other. The conflicting factor in (6.6) is K = 1, which reports that evidences from sensor 1 and sensor 2 are completely conflicting. Under such circumstances, the DS combination rule cannot be applied.

6.2.2 "One Ballot Veto" Paradox:

For a multi-sensor system (more than two sensors), one sensor completely contradicts all other sensor output. Following example depicts the situation:

Example 6.3: In the multi-sensor system, assume that there are four evidences in the frame, $\Theta = \{A, B, C\}$ and proposition A is true.
Sensor 1 : $m_1(\text{A}) = 0.7$, $m_1(\text{B}) = 0.2$, $m_1(\text{C}) = 0.1$,
Sensor 2 : $m_2(\text{A}) = 0$, $m_2(\text{B}) = 0.9$, $m_2(\text{C}) = 0.1$,
Sensor 3 : $m_3(\text{A}) = 0.75$, $m_3(\text{B}) = 0.15$, $m_3(\text{C}) = 0.1$,
Sensor 4 : $m_4(\text{A}) = 0.8$, $m_4(\text{B}) = 0.1$, $m_4(\text{C}) = 0.1$,
Clearly, sensor 2 is faulty and contradicts the results of other 3 sensors. Applying DS combination rule we get: $K = 0.9$, $m_{1-2-3-4}(A) = 0/0.1 = 0$, $m_{1-2-3-4}(B) = 0.097/.01 = 0.97$, $m_{1-2-3-4}(C) = 0.003/0.1 = .03$.

Fusing results are contrary to the assumed proposition that A is true. High value of K proposes high contradiction among sensors. This counterintuitive result is caused by the erroneous sensor 2 values. Interestingly, DS combinatio rule completely omits a proposition even if a single sensor outputs zero evidence.

6.2.3 "Total Trust" Paradox:

Here, one sensor highly contradicts the other sensor. But both of them have common focal element with low evidence. Following example depicts the situation:

Example 6.4: In the multi-sensor system, assume that there are two evidences in the frame, $\Theta = \{A, B, C\}$.
Sensor 1 : $m_1(\mathrm{A}) = 0.95$, $m_1(\mathrm{B}) = 0.05$, $m_1(\mathrm{C}) = 0$,
Sensor 2 : $m_2(\mathrm{A}) = 0.0$, $m_2(\mathrm{B}) = 0.1$, $m_2(\mathrm{C}) = 0.9$,
Applying DS combination rule: $m_{1-2}(A) = 0$, $m_{1-2}(B) = 1$, $m_{1-2}(C) = 0$, $K = 0.99$. Here, common sense suggests that either $m(A)$ or $m(C)$ is correct. But, the wrong proposition B is identified to be true with total confidence even though sensor 1 and 2 nearly negates this idea.

6.3 Eliminating the Paradoxes of DS Combination Rule

Existing modified methods are divided mainly into three categories:

1. Modification of DS Combination Rule

Smet's rule [121] is essentially the Dempster rule applied in Smet's transferable belief model. Smet believed that conflict is caused by incompleteness of frame of discernment, Θ and moved mass of conflict directly to ϕ as an unknown proposition. This model is slightly different formulation of DS theory, but the ideas are essentially the same. In Yager's rule [122], the mass associated with conflict is directly given to universal set Θ. Yager's rule provide the same results when conflict is zero. Although these two methods solve the conflict situation theoretically, the uncertainty of the system still exists. Bicheng et al. [123] modified Yager's rule and conflicting probability of the evidences are distributed to every

proposition based on average support. Inagaki [124] defined a continuous parameter class of combination operations which subsumes both DS and Yager's rule. Depending on conflict of information his combination rule changed between DS and Yager combination rule. But, based on experience, if an engineer applied a weighting factor to one of the sensors credibility, this rule can't be applied. Zhang [125] pointed out that DS rule fails to take into account the focal element intersection. He presented the 'two frame' representation of DS theory where he measured focal element intersections based on cardinality. Li [126] used the interaction between focal elements and proposed two weighted redistribution method which considers the associative relationship among the evidences collected from multi-sources. His argument was, if a body of evidence is greatly supported by others, this piece of evidence should be more important and has great effect on the final combination results. On the contrary, if a body of evidence is highly conflicting with others, this piece of evidence should be less important and has little effect on the final combination results. But, all these methods sometimes violate the theoretical properties of DS combination rule like commutativity and associativity.

2. Revision of Original Evidence before Combination

Commutative and associative properties of DS rule are important for multi-sensor information fusion which may get lost when the original rule is tampered with. As a result, the propositions are modified so that conflict among the evidences are resolved before applying them in DS combination rule. Chen et al. [127] used triangular functions to set a fuzzy model for each sensor. Assuming each sensor output is gaussian, BPA was determined from the sensor outputs using the fuzzy model. Then the raw BPA was weighted using the credibility of each BPA before fusing. Sun [128] also used fuzzy membership function to convert sensor value to fuzzy value. Support degree was calculated using an error distance function. If sensor output is not gaussian, then fuzzy set methods can't be applied. Instead of distance function, an entropy function (Deng entropy [119]) was used to calculate the credibility of evidence in [129]. This was inspired by Murphy's method [130], which used an average of BPAs. Murphy's method had fast convergence rate but failed to consider the relation between focal elements. Jiang [131] used an entropy function to measure the weight

of the evidence to modify them before applying to DS rule. Xiao [132] used almost the same procedure as Jiang but with a different distance function to measure the credibility. Murphy's method is the simplest to implement and most of the methods within this type is inspired by his method.

3. Hybrid Technique Combining both Modification of DS Rule and Original Evidence

Through the comparison between two kinds of conflict resolutions, it is easy to see the underlying logic of two methods. Method 1 cancels the normalization step in DS theory and redistribute the conflict with different measure. Method 2 consider the essential differences between propositions of each sensor in multi-sensor systems and solve the conflict by modifying the original evidence. If method 1 and 2 is combined, then the inherent paradoxes of DS rule are solved. Building on this idea, Lin et al.[133] and Ye Fang et al. [134] published several new improvements of original DS combination rule. They improve the fusion results but often too complicated and over-engineered to apply for real-time use. These methods also loose commutative and associative properties of DS rule.

How to accurately measure the conflicting evidences under DS framework is still an open issue. Keeping the commutative and associative properties of the original DS combination rule and eliminating the paradoxes are critical for multi-sensor fusion. There is still room for improvement to properly measure the conflicts between evidences and obtain appropriate weights for each evidence. Based on this, an improved combination method is proposed which follows "Revision of Original Evidence before Combination" method. A novel entropy function is proposed which can better capture the conflicts between evidences. Reward and penalty is imposed to evidences based on how they agree or disagree with each other. The amount of reward or penalty is determined by the entropy function. Then, the modified weight value (reward or penalty) is applied in adjusting the body of the evidences before using the Dempster's combination rule (n - 1) times, when there are n number of evidences (sensors). The experiments illustrate that the proposed method is reasonable and efficient in coping with the conflicting evidences.

6.4 Entropy in Information Theory under DS Framework

Information is a measure of the compactness of a distribution; logically if a probability distribution is spread evenly across many states, then it's information content is low, and conversely, if a probability distribution is highly peaked on a few states, then it's information content is high [135]. Information is a function of distribution. Entropy measures the compactness of a distribution of information. Entropy is zero when BPA is assigned to a single element, thus creating the most informative distribution. When BPA is uniformly distributed, entropy is at maximum, agrees with the idea of least informative distribution.

In information theory, Shannon entropy [136] is often used to measure the "amount of information" in a variable.

$$E_{Sh} = -\sum_{\mathrm{i}=1}^{n} p_{\mathrm{i}}.log_2(p_{\mathrm{i}}) \tag{6.9}$$

where n is the amount of basic states in a state space, p_{i} is the probability of state i. It is clear that the quantity of entropy is always associated with the amount of states in a system. In the framework of DS evidence theory, the uncertain information is represented by both mass functions and the FOD. Deng entropy [119] considers both.

$$E_{Deng} = -\sum m(A).log_2\frac{m(A)}{(2^{|A|}-1)} \tag{6.10}$$

where $|A|$ denotes the cardinality of the focal element A. Other works related to entropy under DS framework can be found in the literature [137]. Based on Shannon and Deng entropy, We propose a new belief entropy, which considers Bel and Pl of mass function, cardinality of focal elements and number of elements in FOD. The goal of the proposed entropy is to capture the uncertainty of information under DS framework which are omitted by Shannon and Deng entropy.

$$Proposed\ Entropy, E_p = -\sum \frac{Bel(A)+Pl(A)}{2}.\ log_2(\frac{Bel(A)+Pl(A)}{2.(2^{|A|}-1)}.\ \exp^{(\frac{|A|-1}{|X|})}) \tag{6.11}$$

where $|X|$ denotes the cardinality of X, which represents the number of element in FOD. The exponential factor $\exp^{(\frac{|A|-1}{|X|})}$ in the new belief entropy represents the uncertain information in number of elements of FOD that has been ignored by Deng entropy. This

probability interval considers lower and upper bounds of evidence that are the Bel and the Pl, respectively. The new belief entropy can better measure the uncertainty of BPA.

6.4.1 Properties of Proposed Entropy Function

Property 1: Mathematically, the value range of the new belief entropy is (0, $+\infty$). According to DS evidence theory, a focal element A consists at least one element and the limit of it's element number is the scale of FOD. FOD consists at least one element and there is no maximum limit, thus the range of $|A|$ and $|X|$ are the same, denoted as [1, $+\infty$). The range of a mass function m(A) is (0, 1]. Depending on value of $|A|$, believe (Bel) and plausibility (Pl) range could be between (0, $+\infty$]. In proposed entropy equation, where $|A| \in [1, +\infty)$, $|X| \in [1, +\infty)$, $Bel(A) \in (0, +\infty]$ and $Pl(A) \in (0, +\infty]$. Thus the range of the proposed entropy can be denoted (0, $+\infty$).

Property 2: New belief entropy can degenerate to the Shannon entropy when the mass function is Bayesian. If the mass function m(A) is Bayesian, then BPA is assigned only on single element subset, then $|A| = 1$. In this case, the new belief entropy can degenerate to the following equation which is exactly equal to Shannon entropy:

$$= -\sum \frac{m_i+m_i}{2}.log_2((\frac{m_i+m_i}{2.(2^1-1)}).\exp^{(1-1)/|X|}) = -\sum m(A).log_2(m(A))$$

Property 3: Non-negativity. We know that, $0 < (Bel(m_i) + Pl(m_i))/2 < 1$. As a result, $Entropy(m) > 0$. Only if m(A) = 1 and only if A is Bayesian, then Entropy (m) = 0. Thus, new entropy satisfies the non-negativity property.

Property 4: Consistency with DS theory framework. The new entropy is consistent with DS theory framework. Thus, it satisfies the consistency with DS theory framework properties.

Property 5: Probability consistency. If m is Bayesian, then $m(A) = Bel(A) = Pl(A)$, for all $A \in X$. Thus, new entropy satisfies the probability consistency property.

The following example shows the properties of proposed entropy and how it is better at capturing uncertainties compared to Shannon and Deng entropy.

Example 6.5: Given a frame of discernment Θ = {a, b, c}, for a mass function m(a) = m(b) = m(c) = 1/3.

$E_{Sh} = -(\frac{1}{3}log_2\frac{1}{3} + \frac{1}{3}log_2\frac{1}{3} + \frac{1}{3}log_2\frac{1}{3}) = 1.585$

$E_{Deng} = (\frac{1}{3}log_2\frac{1/3}{2^1-1} + \frac{1}{3}log_2\frac{1/3}{2^1-1} + \frac{1}{3}log_2\frac{1/3}{2^1-1} = 1.585$

$E_P = -(\frac{(1/3+1/3)}{2}log_2\frac{(1/3+1/3)}{2.(2^1-1)}.\exp^{(\frac{1-1}{3})} + \frac{(1/3+1/3)}{2}log_2\frac{(1/3+1/3)}{2.(2^1-1)}.\exp^{(\frac{1-1}{3})} +$

$\frac{(1/3+1/3)}{2}log_2\frac{(1/3+1/3)}{2.(2^1-1)}.\exp^{(\frac{1-1}{3})}) = 1.585$

This showed that the result of proposed entropy is identical to Shannon entropy and Deng entropy when the belief is only assigned on single elements (or Bayesian).

Example 6.6: Given a frame of discernment Θ = {a, b, c}, mass function m(a) = m(b) = m(c) = m(a, b) = m(a, c) = m(b, c) = m(a, b, c) = 1/7.

Table 6.2. Bel and Pl values for example 6.6

	a	**b**	**c**	**a,b**	**a,c**	**b,c**	**a,b,c**
Bel(.)	1/7	1/7	1/7	3/7	3/7	3/7	1.0
Pl(.)	4/7	4/7	4/7	6/7	6/7	6/7	1.0

Shannon and Deng entropy is calculated to compare with the values from proposed entropy.

$E_{Sh} = -(\frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7}) = 2.8074$

$E_{Deng} = -(\frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{7} + \frac{1}{7}log_2\frac{1}{3*7} + \frac{1}{7}log_2\frac{1}{3*7} + \frac{1}{7}log_2\frac{1}{3*7} + \frac{1}{7}log_2\frac{1}{7*7}) = 3.887$

$E_P = -(\frac{5}{2*7}log_2\frac{5}{2*7} + \frac{5}{2*7}log_2\frac{5}{2*7} + \frac{5}{2*7}log_2\frac{5}{2*7} + \frac{9}{2*7}log_2(\frac{9}{2*3*7}.\exp^{(1/3)}) +$

$\frac{9}{2*7}log_2(\frac{9}{2*3*7}.\exp^{(1/3)}) + \frac{9}{2*7}log_2(\frac{9}{2*3*7}.\exp^{(1/3)}) + log_2(\frac{1}{7}.\exp^{(2/3)})) = 6.79$

Shannon entropy only considers mass function value and has the lowest entropy. Deng entropy considers both mass function value and cardinality on focal elements. It calculates higher entropy than Shannon. Proposed entropy considers mass function value (central value of probability interval), cardinality of both focal elements and FOD. It results into highest entropy value compared to Shannon and Deng. If a FOD consists of 7 elements compared to say 3 elements, intuitively it can be said, 7 elements FOD should have higher entropy because it is less compact. Also, because proposed entropy considers central value of probability interval $\frac{(Bel+Pl)}{2}$, it is capturing more uncertainty compared to only mass function. As a result, the proposed entropy function abides by the DS framework and superior in capturing uncertainty compared to Shannon and Deng entropy.

6.5 Proposed Steps to Eliminate Paradoxes in Space Domain

With increasing use of sensors application in real-time decision making, we need algorithm which can fuse sensor outputs both in space-domain and time-domain. The goal of the proposed method is to eliminate the paradoxes of original DS combination rule and work as a decision level sensor fusion algorithm in both space and time domain. We are adopting 'revision of original evidence before combination' because we don't want to lose the associative and commutative properties of the original DS rule. The proposed method is a distance-based method. It calculates the relative distances between the sensor evidences (classification output). Then based on average distance, it classifies which sensor output is credible and which sensor output is incredible. Then it penalizes the incredible sensor output using the novel entropy function so that incredible sensor has less effect on fused output. It also rewards the credible sensor input so that credible sensor carries more weight towards fused output. At the end, modified evidence is fused using original DS sensor fusion equation. Following example is used to showcase the steps and compare the final fused results with works from open literature.

Example 6.7: In a multi-sensor target recognition system, assume there are totally three types of targets to be recognized = $\{A, B, C\}$. Suppose there are five sensors. They could be any types of sensors. After data acquisition at a specific moment by five sensors, data are processed and classification IDs are generated. Generated IDs from five sensors are listed as BPAs:

Sensor 1: $m_1 : m_1(A) = 0.41, m_1(B) = 0.29, m_1(C) = 0.30$

Sensor 2: $m_2 : m_2(A) = 0.00, m_2(B) = 0.90, m_2(C) = 0.10$

Sensor 3: $m_3 : m_3(A) = 0.58, m_3(B) = 0.07, m_3(A, C) = 0.35$

Sensor 4: $m_4 : m_4(A) = 0.55, m_4(B) = 0.10, m_4(A, C) = 0.35$

Sensor 5: $m_5 : m_5(A) = 0.60, m_5(B) = 0.10, m_5(A, C) = 0.30$

This is a classic example of 'one-ballot veto' paradox.

Step 1: Build a multi-sensor information matrix. Assume for a multi-sensor system, there are N evidences (sensors) in the frame $\Theta = \{H_1, H_2, ..., H_M\}$ (objects to be detected).

$$\begin{pmatrix} m_1(H_1) & m_1(H_2) & \dots & m_1(H_M) \\ m_2(H_1) & m_2(H_2) & \dots & m_2(H_M) \\ \vdots & \vdots & \ddots & \vdots \\ m_N(H_1) & m_N(H_2) & \dots & m_N(H_M) \end{pmatrix} = \begin{pmatrix} .41 & .29 & .3 & 0 \\ .0 & .9 & .1 & 0 \\ .58 & .07 & 0 & .35 \\ .55 & .1 & 0 & .35 \\ .6 & .1 & 0 & .3 \end{pmatrix} \tag{6.12}$$

Step 2: Measure the relative distance between evidences. Several distance function can be used to measure the relative distance. They all have their own advantages and disadvantages regarding runtime and accuracy. We have used Jousselme's distance [138] function. Jousselme's distance function uses cardinality in measuring distance which is an important metric when multiple elements are present in one BPA under DS framework. Effect of different distance functions (Euclidean, Jousselme, Minkowsky, Manhattan and Camberra distance function) on simulation time and information fusion can be found in literature [139]. Assuming that there are two mass functions indicated by m_{i} and m_{j} on the discriminant frame Θ, the Jousselme distance between m_{i} and m_{j} is defined as:

$$DM(m_{\mathrm{i}}, m_{\mathrm{j}}) \;=\; \sqrt{\frac{1}{2}.(m_{\mathrm{i}} - m_{\mathrm{j}}).D.(m_{\mathrm{i}} - m_{\mathrm{j}})^T} \tag{6.13}$$

Where $D = \frac{|A \cap B|}{|A \cup B|}$, and $|.|$ represents cardinality.

Step 3: Calculate sum of evidence distance for each sensor.

$$d_{\mathrm{i}} \;=\; \sum_{\mathrm{j}=1 \;\&\; \mathrm{j}\neq\mathrm{i}}^{N} DM(m_{\mathrm{i}}, m_{\mathrm{j}}) = \begin{pmatrix} 1.5449 \\ 2.9284 \\ 1.2311 \\ 1.1761 \\ 1.1944 \end{pmatrix} \tag{6.14}$$

Step 4: Calculate global average of evidence distance.

$$\bar{d} \;=\; \frac{\sum_{\mathrm{i}=1}^{N} d_{\mathrm{i}}}{N} \;=\; 1.615 \tag{6.15}$$

Step 5: Calculate belief entropy for each sensor using (6.11) and normalize.

Table 6.3. Bel and Pl values for Example 6.7

	m(A)	**m(B)**	**m(C)**	**m(A,C)**
m_1	Bel=0.42, Pl=0.41	Bel=0.29,Pl=0.29	Bel=0.3,Pl=0.3	Bel=0,Pl=0
m_2	Bel=0, Pl=0	Bel=0.9,Pl=0.9	Bel=0.1,Pl=0.1	Bel=0,Pl=0
m_3	Bel=0.93, Pl=0.93	Bel=0.07,Pl=0.07	Bel=0,Pl=0.35	Bel=0.93,Pl=0.93
m_4	Bel=0.9, Pl=0.9	Bel=0.1,Pl=0.1	Bel=0,Pl=0.35	Bel=0.9,Pl=0.9
m_5	Bel=0.9, Pl=0.9	Bel=0.1,Pl=0.1	Bel=0,Pl=0.3	Bel=0.9,Pl=0.9

It is interesting to note that, although m_3, m_4, m_5 has zero m(C) values, it has non-zero Pl values. As a result, it will consider non-zero Pl values of m(C) when calculating entropy. $E_P(m_1) = 1.5664,\ E_P(m_2) = 0.469,\ E_P(m_3) = 1.3861,\ E_P(m_4) = 1.513,\ E_P(m_5) = 1.483.$ Normalize the entropy:

$$\overline{E_P(m_{\mathrm{i}})} \;=\; \frac{E_p(m_{\mathrm{i}})}{\sum E_p(m_{\mathrm{i}})} \tag{6.16}$$

Step 6: The evidence set is divided into two parts: the credible evidence and the incredible evidence. From (14) and (15):

$$\begin{gathered} If\ d_{\mathrm{i}} \leq \bar{d}, m_{\mathrm{i}}\ \mathrm{is}\ credible\ evidence \\ If\ d_{\mathrm{i}} > \bar{d}, m_{\mathrm{i}}\ \mathrm{is}\ incredible\ evidence \end{gathered} \tag{6.17}$$

The intuition is that, if an evidence has higher distance than average distance (which is calculated using all the evidences) then probably that evidence is faulty and should be penalized (incredible evidence). If an evidence distance is lower than average, then that evidence is in harmony with other evidence and should be rewarded (credible evidence). Lower entropy means lower uncertainty and that evidence should be rewarded more for

credible evidence. Opposite is true for incredible evidence. So, we needed a function which has large slope as it goes near to zero. Natural log function fits the bill. As a result, following Reward and Penalty function is proposed:

$$\begin{aligned} For\ credible\ evidence,\ Reward\ function &= -ln(\overline{E_P(m)}) \\ For\ incredible\ evidence,\ Penalty\ function &= -ln(1-\overline{E_P(m)}) \end{aligned} \tag{6.18}$$

Using (6.16), (6.17) and (6.18) calculate reward and penalty value for each evidence.
$Reward_1 = 1.4103$, $Penalty_2 = 0.0759$, $Reward_3 = 1.5326$, $Reward_4 = 1.445$, $Reward_5 = 1.4647$.
Normalize Reward and Penalty values to get evidence weights.
$w_1 = 0.2379$, $w_2 = 0.0128$, $w_3 = 0.2585$, $w_4 = 0.2437$, $w_5 = 0.2471$. Obviously, we can observe that there is a high conflict between the evidence m_2 and other evidences. So m_2 is defined as an incredible evidence, and has very low weight. Other evidences are supported by each other, so their weights are higher than m_2.

Step 7: Modify the original evidences.

$$m(A) = \sum_{\mathrm{i}=1}^{N} m_{\mathrm{i}}(A).w_{\mathrm{i}} \tag{6.19}$$

Resulting modified evidence, m(A) = 0.5298, m(B) = 0.1477, m(C) = 0.0726, m(A,C) = 0.2499.

Step 8: : Combine modified evidence for (n-1) times (for this example 4 times) with DS combination rule using (6.5) and (6.6). How to apply the fusion rule is important. For this example, if evidence m_1 and m_2 is fused with modified evidence, then, $m_{12}(A) = 0.8125$, $m_{12}(B) = 0.0325$, $m_{12}(C) = 0.062$, $m_{12}(A,C) = 0.093$. Now to get m_{123}, if m_{12} values are fused with m_{12} values using (6.5) and (6.6), that would be wrong. To get m_{123}, m_{12} values should be fused with the original modified evidence from step 7. It is also evident that, for single elements, if that element has higher value after step 7, it will have highest value after fusing (n-1) times. The higher the value after step 7, the higher the value after fusion.

Table 6.4 compares the results of the proposed algorithm with other combination methods from open literature for Example 6.7.

