IMAGE-BASED PLANT PHENOTYPING USING MACHINE LEARNING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Javier Ribera Prat

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Edward J. Delp, Chair

School of Electrical and Computer Engineering

Dr. Amy R. Reibman

School of Electrical and Computer Engineering

Dr. Charles A. Bouman

School of Electrical and Computer Engineering

Dr. Melba M. Crawford

Department of Agronomy

Approved by:

Dr. Pedro Irazoqui

Head of the School Graduate Program

ACKNOWLEDGMENTS

Most importantly, I would like to express my deepest gratitude to my doctoral adviser Professor Edward J. Delp. Without his support, direction and guidance, this dissertation would not have been possible. I am grateful for having the opportunity to work under his supervision, and for becoming a member of the Video and Image Processing Laboratory (VIPER). His determination to overcome any difficulty along my studies and his technical expertise have been priceless.

I would like to thank Professor Amy R. Reibman for her cordiality, extraordinary technical skills, and her invaluable criticism that resulted in very fruitful feedback.

I would like to show my appreciation to Professor Charles A. Bouman for his educational approach in ECE637 and ECE641. His passion for the field, and his unique intuition is a source of inspiration.

It is a pleasure to express my appreciation to Professor Melba M. Crawford, for her enthusiasm, and her suggestions that have shaped this work from an expert in this field.

I would like to thank Dr. Neeraj Gadgil, Dr. Albert Parra Pozo, and Dr. Khalid Tahboub for their warm welcome to West Lafayette. I would also like to show my thankfulness to Yuhao Chen, Blanca Delgado, David Güera Cobo, and Daniel (Dani) Mas Montserrat for their friendship throughout my studies at Purdue, and for being great teammates. I also want to thank the rest of the former and current VIPER lab colleagues Sriram Baireddy, Di Chen, Enyu Cai, Qingchaung (Cici) Chen, Dahjung Chung, Jeehyun Choe, Shaobo Fang, Chichen Fu, Shuo Han, Hanxiang (Hans) Hao, Dr. Ye He, David Ho, János Horváth, Dr. Joonsoo Kim, Soonam Lee, He Li, Chang Liu, Daniel Mas, Ruiting Shao, Zeman Shao, Dr. Yu Wang, Changye (Chad) Yang, Sri Kalyan Yarlagadda, Jiaju Yue, Dr. Bin Zhao, and Yifan Zhao, and our lab visitor Dr. Thitiporn (Bee) Pramoun.

I would also like to thank Professor Ayman F. Habib and the members of the Digital Photogrammetry Research Group (DPRG) for collecting the Unmanned Aerial Vehicle (UAV) and PhenoRover data used in this dissertation.

And thank you to my parents, without whom I could not have achieved this accomplishment. Their unconditional support and role model have been indispensable in my professional career and personal life.

This work was funded in part by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy, under Award Number DE-AR0000593. This work was also partially supported by the U.S. Department of Homeland Security's VACCINE Center under Award Number 2009-ST-061-CI0001. The views and opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

TABLE OF CONTENTS

Page

LIST OF TABLES viii

LIST OF FIGURES ix

NOMENCLATURE xiii

ABSTRACT xv

1 INTRODUCTION 1

1.1 Image-based Phenotyping 1

1.2 Crowdflow Estimation Enhanced by Crowdsourcing 6

1.3 Contributions of This Thesis 9

1.4 Publications Resulting From This Work 11

2 PHENOTYPING: PLOT-LEVEL TRAITS 13

2.1 Introduction 13

2.2 Overview Of Previous Work 15

2.3 Datasets For Geometric Phenotypes 16

2.4 Plot Extraction From UAVs 21

2.4.1 Non-constant spacing 30

2.5 Plot Extraction From Ground-Based Platforms 34

2.6 Canopy Coverage 40

2.7 Leaf Counting 42

2.7.1 Experimental Results 47

3 PHENOTYPING: PLANT LOCALIZATION 53

3.1 Introduction 53

3.2 Overview Of Previous Work 54

3.3 A Loss Function For Object Localization With Deep Learning 57

3.3.1 The Average Hausdorff Distance 57

Page

3.3.2 The Weighted Hausdorff Distance . . . 60
3.3.3 Advantage Over Pixelwise Losses . . . 62
3.3.4 CNN Architecture . . . 64
3.3.5 Location Estimation . . . 66
3.3.6 Experimental Results . . . 69
3.4 Counting With Deep Learning . . . 78
3.4.1 Modifications To The CNN Architecture . . . 80
3.4.2 Experimental Results . . . 80
3.5 Plant Location Model Using A Statistical Model . . . 84
3.5.1 Experimental Results . . . 90
4 AN API AND WEB PLATFORM FOR PHENOTYPING . . . 92
4.1 An API For Phenotypic Data And Metadata . . . 92
4.1.1 XML specification . . . 93
4.2 A Web Platform For Phenotyping . . . 98
5 CROWDSOURCING TO ENHANCE CROWDFLOW ESTIMATION . . . 105
5.1 Crowd Flow Estimation . . . 105
5.1.1 Key Ideas and Overall Scheme . . . 105
5.1.2 Foreground Pixel Count . . . 106
5.1.3 Weighting Function . . . 109
5.1.4 Crowdedness Estimation . . . 111
5.1.5 Training Data . . . 116
5.2 Uncertainty Characterization and Crowdsourcing . . . 117
5.2.1 Uncertainty of The Classifier . . . 117
5.2.2 Use of Crowdsourcing . . . 118
5.2.3 Crowdsourcing Task . . . 120
5.2.4 Results Aggregation . . . 122
5.3 Web Platform . . . 123
5.3.1 Control by The Operator . . . 123

Page

5.3.2 Crowdsourcing Tasks 127
5.3.3 Training 128
5.4 Experimental Results 136
6 SUMMARY AND FUTURE WORK 143
6.1 Summary 143
6.2 Future Work 145
REFERENCES 147
VITA 162