Table 6.4. Evidence combination results based on different combination methods for Example 6.7

Methods	m_{12}	m_{123}	m_{1234}	m_{12345}
Dempster [118]	m(A) = 0, m(B) = 0.8969, m(C) = 0.1031	m(A) = 0, m(B) = 0.8969, m(C) = 0.1031	m(A) = 0, m(B) = 0.8969, m(C) = 0.1031	**m(A) = 0**, m(B) = 0.8969, m(C) = 0.1031
Murphy [130]	m(A) = 0.0964, m(B) = 0.8119, m(C) = 0.0917, m(AC) = 0	m(A) = 0.4619, m(B) = 0.4497, m(C) = 0.0794, m(AC) = 0.0090	m(A) = 0.8362, m(B) = 0.1147, m(C) = 0.0410, m(AC) = 0.0081	**m(A) = 0.9620**, m(B) = 0.0210, m(C) = 0.0138, m(AC) = 0.0032
Deng [140]	m(A) = 0.0964, m(B) = 0.8119, m(C) = 0.0917, m(AC) = 0	m(A) = 0.4974, m(B) = 0.4054, m(C) = 0.0888, m(AC) = 0.0084	m(A) = 0.9089, m(B) = 0.0444, m(C) = 0.0379, m(AC) = 0.0089	**m(A) = 0.9820**, m(B) = 0.0039, m(C) = 0.0107, m(AC) = 0.0034
Han [141]	m(A) = 0.0964, m(B) = 0.8119, m(C) = 0.0917, m(AC) = 0	m(A) = 0.5188, m(B) = 0.3802, m(C) = 0.0926, m(AC) = 0.0084	m(A) = 0.9246, m(B) = 0.0300, m(C) = 0.0362, m(AC) = 0.0092	**m(A) = 0.9844**, m(B) = 0.0023, m(C) = 0.0099, m(AC) = 0.0034
Wang[142] recalculated	m(A) = 0.0964, m(B) = 0.8119, m(C) = 0.0917, m(AC) = 0	m(A) = 0.6495, m(B) = 0.2367, m(C) = 0.1065, m(AC) = 0.0079	m(A) = 0.9577, m(B) = 0.0129, m(C) = 0.0200, m(AC) = 0.0094	**m(A) = 0.9867**, m(B) = 0.0008, m(C) = 0.0087, m(AC) = 0.0035
Jiang [129]	m(A) = 0.0964, m(B) = 0.8119, m(C) = 0.0917, m(AC) = 0	m(A) = 0.7614, m(B) = 0.1295, m(C) = 0.0961, m(AC) = 0.0130	m(A) = 0.9379, m(B) = 0.0173, m(C) = 0.0361, m(AC) = 0.0087	**m(A) = 0.9837**, m(B) = 0.0021, m(C) = 0.0110, m(AC) = 0.0032
Proposed	m(A) = 0.00573, m(B) = 0.96906, m(C) = 0.02522, m(AC) = 0	m(A) = 0.7207, m(B) = 0.1541, m(C) = 0.1178, m(AC) = 0.007	m(A) = 0.9638, m(B) = 0.0019, m(C) = 0.0224, m(AC) = 0.0117	**m(A) = 0.9877**, m(B) = 0.0002, m(C) = 0.0087, m(AC) = 0.0034

As seen from Table 6.4, when evidences are in high conflict, classical Dempster's combination rule produces counterintuitive results that is not correct. With increase in number of sensors, Murphy's simple averaging, Deng's weighted averaging, and Han's novel weight averaging, Wang's weighted evidence, Jiang's uncertainty measure all give reasonable results, although their final combination results are slightly inferior to the outcomes of our proposed approach. Wang et al. [142] showed in his paper that the modified evidences before the fusion steps are $m(A) = 0.5048, m(B) = 0.184, m(C) = 0.068, m(AC) = 0.243$. Now the

modified evidence for m(A) is lower than our proposed method as stated in step 7. Also as explained in step 8, it is unlikely that after fusing these evidences (n - 1) times (4 for this example) using original DS combination rule, the fused $m_{12345}(A)$ will be higher than our proposed method. Using the evidences presented in Wang's work, the recalculated fused evidences are presented in Table 6.4. The proposed method also has the highest convergence rate (rate of m(A) value goes towards 1) after sensor 3. It is reasonable to say that, proposed method overcomes the paradoxes of classical DS rule and produces competitive fusion result compared to other combination rules available in open literature. Fig. 6.1 shows how fused evidence of m(A) changes with addition of new sensors and compares multiple methods from the literature. As m(A) is the correct evidence, how it is changing with inclusion of new sensor evidence is important for justification of fused result. The proposed method penalizes m(A) when only two sensors are used. As a result, m(A) starts with lower evidence for the proposed method compared to other methods (number of sensors = 2). But with inclusion of correct evidences from sensor 3 and 4, m(A) converges towards 1 quickly for the proposed method, compared to other methods. As m(A) evidence converges towards 1, convergence rate becomes slow for all the methods. Zoomed in view shows that the proposed method has higher m(A) evidence after fusing 5 sensor evidences, compared to other methods from literature.

It can be seen from Example 6.7 that, this method is applicable for any multi-sensor system because fusion occurs after classification ID is created from sensor output. As an example, let's assume multiple cameras are used for object classification. Camera output (video/image) will go through a classifier (example: neural-network) for object ID classification. After classification, output may have similar syntax to Example 6.7. Then the proposed method can be applied to find out which sensor is providing erroneous data and fuse them accordingly.

6.6 Proposed Steps for Time-domain Data Fusion

The same eight-step algorithm is slightly modified to fit time-domain sensor fusion paradigm. It calculates the relative distances between the sensor data at each time step (classification output). Then based on average distance, it classifies which time-step output

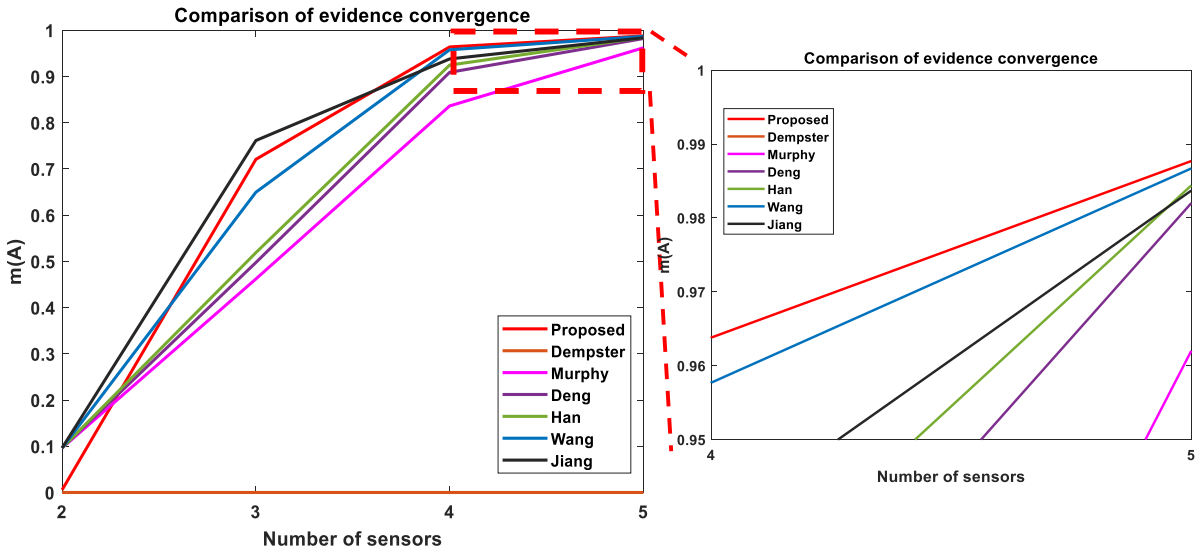


Figure 6.1. Comparison of convergence of evidence m(A) for Example 6.7.

is credible and which time-step output is incredible. Then it penalizes the incredible time-step output using the entropy function so that incredible time-step has less effect on fused output. It also rewards the credible time-step input so that credible time-step carries more weight towards fused output. Credible time step is the time-step which contains credible or true data. Incredible time step contains untrue or unreliable data. At the end, modified evidence is fused using original DS sensor fusion equation. The proposed algorithm can also be applied to space-domain sensor fusion with some minor modification [143]. As an example, each sensor could be a camera. From each camera, multiple object classification output can be obtained when video feed goes through a neural-network type classifier. Classification ID's from multiple cameras can be fused together using the proposed algorithm. Due to faulty sensors or obstructed view, neural-network output from cameras can generate wrong classification ID. In space-domain, the proposed algorithm finds out which sensors are generating wrong classification ID and penalizes the wrong classification ID by assigning less weight to that sensor output.

The proposed decision level sensor fusion algorithm has multiple advantages:

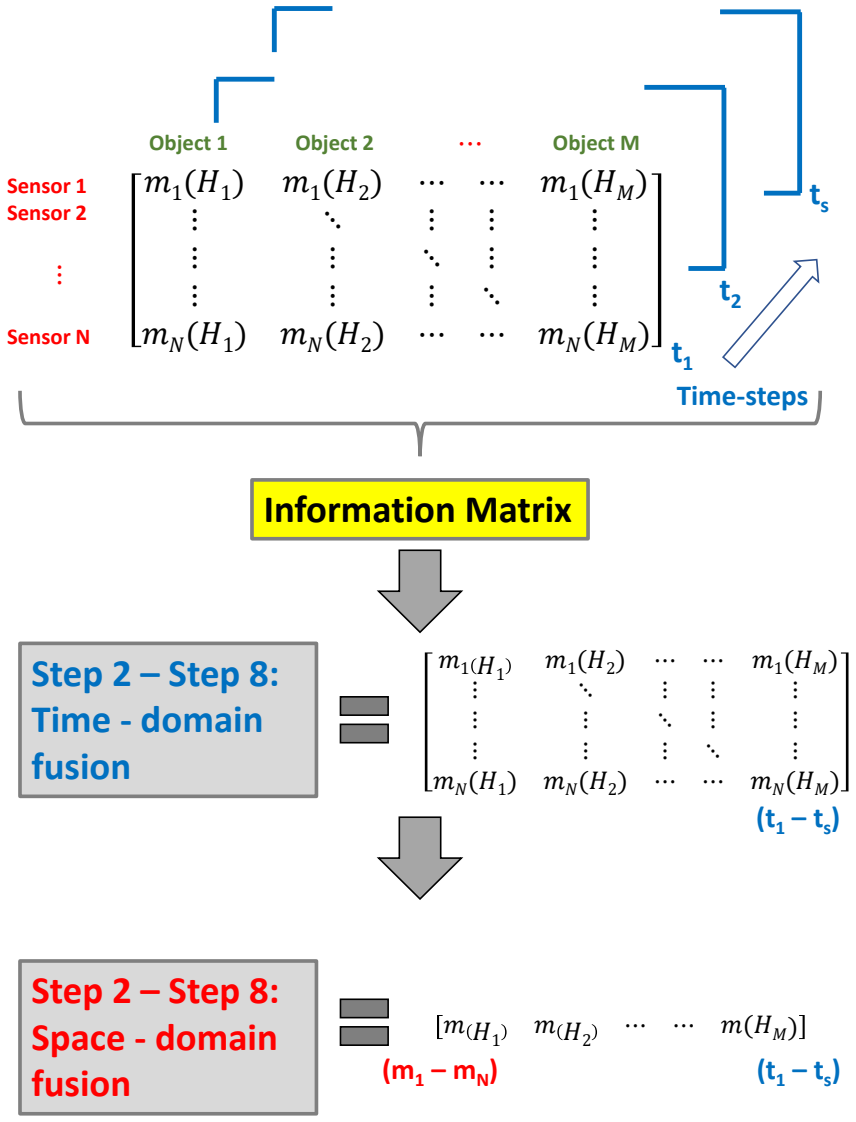


Figure 6.2. Simple representation of sensor-fusion in space and time domain.

- Improves the limitations of the original DS combination rule.
- Combines results from both homo-genius and hetero-genius sensors at decision level.
- Can detect faulty sensors in a system and compensate so that faulty sensor doesn't affect fused results.

Simplified format of this process with resulting output is presented in Fig. 6.2. First, a 3-dimensional information matrix is created which contains all the sensor evidences from time step $t_1 - t_s$. At time-domain fusion, information from all the selected time-steps are fused and output is a 2-dimensional matrix. At space-domain fusion, information from all the sensors are fused and output is a vector which contains classification evidence (% output) for each object. Following steps show the modified algorithm for time-domain fusion:

Step 1: Build a multi-time-steps information matrix. Assume there are N time-steps in the frame $\Theta = \{H_1, H_2, ..., H_M\}$.

$$\begin{pmatrix} m_1(H_1) & m_1(H_2) & \dots & m_1(H_M) \\ m_2(H_1) & m_2(H_2) & \dots & m_2(H_M) \\ \vdots & \vdots & \ddots & \vdots \\ m_N(H_1) & m_N(H_2) & \dots & m_N(H_M) \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_s \end{pmatrix} \tag{6.20}$$

Step 2: Measure the relative distance between each time-step data. Assuming that there are two mass functions indicated by m_i and m_j on the discriminant frame Θ, the Jousselme distance between m_i and m_j is defined as:

$$DM(m_\mathrm{i}, m_\mathrm{j}) \;=\; \sqrt{\frac{1}{2} \cdot (m_\mathrm{i} - m_\mathrm{j}) \cdot D \cdot (m_\mathrm{i} - m_\mathrm{j})^T} \tag{6.21}$$

where $D = \frac{|A \cap B|}{|A \cup B|}$, and $|.|$ represents cardinality.

Step 3: Calculate sum of distance for each time-step.

$$d_\mathrm{i} \;=\; \sum_{\mathrm{j}=1 \;\&\; \mathrm{j} \neq \mathrm{i}}^{N} DM(m_\mathrm{i}, m_\mathrm{j}) \tag{6.22}$$

Step 4: Calculate global average of distance for all time-steps considered.

$$\bar{d} \;=\; \frac{\sum_{\mathrm{i}=1}^{N} d_{\mathrm{i}}}{N} \tag{6.23}$$

Step 5: Calculate belief entropy for each time-step using (6.11) and normalize:

$$\overline{E_P(m_{\mathrm{i}})} \;=\; \frac{E_p(m_{\mathrm{i}})}{\sum E_p(m_{\mathrm{i}})} \tag{6.24}$$

Step 6: The time-step data set is divided into two parts: the credible time-step and the incredible time-step.

$$\begin{gathered} If\ d_{\mathrm{i}} \leq \bar{d}, m_{\mathrm{i}}\ \mathrm{is}\ credible\ time\ step \\ If\ d_{\mathrm{i}} > \bar{d}, m_{\mathrm{i}}\ \mathrm{is}\ incredible\ time\ step \end{gathered} \tag{6.25}$$

The intuition is that, if data at a specific time-step has higher distance than average distance then probably that time-step is faulty and should be penalized (incredible time-step). If data at a specific time-step is lower than average, then that time-step is in harmony with other time-steps and should be rewarded (credible time-step). As a result, following Reward and Penalty function is proposed:

$$\begin{gathered} For\ credible\ time\ step,\ Reward\ function\ = -ln(\overline{E_P(m)}) \\ For\ incredible\ time\ step,\ Penalty\ function\ = -ln(1-\overline{E_P(m)}) \end{gathered} \tag{6.26}$$

Reward and penalty values are normalised to get the weight of each time-step.

$$\overline{w_{\mathrm{i}}} \;=\; \frac{Reward\ or\ Penalty}{\sum Reward\ or\ Penalty} \tag{6.27}$$

Step 7: Modify the original data of each time-step.

$$m(A) \;=\; \sum_{\mathrm{i}=1}^{s} m_{\mathrm{i}}(A).w_{\mathrm{i}} \tag{6.28}$$

Step 8: : Combine modified data of s time-steps for $(s-1)$ times with DS combination rule using (6.5) and (6.6).

6.6.1 Anti-disturbing Ability and Transition Property of Proposed Algorithm

The goal of the time domain fusion is to deal with the conflict between time domain data. Time domain fusion works as a damper for sudden changes. It also improves accuracy if evidences are credible. Following example is used to compare results from the literature.

Example 6.8: Assuming that the discriminant frame of a mid-course ballistic target integrated identification system is $\Theta = \{A, B, C\}$. Multiple sensors have identified the targeted category at five consecutive moments, as shown in Table 6.5.

Table 6.5. Input data of each time step for Example 6.8

time-steps	**m(A)**	**m(B)**	**m(C)**
$T_1 = 0$	0.6	0.1	0.3
$T_2 = 1$	0.65	0.15	0.2
$T_3 = 2$	0.6	0.2	0.2
$T_4 = 3$	0	0.85	0.15
$T_5 = 4$	0.55	0.2	0.25

From Table 6.6, it is evident that when the sensors provide the undisturbed data at time step $T_1 - T_3$, Dempster's rule, Song-1 method, Song-2 method, Chengkun's method and the proposed method can make a correct decision at any moment. When the sensors are disturbed at time T_4, Dempster's rule has fallen into the trouble of 'one-ballot veto' paradox and failed to correctly identify $m(A)$. Song-2 method shows slightly better performance against the adversarial information at T_4 compared to Song-1 but fails to correctly identify $m(A)$. Chenkung's method shows better robustness against change compared to Song's method.

Table 6.6. Data combination results based on different combination methods for Example 6.8

Methods	$T_2 = 1$	$T_3 = 2$	$T_4 = 3$	$T_5 = 4$
Dempster [51]	m(A) = 0.838, m(B) = 0.032, m(C) = 0.129	m(A) = 0.939, m(B) = 0.012, m(C) = 0.048	m(A) = 0, m(B) = 0.586, m(C) = 0.413	m(A) = 0, m(B) = 0.531, m(C) = 0.468
Song-1 [60]	m(A) = 0.767, m(B) = 0.076, m(C) = 0.155	m(A) = 0.797, m(B) = 0.091, m(C) = 0.111	m(A) = 0.0, m(B) = 0.843, m(C) = 0.157	m(A) = 0.317, m(B) = 0.458, m(C) = 0.224
Song-2 [59]	m(A) = 0.665, m(B) = 0.077, m(C) = 0.182, m(ϕ) = 0.075	m(A) = 0.664, m(B) = 0.089, m(C) = 0.137, m(ϕ) = 0.109	m(A) = 0.246, m(B) = 0.471, m(C) = 0.135, m(ϕ) = 0.146	m(A) = 0.503, m(B) = 0.27, m(C) = 0.194, m(ϕ) = 0.032
Chengkun [61]	m(A) = 0.746, m(B) = 0.09, m(C) = 0.163	m(A) = 0.771, m(B) = 0.106, m(C) = 0.123	m(A) = 0.679, m(B) = 0.191, m(C) = 0.128	m(A) = 0.708, m(B) = 0.138, m(C) = 0.153
Proposed	m(A) = 0.833, m(B) = 0.033, m(C) = 0.133	m(A) = 0.943, m(B) = 0.017, m(C) = 0.039	m(A) = 0.971, m(B) = 0.007, m(C) = 0.022	m(A) = 0.987, m(B) = 0.002, m(C) = 0.01

From Fig. 6.3, it is obvious that at time step $T_4 = 3$, the fluctuation of $m(A)$ in the proposed method is non-existent. All the other methods show some extent of fluctuation towards lower $m(A)$ data. But the proposed method penalizes that time-step data and keeps improving $m(A)$. This shows that the proposed method can effectively handle the conflicting information of time-domain data and has stronger anti-disturbing ability.

Proposed algorithm puts higher weights on time-steps when data agrees with one another. Also if at T_4, a small value of m(A) was used instead of zero (say, m(A) = 0.05), the proposed algorithm would still produce superior fusion results. At T_4, m(A) value for original DS fusion rule wouldn't go to zero but still would be a very small number. Chengkun, Song-1 and Song-

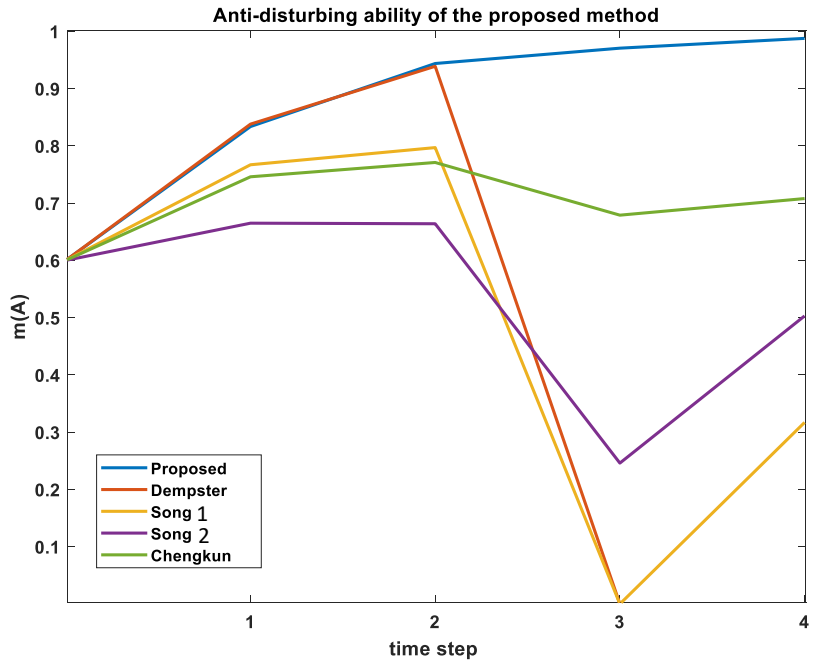


Figure 6.3. Comparison of anti-disturbing ability of several combination rules for Example 6.8.

2 would produce slightly better fused result with less deviation (or dip) compared to current result but would still contain downward deviation for fused m(A). On the other hand, the proposed method would use the small value of m(A) as a positive reinforcement and would produce higher fused m(A) value compared to the results from Fig. 6.3. When time-step data are transitioning from one BPA to another in time domain, it would be interesting to see how the proposed algorithm cope with the new time-steps which have higher evidence on a different BPA. Also, how quickly the algorithm can response along the transition. Example 6.9 shows the transition property of our algorithm.

Example 6.9: Assuming that the discriminant frame of a mid-course ballistic target integrated identification system is $\Theta = \{A, B, C\}$. Multiple sensors have identified the targeted category at five consecutive moments. As shown in Table 6.7. The target has changed from A to B after time step T_2.

Table 6.7. Input data of each time step for Example 6.9

time-steps	m(A)	m(B)	m(C)
$T_1 = 0$	0.6	0.1	0.3
$T_2 = 1$	0.65	0.15	0.2
$T_3 = 2$	0.2	0.6	0.2
$T_4 = 3$	0.1	0.8	0.1
$T_5 = 4$	0.15	0.75	0.1

At time domain fusion, another important goal is to make the system robust so that time-step data can transition quickly between one another. As seen from Fig. 6.4, proposed method shows reasonable results for transition between time-step data. Time-step data of m(B) stars to rise after T_2. But fused m(B) stars to rise after T_4. Based on input evidences, it can be said that the proposed method takes 2 time steps for time-step data transition which is quite robust for real time object classification application. As an example, let's say, camera can process video at 30 frames per second (FPS). Each time-step for this case is, time needed to process each frame. Because new time-step data are gathered with each frame. For 30 FPS, each time-step = (1/30) = 0.03 sec. If 2 time-steps are needed for proper transition from one fused time-step data to another, it will take (2*0.03) = 0.06 sec, which is quite robust for real-time application.

6.6.2 Modification of BPA for CNN Based Object Classification under DS Framework

The goal is to understand the uncertainties related to CNN classifier and include that within our DS framework. Classification report of model-2 from chapter 4 is presented in Table 6.8. Fig. 6.5 shows how the CNN classifier performs when a Pigweed plant and a Ragweed plant placed separately. This is same as Fig. 4.9. As we have seen, precision and recall are good measures of how well the classifier works at weed classification. The intuition is, the classifier is never 100% certain about any classification even if it shows 100% classification output for any object, because the recall and precision value is not 1.0. Say

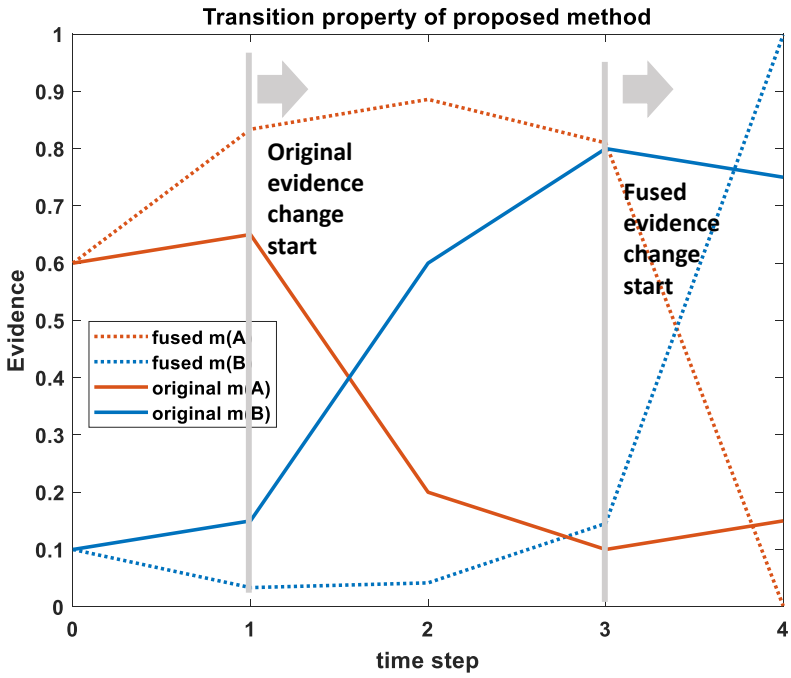


Figure 6.4. Transition property of the proposed algorithm for Example 6.9.

for an image, the classifier outputs 100% that it is Ragweed. But among those 100%, only 96% are possibly relevant (Ragweed has 0.96 precision). And among those 96%, only 94% is correctly classified (Ragweed has 0.94 recall). We can include these uncertainties into our BPA using the following equations:

$$m(H_{\mathrm{i}})_{updated} \;=\; Precision_{\mathrm{i}} * Recall_{\mathrm{i}} * m(H_{\mathrm{i}}) \tag{6.29}$$

$$m(\Theta) \;=\; 1 - \sum_{\mathrm{i}=1}^{n} m(H_{\mathrm{i}})_{updated} \tag{6.30}$$

where $\Theta = \{Ragweed, Pigweed, Cocklebur\}$, is the universal set ($\Theta$ in Fig. 6.6) which contains evidence for all three weeds.

Table 6.8. Classification report of CNN classifier (Model-2).

	Cocklebur	**Pigweed**	**Ragweed**
Precision	0.94	0.94	0.96
Recall	1.0	0.89	0.94
Training accuracy	0.99	0.99	0.99

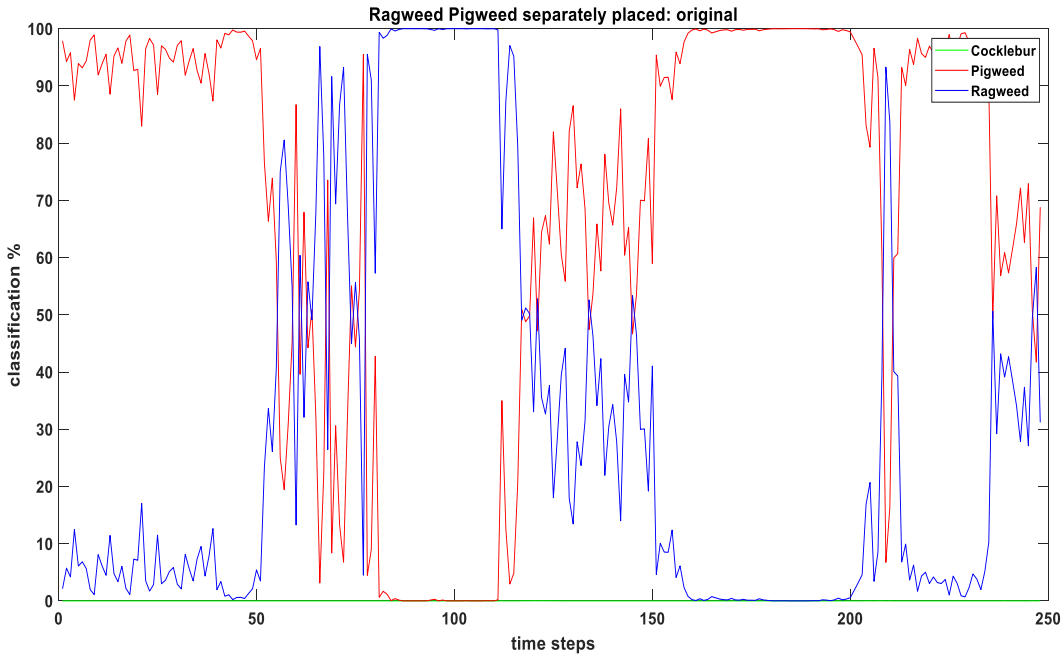


Figure 6.5. Real-time weed classification from video input using CNN classifier (Model-2). Classification % is showing CNN output of video feed at each time step. This CNN output is used as BPA in fusion algorithm.

Fig. 6.6 shows how incorporating precision and recall into BPA affects real-time weed classification using CNN classifier. Top figure shows the classification results before fusion. As expected, around 20% evidence is placed on Θ (universal set) for all time steps, which contains evidence for all three weeds. Also, classification accuracy for Ragweed or Pigweed never reaches 100% because remaining evidences are placed in Θ. Effectively, the summation of uncertainties related to CNN classifier is placed into Θ. Which is another way of saying that the classifier is not sure about the exact type of weed, so that percentage is placed into

Θ because it contains possibility of all three weeds. Bottom figure shows the time-domain sensor fusion (time-step, $t_s = 5$ used) after evidence update. With updated evidence, time-domain fusion is still successful in filtering noise (reduce sudden changes – smoother curves) from weed classification output and transition from one weed to another. One interesting thing is, Θ value goes to zero which seems counter-intuitive. But Θ is a set which contains evidence for all three weeds under closed world assumption. During each fusion step (we are using $t_s = 5$, we have 4 time-domain fusion steps), evidences from the universal set (Θ) is distributed among all three weed evidences. As a result, Θ value goes down with each fusion step.

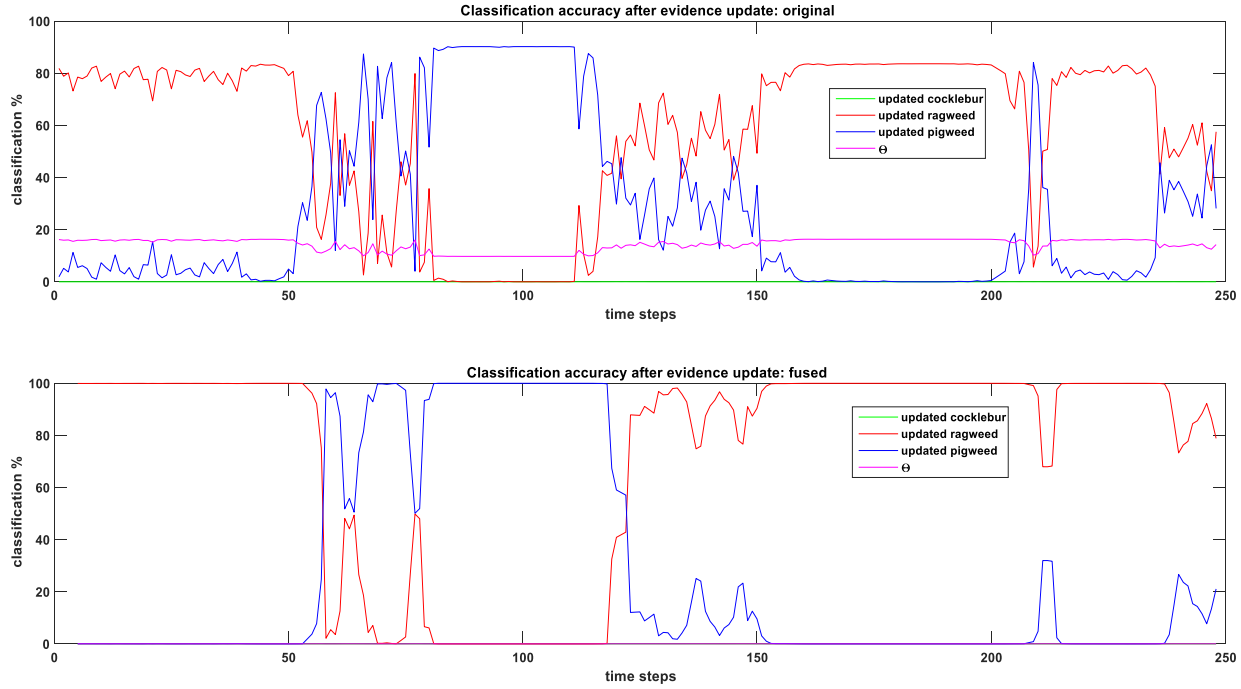


Figure 6.6. Effect of considering precision and recall on updating BPA on real-time weed classification. Classification % are showing BPA from equations (6.29) and (6.30) [top figure]. Time-domain fusion of updated BPA for $t_s = 5$. Classification % showing fused results when BPA from equations (6.29) and (6.30) goes through the proposed fusion algorithm [bottom figure].

6.6.3 Effect of Number of Time-steps (fusion-time) on Fused Output

Number of time-steps considered for time-domain fusion has direct impact on fused output because with increased number of time-steps, higher volume of data is considered for fusion. Fig. 6.7 shows the effect of fusion-time on time-domain sensor fusion. In this figure, fusion-time, $t_s = 3$ means we consider time steps (t_1, t_2, t_3) for time-domain sensor fusion. At the next time step t_4, we discard t_1 and consider (t_2, t_3, t_4) for time-domain fusion and the classification output is the fused result of this three time-steps. And this goes on until we reach the end. In this same manner, $t_s = 5$ means we consider five time steps $(t_1, t_2, t_3, t_4, t_5)$ for time-domain sensor fusion. From Fig. 6.7 it can be seen that, fused results for all the time steps are successful in filtering noise (reduce sudden changes – smoother curves) from weed classification output and transition from one weed to another. In other words, proposed algorithm captures the weed classification dynamics from video feed. Lower t_s ($t_s = 3\,and\,5$) results are more responsive to changes compared to higher t_s ($t_s = 10\,and\,15$) results. Fused weed classification outputs are basically weighted average of the classification values of the fusion-time (t_s) considered. As a result, if we consider high t_s (say $t_s = 15$), fused output wouldn't change much when at each step, only 1 set of new data is added to a set already containing 15 sets of classification data. This shows that, t_s is a tuning parameter for time-domain fusion and this can be tuned based on better response (lower t_s) or better robustness (higher t_s).

6.7 Application of the Proposed Algorithm

Using the proposed sensor fusion algorithm, we will try to analyze and improve three limitations of single-camera based CNN classifications system:

- When faulty sensor is present.
- Unstable and possibly wrong classification ID from video feed.
- Unstable and possibly wrong classification due to partial occlusion of weed from camera view.

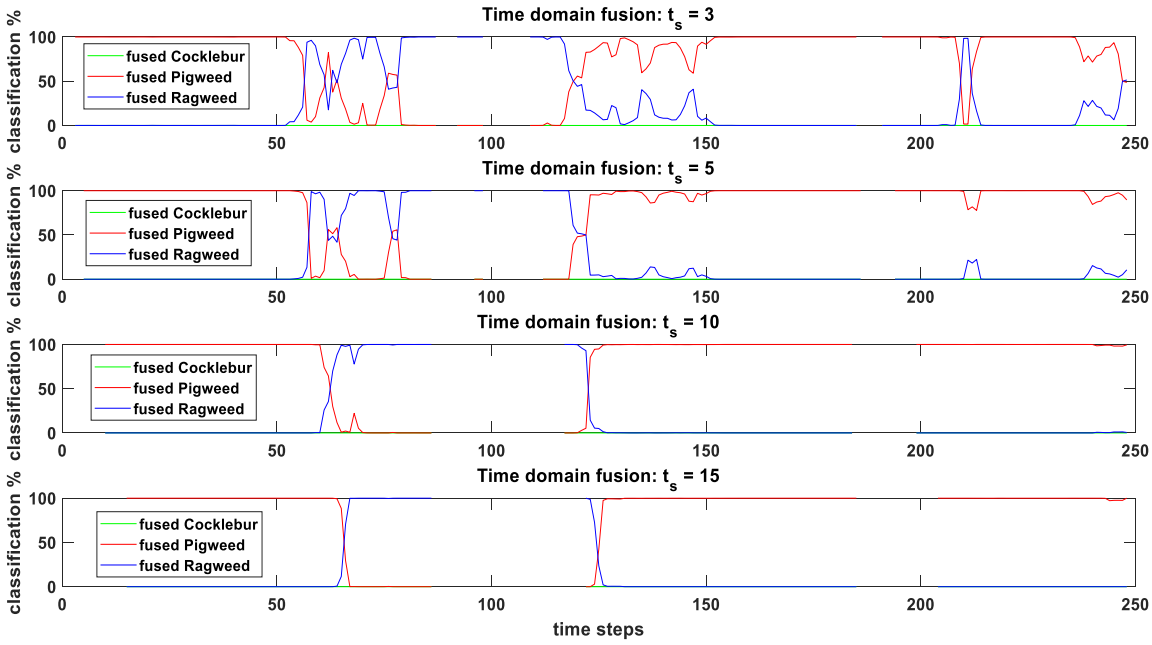


Figure 6.7. Effect of fusion-time (t_s) during time-domain sensor fusion on real-time weed classification from video input. Classification % showing fused results when BPA from Fig. 6.5 goes through the proposed time-domain fusion algorithm (step 1 - step 8). from section 6.6

6.7.1 Fusion when Faulty Sensor is Present in Sensor-array

In a perfect world, the moment a CNN classifier sees an object, it should classify that object with 100% classification accuracy. But in real world, the classification accuracy varies a lot due to noise, lighting, vibration, occlusion etc. A steady output is needed from object classification system to spray system to spray herbicides accurately. A multi-sensor fusion architecture with capability to fuse evidence in time and space domain is proposed (Fig. 6.9) to create a steady classification output. In time domain, a certain number of time-steps ($t_s = 10$ for this case) is chosen to fuse the evidences for each sensor. Because evidences are weighted by the fusion algorithm, if at a certain time step, a sensor output disagrees with other time steps (wrong classification), it will be given less weight. If higher number of t_s is considered for time domain fusion, the output will be more robust against wrong classification (smoother curve) but the response time (rate of change of classification output) will be slower. t_s is a tuning parameter for time-domain fusion and this can be tuned based

on faster response (lower t_s) or better robustness (higher t_s). An artistic representation of how multiple cameras can be placed on an AgBot is shown in Fig. 6.8.

Figure 6.8. Placement of multiple cameras on an AgBot.

With increasing number of sensors, the possibility of finding a faulty sensor also increases. Any multi-sensor fusion algorithm should be able to find the faulty sensor in the system and compensate for that in fusion architecture. The proposed algorithm is able to find the faulty sensor in space domain fusion step because faulty sensor disagrees with the evidences from other sensors. The algorithm compensates for that by applying less weight to the faulty sensor evidences so that it has less effect on final classification output. Fig. 6.10 shows the classification outputs at each step depicted by Fig. 6.9. At the top, Fig. 6.10 shows the time steps where Ragweed and Pigweed should be classified with 100% accuracy (ground truth). Fig. 6.10 (a) and (b) shows the classification output from each camera for Pigweed and Ragweed respectively. Between time steps 20 - 40 and 60 - 100, CNN is showing wrong classification (between 20 - 40 it's not 100% classification output for Ragweed) possibly due to noise and vibration. Also in this scenario, right camera (synthetically introduced data: faulty sensor) is showing contradicting output compared to center and left camera. The goal is to observe how the fusion algorithm handles the faulty evidence from right camera. Right camera is showing higher accuracy of Pigweed for time steps 20 - 80 (where it should be high accuracy for Ragweed from ground truth) and for time steps 100 - 150 showing higher

accuracy for Ragweed (where it should be Pigweed). After time step fusion (Fig. 6.10 (c)), unstable outputs are reduced (smoother curve) but right camera still showing wrong output. After space fusion (Fig. 6.10 (d)), the faulty evidence from right camera is compensated with lower weights. Final output shows steady and accurate weed classification depicting ground truth values.

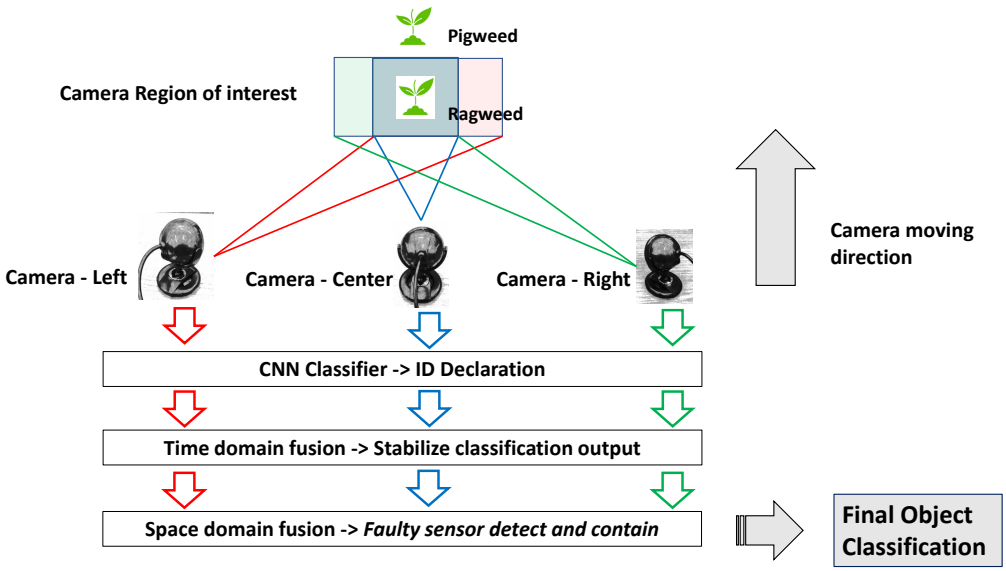


Figure 6.9. Reduced unstable classification with time domain sensor fusion. Reduce the effect of faulty sensor evidence with space domain sensor fusion.

6.7.2 Fusion when Weed is Partially Occluded

Partial occlusion can occur in the field due to wind, bumpiness or weed growing outside the region of interest of the camera. Fig. 6.11 shows partial occlusion scenario for our multi-sensor setup. Zoomed in image showing the views of different cameras. For the first half of the video, Pigweed is partially occluded for center and left camera but full exposure for right camera. For the last half of the video, Ragweed is partially occluded for center and right camera but full exposure for left camera. Potted plants are carefully placed to create this scenario of partial occlusion.

Classification accuracy and fusion steps are shown in Fig. 6.12. Ground truth values for the time steps are shown at the top. Between time step 20 - 30, left camera shows loss in

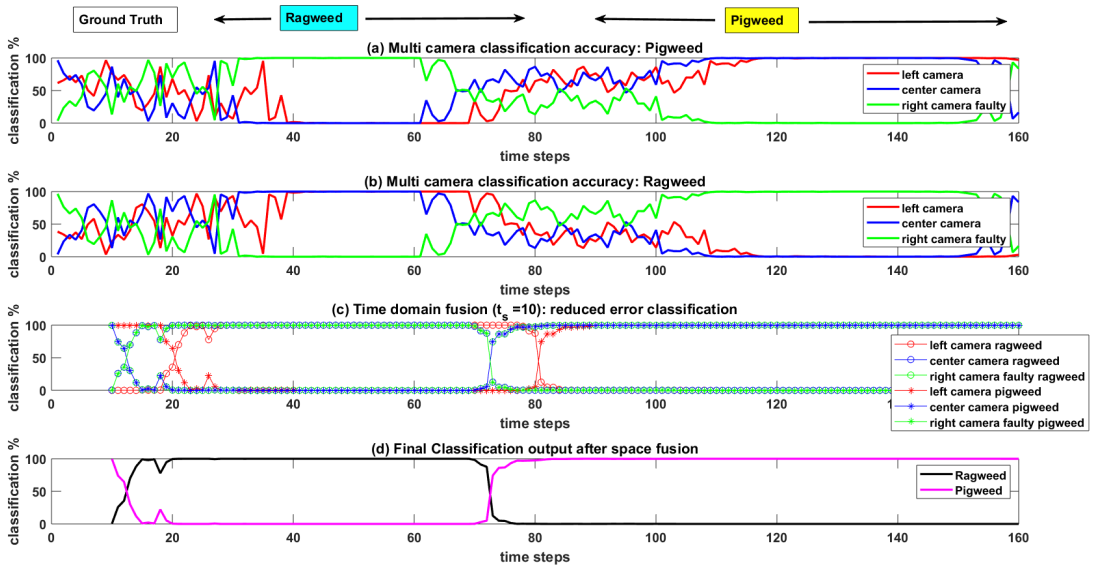


Figure 6.10. Ground truth value for Ragweed and Pigweed at the top. (a) Classification accuracy from left, center and right camera for Pigweed. (b) Classification accuracy from left, center and right camera for Rigweed. (c) Reduced unstable output (smooth curve) with time domain fusion. Each line shows the classification % of a specific weed for a specific camera. (d) Eliminated the effect of faulty sensor (right camera) evidence on final classification output with space domain fusion.

classification accuracy (Fig. 6.12 (a)) for Pigweed, possibly due to partial occlusion. Between time step 70 - 80, right and center camera show loss in classification accuracy (Fig. 6.12 (b)) for Pigweed. A small time step ($t_s = 3$) is used for time domain fusion step to capture the dynamics of the system. In space domain fusion, error classification due to partial occlusion is reduced by the algorithm. As an example, between time step 20 - 30, evidence from left camera contradicts with right and center camera. Right and center camera are showing high evidence for Pigweed, whereas left camera is showing contradicting evidence. As a result, evidences from left camera are penalized during these time steps. Final fused classification output is able to capture the classification accuracy of the ground truth value which is marked at the top of the figure.

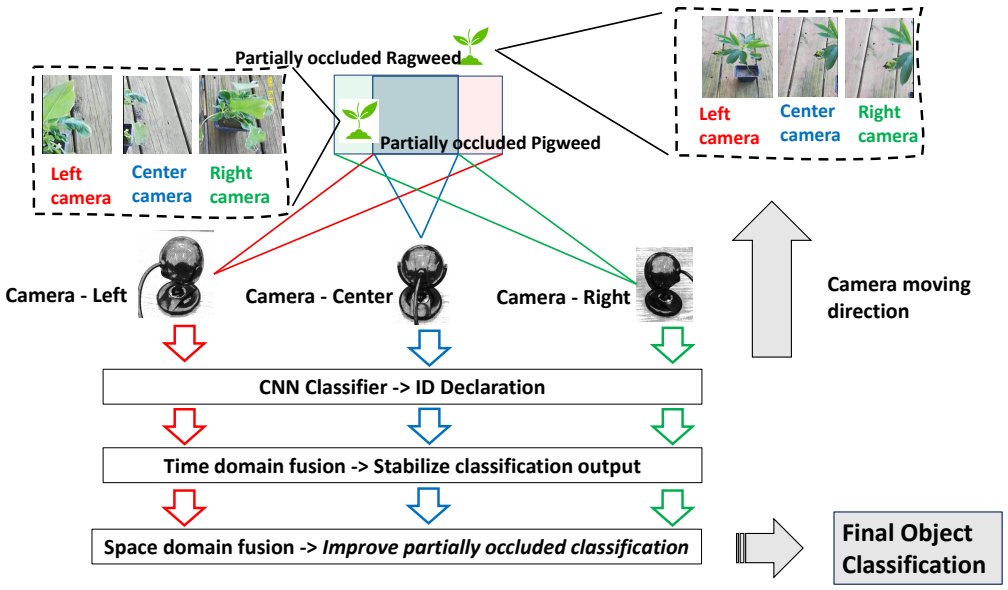


Figure 6.11. Improved classification for partially occluded weed data with space domain sensor fusion.