LIST OF TABLES

Table Page

3.1 Summary of object localization results 76

3.2 Head location results using the mall dataset, using $r = 5$. 76

3.3 Pupil center localization results 77

3.4 Plant location results using the plant dataset, using $r = 5$. 77

3.5 Impact of p on the regression results 83

3.7 Comparison of different CNN architectures for counting 84

4.1 Description of the elements of our API. 95

4.2 Description of the attributes of our API 98

5.1 Effect of quality degradation on the average error rate 139

5.3 Results of the crowdsourcing incorporation 139

5.5 Comparison of uncertainty characterizations 141

5.7 Comparison of aggregation criterias 142

LIST OF FIGURES

Figure Page

1.1 Photograph of different varieties of sorghum plants. 3
1.2 Anatomy of a sorghum plant 4
1.3 Sorghum panicle 5
2.1 Structure of a crop field. 14
2.2 UAVs used during data collection 17
2.3 Perspective image 18
2.4 Distortion-free image 19
2.5 Orthorectified mosaic 19
2.6 Picture of the PhenoRover 20
2.7 Example of row segments 21
2.8 A very well orthophoto 22
2.9 Example of a Region Of Interest (ROI), a field panel 23
2.10 Example of horizontal profile. 24
2.11 Estimated lines separating ranges in an orthophoto 25
2.12 Estimated plot boundaries, before shrinking 27
2.13 Plot boundaries after horizontal shrinking. 29
2.14 Example of poorly rectified orthophoto. 31
2.15 Estimated plot boundaries, zoomed sections of Figure 2.14(b). 33
2.16 Images acquired from our ground-based platform, the PhenoRover 35
2.17 Intermediate steps of the plot extraction using a ground-based image. . . . 36
2.18 Radon transform diagram 38
2.19 Example of low canopy coverage. 40
2.20 Example of high canopy coverage. 41
2.21 Definition of canopy coverage. 42

Figure Page

2.22 Segmentation mask 43

2.23 Leaf count ground truth 45

2.24 Relation between leaf count and number of leaf pixels 46

2.25 Accuracies using each subrow as training data 47

2.26 Perspective image analysis 48

2.27 Distortion-free image analysis 49

2.28 Orthomosaic image analysis 50

2.29 Evolution of leaf count 52

3.1 Ground truth of plant locations 54

3.2 Example of object localization 58

3.3 Illustration of the Hausdorff distance 59

3.4 Our Fully Convolutional Network (FCN) architecture 64

3.5 Estimated locations using the Weighted Hausdorff Distance (WHD) 67

3.6 Splitting an orthophoto into training, validation, and testing 70

3.7 Effect on the F-score of the threshold τ 73

3.8 Beta mixture model and adaptive thresholds 74

3.9 F-score as a function of r 75

3.10 Examples of row segments 79

3.11 Modifications in our Convolutional Neural Network (CNN) architecture . . 81

3.12 Orthophoto of a sorghum field 81

3.13 Data augmentation techniques 82

3.14 A single sorghum plant 86

3.15 Scheme of the prior distribution of plant locations 87

3.16 Visualization of the cost function for plant localization 91

4.1 DIBPS. Login page 100

4.2 DIBPS. Upload page 101

4.3 DIBPS. Phenotyping methods page 102

4.4 DIBPS. Page to download the results 103

Figure Page

4.5 DIBPS. Parameters needed for plot extraction 104

5.1 ROI and Tripwire examples 107

5.2 Overall scheme of the automatic crowd flow estimation method 107

5.3 Scheme of the foreground pixel count 108

5.4 Foreground mask of a Tripwire 108

5.5 Weighting scheme illustration 110

5.6 Different levels of crowdedness 112

5.7 GLCM calculations 113

5.8 Texture feature scheme 114

5.9 Nearest neighbor texture feature vector 115

5.10 First characterization of uncertainty 119

5.11 Third characterization of uncertainty 120

5.12 Crowdsourcing task 121

5.13 Web platform: processes monitor 125

5.14 Web platform: processes invoker selecting a video file 126

5.15 Web platform: processes invoker selecting a real-time video stream . . . 127

5.16 Web platform: crowdsourcing tasks 128

5.17 Web platform: training 129

5.18 Web platform: new training data, step 2 130

5.19 Web platform: new training data, step 3 131

5.20 Web platform: new training data, step 4 132

5.21 Web platform: new training data, step 6 133

5.22 Web platform: new training data, step 7 134

5.23 Web platform: new training data, step 8 134

5.24 Web platform: new training data, Background Subtraction (BS) preview 135

5.25 University of California, San Diego (UCSD) pedestrian dataset 136

5.26 Video segments of the UCSD dataset used for testing 137

5.27 Error rates with no crowdsourcing, in the first experiment 138

Figure Page

5.28 Evolution of the o-crowd utilization in the first experiment 140

5.29 Evolution of the o-crowd utilization in the second experiment 140

NOMENCLATURE

ACRE	Agronomy Center for Research and Education
API	Application Programming Interface
BMM	Beta Mixture Model
BS	Background Subtraction
CRF	Constant Rate Factor
CNN	Convolutional Neural Network
CSV	Comma-Separated Values
DIBPS	Distributed Image-Based Phenotyping System
EM	Expectation Maximization
FCN	Fully Convolutional Network
GIS	Geographic Information System
GLCM	Gray Level Co-occurrence Matrix
GMM	Gaussian Mixture Model
GUI	Graphical User Interface
HSV	Hue, Saturation, and Value
IR	Infrared
ICD	Iterative Coordinate Descent
JSON	JavaScript Object Notation
MAP	Maximum A Posteriori
MAPE	Mean Absolute Percent Error
ML	Machine Learning
MLE	Maximum Likelihood Estimator
MSE	Mean Squared Error
NDVI	Normalized Difference Vegetation Index

NIR	Near Infra-Red
LAI	Leaf Area Index
PID	Process Identifier
RGB	Red, Green, and Blue
RMSE	Root Mean Squared Error
ROI	Region Of Interest
SVM	Support Vector Machine
TIFF	Tagged Image File Format
UAV	Unmanned Aerial Vehicle
UCSD	University of California, San Diego
WHD	Weighted Hausdorff Distance
XML	Extensible Markup Language

ABSTRACT

Ribera Prat, Javier Ph.D., Purdue University, May 2019. Image-Based Plant Phenotyping Using Machine Learning. Major Professor: Edward J. Delp.

Phenotypic data is of crucial importance for plant breeding in estimating a plant's biomass. Traits such as leaf area and plant height are known to be correlated with biomass. Image analysis and computer vision methods can automate data analysis for high-throughput phenotyping. Many methods have been proposed for plant phenotyping in controlled environments such as greenhouses. In this thesis, we present multiple methods to estimate traits of the plant crop sorghum from images acquired from UAV and field-based sensors. We describe machine learning techniques to extract the plots of a crop field, a method for leaf counting from low-resolution images, and a statistical model that uses prior information about the field structure to estimate the center of each plant. We also develop a new loss function to train Convolutional Neural Networks (CNNs) to count and locate objects of any type and use it to estimate plant centers. Our methods are evaluated with ground truth of sorghum fields and publicly available datasets and are shown to outperform the state of the art in generic object detection and domain-specific tasks.

This thesis also examines the use of crowdsourcing information in video analytics. The large number of cameras deployed for public safety surveillance systems requires intelligent processing capable of automatically analyzing video in real time. We incorporate crowdsourcing in an online basis to improve a crowdflow estimation method. We present various approaches to characterize this uncertainty and to aggregate crowdsourcing results. Our techniques are evaluated using publicly available datasets.

1. INTRODUCTION

1.1 Image-based Phenotyping

In many agricultural applications, one wants to characterize physical properties of plants. This process is known as phenotyping [1]. Plant breeders collect phenotypic information in order to study the performance of a crop [2,3]. Phenotyping is formally defined in [1] as "characterizing the performance of the plants for desired trait(s)". For example, some phenotypic traits such as leaf area have been shown to be correlated with above-ground biomass [4–6]. Also, agronomists can use plant spacing and plant density to predict the future yield of their crops [7–11]. Other remote-sensed data such as LiDAR point clouds or hyperspectral data can be used by crop models to estimate biomass [12]. Other phenotypic traits of a plant include height, leaf color, canopy aperture, or chlorophyll fluorescence intensity [13,14]. Obtaining high quality phenotypic data plays a crucial role in phenotypic studies.

However, in many research studies, phenotypic information is still being collected manually [15]. Examples of traditional procedures include obtaining in-field measurements with portable instrumentation [16, 17]. Samples can also be taken to the laboratory for exhaustive examination. These methods are not only labor-intensive and time-consuming, but also destructive, which increases the plant population size required in phenotyping studies [18].

Advances in genotyping technologies such as marker-assisted selection [19] have boosted the ability to sequence a plant's DNA and greatly reduced genotyping costs [20, 21]. However, current phenotyping capabilities limit the potential of linking genotype with phenotypic traits [2].

Future high-throughput phenotyping platforms are expected to easily provide precise phenotypic data in a non-destructive way [18]. Computer-assisted methods for

phenotyping, and in particular imaging techniques are becoming more popular [22,23]. In particular, Machine Learning (ML) is becoming a promising approach to analyze the enormous amount of data generated by phenotyping platforms [24]. In [25], Zhou et al. compare different ML techniques in the prediction of sorghum biomass, using remote-sensed hyperspectral and RGB data.

In [26], a camera-based growth chamber is described for plant phenotyping using micropots. The plant leaves are segmented by selecting a green cluster in the YIQ colorspace [27]. Leaf morphology is determined by using morphological processing and connected components. The number of leaves per plant and the length of each leaf are estimated. In [28], the plant *Setaria* is phenotyped in a highly controlled setting. The phenotypic traits are estimated from RGB images include plant height, convex hull, and plant area using morphological and watershed methods. In [29], the circular geometry and overlap between the leaves of rosette plants are used in order to individually segment each leaf. In [30], rosette plants in a laboratory are automatically segmented and analyzed by using active contours and a Gaussian Mixture Model (GMM). Other image-based methods are described in several review papers [18, 22]. The web site `https://plant-image-analysis.org`, described in [31], references more than 100 software tools for image-based plant phenotyping.

There exist commercial image-based systems for automated plant phenotyping [32, 33]. These systems have been mainly used indoors [28, 34]. In fact, most image-based phenotyping methods that have been proposed are designed for indoor or greenhouse settings, while real-life crops are mainly grown outdoors [35]. Many different sensor types have been proposed for phenotyping, including multi-spectral, hyper-spectral, Infrared (IR) and RGB cameras. The techniques that we describe in this thesis employ RGB images acquired from a UAV flying over the field, or from a ground-based platform driving over the field.

In this thesis, the crop fields we analyze consist of sorghum plants. Sorghum (*Sorghum bicolour* (L.) Moench) is the fifth most important crop in the world [1]. Figure 1.1 shows different varieties of sorghum plants.



Fig. 1.1.: Photograph of different varieties of sorghum plants.

Typically cultivated for food in dry areas of South Africa, sorghum is gaining importance in wetter eastern areas [36]. Attempts to grow sorghum in cooler regions motivate the necessity to identify varieties with higher cold tolerance [37,38]. Sorghum belongs to the Gramineae (or Poaceae) family, commonly known as weeds or grasses. Other members of the Gramineae family are corn, rice, wheat, barley, bamboo, or *setaria*. Uses of sorghum include food [39], beverages [40], fodder, or energy [41–44]. Sorghum has attracted interest for biofuel production due to its high tolerance to drought [43]. This poses sorghum as a good candidate for crops that can adapt to climate change conditions [1]. The anatomical parts of a sorghum plant are shown in Figure 1.2. Leaves are flat and generally green. Figure 1.3 shows a close-up of the panicle (or inflorescence), clearly showing the grains. The grains are of oval shape, and, although usually brown, they can also be red, white, or yellow [36].

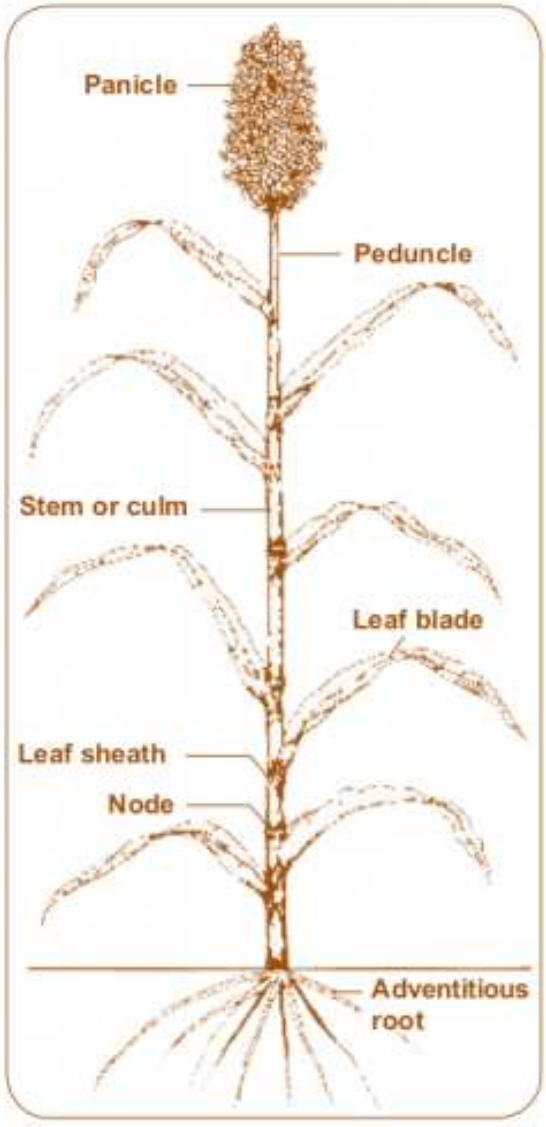


Fig. 1.2.: Anatomy of a sorghum plant. Source: [36].

The canopy of a group of plants is defined as the upper part of the plant material [45]. In forests, the canopy is formed by the crowns of the trees. In the case of sorghum, the canopy of the crop can be very dense, affecting the amount of light that leaves in lower layers receive. Canopy structure is thus an important trait of a sorghum crop field.