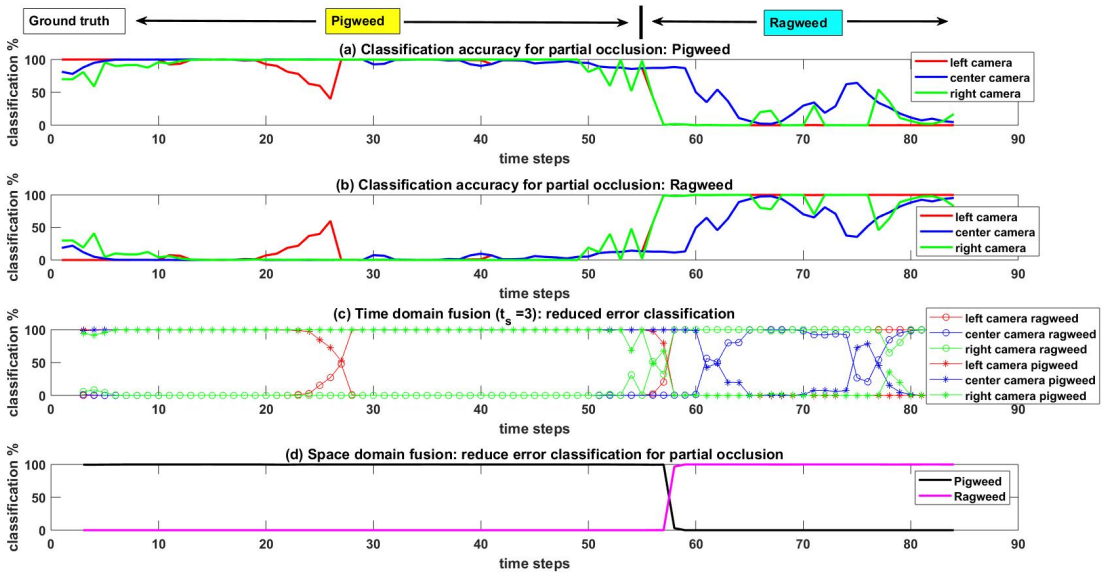


Figure 6.12. Ground truth value for Ragweed and Pigweed at the top. (a) Classification accuracy from left, center and right camera for Pigweed. (b) Classification accuracy from left, center and right camera for Rigweed. (c) Reduced classification error (smooth curve) with time domain fusion. (d) Eliminated the effect of partial occlusion on final classification output with space domain fusion.

6.8 Conclusions

An eight step algorithm under DS framework is introduced as an innovative methodology that can be used to better capture uncertainties related to decision level multi-sensor fusion. A novel entropy function is proposed based on Shannon entropy which takes into account the central value of probability interval and cardinality of both focal elements and FOD. As a result, it is better at capturing uncertainties under DS framework compared to Shannon and Deng entropy. The proposed algorithm calculates distances between multiple evidences (sensors). Based on evidence distance, it rewards the evidence which agrees with one another and penalizes the evidences which disagrees. The proposed entropy function is used to calculate the weights of the evidences. Conflicting evidences are modified before using them for spatial domain fusion. Classical DS combination rule is used for decision-level sensor fusion; as a result, associative and commutative properties are kept. The proposed method is able to suppress the paradoxes of classical DS combination rule. Detailed example shows that the proposed method produces competitive convergence rate and fusion accuracy in terms of combining the conflicting evidences in spatial domain. In time domain, transition property of proposed method from one evidence to another proved to be compatible for real time application. Uncertainties of CNN based classifier are included into the fusion algorithm using reconstructed BPA with precision and recall values from the classifier. Evidence fusion algorithm is tested with real-time video input for weed classification. Number of times-steps (t_s) considered for time-domain data fusion is an important tuning parameter. Smaller t_s value shows fast-response in classification output, bigger t_s value shows more robustness. Results show that proposed algorithm is able to include CNN uncertainties into the evidence fusion framework and applicable for real-time object classification from video feed. Also for real life application, multiple sensors are needed to correctly classify weeds from multiple points of view. The fusion algorithm is able to stabilize classification output from CNN with time domain fusion. Using space domain fusion, faulty sensor can be detected (if present in the sensor array) and its effect on final classification output is successfully minimized. The proposed fusion algorithm is also able to reduce classification error introduced by partial

occlusion of weeds from camera view. Final weed classification output from the fusion algorithm is correct, robust and stable in these challenging scenarios.

7. IMPROVING THE ROBUSTNESS OF OBJECT DETECTION THROUGH A MULTI-CAMERA BASED FUSION ALGORITHM USING FUZZY LOGIC

A single camera creates bounding box (BB) for the detected object with certain accuracy through Convolution neural network (CNN). However, a single RGB camera may not be able to capture the actual object within the BB even if the CNN detector accuracy is high for the object. In this research, we present a solution to this limitation through the usage of multiple cameras, projective transformation, and a fuzzy-logic based fusion. The proposed algorithm generates a 'confidence score' for each frame to check the trustworthiness of the BB's generated by the CNN detector. As a first step toward this solution, we created a two-camera setup to detect objects. Agricultural weed is used as objects to be detected. A CNN detector generates BB for each camera when weed is present. Then a projective transformation is used to project one camera's image plane to another camera's image plane. The intersect over union (IOU) overlap of the BB's is computed when objects are detected correctly. Four different scenarios are generated based on how far the object is from the multi-camera setup and IOU overlap is calculated for each scenario (ground truth). When objects are detected correctly and bounding boxes are at correct distance, the IOU overlap value should be close to the ground truth IOU overlap value. On the other hand, IOU overlap value should differ if BB's are at incorrect positions. Mamdani fuzzy rules are generated using this reasoning and three different confidence scores ('High', 'OK', 'Low') are given to each frame based on accuracy and position of BB's. The proposed algorithm was then tested under different conditions to check it's validity. The confidence score of the proposed fuzzy system for three different scenarios supports the hypothesis that the multi-camera based fusion algorithm improved the overall robustness of the detection system.

7.1 Objectives

Weed can be detected with a CNN based detector with very high accuracy and at real-time. If we use multiple cameras then the robustness of the overall system increases. When we use a detector to create a BB, it gives us the accuracy percentage of objects inside the

BB and the position of the BB. In this chapter, we explore the possibility of improving the overall system robustness by measuring the "confidence" of BB position of multiple cameras. The detector will give us the position of BB on each image plane. But how do we know for sure that the position of the BB is correct? When we are detecting an object using multiple cameras, the position of the BB on the camera image plane may appear in different places based on the detection accuracy and the position of the cameras. But in 3D space, the object is at the exact same position for both cameras. We will manipulate this information to calculate the confidence score of the BB position using projective transformation, IOU overlap value and fuzzy logic based fusion. We will complete the following steps to calculate the confidence score:

- Create a two-camera setup and calculate the homography between the two cameras.
- Place weed (object) at different specific distances from the camera setup.
- Use a CNN detector to create BB of detected weed on both camera. Use best possible detection BB so that the weed is perfectly detected and inside the BB.
- Use homography matrix to project one camera image plane over another and then calculate the IOU overlap values of the BBes after projection for different weed position.
- The logic is, if perfect BB is created at a certain distance of weed position then it will have a specific IOU overlap value. We will have "High" confidence for this scenario. If the position or IOU value deviates from perfect condition then we will have "less than High" confidence on the detection. Use fuzzy "If-then" rules to capture these conditions.
- Use defuzzification to get the crisp "confidence" value for different scenarios.

7.2 Projective Transformation

In a pinhole camera model, a point in 3D space is projected onto an imaging surface which is called image plane [144]. All rays (or points) of light passes through a single point which is called camera center. The size of the object on image plane can be calculated from

similar triangle. Assuming the camera is calibrated or there are no distortions (radial and barrel distortion), a point (X, Y, Z) in physical world is projected onto image plane at (x, y) location with following equations:

$$x = f.X/Z \tag{7.1}$$

$$y = f.Y/Z \tag{7.2}$$

where, f is the focal length.

The relationship that maps a set of points from one image plane to a set of points to another image plane is called projective transformation. Planar homography is the projective mapping from one plane to another. According to Hartley and Zisserman [86], projective transformation is defined as, "A planar projective transformation is a linear transformation on homogeneous 3-vectors represented by a non-singular 3-by-3 matrix."

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \tag{7.3}$$

Or in short, $x = H.x$, where H is the homography matrix. This homography matrix relates the position of a point from source image plane (image plane 1 in Fig. 7.1(a)) to a destination image plane (image plane 2 in Fig. 7.1(a)). $(x_1.x_2, x_3)^T$ and $(x_1.x_2, x_3)^T$ are coordinates of a single point on two image planes. Also, H is a homogeneous matrix, which means only the ratio of the matrix elements are important. There are eight independent ratios in H (h_{33} is a scaling factor) which means a projective transformation has eight degrees of freedom [86]. Here, homogeneous coordinates are used to represent the points. In homogeneous coordinate system n dimensional vector is represented as a $(n+1)$ dimensional vector ((x, y) coordinate becomes (x, y, z)). Homogeneous coordinates can be converted to inhomogeneous coordinates very easily ($x = x/z$ *and* $y = y/z$). This is a mathematical trick to represent rays of light as points, because image plane projects a ray of light on a point.

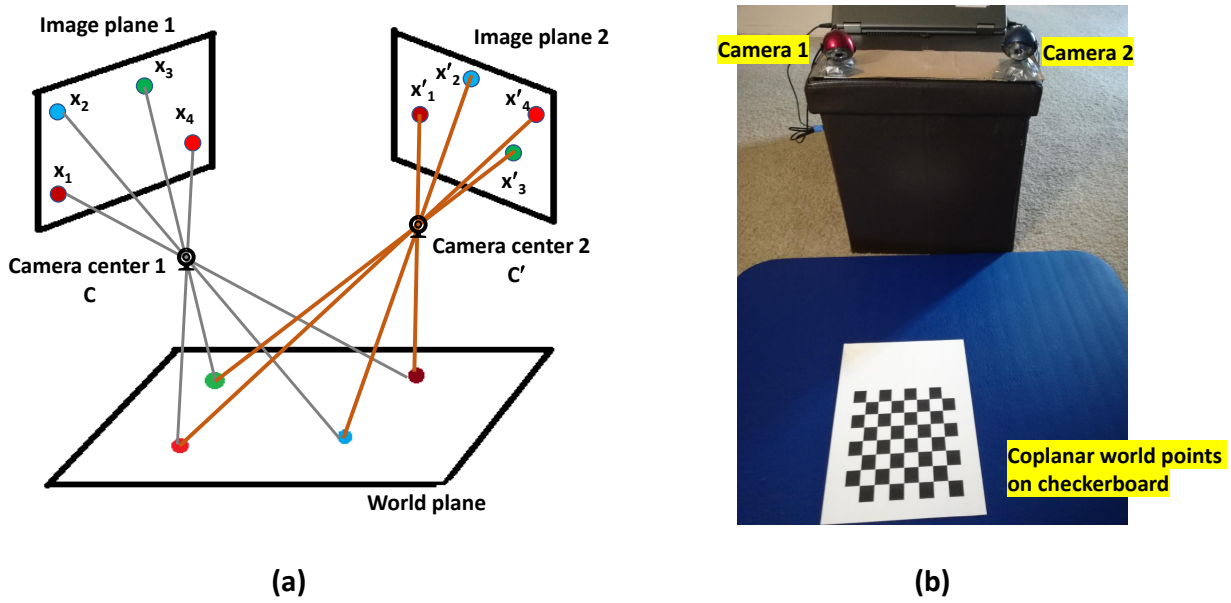


Figure 7.1. (a) For multiple cameras, image planes are related by projective transformation when all world points are co-planer. (b) Actual setup of two cameras with co-planer checkerboard points.

Let's consider a pair of inhomogenIOUs matching points (x, y) and (x, y) on image plane 1 and 2 respectively. We are considering inhomogenIOUs coordinates because they can be measured directly from image plane (coordinates of points in pixel). From equation (7.3):

$$x = \frac{x_1}{x_3} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}} \tag{7.4}$$

$$y = \frac{x_2}{x_3} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}} \tag{7.5}$$

After rearranging:

$$x(h_{31}x + h_{32}y + h_{33}) = h_{11}x + h_{12}y + h_{13} \tag{7.6}$$

$$y(h_{31}x + h_{32}y + h_{33}) = h_{21}x + h_{22}y + h_{23} \tag{7.7}$$

Four points on each image plane will create eight linear equations and four points are sufficient to solve for H between two image planes. The only condition is, no three points can be collinear [86]. After H matrix is calculated, it is then applied to the whole image plane 1 to convert it to image plane 2.

Few important remarks:

1. Camera intrinsic parameters or pose are not needed to calculate H.
2. If only four points are used to calculate H, outliers can create incorrect H matrix.
3. Including more points and using robust method to minimize reprojection error will help to calculate correct H matrix.

7.2.1 Homography Matrix Calculation

We have established that no camera or pose parameters are needed to calculate homography matrix, H. As a result the following steps are followed to calculate H:

1. Place a printed checkerboard pattern infront of the cameras (Fig. 7.1 (b)).
2. Measure the pixel locations of checkerboard corners for each camera (Fig. 7.2 and Fig. 7.3).
3. Calculate the reprojection error which is the sum of squared euclidean distances between H times the camera 2 checkerboard corner points and camera 1 checkerboard corner points.
4. Use an optimization algorithm to minimize reprojection error for a specific H matrix.

We have tested four different optimization algorithm for H matrix calculation. Then we calculate the projection error between the actual checkerboard corner location and projected checkerboard corner location. Error is calculated interms of how much each corner point deviate from actual corner location in terms of pixels. All results are presented in Table 7.1. Because we are measuring the corner locations offline, robust methods are specifically necessary for our use case. Once the H matrix is calculated, that will remain same as long as

camera positions are constant. As table 7.1 shows both least mean square and least median square resulted minimum reprojection error. The H matrix calculated is:

$$H = \begin{bmatrix} 9.29968576e-01 & -8.34785721e-01 & 3.28368011e+02 \\ 3.21580417e-01 & 9.89425377e-01 & -3.15245367e+02 \\ -7.93435882e-05 & 5.73575359e-05 & 1.0 \end{bmatrix} \tag{7.8}$$

Fig. 7.4 shows how H matrix is used to reproject camera 2 view into camera 1 view. Left image of Fig. 7.4 is the actual left camera view (Fig. 7.2) multiplied by H which projects the left camera view into right camera view.

Table 7.1. Optimization methods and reprojection error.

Method	Reprojection error (pixel/point)
Least mean square	1.621
RANSAC [94]	1.715
Least median square [145]	1.621
PROSAC [146]	1.961

7.3 CNN based Weed Detection from Classification

We want to calculate the value of IOU overlap when weed is detected on both camera. To do that, first we need the bounding box (BB) for the detected weed. Until now we have trained a CNN classifier (VGG16 with transfer learning) in Chaper 4. Now we will fit a detector on top of the classifier to get the BB. We should mention that, we are just fitting a detector, we are not training the detector for better performance. The steps to fit a detector on top of the classifier is shown in Fig. 7.5.

First an image pyramid is created to deal with different scale factors. The reason is to detect objects at different scale. As image pyramid grows bigger, it will help to detect bigger objects. Then we run a sliding window at each scale of the pyramid. The size of the sliding window should depend on the size of the object to be detected. At each position of the

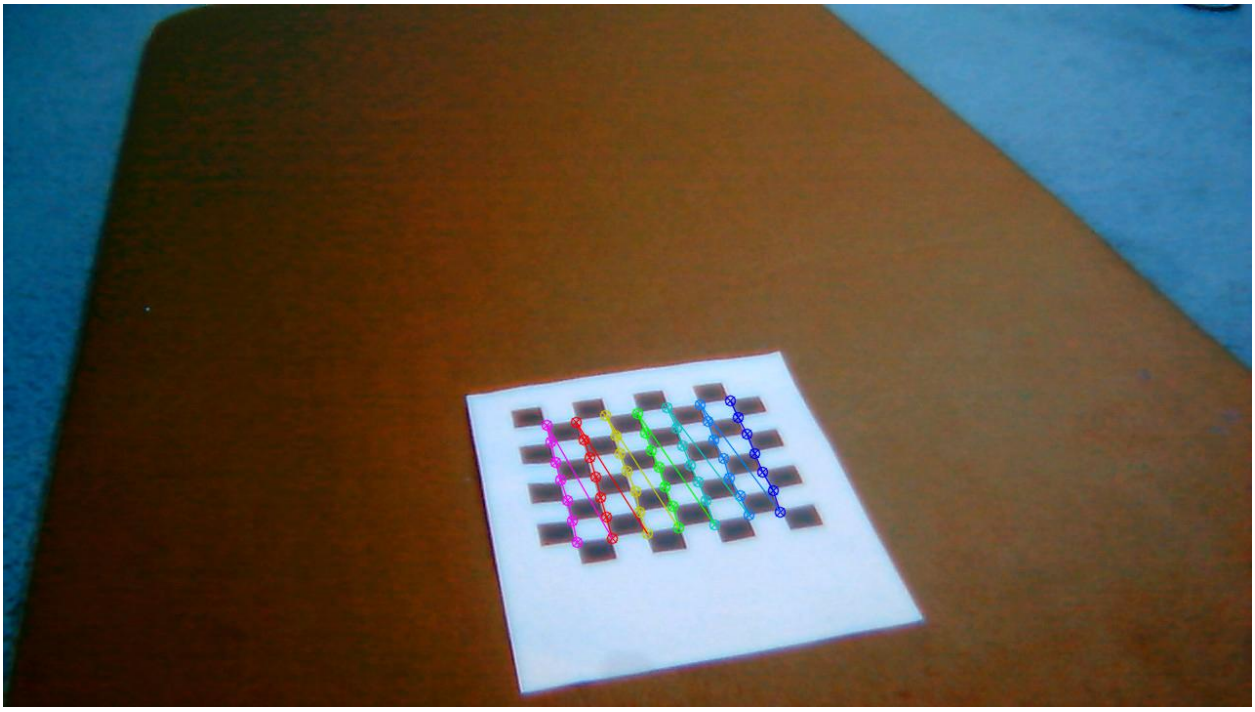


Figure 7.2. Checkerboard corner points for left camera (camera 2).

sliding window, we are passing it through the CNN classifier and saving the class label with % accuracy. Basically we are creating a lot of bounding boxes with class label accuracy and saving their location. Then we pass all the BBs through an algorithm called non maximum suppression (NMS). NMS takes a BB, then calculates IOU with all the other BBs. If IOU value is over some predetermined threshold then that BB is discarded, else that BB is kept. Basically NMS tries to figure out which one is the unique BB. These steps are standard and close to the steps followed in SSD [147] but without the training.

7.3.1 IOU Overlap Calculation

Now that we have the BB from detector, we want to calculate the IOU value for left and right camera BB. We want to see how the IOU value changes before and after reprojection when weed is detected correctly and BB is created. We also want to see how IOU value changes when weed is detected at different distances from camera.

Fig. 7.6 shows how BBes from both cameras overlap each other at different distances. "Too far" is about 36 inches away from camera and "too close" is about 8 inches away from

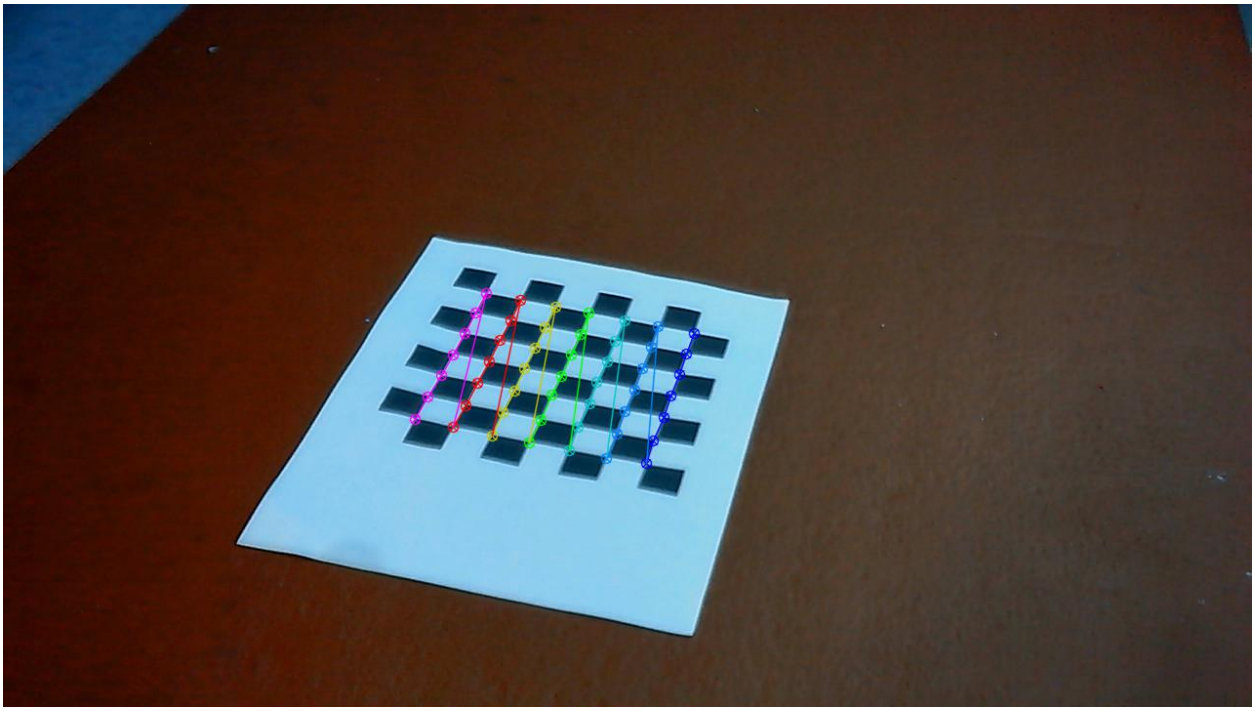


Figure 7.3. Checkerboard corner points for right camera (camera 1).

camera. BB sizes are fixed for both camera. We also assume perfect detection. As the weed move closer to the camera, we can see that the BB overlap is decreasing and for "Too close" position, without reprojection BB overlap is zero. With reprojection (Left camera image is reprojected using H) BB overlap still decreases as weed moves closer to camera. Homography is 2D transformation from one camera image plane to another camera image plane. But the object (weed) we are detecting is actually 3D. As a result, as we move closer to camera, BB overlap decreases because projection of the 3D object onto a 2D plane deviates more. If the objects we are detecting were 2D then after reprojection, BB overlap would remained more or less the same.

Fig. 7.7 shows how IOU value changes when weed distance changes for two different BB sizes. IOU overlap after reprojection is always bigger than before reprojection. As we move farther, the difference between two different systems reduces. Also for smaller BB size, IOU before projection goes to zero when weed at "close" position. This makes sense because smaller BB has smaller chance of overlap. IOU overlap between two different camera BB changes with BB size. In actual scenario, a CNN detector will create BB based on the size

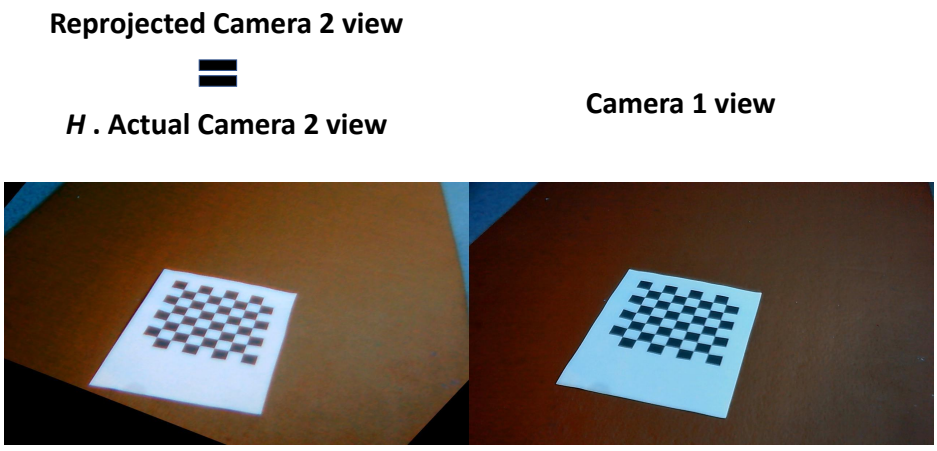


Figure 7.4. Reprojection of camera 2 image to camera 1 image using homography matrix.

or shape of the object and they will be different based on scenario. But for this study, we have kept the BB size constant (BB width: 500 pixels, height: 350 pixels). But in future, different BB sizes could be considered and IOU value should be calculated for a range of BB sizes.