Fig. 1.3.: Sorghum panicle. The panicle is formed by spikelets, which contain the grains.

In this thesis, we propose multiple machine learning and image processing techniques to obtain phenotypic traits using images acquired from a UAV or a ground-based platform. We describe methods to segment and extract plots from a crop field, estimate the number of leaves in low resolution images, estimate canopy coverage, and count and locate plant centers. We have also developed a web platform that includes these tools. Plant scientists and agronomists can run these phenotyping tools from their browser and analyze their own data. Validated with data collected in a sorghum field at Purdue University, our techniques are shown to be of great value and easy to use in phenotypic studies.

1.2 Crowdflow Estimation Enhanced by Crowdsourcing

The use of video surveillance systems is becoming more popular for a variety of commercial, law enforcement and military applications [46]. One can observe the tremendous increase in the number of deployed cameras in such systems [47]. This makes the continuous monitoring of all the video feeds by human operators an impracticable task. As a consequence, in most video surveillance systems, the recorded videos are stored and only used in an after-the-fact investigation. In this situations, these surveillance systems are incapable of preventing or alerting about security issues in real time [48]. Automatic analysis of video feeds may help overcome this limitation and address the scalability issue of video surveillance systems [49].

Intelligent surveillance systems require video analytics capable to understand the scenes being recorded, and to warn the operators in real time. Some examples of automatic video analysis include fight, abandoned baggage, intrusion detection, and crowd analysis. Crowd analysis consists in estimating the attributes of a crowd of people, such as direction of movement, speed, density, or any other pattern. Surveys of computer vision methods for crowd analysis can be found in [50, 51].

One of the attributes of a crowd that can be analyzed is crowd flow. Crowd flow is defined as the number of people crossing a specific spatial zone. Crowd flow estimation can be used to avoid surpass the capacity of a building, and can provide useful information when designing entry or exit nodes of a building.

Many methods have been proposed to estimate a crowd's flow. One type of approach is to identify and track every single individual in the scene, and count how many people cross the desired region. The literature on human motion tracking is extensive [52,53]. However, this approach may arise scalability issues when dealing with large crowds. Simultaneous tracking of hundreds or thousands of targets may become too computationally expensive. Indirect methods estimate attributes of crowds such as the crowd flow without tracking. In these indirect methods, the characteristics of a crowd are related to low level features extracted from the video. In [54], pedestrians

are counted by employing a linear relationship between the number of pixels in the foreground segmentation and the number of pedestrians. A constant level of occlusion between people is assumed. In [55, 56], the crowd density or occlusion level is related to the texture of the image. In [57], these two ideas are combined to estimate the crowd flow allowing different levels of occlusion. Texture features are incorporated to consider changing crowd densities. In [58], pedestrians are counted by using geometric, edge, and texture features. In [59], other features such as edge orientation and blob size histograms are used. In [60], a novel spatial-temporal matrix, support vector machine (SVM), and mean-shift clustering are introduced to count pedestrians.

Real-life scenarios introduce challenges that can undermine the performance of real-time video analytics. Video processing techniques must be resilient to gradual and sudden changes in illumination, occlusions, and shadows. Distortion in the transmitted video due to compression or packet losses can also negatively impact the video analytics.

Crowdsourcing has been effectively used in many situations to solve problems that involve cognitive tasks. Originally devised by J. Howe [61], crowdsourcing was later formally defined in [62] as "a type of participative online activity in which an individual, an institution, a non-profit organization, or company proposes to a group of individuals of varying knowledge, heterogeneity, and number, via a flexible open call, the voluntary undertaking of a task". Crowdsourcing has also been referred to as collective intelligence or the wisdom of crowds [63]. The wisdom of the crowd can sometimes outperform an expert annotator if the crowd satisfies the conditions of diversity, independence, decentralization and aggregation [63]. Many studies have employed crowdsourcing to make human intelligence improve a machine's performance. In [64], many different crowdsourcing systems on the world-wide web are reviewed. Crowdsourcing has been widely used in the computer vision community to build up training data for video analysis techniques. In [65], a model for object detection is iteratively refined using crowdsourcing. In [66], machine learning and crowdsourc-

ing enhance each other. Robot's apprehension of unknown objects is supervised by human-provided segmentations in an online basis.

Often, the public crowd is reached through commercial platforms such as Amazon Mechanical Turk [67], Mob4hire [68], Freelancer [69], or uTest [70]. However, employing online commercial platforms for law enforcement may arise privacy concerns about video content protection [71]. In [72], a web-based tool that allows the crowd to annotate surveillance videos is described. In this tool, the operator has fine control on the crowd's access to each video, and the tasks assigned to each crowd member. Also, new annotators are trained before being assigned a specific role. In [73], a hierarchical pyramid model of crowd members is built in order to distinguish the performance and experience of each member.

To avoid confusion, in this thesis we refer to the crowdsourcing crowd as "observation crowd" or "o-crowd". The "o-crowd" comprise the humans that assist the automatic analysis in order to enhance its performance. The crowd of people recorded by the cameras, and whose attributes are estimated by the automatic method, will be referred to as the "crowd".

In this thesis, our approach is to "ask" the o-crowd when the automatic method is uncertain about making a particular decision. Also, a web platform for this crowdsourcing scheme is described. Using this web platform, the o-crowd annotates unlabeled data to enhance the accuracy of the automatic crowd flow estimation method.

In many situations, labeled data is very expensive to obtain, while unlabeled data is abundant. The goal of active learning is to properly select the optimal training data to be queried [74]. Effectively selecting the most crucial unlabeled data can significantly increase the accuracy or decrease the amount of labels needed. Active learning has been used in various situations to intelligently learn from an heterogeneous o-crowd [75].

Active learning can be categorized into two distinct types: online active learning, and pool-based active learning. In online learning, the decision whether the data must be labeled or not is taken in real-time, as data become available. In contrast,

in pool-based learning, many sample data are delivered, and the optimal data points must be selected.

In this thesis, we employ online active learning when the confidence of our automatic method is too low. Different methods to characterize the uncertainty of classification methods are proposed. The automatic crowd flow estimation method analyzes frame by frame the input video and classifies every frame into different classes. If the uncertainty of the classifier is above a predefined threshold, the uncertain frame is queried to the o-crowd. We propose and compare different ways to characterize this uncertainty.

O-crowd members typically have different levels of expertise and bias [75, 76]. Thus, even when the same frame is shown to all the members of the o-crowd, one should expect different labels for the same unlabeled frame. This suggests that the optimal way to combine the labels may not be to treat them equally. We propose and evaluate criteria to aggregate the o-crowd answers into one final label. Finally, the aggregated result is incorporated into the classifier to reduce its uncertainty. Hence, future classifications are enhanced and the o-crowd is reached less often.

Experimental evaluation is conducted using a publicly available surveillance video dataset.

1.3 Contributions of This Thesis

In this thesis, we developed new methods for plant phenotyping from UAV and ground-based platforms, object localization, and reduction of uncertainty in classification methods using crowdsourcing. The main contributions of this work are:

- Plot Extraction

  We address the problem of extracting sections of an image that belong to different field plots. This is known as "plot extraction" and enables further phenotypic analyses. For example, one can use the extracted plots to estimate the canopy coverage or leaf count of each plot separately. We describe two methods

for plot extraction. One method extracts plots from an orthophoto of UAV images, by minimizing an energy function that finds straight prarallel lines between the rows of plants. The other method extracts row segments from the "PhenoRover", acquiring images from the top of the crop canopy. This method uses the Radon Transform to find the most dominant almost-parallel lines in the non-rectified image.

- Leaf count and canopy coverage estimation

  We describe a technique to segment plant material, and estimate canopy coverage and leaf count at the plot level. We use this technique to estimate the leaf count of a crop field without individually segmenting each leaf. This method assumes that leaves have approximately the same area. We evaluate this technique with perspective, distortion-free, and orthorectified images of sorghum plants, achieving an average accuracy of 87.7 % using ground truth provided by manual leaf count from the images.

- Plant Counting and Location

  We investigate the problem of counting and locating plants from UAV imagery. Counting and locating are usually considered two sides of the same coin. We propose methods to count and locate plants as separate tasks, and a method to simultaneously locate and count generic objects. A statistical model allows to estimate of the location of each plant, by making use of prior information that accounts for the alignment of the plants in the field. We count the number of plants in a plot by using a regression loss and a CNN. We also design a novel loss function, which we call *Weighted Hausdorff Distance*, and employ it to locate sorghum plants and estimate intra-row plant spacing.

- Crowdsourcing

  We incorporate crowdsourcing to improve the accuracy of a crowd flow estimation method. Diverse characterizations of the uncertainty of a classifier are proposed and evaluated, as well as different criteria to aggregate the la-

bels provided by heterogeneous crowdsourcing labelers. The method uses the crowdsourced label to reduce the error rate, and retrains a classifier to reduce how often to ask the crowdsourcing crowd. Our experimental evaluation using a publicly available dataset suggests that crowdsourcing reduces the error rate, and that the utilization of the o-crowd is reduced with time.

- Web phenotyping system, and online crowdsourcing tool
  We develop an online platform that allows plant scientists and agronomists to make use of our phenotyping tools and estimate plant traits from their own data. We also develop an online system to reach the crowdsourcing labelers. Crowdsourcing members can annotate unlabeled data for which the automatic method has low confidence.

1.4 Publications Resulting From This Work

1. **J. Ribera**, D. Güera, Y. Chen, and E. J. Delp, "Locating Objects Without Bounding Boxes", *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, Long Beach, CA (to appear)

2. **J. Ribera**, Y. Chen, C. Boomsma, and E. J. Delp, "Counting Plants Using Deep Learning", *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, November 2017, Montreal, Canada
   URL: https://doi.org/10.1109/GlobalSIP.2017.8309180

3. Y. Chen, **J. Ribera**, C. Boomsma, and E. J. Delp, "Locating Crop Plant Centers from UAV-Based RGB Imagery", *IEEE International Conference on Computer Vision Workshops (ICCVW)*, October 2017, Venice, Italy
   URL: https://doi.org/10.1109/ICCVW.2017.238

4. Y. Chen, **J. Ribera**, C. Boomsma, and E. J. Delp, "Plant Leaf Segmentation For Estimating Phenotypic Traits", *IEEE International Conference on Image*

Processing (ICIP), September 2017, Beijing, China
URL: https://doi.org/10.1109/ICIP.2017.8297010

5. Y. Wang, **J. Ribera**, C. Liu, F. Zhu, and E. J. Delp, "Pill Recognition Using Minimal Labeled Data", *IEEE International Conference on Multimedia Big Data (ICMBD)*, April 2017, Laguna Hills, CA
URL: https://doi.org/10.1109/BigMM.2017.61

6. **J. Ribera**, F. He, Y. Chen, A. F. Habib, and E. J. Delp, "Estimating Phenotypic Traits From UAV Based RGB Imagery", *ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on Data Science for Food, Energy, and Water*, August 2016, San Francisco, CA
URL: https://arxiv.org/abs/1807.00498

7. J. Kim, H. Li, J. Yue, **J. Ribera**, L. Huffman, and E. J. Delp, "Automatic and Manual Tattoo Localization", *IEEE International Conference on Technologies for Homeland Security (HST)*, May 2016, Waltham, MA
URL: https://doi.org/10.1109/THS.2016.7568950

8. **J. Ribera**, K. Tahboub, and E. J. Delp, "Characterizing the uncertainty of classification methods and its impact on the performance of crowdsourcing", *IS&T/SPIE Electronic Imaging*, pp. 94080A–94080A, January 2015, San Francisco, CA
URL: http://doi.org/10.1117/12.2085415

9. **J. Ribera**, K. Tahboub, and E. J. Delp, "Automated Crowd Flow Estimation Enhanced by Crowdsourcing", *Proceedings of the IEEE National Aerospace & Electronics Conference*, pp. 174-179, June 2014, Dayton, OH
URL: https://doi.org/10.1109/NAECON.2014.7045798

2. PHENOTYPING: PLOT-LEVEL TRAITS

2.1 Introduction

In this section we define some agronomic terms for a clearer understanding of this chapter. We describe the structure of a crop field and different types of phenotypic traits.

In agriculture and plant breeding, crop fields are commonly organized in the following manner [77]. A field is an extensive area of land, usually multiple acres, used for agricultural purposes. In a research farm we denote fields by numbers such as F51 or F47. Figure 2.1 shows a depiction of a typical structure of a crop field. A field can be composed of smaller areas called panels, where plants are planted in a consistent way for example in terms of plant density or set of plant varieties. In these panels a particular experiment is conducted. For example, the name of a panel can be "Hybrid Calibration," containing plants that have been crossbred from different genetic material. Within a panel, the plants are planted in straight lines known as "rows." Rows are remarkably straight because the planting is usually done with a GPS-enabled precision planter that is highly accurate. Examples of rows are marked in Figure 2.1 as vertical dashed rectangles. Along one row, there might be periodic spacing that breaks the rows into a smaller sets of plants known as "row segments." All the row segments in the perpendicular direction of a row are known as a "range." Examples of ranges are marked in Figure 2.1 as horizontal dashed rectangles.