7.4 Fuzzy Logic

Fuzzy logic was invented by Lotfi Zadeh [148] by combining crisp logic and set theory. In reality, many concepts are better defined by human words than by mathematics. Zadeh tried to capture that link between human language and mathematics. Zadeh used fuzzy sets to capture that relationship. If there is uncertainty about membership data regarding that data belonging to a particular set, fuzzy sets are used to define that data to a partial set. As an example, if fuzzy sets are used to define the bounding box position of weed that can be defined as "too close", "close", "far" or "too far"; and for confidence of fused bounding box as "low", "OK" and "high". The membership degree quantifies the level of how that data belongs to that set. As an example, in the case of bounding box distance from camera center:

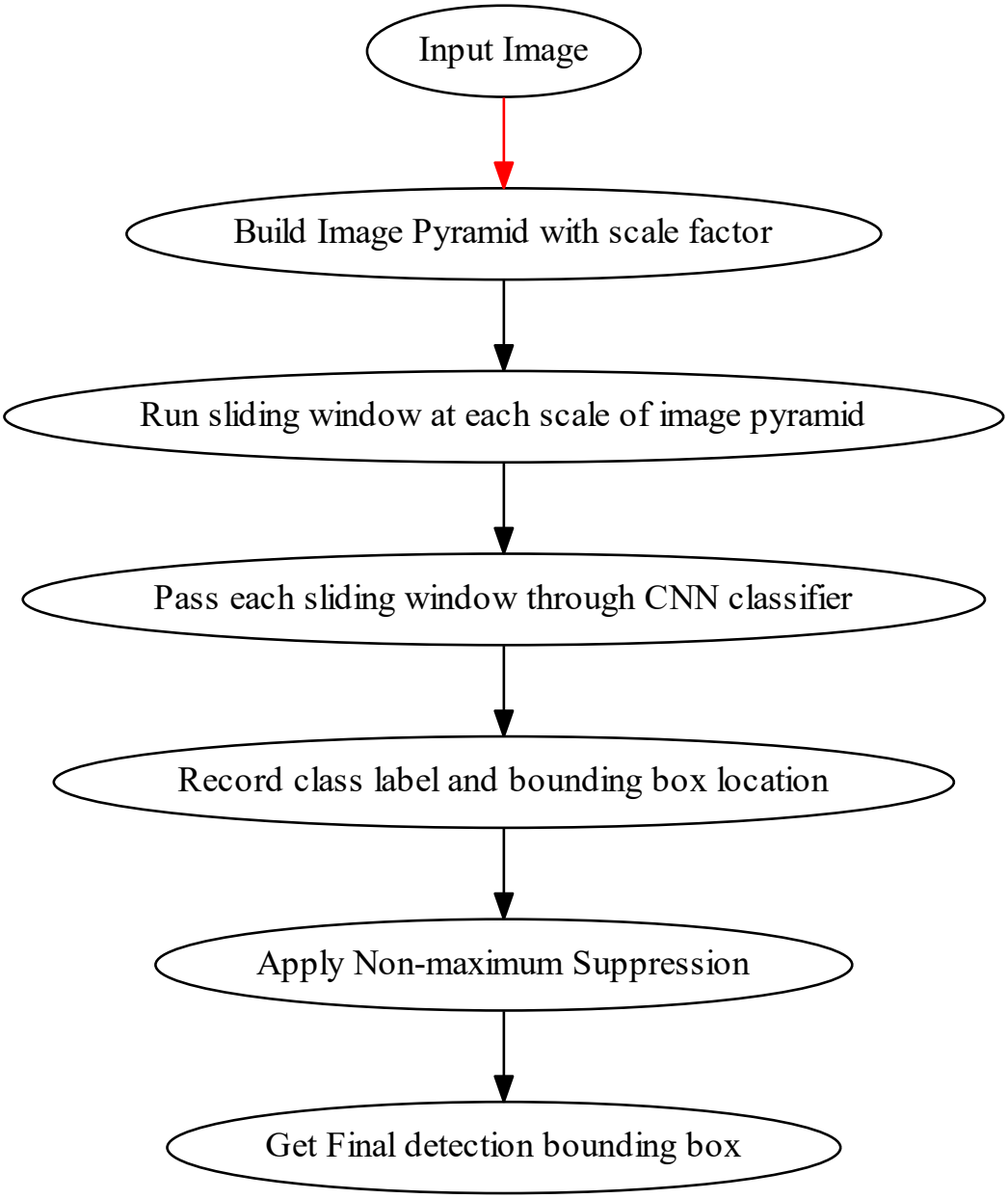


Figure 7.5. Steps to create a object detector from CNN classifier.

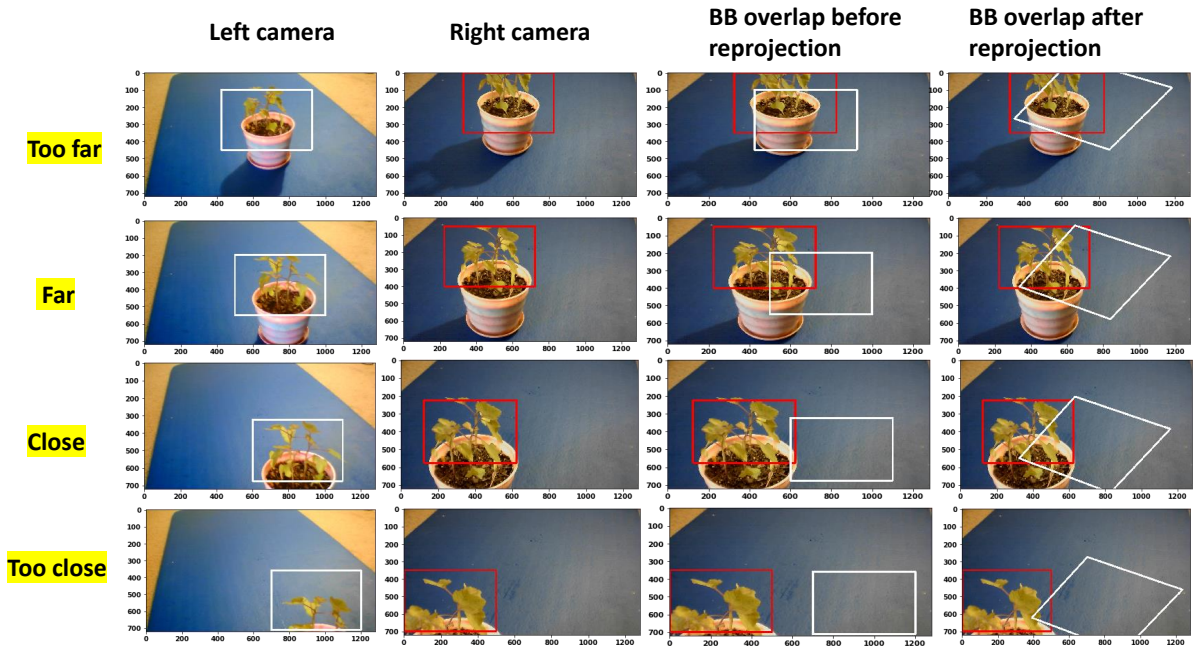


Figure 7.6. How BB overlap changes when weed is placed at different distance from camera. "Too far", "Far", "Close" and "Too close" represents the distance of the weed from camera. "After reprojection" shows the BB overlap when left camera image is reprojected using H. BB size, width: 500 pixels; height: 350 pixels.

$$m_{distance}(y) \in [0, 1] \tag{7.9}$$

where $m_{distance}(y)$ is the degree of membership y has in fuzzy set of "distance in camera view" and y is the vertical distance from top of the camera view in pixel value. Fig. 7.8 shows this.

Membership function expresses various degrees of strength between the elements in fuzzy set. If likelihood is higher that an element belongs to a certain set then the membership strength is also higher. Membership strength of zero means that the element doesn't belong to that set, and membership strength of one means that the element definitely belongs to that set. In this study, fuzzy sets are used to define the distance of bounding box, IOU overlap and confidence in bounding box position.

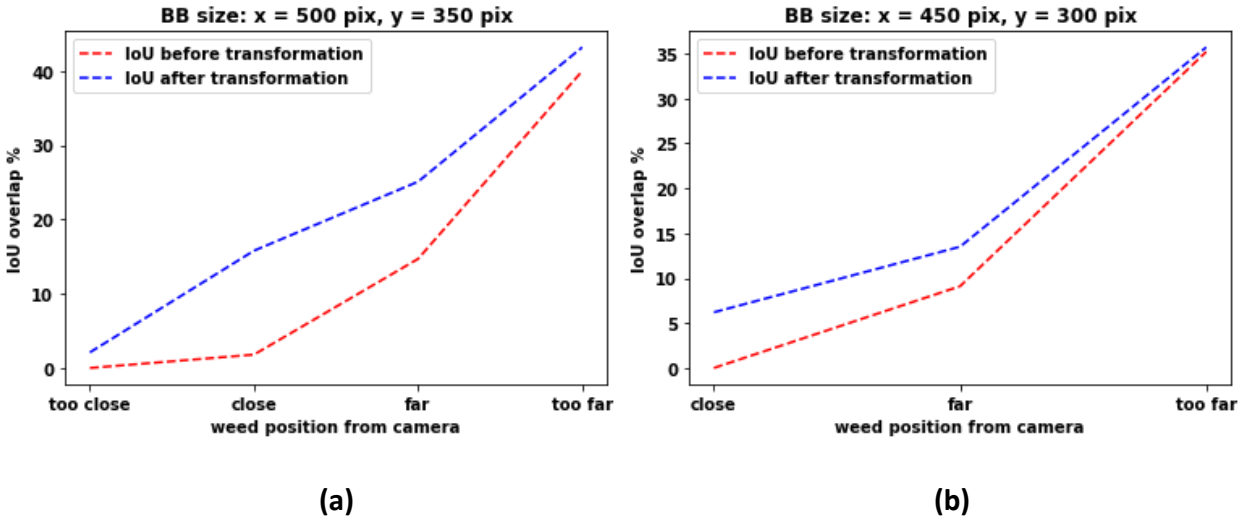


Figure 7.7. IOU overlap value with respect to distance from camera.(a) BB size, width: 500 pixels; height: 350 pixels.(b) BB size, width: 450 pixels; height: 300 pixels.

The membership function design is based on a combination of our personal experience and the knowledge we gained from the camera setup testing. In Fig. 7.8 and 7.9 the membership function for right and left camera bounding box position is presented respectively. Weed is placed at four different distances in-front of the camera as shown in Fig. 7.6. Fuzzy names are selected based on their position ("too far", "close" etc.). Triangular membership functions are selected to represent their strength. As an example, in Fig. 7.8, for "too far" position, BB distance has maximum degree of membership (one) at position 150 (pixel). Which means when weed is placed at "too far" position and the best BB is generated by the CNN detector, the BB y-position center (vertical distance) is found at 150 pixels from the top position of the camera view. In Fig. 7.10 the membership function for IOU overlap is presented. It's derived from Fig. 7.7(a). As an example, for "close" position the IOU overlap value from Fig. 7.7(a) is 15%. Which means, when weed is placed at "close" position and BB is created from CNN detector, for perfect detection (best possible fit of BB over the weed for our test set) the IOU overlap value will be 15%. The overlap value will be different if it's not

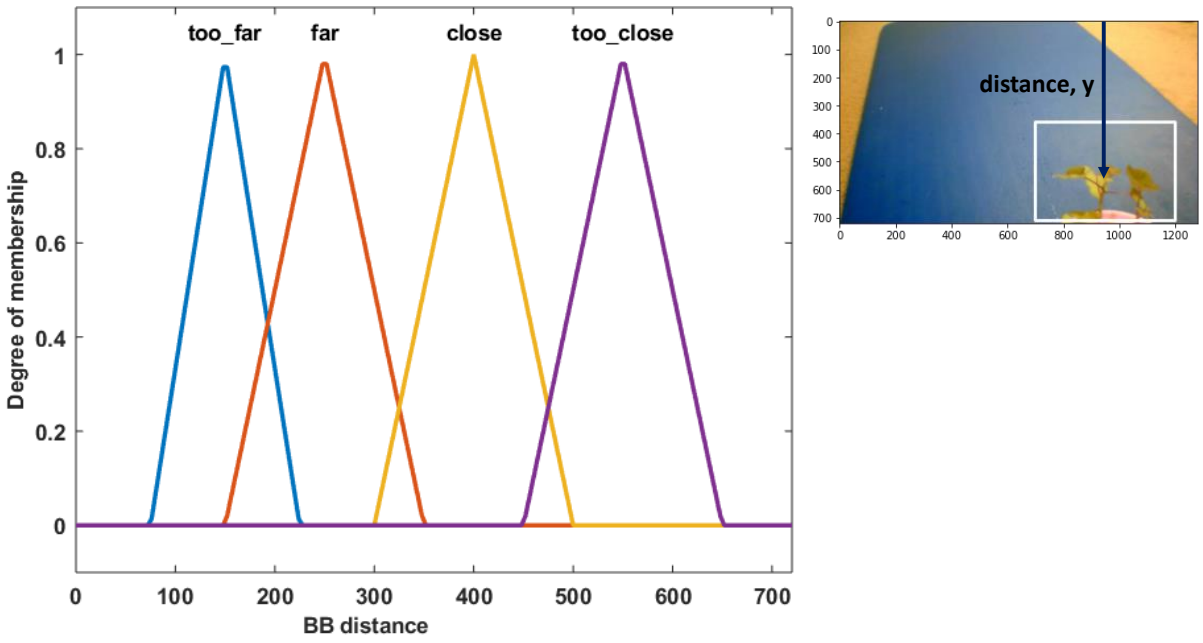


Figure 7.8. Fuzzy set for weed bounding box distance in camera view for right camera. [Input].

a perfect detection at this position. Gaussian membership function is used to represent this. Fig. 7.11 shows the confidence membership function. If the BB distance and IOU overlap value perfectly match then they will have "High" confidence. And based on their deviation from perfect condition and the fuzzy rule-set, they can be "OK" or "Low".

7.4.1 Fuzzy Steps and Rule-set:

The overall fuzzy system with inputs, rule evaluations, defuzzification (output) is presented in Fig. 7.12. Following steps are followed for the application of fuzzy analysis:

1. Identify inputs with their ranges and name them: In this study, the subsets for weed distance and IOU overlap are too close, close, far, too far.

2. Identify output with their ranges and name them: In this study, the subsets for confidence are Low, OK and High.

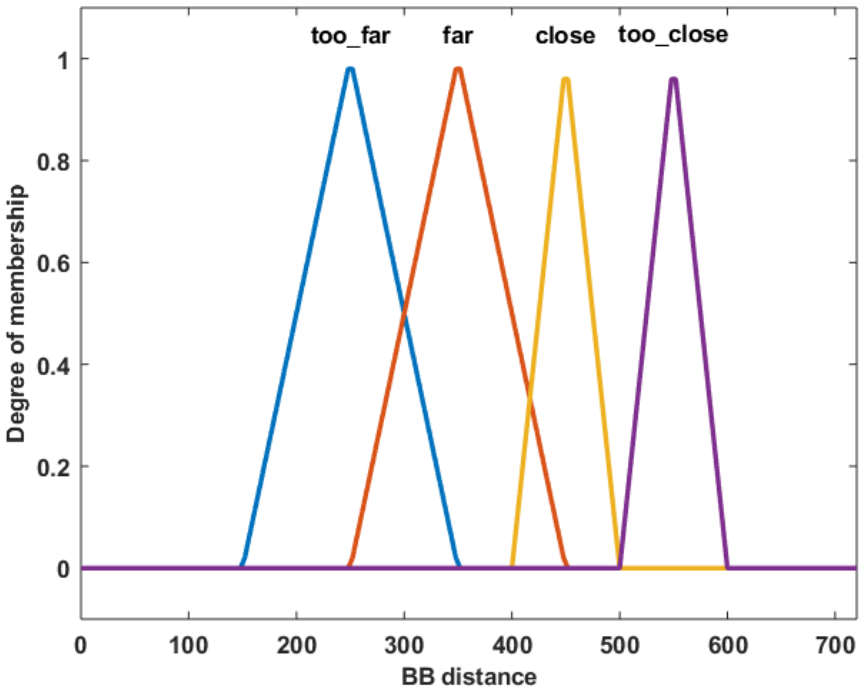


Figure 7.9. Fuzzy set for weed bounding box distance in camera view for left camera. [Input].

3. Create degree of fuzzy membership function for inputs and output: Fig. 7.7(a), 7.8, 7.9, 7.10 and Fig. 7.11 shows this.

4. Construct the rule base for the system based on expert judgement: The rules creates a linguistic relationship between the input variables and the output. In a fuzzy "IF..THEN" rule, the IF part is the premise and the THEN part is the output based on premise. The rules can be combined with logical "or" or logical "and". Here, the three input variables are: RightCam BB (right camera BB distance), LeftCam BB (left camera BB distance, and IOU overlap, and the output variable is Confidence. The generic conditional statement used in this study is:

$$\begin{aligned} R_n : IF\ RightCam\ BB \text{ is } A(n)\ and\ LeftCam\ BB \text{ is } B(n) \\ and\ IOU\ overlap \text{ is } C(n) \\ THEN\ Confidence \text{ is } D(n) \end{aligned} \tag{7.10}$$

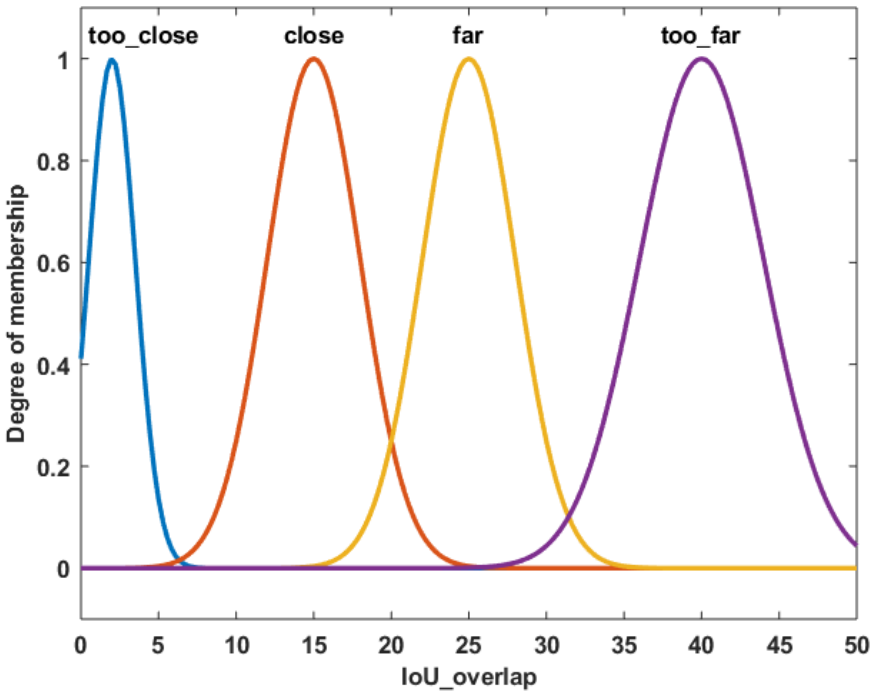


Figure 7.10. Fuzzy set for bounding box overlap between left and right camera [Input].

here, A(n), B(n) and C(n) are too close, close, far, too far and D(n) is Low, OK and High. Twenty fuzzy rules are designed to optimize for the relationship between output and inputs. All rules are not valid for this problem. Some rules might not get triggered at all based on BB position and IOU overlap value. All rules are given equal weight of one. The final output, Confidence is the union of the output fuzzy subsets for the activated rules. In this research, Mamdani [149] inference is used. All the rules are presented in Table 7.2.

5. Defuzzification: Centroid defuzzification method is used in this research [77].

7.5 Results

Three scenarios are created and tested to check the performance of the fuzzy system. In all the following scenario we are assuming that, the CNN detector is detecting the weed with

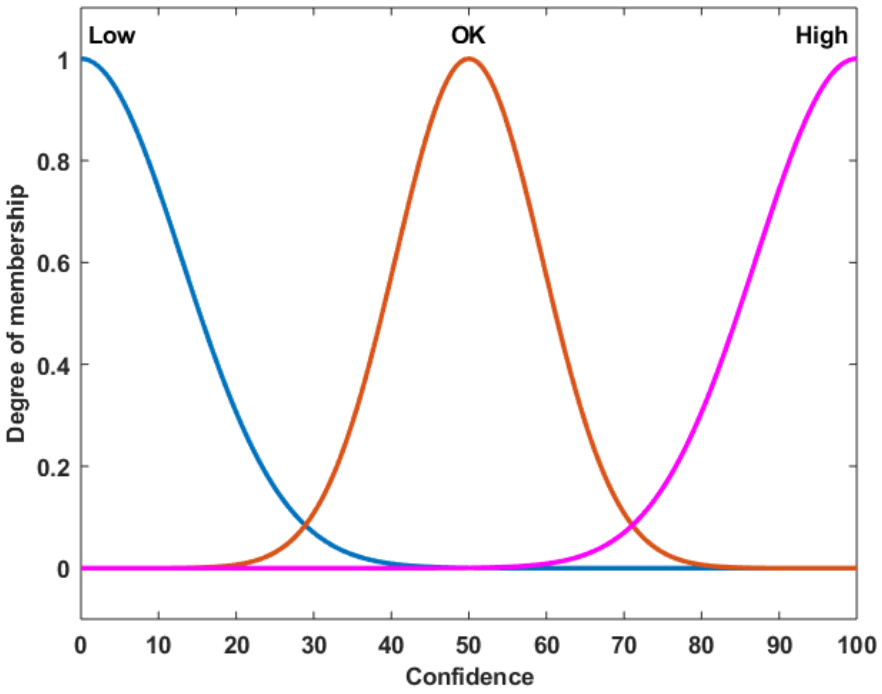


Figure 7.11. Fuzzy set for fusion confidence [Output].

100% accuracy inside the BB, the whole weed plant is visible from both cameras and there is no occlusion.

7.5.1 Scenario 1: High Confidence

Weed at position "too close" is chosen to test the performance of the system. Fig. 7.13 shows all the steps of the fuzzy confidence score measurement system. For 'High confidence' situation, the CNN detector detects the weed correctly. The BB covers the whole weed for both cameras and the weed is usually at the center of the BB. As three inputs, we measure the vertical center distance of the BB for both cameras and the IOU overlap score. All three inputs goes into the fuzzy rule-set. For a specific position of the weed in 3D world, if the weed is detected by both cameras correctly then they should have a specific IOU overlap value which we receive from Fig. 7.7(a). As a result, all these inputs trigger the fuzzy-rule 4 (R_4 in Table 7.2). This follows one of the rules for High confidence. After defuzzification we receive a confidence score of 88.6%. The fuzzy system gives a high confidence score

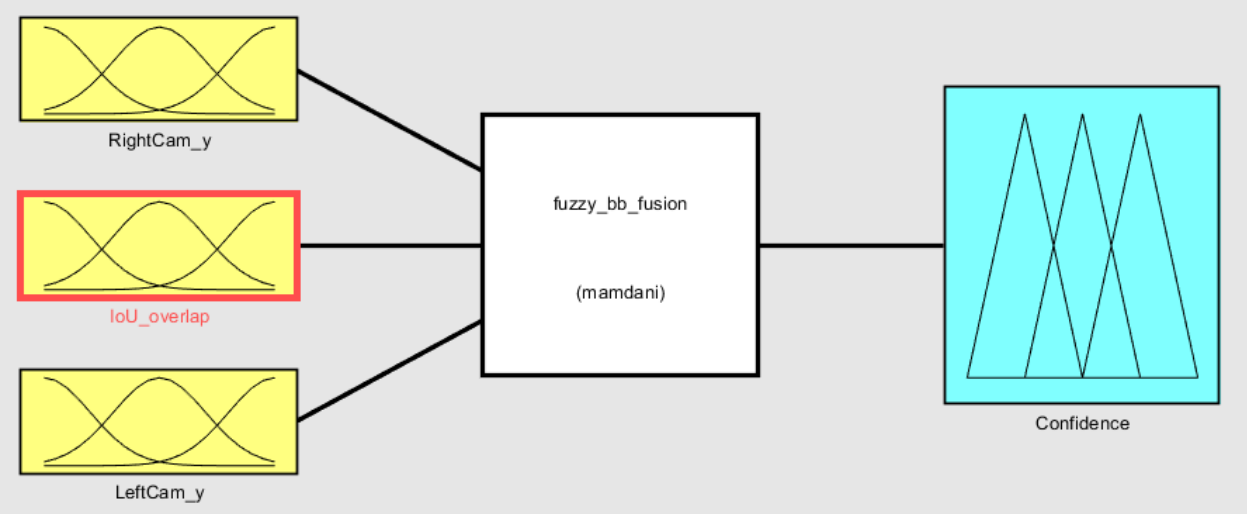


Figure 7.12. Basic configuration of the fuzzy system.

because the weed is detected correctly by both cameras. One important thing is, because we are using gaussian membership function and centroid defuzzification, for high confidence score will always provide a value between 70% and 100% but not exactly 100% even if the detection meets all the criteria (rules) for perfect detection.