Together, multiple row segments create a "plot" comprised of plants from a given variety. Plants within the same plot are always of the same variety and are treated equally. The more row segments a plot is composed of, the more samples of the same type are available. However, designing experiments with a high number of row segments per plot is expensive because of the additional land and labor it would

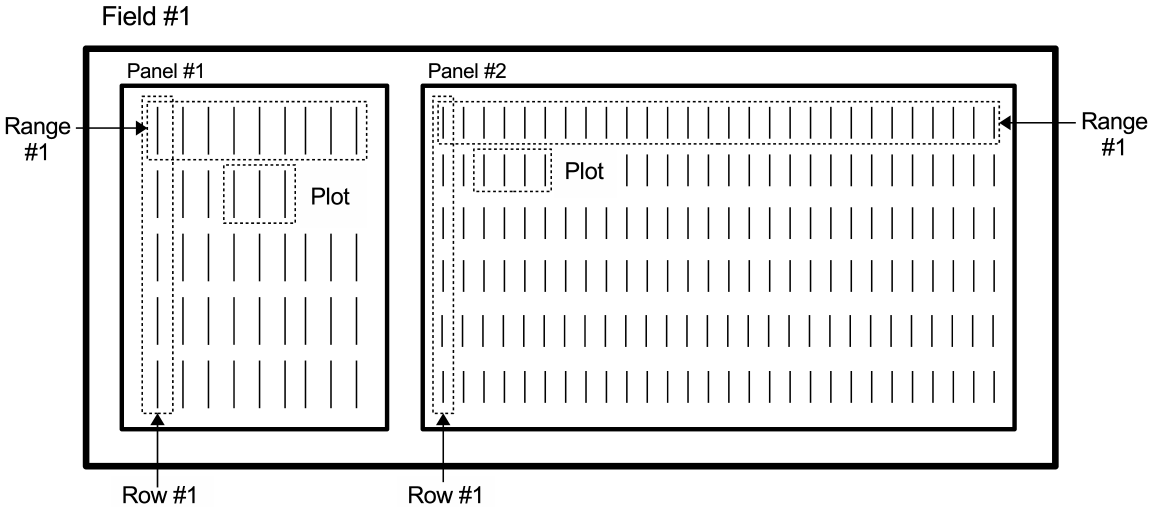


Fig. 2.1.: Scheme of the structure of a crop field. A field is formed by "panels", which are formed by "rows" (vertical in the figure) and "ranges" (horizontal in the figure). This forms a grid of "row segments." Groups of row segments consitute "plots.

require. Plots are assigned to experiments following a formal experimental design, laid out to test hypothesis about different plant varieties, treatments, or field management strategies. The rows within a plot that are selected for sampling affects the estimate of the plot-level phenotypes. Some research suggests that repeatability and prediction accuracy is lowered by using the entire plot data as opposed to only the interior rows. This has potential impact in the future of experimental design when using remote sensing for plant breeding.

Note that this is a very generic description of a crop field. In many field layouts, a field may contain a single panel, making the notion of panels unnecessary. Also, rows may not be subdivided into ranges, resulting in very long rows.

We categorize the phenotypic traits of crop fields into two types: plot-level traits and plant-level traits. Plot-level traits are traits of crop fields that correspond to entire plots, and not to particular plants. These include traits such as canopy coverage, the total number of leaves in a plot, or Leaf Area Index (LAI). Canopy coverage is defined with detail in Section 2.6. The plot-level LAI is defined as the ratio between the total

leaf area in a plot and the area of that plot. This quantity can be higher than one if the leaves overlap. One can also assign these traits to individual row segments, or single-row plots. Plant-level traits include plant location, the number of leaves of a single plant, or intra-row spacing (the distance from a plant to its neighbouring plants in the same row segment).

In this chapter, we describe methods to estimate plot-level traits.

2.2 Overview Of Previous Work

Many methods have been proposed to analyze plot-level traits of crop fields. An important part of the research in estimating plot-level traits has traditionally used non-RGB sensors. Sensors that capture wavelengths beyond the visible spectrum can capture properties of plants that RGB cameras cannot. For example, a common index known as Normalized Difference Vegetation Index (NDVI) is defined as

$$\text{NDVI} = \frac{R_{\text{NIR}} - R_{\text{red}}}{R_{\text{NIR}} - R_{\text{red}}}, \tag{2.1}$$

where R_{NIR} is the reflectance at a wavelength of 0.8µm, and R_{red} is the reflectance at a wavelength of 0.6µm. This requires an Near Infra-Red (NIR) that can capture wavelengths at 0.8µm, or a multispectral camera with a filter on the blue band. NDVI is commonly used in remote sensing to estimate traits such as LAI and canopy coverage [78] (canopy coverage is defined in Section 2.6). However, there exist many other indices that rely only on visible wavelengths [79]. In this dissertation, we focus on RGB cameras, which are more appropriate for geometric traits, and are easier to use off the shelf than NIR or hyperspectral cameras.

A typical first step in an image processing pipeline for remote plant phenotyping is to classify pixels into plant material (vegetation) or not (background). The result is called a "vegetation mask." Multiple methods have been proposed to obtain such vegetation mask. In [80], they use RGB, geometrical, and optionally NIR features, and a Random Forest to classify vegetation pixels. In [81], a decision tree using features of multiple color spaces is used to segment vegetation in a variety of illumination

conditions. In [82] and [83], they describe a method that iteratively groups spatially adjacent pixels in a bottom-up approach, and uses RGB and NIR features and Otsu's method [84]. In [85], a Support Vector Machine (SVM) with only RGB features is used to segment vegetation when rain makes the green channel unreliable. In [86], pixels in RGB drone imagery are segmented into three classes: crop plants, weeds, and others, obtaining the highest accuracy using a FCN. In [87], a custom FCN combined with pre-computed features is used to separate crop plants from weeds in RGB-only images. In [88], a FCN is also used to segment drone images, but using NIR and NDVI features. In [89], an dataset of labeled weeds and plants is collected in an unsupervised manner [89] to avoid the high labeling cost. This dataset is then processed using a CNN. In [90], they estimate the canopy coverage of a forest, using only indices computed from the visible spectrum. A survey of image processing techniques for obtaining a vegetation mask can be found in [79].

2.3 Datasets For Geometric Phenotypes

In this section, we introduce the datasets we used in this thesis to estimate phenotypic traits. These datasets were acquired at the Agronomy Center for Research and Education (ACRE) at Purdue University over a field of sorghum plants. ACRE is a research farm field operated by Purdue University and located in West Lafayette, Indiana [91]. The data were collected by Professor Ayman F. Habib and the Digital Photogrammetry Research Group (DPRG).

Some of the aerial images, and particularly those used in Section 2.7, were collected from a DJI Phantom 2 UAV equipped with a GoPro Hero 3+ Black Edition RGB camera. Other images were collected from a DJI S1000+ octocopter equipped with a Sony Alpha 7R RGB camera with a 35 mm lens. Figure 2.2(a) shows the DJI Phantom 2, and Figure 2.2(b) shows the DJI S1000+. A gimball guarantees that the camera is always pointing at the nadir direction.



(a) (b)

Fig. 2.2.: (a) DJI Phantom 2 (b) DJI S1000+

Figure 2.3 shows an example of an image of our datasets from a GoPro camera. One can easily observe that the lens distortion is significant. In [92] the procedure to remove this distortion is explained. This process requires the calibration of the UAV system (camera and GNSS/INS system), and also generates an orthorectified mosaic image of the entire field. The system calibration procedure is described in [93, 94].

Summarizing, we analyze three different types of images:

- Perspective images. Original, distorted images as taken by the camera. Figure 2.3 shows an example.
- Distortion-free images. Perspective images whose lens distortion has been corrected. Figure 2.4 shows an example.
- Orthorectified mosaics. Combination of many of distortion-free images into a mosaic of the entire field, where every pixel is at the same distance to the camera. This is the result of the system calibration and orthorectification process described in [92–94]. Figure 2.5 shows an example.

Finally, for the method described in Section 2.5, we also analyze images taken from the "PhenoRover." The PhenoRover is our ground-based platform, originally converted from a crop sprayer platform with a custom boom. The PhenoRover is



Fig. 2.3.: Example of a perspective image, an original image with distortion, as taken from a GoPro camera. This is an extreme case of perspective distortion from the 2015 dataset, used for illustration purposes. Non-GoPro cameras used in posterior years do not show such high distortion.

driven by an operator along the field rows, and is equipped with a variety of sensors such as RGB cameras, hyperspectral cameras, LiDARs, and video cameras. Figure 2.6 shows a picture of the PhenoRover collecting data in the field.

It is worth noting that, even though color is an important feature in some of the techniques described in this thesis, we do not perform color calibration to our datasets. This would require a color calibration panel to be present in the field during data acquisition. Although this is typically required for hyperspectral data

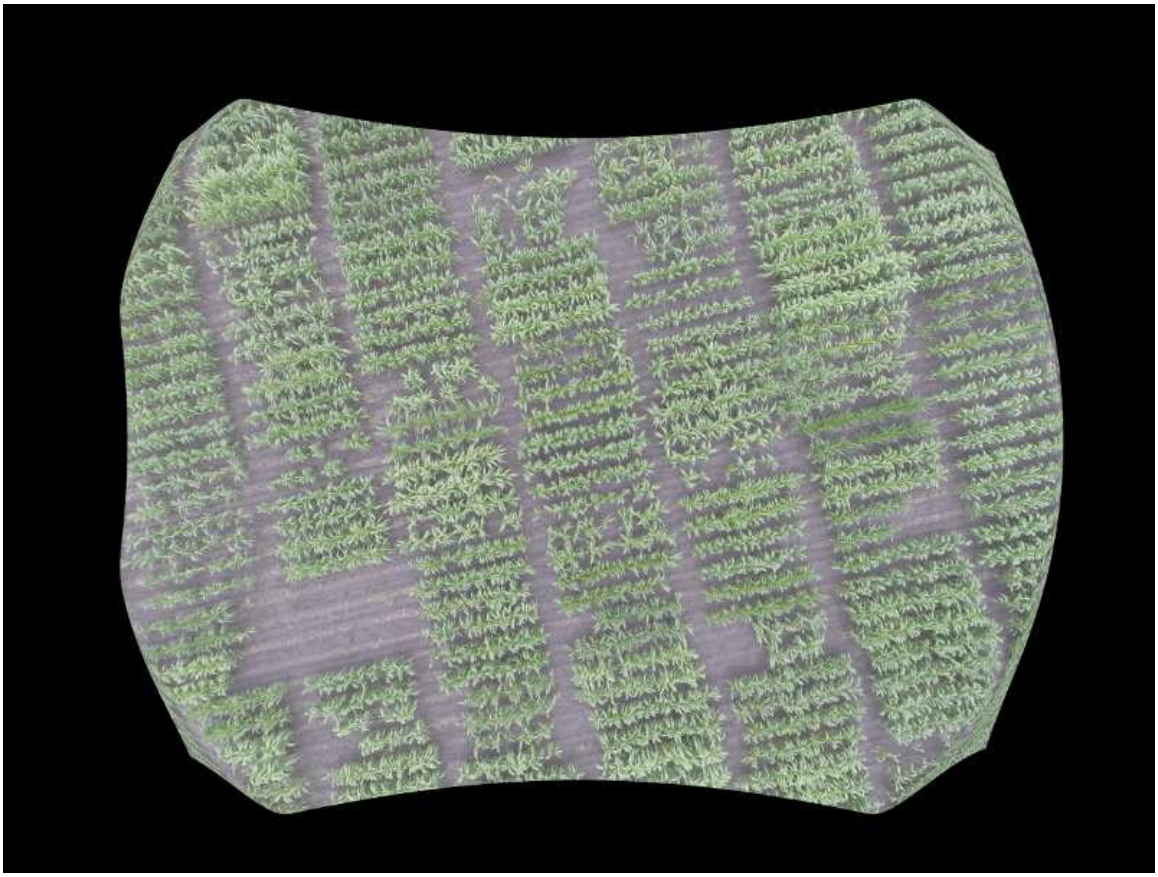


Fig. 2.4.: Example of a distortion-free image, where lens distortion has been removed.

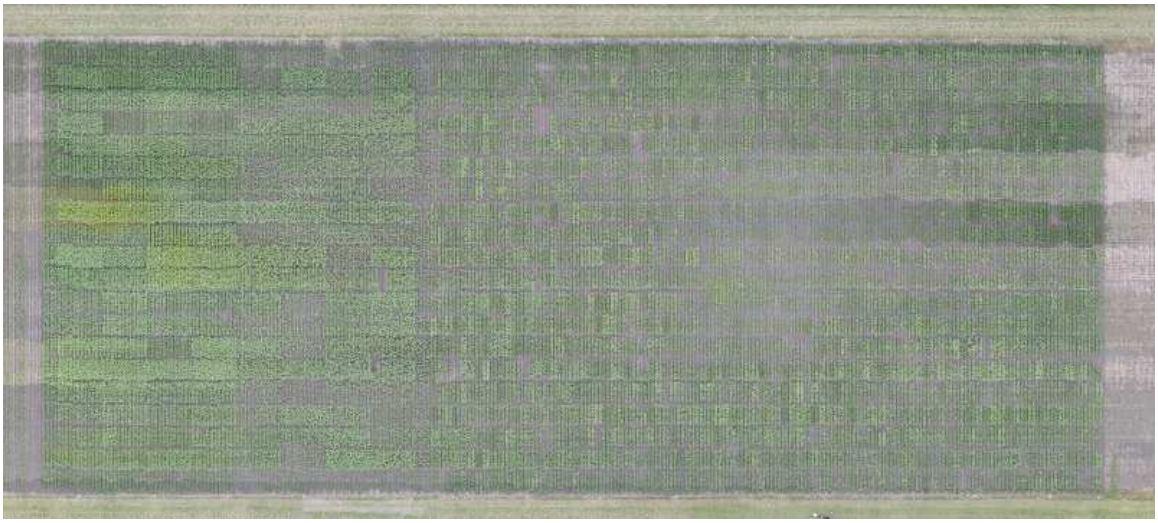


Fig. 2.5.: Example of an orthorectified mosaic of the entire sorghum field.