7.5.2 Scenario 2: OK Confidence

Weed at position "too close" is chosen to test the performance of the system. Fig. 7.14 shows all the steps of the fuzzy confidence score measurement system for 'OK confidence' situation. Here the right camera detects the weed correctly but the left camera detection is partially correct and detecting the weed at a slightly left position than the actual position. Although the BB center vertical distance for both right and left camera is correct and comparable to 'High confidence' situation. But because left camera is detecting at a partially correct position, this deviates the IOU overlap value and triggers rule 20 (R_{20} in Table 7.2). As a result the fuzzy system gives it a confidence score of 50% which is a reasonable value given the partial detection.

Table 7.2. Fuzzy rules.

	Rules description	
R_1	IF (RightCam BB is too far) and (IOU overlap is too far) and (LeftCam BB is too far) THEN	(Confidence is High)
R_2	IF (RightCam BB is far) and (IOU overlap is far) and (LeftCam BB is far) THEN	(Confidence is High)
R_3	IF (RightCam BB is close) and (IOU overlap is close) and (LeftCam BB is close) THEN	(Confidence is High)
R_4	IF (RightCam BB is too close) and (IOU overlap is too close) and (LeftCam BB is too close) THEN	(Confidence is High)
R_5	IF (RightCam BB is too far) and (LeftCam BB is too close) THEN	(Confidence is Low)
R_6	IF (RightCam BB is too far) and (LeftCam BB is close) THEN	(Confidence is Low)
R_7	IF (RightCam BB is too far) and (LeftCam BB is far) THEN	(Confidence is Low)
R_8	IF (RightCam BB is far) and (LeftCam BB is too far) THEN	(Confidence is Low)
R_9	IF (RightCam BB is far) and (LeftCam BB is close) THEN	(Confidence is Low)
R_{10}	IF (RightCam BB is far) and (LeftCam BB is too close) THEN	(Confidence is Low)
R_{11}	IF (RightCam BB is close) and (LeftCam BB is too far) THEN	(Confidence is Low)
R_{12}	IF (RightCam BB is close) and (LeftCam BB is far) THEN	(Confidence is Low)
R_{13}	IF (RightCam BB is close) and (LeftCam BB is too close) THEN	(Confidence is Low)
R_{14}	IF (RightCam BB is too close) and (LeftCam BB is close) THEN	(Confidence is Low)
R_{15}	IF (RightCam BB is too close) and (LeftCam BB is far) THEN	(Confidence is Low)
R_{16}	IF (RightCam BB is too close) and (LeftCam BB is too far) THEN	(Confidence is Low)
R_{17}	IF (RightCam BB is too far) and (IOU overlap is Not too far) and (LeftCam BB is too far) THEN	(Confidence is OK)
R_{18}	IF (RightCam BB is far) and (IOU overlap is Not far) and (LeftCam BB is far) THEN	(Confidence is OK)
R_{19}	IF (RightCam BB is close) and (IOU overlap is Not close) and (LeftCam BB is close) THEN	(Confidence is OK)
R_{20}	IF (RightCam BB is too close) and (IOU overlap is Not too close) and (LeftCam BB is too close) THEN	(Confidence is OK)

7.5.3 Scenario 3: Low confidence

Weed at position "too close" is chosen to test the performance of the system. Fig. 7.15 shows all the steps of the fuzzy confidence score measurement system for 'Low confidence' situation. Here the right camera detects the weed correctly but the left camera detection is less than partially correct and detecting the weed at a higher position than the actual position. As a result, the BB center vertical distance is not comparable with 'High confidence' situation. This triggers rule 15 (R_{15} in Table 7.2). As a result the fuzzy system gives it a confidence score of 10.8% which is a reasonable value given the less than partial detection.

One limitation of this fuzzy confidence measurement system is, this inherently assumes one of the camera detection is correct. But if both of them are incorrect then this system

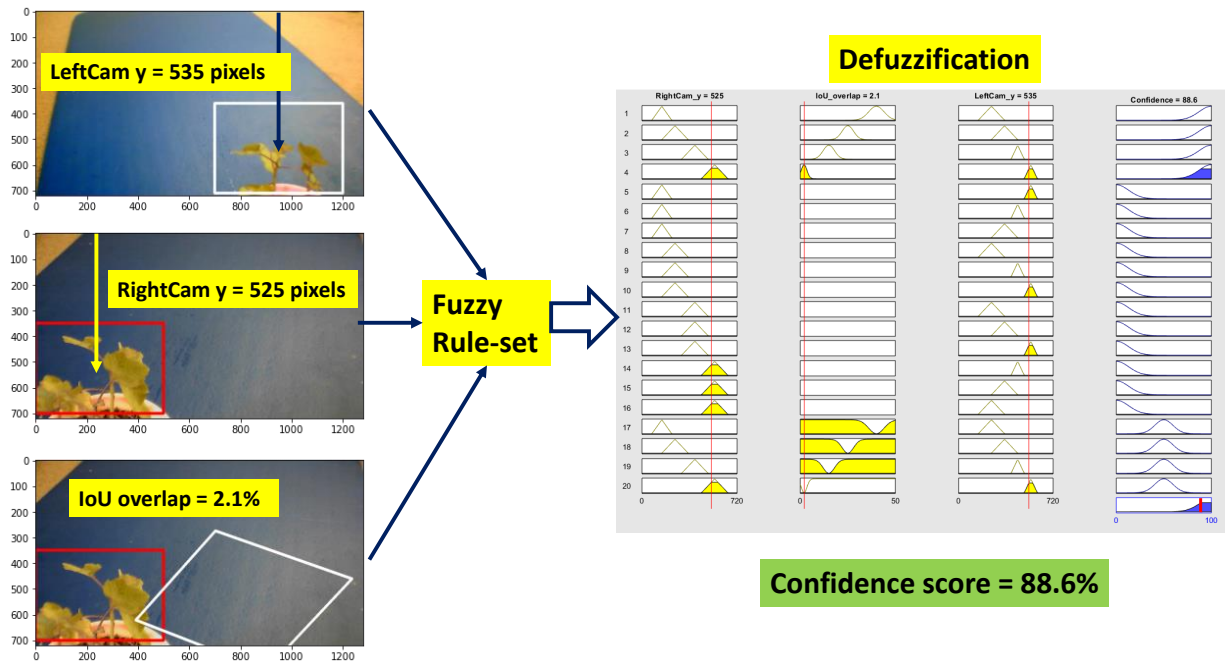


Figure 7.13. Fuzzy confidence score measurement (High confidence).

may produce less than ideal confidence score. There are two ways to tackle this limitation. First, for video input, we can incorporate a tracking algorithm for each BB. If a BB position deviates more than a threshold value than previous position and doesn't follow a trend then we can either discard that frame or discard that BB from fuzzy system to produce better results. Secondly, we can incorporate more that two cameras into the system. Then we can calculate the evidence distance (in this case BB position) for each camera and then discard or give lower weight to the BB which deviates from usual norm. If we incorporate these into the fuzzy system, then the system becomes more complex and losses the inherent advantage of a fuzzy system which is easy to interpret.

7.6 Conclusions

In this research we use a fuzzy system to calculate the confidence score of the multi-camera CNN object detector based on BB position and IOU overlap. First, we use a CNN based object detector to detect weed at multiple positions in-front of the multi-camera setup. Then we use projective transformation to project one camera's image plane to another cam-

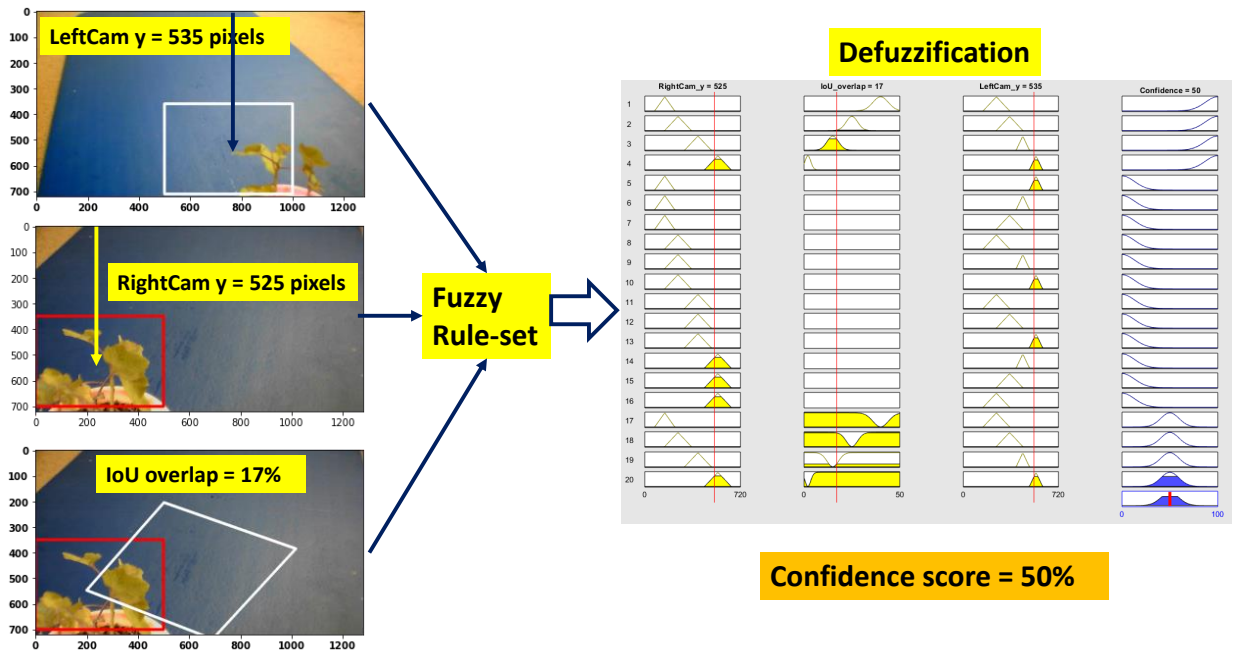


Figure 7.14. Fuzzy confidence score measurement (OK confidence).

era's image plane. Then we calculate the IOU overlap value at each different position of the weed for perfect detection. When we are detecting an object using multiple cameras, the position of the BB on the camera image plane may appear in different places based on the detection accuracy and the position of the cameras. But in 3D space, the object is at the exact same position for all cameras. As a result, there is a relationship between IOU overlap value of the BBes and the position of the BB on a camera image plane. If BB position or IOU overlap value deviate from ideal condition, that gives us information regarding less than perfect detection. We generate fuzzy rule-set using the relationship between BB position of each camera and IOU overlap value. We test our fuzzy system for three different scenarios which are ideal detection (High confidence), less than ideal detection (OK confidence) and wrong detection (Low confidence). The confidence score of our fuzzy system for three different scenarios proves our hypothesis regarding a relationship between IOU overlap and BB position and makes a more robust overall multi-camera detection system.

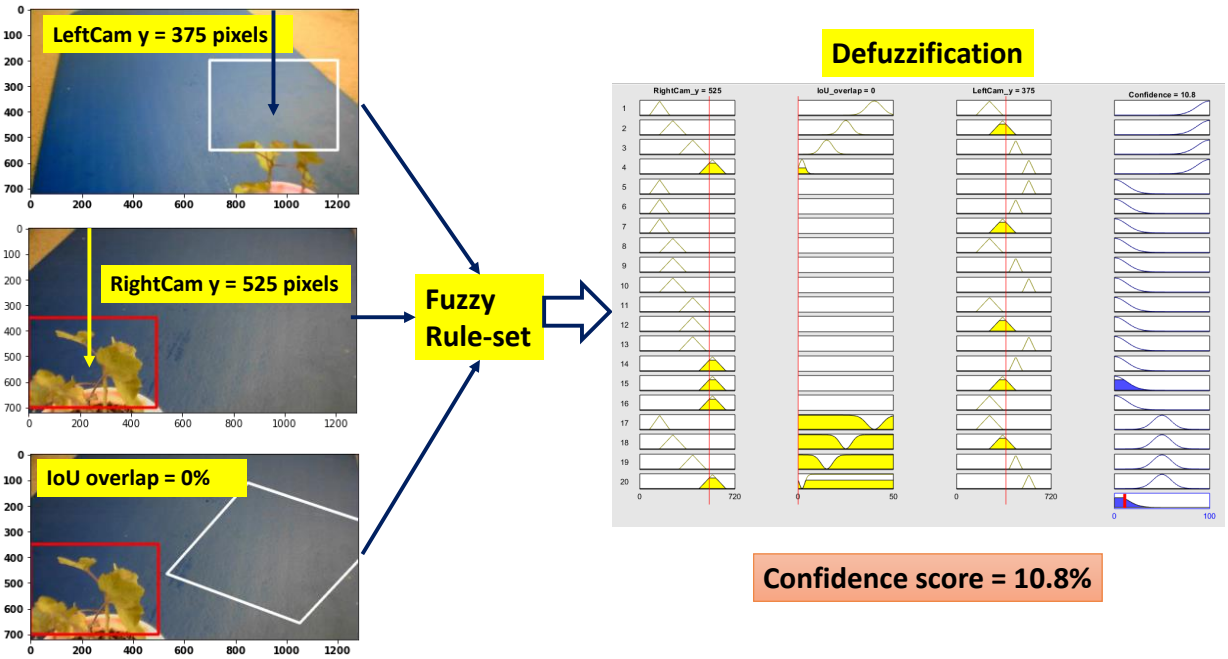


Figure 7.15. Fuzzy confidence score measurement (Low confidence).

8. CONCLUSIONS

1. In this research, we propose a complete methodology to convert an agricultural vehicle into an autonomous weeding robot. This robot should be able to detect crop rows for autonomous navigation through crop fields, classify multiple weeds using a multi-sensor array and send a robust, stable signal to the spray system for accurate herbicide spray.

2. We propose and apply an unsupervised machine learning (clustering) based approach for crop row detection. We incorporate classical computer vision technique to transform camera image plane for better crop row detection. We also incorporate geometric domain knowledge (crop row distance, size of weed and crop etc.) into our algorithm to reduce false positive detection of crop rows. The proposed algorithm is applicable for real-time usage on low spec hardware (without GPU) and shows good crop row detection accuracy at very challenging scenarios. CAROLIF has average 9.2 FPS inference on a dual-core cpu which is reasonable for real-time implementation in agricultural robotics. Compared to Hough transform (IOU 0.58), sliding window (IOU 0.61), cluster-least sq. (IOU 0.7), CAROLIF shows superior performance (IOU 0.73). Test on video data from an agricultural vehicle shows promising result for real-time application. We achieve 90.5% accuracy, 96.6% precision and 93.3% recall on a video containing very complex scenarios with shadow, intermittent crop growth and high weed pressure. In situations, when even naked eyes may fail to detect crop rows correctly, detection accuracy deteriorates. But due to incorporating geometric domain knowledge into algorithm, false positive detection of crop rows is low even in very challenging scenarios.

3. We use histogram-based image statistics for image quality determination. Color based image segmentation (segmenting green pixels from background) methods have reduced performance under extreme lighting (solar glaring or dark shadows). We calculate the quality of an image using the mean, variance, skew and kurtosis values of that image. Good quality image has mean at the center (relatively), higher variance and lower skew and lower kurtosis compared to bad quality image. If an image has bad quality criteria,

then we apply CLAHE to shift the pixel values towards good quality criteria. Test shows that, changing image quality helps to improve the performance of segmentation methods under extreme conditions. The processing time is fast (0.214 seconds) and suitable for real-time control application.

4. For real-time weed classification, we propose and implement a transfer learning based CNN method. We collect weed images from fields and greenhouse plants and create our own dataset. The dataset is important because this dataset contains weed images which captures real life scenarios, which is not always the case for published works in weed classification. We train three different architectures on the dataset. End-to-end CNN has the smallest inference time (0.064 seconds) but also the lowest test accuracy (81%). VGG16 with transfer learning has highest test accuracy (94%) and moderate inference time (0.266 seconds). Inception-ResNetV2 with transfer learning has moderate test accuracy (88%) and highest inference time (3.68 seconds). This study shows how different CNN architectures can affect the inference time and weed classification accuracy when trained on a small dataset. Test on video data shows that, CNN model is usually good at classifying weeds. But if there is a noise or foreign objects that may be present, classification accuracy deteriorates and sometimes become unstable. It shows the necessity of a decision level sensor fusion algorithm for robust and accurate weed classification.

5. For robust and stable weed classification, we propose a novel entropy function based space and time domain decision level sensor fusion algorithm. Our novel entropy function shows better result at capturing uncertainties under DS framework compared to Shannon and Deng entropy. In space domain fusion, our algorithm is able to fuse conflicting evidences from sensor array and overcomes the limitations of original DS fusion algorithm. Results show that, our algorithm has higher fusion evidence ($m(A) = 98.77\%$) compared to other fusion algorithms from literature (Dempster: $m(A) = 0\%$, Murphy: $m(A) = 96.2\%$, Deng: $m(A) = 98.2\%$, Han: $m(A) = 98.44\%$, Wang: $m(A) = 98.67\%$, Jiang: $m(A) = 98.37\%$). In space domain fusion, our algorithm shows better convergence rate.

6. In time domain fusion, the proposed algorithm is able to capture the dynamics of the system. It is also able to suppress noise and instabilities from the classification ID signal, which comes from the CNN. Results show that the proposed algorithm is better at capturing system dynamics when conflicting evidence is present ($m(A) = 98.7\%$) compared to other well established methods (Dempster: $m(A) = 0\%$, Song-1: $m(A) = 31.7\%$, Song-2: $m(A) = 50.3\%$, Chengkun: $m(A) = 70.8\%$). The only tuning parameter is number of time-steps (t_s) considered for time domain fusion. Lower t_s value shows faster response and higher t_s value shows better robustness.

7. Space and time domain fusion algorithm is applied to real-time output from CNN video feed. The algorithm is tested in two challenging scenarios: (1) when faulty sensor is present in sensor array and, (2) weed is partially occluded. Space domain fusion is able to catch the faulty sensor from sensor array and reduce it's effect on final classification result. Time domain fusion is able to compensate for performance deterioration introduced by partial occlusion. Final weed classification output from the fusion algorithm is correct, robust and stable in these challenging scenarios.

8. CNN based object detector will provide us the position of BBes of the detected objects. But how do we know for sure that the position of the BB is correct? To answer this question, we implement a fuzzy-fusion expert system. We use the predefined BB positions from multiple cameras and their IoU overlap values to generate fuzzy rules. Experimental test shows that, for perfect detection in both cameras, the fuzzy system gives us a confidence score of 88.6%. For partial detection in one of the cameras, the system gives a confidence score of 50%. For wrong detection in one of the cameras, the system gives a score of 10.8%. This confidence score will create an extra layer of information, which will help us to make the overall system more robust. Moreover, this confidence score can be used as a weight in time domain fusion for that specific frame when video data is processed. One limitation of this system is that, it always considers one of the sensor is correct. More sensors or tracking algorithm can be incorporated to mitigate this limitation, but will create a more complex set of fuzzy rules.

9. FUTURE WORKS

This research can further be enhanced by addressing the following ideas:

1. Agricultural navigation has four operations: field layout planning, route planning, vehicle path planning and vehicle auto guidance. Most of the navigation is done with fixed sensors on an AgBot. But fixed position of sensors introduce problems like vibration and limited maneuverability. Also fast storage and processing units for real-time application are costly. As a result, field data can be collected at first stage. Then at second stage, processing can be done offline. Finally application of fertilizers or chemicals can be done using the AgBots. To reduce fuel and operational costs, unmanned aerial vehicles (UAVs) can be used to solve some of these problems. A group of UAVs can be used to complete the field planning, route planning, and path planning. Then those GPS information can be uploaded to the big AgBot. Only sensors needed on the AgBot are for vehicle auto guidance and obstacle detection. Research is needed on UAV deployment, UAV and AgBot positioning alignment, and optimizing the number of sensors on AgBot.

2. For route and path planning, it is usually assumed that rows are traversed consecutively. It is also assumed that the field will be covered by one AgBot or a group of AgBots with same operating conditions [67]. Research is needed on optimizing AgBot operating conditions and field traversing conditions.

3. Crop rows are usually straight because it was easier to achieve with animals and simple machines. For better growth, equal nourishment is needed for crops. Optimal grid patterns suitable for field/terrain conditions can be achieved with smaller AgBots. Research is needed in optimal seeding pattern recognition.

4. For crop row detection, most of the published work use monocular camera. Few stereo camera work is also published. But at early crop growth stage, stereo data is not always reliable for crop height measurement. Incorporation of cheap 3D lidar can improve performance and robustness of crop row detection algorithms.

5. Diversity of crops and cropping systems are quite extensive. It makes sense that application targeted algorithms will perform better than generic algorithms. How user inputs can be incorporated with generic algorithms to make them target-specific is also an open question. As an example, CAROLIF can be tuned by changing the 'cluster size' parameter. But data is needed to make a connection between crop growth stage and the 'cluster size' parameter. Research work is needed regarding what user inputs can improve the performance of crop row detection algorithm. The generation of publicly available data sets with accompanying ground truth for crop lines would also help to evaluate and compare approaches. At the time of this writing, no video data with ground truth is available for row detection. Also, no research is conducted to evaluate how tracking algorithms can affect detection accuracy (due to lack of publicly available video data).

6. For crop/weed detection, accuracy of CNN may deteriorate due to occlusion, damaged leaves, varying lighting conditions, shadow, different growth stages of crops/weeds. An encoder-decoder based CNN can be used to learn the position of weeds on an image and another CNN trained to classify weeds can be applied to those specific pixels to improve classification accuracy.

7. A wide variety of research is trying to retrofit current conventional agricultural vehicle with automation technologies. Proper research is needed towards the optimized shape, size and design of the AgBot. Topology of the AgBot can be optimized (in terms of maximum load carrying capacity, range of vehicle, sensor placement etc.) because cabin space is not needed. Can small, lightweight robots be a viable option to replace large, heavy machines, primarily to reduce soil compaction is an open question.

8. "As much as USD 319,864 for an 850 hector farm is required for investment in intelligent machines to achieve maximum break-even point. Fortunately, farming robots would bring profitable business to farmers because robots can reduce 20% of the scouting costs for cereals, 12% for sugar beet weeding, and 24% for inter-row weeding [150]." More comprehensive research is needed in this field to convince the farmers about the cost benefits of using robots.

REFERENCES

[1] U. G. Assembly, "Food production must double by 2050 to meet demand from world's growing population," *Press Release, October*, vol. 9, p. 2009, 2009.

[2] N. Gilbert, "A hard look at gm crops," *Nature*, vol. 497, no. 7447, p. 24, 2013.

[3] I. Heap, "The international survey of herbicide resistant weeds," *http://www. weedscience. com*, 2009.

[4] D. R. Hall, "A rapidly deployable approach for automated visual weed classification without prior species knowledge," Ph.D. dissertation, Queensland University of Technology, 2018.

[5] P. Kudsk and J. Streibig, "Herbicides–a two-edged sword," *Weed Research*, vol. 43, no. 2, pp. 90–102, 2003.

[6] I. Graham-Bryce, "Crop protection: A consideration of the effectiveness and disadvantages of current methods and of the scope for improvement," *Philosophical Transactions of the Royal Society of London. B, Biological Sciences*, vol. 281, no. 980, pp. 163–179, 1977.

[7] I. Rutherford, "Characteristics of boom and nozzle spraying—a robust, safe and efficient system for the future?" *Application and Biology*, pp. 5–9, 1985.

[8] I. Heap, "Global perspective of herbicide-resistant weeds," *Pest management science*, vol. 70, no. 9, pp. 1306–1315, 2014.

[9] A. Wang, W. Zhang, and X. Wei, "A review on weed detection using ground-based machine vision and image processing techniques," *Computers and electronics in agriculture*, vol. 158, pp. 226–240, 2019.

[10] M. Rodrigo, N. Oturan, and M. A. Oturan, "Electrochemically assisted remediation of pesticides in soils and water: A review," *Chemical reviews*, vol. 114, no. 17, pp. 8720–8745, 2014.

[11] E. Marshall, "Field-scale estimates of grass weed populations in arable land," *Weed Research*, vol. 28, no. 3, pp. 191–198, 1988.

[12] G. A. Johnson, D. Mortensen, and A. Martin, "A simulation of herbicide use based on weed spatial distribution," *Weed Research*, vol. 35, no. 3, pp. 197–205, 1995.

[13] L. Tian, J. F. Reid, and J. W. Hummel, "Development of a precision sprayer for site-specific weed management," *Transactions of the ASAE*, vol. 42, no. 4, p. 893, 1999.

[14] C. R. Medlin and D. R. Shaw, "Economic comparison of broadcast and site-specific herbicide applications in nontransgenic and glyphosate-tolerant glycine max," *Weed Science*, vol. 48, no. 5, pp. 653–661, 2000.

[15] M. Basso and E. P. de Freitas, "A uav guidance system using crop row detection and line follower algorithms," *Journal of Intelligent & Robotic Systems*, pp. 1–17, 2019.

[16] P. V. Hough, *Method and means for recognizing complex patterns*, US Patent 3,069,654, Dec. 1962.

[17] X. Zhang, X. Li, B. Zhang, J. Zhou, G. Tian, Y. Xiong, and B. Gu, "Automated robust crop-row detection in maize fields based on position clustering algorithm and shortest path method," *Computers and electronics in agriculture*, vol. 154, pp. 165–175, 2018.

[18] N. Sainz-Costa, A. Ribeiro, X. P. Burgos-Artizzu, M. Guijarro, and G. Pajares, "Mapping wide row crops with video sequences acquired from a tractor moving at treatment speed," *Sensors*, vol. 11, no. 7, pp. 7095–7109, 2011.

[19] I. Vidović, R. Cupec, and Ž. Hocenski, "Crop row detection by global energy minimization," *Pattern Recognition*, vol. 55, pp. 68–86, 2016.

[20] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey'acm computing surveys (csur)," 2006.

[21] M. Marron, J. Garcia, M. Sotelo, M. Cabello, D. Pizarro, F. Huerta, and J. Cerro, "Comparing a kalman filter and a particle filter in a multiple objects tracking application," in *2007 IEEE International Symposium on Intelligent Signal Processing*, IEEE, 2007, pp. 1–6.

[22] J. Geipel, J. Link, and W. Claupein, "Combined spectral and spatial modeling of corn yield based on aerial images and crop surface models acquired with an unmanned aircraft system," *Remote Sensing*, vol. 6, no. 11, pp. 10 335–10 355, 2014.

[23] M. Herrero-Huerta, D. González-Aguilera, P. Rodriguez-Gonzalvez, and D. Hernández-López, "Vineyard yield estimation by automatic 3d bunch modelling in field conditions," *Computers and electronics in agriculture*, vol. 110, pp. 17–26, 2015.

[24] J. Moonrinta, S. Chaivivatrakul, M. N. Dailey, and M. Ekpanyapong, "Fruit detection, tracking, and 3d reconstruction for crop mapping and yield estimation," in *2010 11th*

International Conference on Control Automation Robotics & Vision, IEEE, 2010, pp. 1181–1186.

[25] Q. Wang, S. Nuske, M. Bergerman, and S. Singh, "Automated crop yield estimation for apple orchards," in *Experimental robotics*, Springer, 2013, pp. 745–758.

[26] M. Vázquez-Arellano, H. Griepentrog, D. Reiser, and D. Paraforos, "3-d imaging systems for agricultural applications—a review," *Sensors*, vol. 16, no. 5, p. 618, 2016.

[27] A. Piron, V. Leemans, F. Lebeau, and M.-F. Destain, "Improving in-row weed detection in multispectral stereoscopic images," *Computers and electronics in agriculture*, vol. 69, no. 1, pp. 73–79, 2009.

[28] D. Seatovic, H. Kutterer, T. Anken, and M. Holpp, "Automatic weed detection in grassland," in *67th Conference Agricultural Engineering, Land-Technik-AgEng 2009, Hannover, Germany, 06-07 November 2009*, VDI Verlag, 2009, pp. 187–192.

[29] W. Strothmann, "Multi-wavelength laser line profile sensing for agricultural applications.," Ph.D. dissertation, University of Osnabrück, Germany, 2016.

[30] U. Weiss and P. Biber, "Plant detection and mapping for agricultural robots using a 3d lidar sensor," *Robotics and autonomous systems*, vol. 59, no. 5, pp. 265–273, 2011.

[31] W. Saeys, B. Lenaerts, G. Craessaerts, and J. De Baerdemaeker, "Estimation of the crop density of small grains using lidar sensors," *Biosystems Engineering*, vol. 102, no. 1, pp. 22–30, 2009.

[32] A. D. Nakarmi and L. Tang, "Inter-plant spacing sensing at early growth stages using a time-of-flight of light based 3d vision sensor," in *2010 Pittsburgh, Pennsylvania, June 20-June 23, 2010*, American Society of Agricultural and Biological Engineers, 2010, p. 1.

[33] B. Adhikari and M. Karkee, "3d reconstruction of apple trees for mechanical pruning," in *2011 Louisville, Kentucky, August 7-10, 2011*, American Society of Agricultural and Biological Engineers, 2011, p. 1.

[34] G. L. Grinblat, L. C. Uzal, M. G. Larese, and P. M. Granitto, "Deep learning for plant identification using vein morphological patterns," *Computers and Electronics in Agriculture*, vol. 127, pp. 418–424, 2016.

[35] C. Potena, D. Nardi, and A. Pretto, "Fast and accurate crop and weed identification with summarized train sets for precision agriculture," in *International Conference on Intelligent Autonomous Systems*, Springer, 2016, pp. 105–121.

[36] J. Yu, S. M. Sharpe, A. W. Schumann, and N. S. Boyd, "Deep learning for image-based weed detection in turfgrass," *European journal of agronomy*, vol. 104, pp. 78–84, 2019.

[37] H. K. Suh, J. Ijsselmuiden, J. W. Hofstee, and E. J. van Henten, "Transfer learning for the classification of sugar beet and volunteer potato under field conditions," *Biosystems Engineering*, vol. 174, pp. 50–65, 2018.

[38] A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. C. Wood, J. Johns, W. Banks, B. Girgenti, O. Kenny, J. Whinney, *et al.*, "Deepweeds: A multiclass weed species image dataset for deep learning," *Scientific reports*, vol. 9, no. 1, pp. 1–12, 2019.

[39] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*, IEEE, 2016, pp. 1–6.

[40] W.-S. Jeon and S.-Y. Rhee, "Plant leaf recognition using a convolution neural network," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 17, no. 1, pp. 26–34, 2017.

[41] S. G. Wu, F. S. Bao, E. Y. Xu, Y.-X. Wang, Y.-F. Chang, and Q.-L. Xiang, "A leaf recognition algorithm for plant classification using probabilistic neural network," in *2007 IEEE international symposium on signal processing and information technology*, IEEE, 2007, pp. 11–16.

[42] A. Kaya, A. S. Keceli, C. Catal, H. Y. Yalic, H. Temucin, and B. Tekinerdogan, "Analysis of transfer learning for deep neural network based plant classification models," *Computers and electronics in agriculture*, vol. 158, pp. 20–29, 2019.

[43] O. Söderkvist, *Computer vision classification of leaves from swedish trees*, 2001.

[44] P. F. Silva, A. R. Marcal, and R. M. A. da Silva, "Evaluation of features for leaf discrimination," in *International Conference Image Analysis and Recognition*, Springer, 2013, pp. 197–204.

[45] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[46] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[47] Y. Ban, H. Hu, and I. M. Rangel, "Fusion of quickbird ms and radarsat sar data for urban land-cover mapping: Object-based and knowledge-based approach," *International Journal of Remote Sensing*, vol. 31, no. 6, pp. 1391–1410, 2010.

[48] W. Jiang, Y. Luo, X.-Y. Qin, and J. Zhan, "An improved method to rank generalized fuzzy numbers with different left heights and right heights," *Journal of Intelligent & Fuzzy Systems*, vol. 28, no. 5, pp. 2343–2355, 2015.

[49] E. Bossé and J. Roy, "Fusion of identity declarations from dissimilar sources using the dempster-shafer theory," *Optical Engineering*, vol. 36, 1997.

[50] K. Coombs, D. Freel, D. Lampert, and S. J. Brahm, "Using dempster-shafer methods for object classification in the theater ballistic missile environment," in *Sensor Fusion: Architectures, Algorithms, and Applications III*, International Society for Optics and Photonics, vol. 3719, 1999, pp. 103–113.

[51] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," in *Classic works of the Dempster-Shafer theory of belief functions*, Springer, 2008, pp. 57–72.

[52] G. Shafer, *A mathematical theory of evidence*. Princeton university press, 1976, vol. 42.

[53] F. Xiao, "Multi-sensor data fusion based on the belief divergence measure of evidences and the belief entropy," *Information Fusion*, vol. 46, pp. 23–32, 2019.

[54] P. Smets, "Data fusion in the transferable belief model," in *Proceedings of the third international conference on information fusion*, IEEE, vol. 1, 2000, PS21–PS33.

[55] R. R. Yager, "On the dempster-shafer framework and new combination rules," *Information sciences*, vol. 41, no. 2, pp. 93–137, 1987.

[56] W. Jiang, M. Zhuang, X. Qin, and Y. Tang, "Conflicting evidence combination based on uncertainty measure and distance of evidence," *SpringerPlus*, vol. 5, no. 1, p. 1217, 2016.

[57] L. Hong and A. Lynch, "Recursive temporal-spatial information fusion with applications to target identification," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 29, no. 2, pp. 435–445, 1993.

[58] F. Ye, J. Chen, and Y. Tian, "A robust ds combination method based on evidence correction and conflict redistribution," *Journal of Sensors*, vol. 2018, 2018.

[59] Y. Song, X. Wang, and L. Lei, "Combination of temporal evidence sources based on intuitionistic fuzzy sets," *Acta Automatica Sinica*, vol. 42, no. 9, pp. 1322–1338, 2016.

[60] Y. Song, X. Wang, L. Lei, and Y. Xing, "Credibility decay model in temporal evidence combination," *Information Processing Letters*, vol. 115, no. 2, pp. 248–252, 2015.

[61] L. Chengkun, C. Yunxiang, X. Huachun, W. Weijia, and W. Zezhou, "Evidence combination method in time domain based on reliability and importance," *Journal of Systems Engineering and Electronics*, vol. 29, no. 6, pp. 1308–1316, 2018.

[62] B. Åstrand and A.-J. Baerveldt, "An agricultural mobile robot with vision-based perception for mechanical weed control," *Autonomous robots*, vol. 13, no. 1, pp. 21–35, 2002.

[63] Y.-K. Choi and S.-J. Lee, "Development of advanced sonar sensor model using data reliability and map evaluation method for grid map building," *Journal of Mechanical Science and Technology*, vol. 29, no. 2, pp. 485–491, 2015.

[64] N. Shalal, T. Low, C. McCarthy, and N. Hancock, "A preliminary evaluation of vision and laser sensing for tree trunk detection and orchard mapping," in *Proceedings of the Australasian Conference on Robotics and Automation (ACRA 2013)*, Australasian Robotics and Automation Association, 2013, pp. 1–10.

[65] M. C. Garcia-Alegre, D. Martin, D. M. Guinea, and D. Guinea, "Real-time fusion of visual images and laser data images for safe navigation in outdoor environments," *Sensor Fusion-Foundation and Applications, Published*, 2011.

[66] J. Gai, L. Tang, and B. Steward, "Plant localization and discrimination using 2d+ 3d computer vision for robotic intra-row weed control," in *2016 ASABE Annual International Meeting*, American Society of Agricultural and Biological Engineers, 2016, p. 1.

[67] S. G. Vougioukas, "Agricultural robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 365–392, 2019.

[68] W. Zhiqiang and L. Jun, "A review of object detection based on convolutional neural network," in *2017 36th Chinese Control Conference (CCC)*, IEEE, 2017, pp. 11 104–11 109.

[69] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[70] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[71] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[72] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[73] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[74] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[75] P. Roy, S. Ghosh, S. Bhattacharya, and U. Pal, "Effects of degradations on deep neural network architectures," *arXiv preprint arXiv:1807.10108*, 2018.

[76] H. Hosseini, B. Xiao, M. Jaiswal, and R. Poovendran, "On the limitation of convolutional neural networks in recognizing negative images," in *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, IEEE, 2017, pp. 352–358.

[77] T. J. Ross *et al.*, *Fuzzy logic with engineering applications*. Wiley Online Library, 2004, vol. 2.

[78] W. Chen, T. Xu, J. Liu, M. Wang, and D. Zhao, "Picking robot visual servo control based on modified fuzzy neural network sliding mode algorithms," *Electronics*, vol. 8, no. 6, p. 605, 2019.

[79] J. Romeo, G. Pajares, M. Montalvo, J. M. Guerrero, M. Guijarro, and J. M. de la Cruz, "A new expert system for greenness identification in agricultural images," *Expert Systems with Applications*, vol. 40, no. 6, pp. 2275–2286, 2013.

[80] G. E. Meyer, J. C. Neto, D. D. Jones, and T. W. Hindman, "Intensified fuzzy clusters for classifying plant, soil, and residue regions of interest from color images," *Computers and electronics in agriculture*, vol. 42, no. 3, pp. 161–180, 2004.

[81] M. Sujaritha, S. Annadurai, J. Satheeshkumar, S. K. Sharan, and L. Mahesh, "Weed detecting robot in sugarcane fields using fuzzy real time classifier," *Computers and electronics in agriculture*, vol. 134, pp. 160–171, 2017.

[82] E. M. López, M. García, M. Schuhmacher, and J. L. Domingo, "A fuzzy expert system for soil characterization," *Environment international*, vol. 34, no. 7, pp. 950–958, 2008.

[83] E. I. Papageorgiou, A. T. Markinos, and T. A. Gemtos, "Fuzzy cognitive map based approach for predicting yield in cotton crop production as a basis for decision support

system in precision agriculture application," *Applied Soft Computing*, vol. 11, no. 4, pp. 3643–3657, 2011.

[84] S. Kolhe, R. Kamal, H. S. Saini, and G. K. Gupta, "An intelligent multimedia interface for fuzzy-logic based inference in crops," *Expert Systems with Applications*, vol. 38, no. 12, pp. 14 592–14 601, 2011.

[85] R. E. Kalman, "A new approach to linear filtering and prediction problems," 1960.

[86] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[87] C. Gée, J. Bossu, G. Jones, and F. Truchetet, "Crop/weed discrimination in perspective agronomic images," *Computers and Electronics in Agriculture*, vol. 60, no. 1, pp. 49–59, 2008.

[88] A. Ribeiro, C. Fernández-Quintanilla, J. Barroso, M. García-Alegre, J. Stafford, *et al.*, "Development of an image analysis system for estimation of weed pressure," *Precision agriculture*, vol. 5, pp. 169–174, 2005.

[89] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," Stanford, Tech. Rep., 2006.

[90] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[91] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[92] L. McInnes, J. Healy, and S. Astels, "Hdbscan: Hierarchical density based clustering," *The Journal of Open Source Software*, vol. 2, no. 11, Mar. 2017. DOI: 10.21105/joss.00205. [Online]. Available: https://doi.org/10.211052Fjoss.00205.

[93] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, 1996, pp. 226–231.

[94] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[95] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[96] I. D. García-Santillán, M. Montalvo, J. M. Guerrero, and G. Pajares, "Automatic detection of curved and straight crop rows from images in maize fields," *Biosystems Engineering*, vol. 156, pp. 61–79, 2017.

[97] S. L. N. Rafael Padilla and E. A. B. da Silva, "Survey on performance metrics for object-detection algorithms," International Conference on Systems, Signals and Image Processing (IWSSIP), 2020.

[98] E. Hamuda, B. Mc Ginley, M. Glavin, and E. Jones, "Improved image processing-based crop detection using kalman filtering and the hungarian algorithm," *Computers and Electronics in Agriculture*, vol. 148, pp. 37–44, 2018.

[99] S. S. Pathan, A. Al-Hamadi, and B. Michaelis, "Intelligent feature-guided multi-object tracking using kalman filter," in *2009 2nd International Conference on Computer, Control and Communication*, IEEE, 2009, pp. 1–6.

[100] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[101] A. Rosebrock, *Deep Learning for Computer Vision with Python: Starter Bundle*. PyImageSearch, 2017.

[102] F. Chollet, *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek*. MITP-Verlags GmbH & Co. KG, 2018.

[103] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[104] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[105] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[106] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. Hughes, "Using transfer learning for image-based cassava disease detection," *arXiv preprint arXiv:1707.03717*, 2017.

[107] F. Chollet, *Keras*, https://github.com/fchollet/keras, 2015.

[108] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[109] M. N. Khan and S. Anwar, "Robust weed recognition through color based image segmentation and convolution neural network based classification," in *ASME International Mechanical Engineering Congress and Exposition*, American Society of Mechanical Engineers, vol. 59414, 2019, V004T05A045.

[110] J. Camargo Neto, *A combined statistical-soft computing approach for classification and mapping weed species in minimum-tillage systems*. The University of Nebraska-Lincoln, 2004.

[111] T. Kataoka, T. Kaneko, H. Okamoto, and S. Hata, "Crop growth estimation system using machine vision," in *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, IEEE, vol. 2, 2003, b1079–b1083.

[112] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[113] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital image processing using MATLAB*. Pearson Education India, 2004.

[114] Y. Chang, C. Jung, P. Ke, H. Song, and J. Hwang, "Automatic contrast-limited adaptive histogram equalization with dual gamma correction," *IEEE Access*, vol. 6, pp. 11 782–11 792, 2018.

[115] M. N. Khan and S. Anwar, "Time-domain data fusion using weighted evidence and dempster–shafer combination rule: Application in object classification," *Sensors*, vol. 19, no. 23, p. 5187, 2019.

[116] M. N. Khan and S. Anwar, "Paradox elimination in dempster–shafer combination rule with novel entropy function: Application in decision-level multi-sensor fusion," *Sensors*, vol. 19, no. 21, p. 4810, 2019.

[117] R. R. Yager and L. Liu, *Classic works of the Dempster-Shafer theory of belief functions*. Springer, 2008, vol. 219.

[118] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," in *Classic works of the Dempster-Shafer theory of belief functions*, Springer, 2008, pp. 57–72.

[119] Y. D. e. Deng, "A generalized shannon entropy to measure uncertainty," *viXra 1502.0222*, vol. 2015,

[120] K. Guo and W. Li, "Combination rule of d–s evidence theory based on the strategy of cross merging between evidences," *Expert Systems with Applications*, vol. 2011, pp. 1360–1336,

[121] P. Smets, "Data fusion in the transferable belief model," *Information Fusion, IEEE*, vol. 2000,

[122] R. R. Yager, "On the dempster-shafer framework and new combination rules," *Information sciences*, vol. 1987, pp. 93–137,

[123] L. Bicheng, H. Jie, and Y. Hujun, "Two efficient combination rules for conflicting belief functions," *Artificial Intelligence and Computational Intelligence, IEEE*, vol. 2009,

[124] T. Inagaki, *Interdependence between safety-control policy and multiple-sensor schemes via Dempster-Shafer theorys*. John Wiley and Sons, Inc, 1994.

[125] L. R. Zhang, "Independence, and combination of evidence in the dempster-shafer theory," *IEEE Transactions on Reliability*, vol. 1991, pp. 182–188,

[126] Y. Li, J. Chen, and L. Feng, "Dealing with uncertainty: A survey of theories and practices," *IEEE Transactions on Knowledge and Data Engineering*, vol. 2013, pp. 449–460,

[127] S. Chen, Y. Deng, and J. Wu, "Fuzzy sensor fusion based on evidence theory and its application," *Applied Artificial Intelligence*, vol. 2013, pp. 235–248,

[128] S. Sun, J. Gao, M. Chen, B. Xu, and Z. Ding, "Fs-ds based multi-sensor data fusion.," *JSW*, vol. 8, no. 5, pp. 1157–1161, 2013.

[129] W. Jiang, B. Wei, C. Xie, and D. Zhou, "Evidential sensor fusion method in fault diagnosis," *Advances in Mechanical Engineering*, vol. 2016,

[130] R. R. Murphy, "Dempster-shafer theory for sensor fusion in autonomous mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 1998, pp. 197–206,

[131] W. Jiang, M. Zhuang, X. Qin, and Y. Tang, "Conflicting evidence combination based on uncertainty measure and distance of evidence," *SpringerPlus*, vol. 5, no. 1, p. 1217, 2016.

[132] F. Xiao, "An improved method for combining conflicting evidences based on the similarity measure and belief function entropy," *International Journal of Fuzzy Systems*, vol. 2018, pp. 1256–1266,

[133] Y. Lin, C. Wang, C. Ma, Z. Dou, and X. Ma, "A new combination method for multisensor conflict information," *The Journal of Supercomputing*, vol. 72, no. 7, pp. 2874–2890, 2016.

[134] F. Ye, J. Chen, and A. Y. Tian, "Robust ds combination method based on evidence correction and conflict redistribution," *Journal of Sensors*, vol. 2018,

[135] H. Durrant-Whyte and T. C. Henderson, "Multisensor data fusion," in handbook of robotics **2008**, pp. 585–610.

[136] C. E. Shannon, "A mathematical theory of communication," *Bell system technical journal*, vol. 1948, pp. 379–423,

[137] L. Pan and Y. Deng, "A new belief entropy to measure uncertainty of basic probability assignments based on belief function and plausibility function," *Entropy*, vol. 20, no. 11, p. 842, 2018.

[138] A.-L. Jousselme and D. a. Grenier, "A new distance between two bodies of evidence," *Information fusion*, vol. 2001, pp. 91–101,

[139] J. Chen, F. Ye, and T. Jiang, ", numerical analyses of modified ds combination methods based on different distance functions," *Progress in Electromagnetics Research Symposium-Fall, IEEE*, vol. 2017, pp. 2169–2175,

[140] D. Yong, S. WenKang, Z. ZhenFu, and L. Qi, "Combining belief functions based on distance of evidence," *Decision support systems*, vol. 38, no. 3, pp. 489–493, 2004.

[141] D.-Q. Han, Y. Deng, C.-Z. Han, and Z. Hou, "Weighted evidence combination based on distance of evidence and uncertainty measure," *J. Infrared Millim. Waves*, vol. 30, no. 5, pp. 396–400, 2011.

[142] J. Wang, F. Xiao, X. Deng, L. Fei, and Y. Deng, "Weighted evidence combination based on distance of evidence and entropy function," *International journal of distributed sensor networks*, vol. 12, no. 7, p. 3 218 784, 2016.

[143] M. N. Khan and S. Anwar, "Paradox elimination in dempster–shafer combination rule with novel entropy function: Application in decision-level multi-sensor fusion," *Sensors*, vol. 19, no. 21, p. 4810, 2019.

[144] A. Kaehler and G. Bradski, *Learning OpenCV 3: computer vision in C++ with the OpenCV library.* " O'Reilly Media, Inc.", 2016.

[145] K. Inui, S. Kaneko, and S. Igarashi, "Robust line fitting using lmeds clustering," *Systems and Computers in Japan*, vol. 34, no. 14, pp. 92–100, 2003.

[146] H. Bazargani, O. Bilaniuk, and R. Laganiere, "A fast and robust homography scheme for real-time planar target detection," *Journal of Real-Time Image Processing*, vol. 15, no. 4, pp. 739–758, 2018.

[147] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[148] L. A. Zadeh, "Fuzzy sets," in *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, World Scientific, 1996, pp. 394–432.

[149] E. H. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," in *Proceedings of the institution of electrical engineers*, IET, vol. 121, 1974, pp. 1585–1588.

[150] K. G. Fue, W. M. Porter, E. M. Barnes, and G. C. Rains, "An extensive review of mobile agricultural robotics for field operations: Focus on cotton harvesting," *AgriEngineering*, vol. 2, no. 1, pp. 150–174, 2020.

A. CODE: CHAPTER 3 - CROP ROW DETECTION

This code shows the crop row detection process described in Chaper 3. More can be found at https://github.com/mdkhan48.

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import matplotlib.gridspec as gridspec
import numpy.polynomial.polynomial as poly
import time
import hdbscan
import pandas as pd
from sklearn import linear_model
import pickle
import math


def combined (img):
    #b, g, r = cv2.split(img)
    b = img[:, :, 0]
    g = img[:, :, 1]
    r = img[:, :, 2]
    r_max = np.amax(r)
    g_max = np.amax(g)
    b_max = np.amax(b)

    red_norm = r/r_max
    green_norm = g/g_max
    blue_norm = b/b_max

    norm = red_norm + blue_norm + green_norm

    small_num = 0.0001
    r = red_norm/(norm+small_num)
    g = green_norm/(norm+small_num)
    b = blue_norm/(norm+small_num)

    ExG = 2*g - r - b #excess green

    return ExG

origimg = cv2.imread(r'image location')
plt.imshow(origimg)
plt.show()
```

```
height = origimg.shape[0]
width = origimg.shape[1]
crop_img = origimg[int(height / 5):int(height / 2), int(width / 3):int(400+(width / 3))]
cropheight = img.shape[0]
cropwidth = img.shape[1]

max_value = np.max(combined(img))
min_value = np.min(combined(img))

new_min = 0
new_max = 255
old_range = max_value - min_value
new_range = new_max - new_min
lin_map = (((combined(img).astype(np.float64) - min_value) * new_range) / old_range) + n
image_map = lin_map.astype(np.uint8)
thresh_val,thresh_img = cv2.threshold(image_map,0,255,cv2.THRESH_OTSU)

kernel = np.ones((3,3),np.uint8)
opening = cv2.morphologyEx(thresh_img, cv2.MORPH_OPEN, kernel)
closing = cv2.morphologyEx(opening, cv2.MORPH_CLOSE, kernel)

indices = np.where(closing == [255])
coordinates = list(zip(indices[0], indices[1]))
data = np.array(coordinates)
y_x_data = np.flip(data)
cluster_size = 500
clusters = hdbscan.HDBSCAN(min_cluster_size=cluster_size).fit(y_x_data)
labels = clusters.labels_
n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
print("clusters:", n_clusters)
good_points = clusters.exemplars_ #getting GOOD points using .exempler_ method
subtract_data = y_x_data

iteration = 0
least_clusters_amount_in_ROI = 3
percent_of_data_to_keep = 0.85
cluster_num_threshold = 2
iteration_break = 3

while (n_clusters < least_clusters_amount_in_ROI): #assuming there will be 3 or 4 cluste

    clusters = hdbscan.HDBSCAN(min_cluster_size= cluster_size).fit(subtract_data)
    labels = clusters.labels_
    n_clusters = len(set(labels)) - (1 if -1 in labels else 0)
```

```
    if (n_clusters > cluster_num_threshold) or (iteration > iteration_break):
        break

    threshold = pd.Series(clusters.outlier_scores_).quantile(percent_of_data_to_keep)
    outliers = np.where(clusters.outlier_scores_ > threshold)[0]
    subtract_data = np.delete(subtract_data,outliers,axis=0)
    iteration = iteration+1
    print("iteration", iteration)
    labels = clusters.labels_
    n_clusters = len(set(labels)) - (1 if -1 in labels else 0)

Xdata_points = subtract_data.T[1]

data_points = [] #store number of (x,y) of a cluster on a list to see if there is big cl
for i in range(0,n_clusters):
    data_points.append(Xdata_points[labels == i].shape[0])

good_data = np.empty((0,2))
for i in range(0,n_clusters):
    good_data = np.append(good_data,good_points[i],axis = 0)

cluster_data_threshold = 5000
data_points = np.sort(data_points)[::-1]

if (data_points[0] > cluster_data_threshold):

    good_clusters = hdbscan.HDBSCAN(min_cluster_size=100).fit(good_data)
    labels_good = good_clusters.labels_
    n_good_clusters = len(set(labels_good)) - (1 if -1 in labels_good else 0)
    Ydata_g = good_data.T[0] #we plot this on x-axis
    Xdata_g = good_data.T[1] #we plot this on y_axis
    for j in range(0,n_good_clusters):
        plt.scatter(np.array(Ydata_g[labels_good == j]), np.array(Xdata_g[labels_good ==

    subtract_data = good_data
    n_clusters = n_good_clusters
    labels = labels_good
subtract_data = np.array(subtract_data)

Ydata = subtract_data.T[0] #we plot this on x-axis
Xdata = subtract_data.T[1] #we plot this on y_axis

y_center = []
x_center = []
```

```
for i in range(0,n_clusters):
    y_center.append(int(np.mean(np.array(Ydata[labels == i]))))
    x_center.append(int(np.mean(np.array(Xdata[labels == i]))))

    plt.scatter(np.array(Ydata[labels == i]), np.array(Xdata[labels == i]))
    plt.plot(y_center[i],x_center[i], "kv",markersize=12)
    plt.text(y_center[i] + 5,x_center[i], s = i+1, fontsize =10)
    print("cluster ",i+1,"center ", y_center[i])
cluster_distance_threshold = 50
center_distance = 0
center_sort = np.sort(y_center)
percentage = 0.3

keep = 10
delete = 10

deleted_cluster = []

for i in range(0,n_clusters-1):
    for j in range(i+1,n_clusters):

        cluster_distance =  np.abs(y_center[i] - y_center[j])

        if (cluster_distance < (cluster_distance_threshold - percentage * cluster_distan
            cluster_i_points = Xdata[labels == i].shape[0]
            cluster_j_points = Xdata[labels == j].shape[0]
            cluster_i_height = np.abs(max(Xdata[labels == i]) - min(Xdata[labels == i]))
            cluster_j_height = np.abs(max(Xdata[labels == j]) - min(Xdata[labels == j]))

            if ((cluster_i_height >= cluster_j_height) & (cluster_i_points >= cluster_j_
                keep = i
                delete = j
            else:
                keep = j
                delete = i
            print("delete",delete+1)
            deleted_cluster.append(delete)

straight = 1
curve = 2
ransac = linear_model.RANSACRegressor()

color = ['red', 'blue', 'cyan', 'magenta', 'yellow','black','green','red', 'blue', 'cyan
vertical_ROI_height = 290
total_points = 1000
```

```
for i in range(0,n_clusters):
    if (any(i == item for item in deleted_cluster)):
        pass
    else:

        ransac.fit(Xdata[labels == i].reshape(-1,1), Ydata[labels == i].reshape(-
1,1))
        xnew_cluster = np.linspace(0, vertical_ROI_height, 1000)
        y1 = xnew_cluster[0]
        y2 = xnew_cluster[999]
        ffit_cluster = ransac.predict(xnew_cluster.reshape(-1,1))
        x1 = ffit_cluster[0]
        x2 = ffit_cluster[999]
```

B. CODE: CHAPTER 5 - STATISTICS PARAMETERS OF IMAGE

This code shows how the mean, variance, skew and kurtosis is calculated from an image histogram.

```
import numpy as np
import matplotlib.pyplot as plt
import cv2

img = cv2.imread('image location')
rows, cols, _ = img.shape


def plot_histogram(img,title):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    plt.figure()
    plt.title(title)
    plt.xlabel("Gray level, a")
    plt.ylabel("# of Pixels with a, n(a)")
    plt.hist(gray.ravel(), 256, [0, 256])
    plt.show()

gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
hist_gray = cv2.calcHist([gray],[0],None,[256],[0,256])
hist_blue = cv2.calcHist([img],[0],None,[256],[0,256])
hist_green = cv2.calcHist([img],[1],None,[256],[0,256])
hist_red = cv2.calcHist([img],[2],None,[256],[0,256])

probability_pa = hist_gray/(rows*cols)
a = np.arange(256)

m = summation (a . pa)
m = np.sum(a*probability_pa)
print("mean", m)

#1st moment
mu_1 = np.sum((a - m) * probability_pa)
print("1st moment mu1",mu_1)

#2nd moment
mu_2 = np.sum(np.power((a - m),2) * probability_pa)
print("2nd moment mu2 (variance)",mu_2)
```

```
#3rd moment
mu_3 = np.sum(np.power((a - m),3) * probability_pa)

#4th moment
mu_4 = np.sum(np.power((a - m),4) * probability_pa)

skewness = mu_3/(np.power(mu_2,1.5))
print("skewness", skewness)
print("abs. skewness", np.absolute(skewness))

kurtosis = mu_4/(np.power(mu_2,2))
print("Kurtosis", kurtosis)
```

C. CODE: CHAPTER 6 - MULTI-SENSOR TIME DOMAIN FUSION

This code shows how time domain fusion is done using information from multiple sensors. This is the time domain fusion step of Fig. 6.11. Matlab 2018b is used.

```
left_camera_weed_data = readtable('load camera data');
center_camera_weed_data = readtable('load camera data');
right_camera_weed_data_faulty = readtable('load camera data');

time_step = left_camera_weed_data(:,1);

time = table2array(time_step);

left_pigweed = table2array(left_camera_weed_data(:,2));
left_ragweed = table2array(left_camera_weed_data(:,3));

center_pigweed = table2array(center_camera_weed_data(:,2));
center_ragweed = table2array(center_camera_weed_data(:,3));

right_pigweed = table2array(right_camera_weed_data_faulty(:,2));
right_ragweed = table2array(right_camera_weed_data_faulty(:,3));


left_M = [left_pigweed left_ragweed]; %information matrix-Left Camera
center_M = [center_pigweed center_ragweed]; %information matrix-Center Camera
right_M = [right_pigweed right_ragweed]; %information matrix-Right Camera

ts = 3; %5 time steps
object = 2; % number of object to be classified
D_jos = [1 0;
         0 1];

M = right_M; % could be left, center, right based on camera

total_time_steps = size(M,1); %number of row of M

%Sensor fusion DS
for count = 1 : (total_time_steps - ts + 1)
    for row_d = count : ts+count-1
        for column_d = count : ts+count-1
            D(row_d-count+1,column_d-count+1) = sqrt (.5 * (M(row_d,:) - M(column_d,:))
        end
```

```
        d(row_d-count+1) = sum(D(row_d-count+1,:));
    end
    d_avg = sum(d)/length(d);
    for e_row = count : ts+count-1
        entropy(e_row-count+1) = 0.0001;
        for col = 1:object
            if (M(e_row,col)==0)
                ;
            else
                entropy(e_row-count+1) = entropy(e_row-count+1) + (-(M(e_row,col)*log2(M
            end
        end
    end

    norm_entropy = entropy./sum(entropy);

    for row = 1:ts
        if (d(row) <= d_avg)
            reward(row) = -log(norm_entropy(row));
        else
            reward(row) = -log(1 - norm_entropy(row)); %penalty
        end
    end

    weight = reward./sum(reward);

    for i = 1:object
        evidence(i) = weight * M(count:ts+count-1,i);
    end

    evidence_new = evidence;

     for fus = 1:ts-1
        for row = 1:object
            for col = 1:object
                fusion(row,col,fus) = evidence(row)*evidence_new(:,col,fus);
            end
        end
        k(fus) = fusion(1,2,fus)+fusion(2,1,fus);
        den(fus) = 1 - k(fus);
        evidence_new(:,:,fus+1) = [fusion(1,1,fus) fusion(2,2,fus)]./den(fus);
     end

     fused_evidence(:,:,count) = evidence_new;
end
```

```
orig_time_steps = 1:total_time_steps;
time_steps = ts:total_time_steps;

fused_ragweed = squeeze(fused_evidence(2,ts,:));
fused_pigweed = squeeze(fused_evidence(1,ts,:));

subplot(2,1,1)
plot(orig_time_steps,right_pigweed*100,'r',orig_time_steps,right_ragweed*100,'b')
xlabel('time steps')
ylabel('classification %')
ylim([0 100])
title('Partial occlusion: original')
legend('Pigweed','Ragweed')

subplot(2,1,2)
plot(time_steps,fused_pigweed*100,'r',time_steps,fused_ragweed*100,'b')
title('Partial occlusion: time fused')
xlabel('time steps')
ylabel('classification %')
ylim([0 100])
legend('fused Pigweed','fused Ragweed')
```

D. CODE: CHAPTER 6 - MULTI-SENSOR SPACE DOMAIN FUSION

This code shows how space domain fusion is done using information from multiple sensors. This is the space domain fusion code for example 6.7. Matlab 2018b is used.

```
% Step 1: Information matrix
%     A   B   C  A,C
M = [.41 .29 .3  0;
      0  .9  .1  0;
     .58 .07  0 .35;
     .55 .1   0 .35;
     .6  .1   0 .3];

 % Step 2: Distance matrix

 % D for jousselme distance
 D_jos = [1 0 0 .5;
          0 1 0  0;
          0 0 1 .5;
         .5 0 .5 1];

D12 = sqrt (.5 * (M(1,:) - M(2,:)) * D_jos * transpose(M(1,:) - M(2,:)));
D13 = sqrt (.5 * (M(1,:) - M(3,:)) * D_jos * transpose(M(1,:) - M(3,:)));
D14 = sqrt (.5 * (M(1,:) - M(4,:)) * D_jos * transpose(M(1,:) - M(4,:)));
D15 = sqrt (.5 * (M(1,:) - M(5,:)) * D_jos * transpose(M(1,:) - M(5,:)));
D23 = sqrt (.5 * (M(2,:) - M(3,:)) * D_jos * transpose(M(2,:) - M(3,:)));
D24 = sqrt (.5 * (M(2,:) - M(4,:)) * D_jos * transpose(M(2,:) - M(4,:)));
D25 = sqrt (.5 * (M(2,:) - M(5,:)) * D_jos * transpose(M(2,:) - M(5,:)));
D34 = sqrt (.5 * (M(3,:) - M(4,:)) * D_jos * transpose(M(3,:) - M(4,:)));
D35 = sqrt (.5 * (M(3,:) - M(5,:)) * D_jos * transpose(M(3,:) - M(5,:)));
D45 = sqrt (.5 * (M(4,:) - M(5,:)) * D_jos * transpose(M(4,:) - M(5,:)));

DM = [0   D12  D13   D14   D15;
     D12   0   D23   D24   D25;
     D13  D23   0    D34   D35;
     D14  D24  D34    0    D45;
     D15  D25  D35   D45    0 ];

%Step 3 calculate average evidence distance
d1 = sum(DM(1,:));
d2 = sum(DM(2,:));
d3 = sum(DM(3,:));
```

```
d4 = sum(DM(4,:));
d5 = sum(DM(5,:));

%Step 4  calculate global average evidence distance
d = (d1+d2+d3+d4+d5)/5;

entropy_m1 = - (.41* log2(.41)+ .29* log2(.29)+ .3* log2(.3))
entropy_m2 = - (.9* log2(.9)+ .1* log2(.1))
entropy_m3 = - (.93* log2(.93)+ .07* log2(.07)+(.35/2)*log2(.35/2)+ .93* log2((.93/2)*ex
entropy_m4 = - (.9* log2(.9)+ .1* log2(.1)+(.35/2)*log2(.35/2)+ .9* log2((.9/2)*exp(1/3)
entropy_m5 = - (.9* log2(.9)+ .1* log2(.1)+(.3/2)*log2(.3/2)+ .9* log2((.9/2)*exp(1/3)))
%normalize entropy
norm_entropy_m1 = entropy_m1/(entropy_m1+entropy_m2+entropy_m3+entropy_m4+entropy_m5);
norm_entropy_m2 = entropy_m2/(entropy_m1+entropy_m2+entropy_m3+entropy_m4+entropy_m5);
norm_entropy_m3 = entropy_m3/(entropy_m1+entropy_m2+entropy_m3+entropy_m4+entropy_m5);
norm_entropy_m4 = entropy_m4/(entropy_m1+entropy_m2+entropy_m3+entropy_m4+entropy_m5);
norm_entropy_m5 = entropy_m5/(entropy_m1+entropy_m2+entropy_m3+entropy_m4+entropy_m5);

%step 6 Normalize evidence Reward and Penalty to get evidence weight.
reward_1 = -log(norm_entropy_m1) %d1>d
penalty_2 = -log(1-norm_entropy_m2)%d2<d
reward_3 = -log(norm_entropy_m3) %d3>d
reward_4 = -log(norm_entropy_m4) %d4>d
reward_5 = -log(norm_entropy_m5) %d5>d

w1 = reward_1/(reward_1+penalty_2+reward_3+reward_4+reward_5);
w2 = penalty_2/(reward_1+penalty_2+reward_3+reward_4+reward_5);
w3 = reward_3/(reward_1+penalty_2+reward_3+reward_4+reward_5);
w4 = reward_4/(reward_1+penalty_2+reward_3+reward_4+reward_5);
w5 = reward_5/(reward_1+penalty_2+reward_3+reward_4+reward_5);


%Step 7: Modify the original evidence
weight = [w1 w2 w3 w4 w5]

m_A = weight * M(:,1);
m_B = weight * M(:,2);
m_C = weight * M(:,3);
m_A_C = weight * M(:,4);

%step 8 combine for (n-1) times with DS combination rule
% m(A)      m(A)   k       k       m(A)
% m(B)      k      m(B)    k        k
% m(C)      k      k      m(C)     m(C)
% m(A,C)   m(A)    k      m(C)     m(A,C)
```

```
%fusion 1-2
m1 = [m_A m_B m_C m_A_C]
fus12 = [m1(1,1)*m1(1,1) m1(1,1)*m1(1,2) m1(1,1)*m1(1,3) m1(1,1)*m1(1,4);
         m1(1,2)*m1(1,1) m1(1,2)*m1(1,2) m1(1,2)*m1(1,3) m1(1,2)*m1(1,4);
         m1(1,3)*m1(1,1) m1(1,3)*m1(1,2) m1(1,3)*m1(1,3) m1(1,3)*m1(1,4);
         m1(1,4)*m1(1,1) m1(1,4)*m1(1,2) m1(1,4)*m1(1,3) m1(1,4)*m1(1,4)];

k12 = fus12(1,2)+fus12(1,3)+fus12(2,1)+fus12(2,3)+fus12(2,4)+fus12(3,1)+fus12(3,2)+fus12
den12 = 1-k12;
m12_A = (fus12(1,1)+fus12(1,4)+fus12(4,1))/den12;
m12_B = (fus12(2,2))/den12;
m12_C = (fus12(3,3)+fus12(3,4)+fus12(4,3))/den12;
m12_A_C = (fus12(4,4))/den12;

%fusion 1-2-3
disp("sensor 1-2 fusion")
m12 = [m12_A m12_B m12_C m12_A_C]
fus123 = [m12(1,1)*m1(1,1) m12(1,1)*m1(1,2) m12(1,1)*m1(1,3) m12(1,1)*m1(1,4);
         m12(1,2)*m1(1,1) m12(1,2)*m1(1,2) m12(1,2)*m1(1,3) m12(1,2)*m1(1,4);
         m12(1,3)*m1(1,1) m12(1,3)*m1(1,2) m12(1,3)*m1(1,3) m12(1,3)*m1(1,4);
         m12(1,4)*m1(1,1) m12(1,4)*m1(1,2) m12(1,4)*m1(1,3) m12(1,4)*m1(1,4)];

k123 = fus123(1,2)+fus123(1,3)+fus123(2,1)+fus123(2,3)+fus123(2,4)+fus123(3,1)+fus123(3,
den123 = 1-k123;
m123_A = (fus123(1,1)+fus123(1,4)+fus123(4,1))/den123;
m123_B = (fus123(2,2))/den123;
m123_C = (fus123(3,3)+fus123(3,4)+fus123(4,3))/den123;
m123_A_C = (fus123(4,4))/den123;

%fusion 1-2-3-4
disp("sensor 1-2-3 fusion")
m123 = [m123_A m123_B m123_C m123_A_C]
%m1_new = M(4,:); %sensor4

fus1234 = [m123(1,1)*m1(1,1) m123(1,1)*m1(1,2) m123(1,1)*m1(1,3) m123(1,1)*m1(1,4);
         m123(1,2)*m1(1,1) m123(1,2)*m1(1,2) m123(1,2)*m1(1,3) m123(1,2)*m1(1,4);
         m123(1,3)*m1(1,1) m123(1,3)*m1(1,2) m123(1,3)*m1(1,3) m123(1,3)*m1(1,4);
         m123(1,4)*m1(1,1) m123(1,4)*m1(1,2) m123(1,4)*m1(1,3) m123(1,4)*m1(1,4)];

k1234 = fus1234(1,2)+fus1234(1,3)+fus1234(2,1)+fus1234(2,3)+fus1234(2,4)+fus1234(3,1)+fu
den1234 = 1-k1234;
m1234_A = (fus1234(1,1)+fus1234(1,4)+fus1234(4,1))/den1234;
m1234_B = (fus1234(2,2))/den1234;
m1234_C = (fus1234(3,3)+fus1234(3,4)+fus1234(4,3))/den1234;
```

```
m1234_A_C = (fus1234(4,4))/den1234;

%fusion 1-2-3-4-5
disp("sensor 1-2-3-4 fusion")
m1234 = [m1234_A m1234_B m1234_C m1234_A_C]
%m1_new = M(5,:); %sensor5

fus12345 = [m1234(1,1)*m1(1,1) m1234(1,1)*m1(1,2) m1234(1,1)*m1(1,3) m1234(1,1)*m1(1,4);
         m1234(1,2)*m1(1,1) m1234(1,2)*m1(1,2) m1234(1,2)*m1(1,3) m1234(1,2)*m1(1,4);
         m1234(1,3)*m1(1,1) m1234(1,3)*m1(1,2) m1234(1,3)*m1(1,3) m1234(1,3)*m1(1,4);
         m1234(1,4)*m1(1,1) m1234(1,4)*m1(1,2) m1234(1,4)*m1(1,3) m1234(1,4)*m1(1,4)];

k12345 = fus12345(1,2)+fus12345(1,3)+fus12345(2,1)+fus12345(2,3)+fus12345(2,4)+fus12345(
den12345 = 1-k12345;
m12345_A = (fus12345(1,1)+fus12345(1,4)+fus12345(4,1))/den12345;
m12345_B = (fus12345(2,2))/den12345;
m12345_C = (fus12345(3,3)+fus12345(3,4)+fus12345(4,3))/den12345;
m12345_A_C = (fus12345(4,4))/den12345;

disp("sensor 1-2-3-4-5 fusion")
m12345 = [m12345_A m12345_B m12345_C m12345_A_C]
```

VITA

Md Nazmuzzaman Khan

Education

- PhD in Mechanical Eng. (GPA: 3.6/4)
  Purdue University
  2016 - 2021 | Indiana, USA

- MSc. in Mechanical Eng. (GPA: 3.7/4)
  Indiana University Purdue University Indianapolis
  2013 - 2015 | Indiana, USA

- BSc. in Mechanical Eng. (GPA: 3.6/4)
  Bangladesh University of Engineering & Technology
  2007 - 2012 | Dhaka, Bangladesh

Experiences

- Sr. Research Engineer, Raven Industries, 2021 - current
  Responsible for the research and development of systems that incorporate the use of various sensors such as camera, radar, and lidar, individually and through the use of sensor fusion, to detect and classify objects encountered in autonomous vehicle operations.

- Team Lead - Agricultural robot, IUPUI, 2017 - 2019
  We participated in three international robotics competitions, where our team won 2^{nd} place twice because our vision module outperformed all other teams.