Fig. 2.6.: The PhenoRover, our ground-based platform.

collection, the analysis of the effect of RGB color calibration on our techniques is left for future work.

2.4 Plot Extraction From UAVs

Before estimating any plot-level phenotypic trait, plot boundaries must be to identified. This involves of segmenting the plots in the aerial images, making it possible to assign plot-level phenotypic traits to individual plots. We call this process "plot extraction." In this section, we describe a method to extract plots from orthophotos.

Our method extracts the plots from a panel with M rows and N ranges, thus containing MN plots. For definitions of panel, row, and range, see Section 2.1. Figure 2.7 shows an image comprised of two ranges and five rows.



Fig. 2.7.: Section of an orthophoto from June 21, 2016, showing two ranges and four rows, forming then eight row segments.

Our method makes four assumptions:

1. Orthophotos are very well georeferenced. Because the planting in the field is performed in straight lines, a well georeferenced orthophoto implies that the field rows are also straight in the orthophoto. Figure 2.8 shows an example of a very well georeferenced orthophoto.

2. The region of the orthophoto to be analyzed (usually a panel) has a constant number of rows (M) and ranges (N). M and N depend on the planting scheme, thus are known beforehand.



Fig. 2.8.: Example of a very georeferenced aligned orthophoto of a sorghum field with three panels, acquired on June 21, 2016.

3. The canopy is not completely closed (see Section 2.6 for the definition of canopy coverage). This means that there is some visible soil between the row segments.

4. For simplicity, our method extracts all row segments from a panel. This is not a limitation, and one can easily extract the plot boundaries by aggregating individual row segments.

First, the user must manually select a ROI of the orthophoto. This ROI must encircle a single panel, and does not have to be very precise, but must completely include the panel and a few pixels of surrounding soil. Figure 2.9 shows a possible ROI, corresponding to the central panel of the orthophoto shown in Figure 2.8. This ROI is necessary to guarantee our assumption of a constant number of rows and ranges (assumption #2). For simplicity, if the rows are not horizontal, we rotate the ROI the required number of degrees to align the rows horizontally. This simplifies the notation below.

Second, we generate a mask

$$I(x,y) = \begin{cases} 1 & \text{if pixel } (x,y) \text{ is plant material} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

using a segmentation technique such as the one described in Section 2.7. It should be noted that the segmentation method used to obtain $I(x,y)$ is not critical and any



Fig. 2.9.: Example of a ROI corresponding to the central panel of the orthophoto shown in Figure 2.8.

other segmentation technique could be employed. We also experimented using other segmentation techniques such as a GMM. However, according to visual inspection, the resulting segmentation mask was not as accurate as using the method described in Section 2.7 to implement Equation (2.2).

We then define the energy function

$$p_h(x) = \sum_{y \in \Omega} I(x, y), \tag{2.3}$$

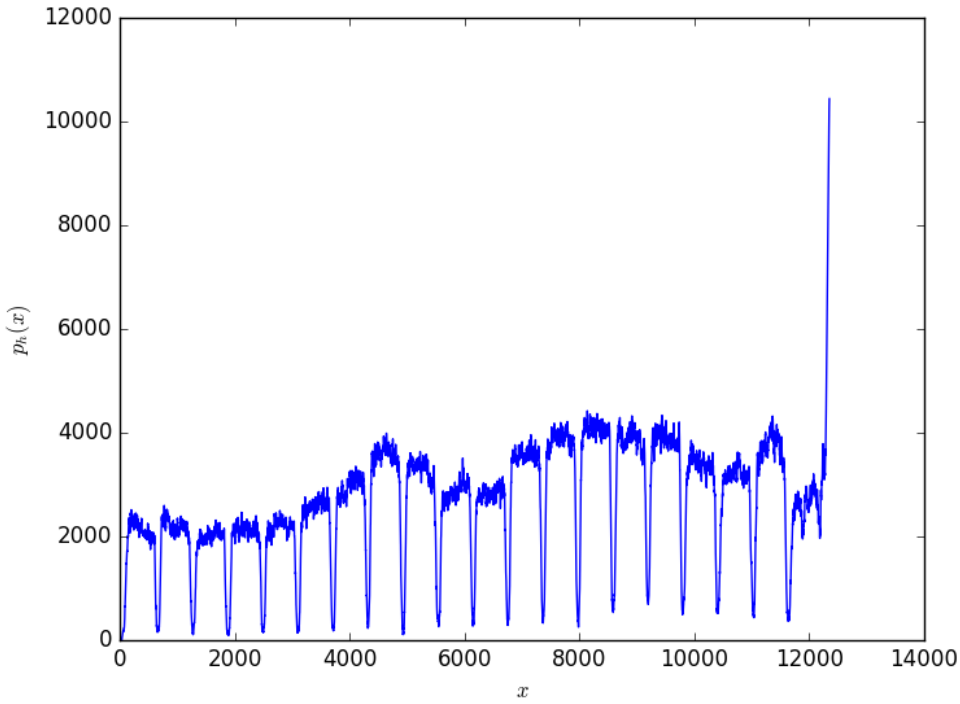


Fig. 2.10.: Horizontal profile, $p_h(x)$, of the ROI shown in Figure 2.9.

where $\Omega \subset \mathbb{Z}^2$ are all the pixels in the ROI, and the sum is performed vertically for each column x. We denote $p_h(\cdot)$ as the "horizontal profile." Figure 2.10 shows an example of a horizontal profile.

Our goal is to obtain a set of vertical lines $X_0, ..., X_N$ that separate the ranges of the panel. The first (X_0) and last (X_N) vertical lines are not between ranges, but between a range and the panel boundaries. The valleys in $p_h(\cdot)$ correspond to all these vertical lines. Ideally, these lines should not intersect with any plant, and only traverse along soil not covered by plants. In practice, $p_h(\cdot)$ at the valleys is greater than than zero because the rows of the field are not completely straight or there may be unexpected plants or weeds. The increase near $x = 12,500$ can be explained by grass and weeds from outside the field growing into the field.

We estimate the location of the range-separating lines as

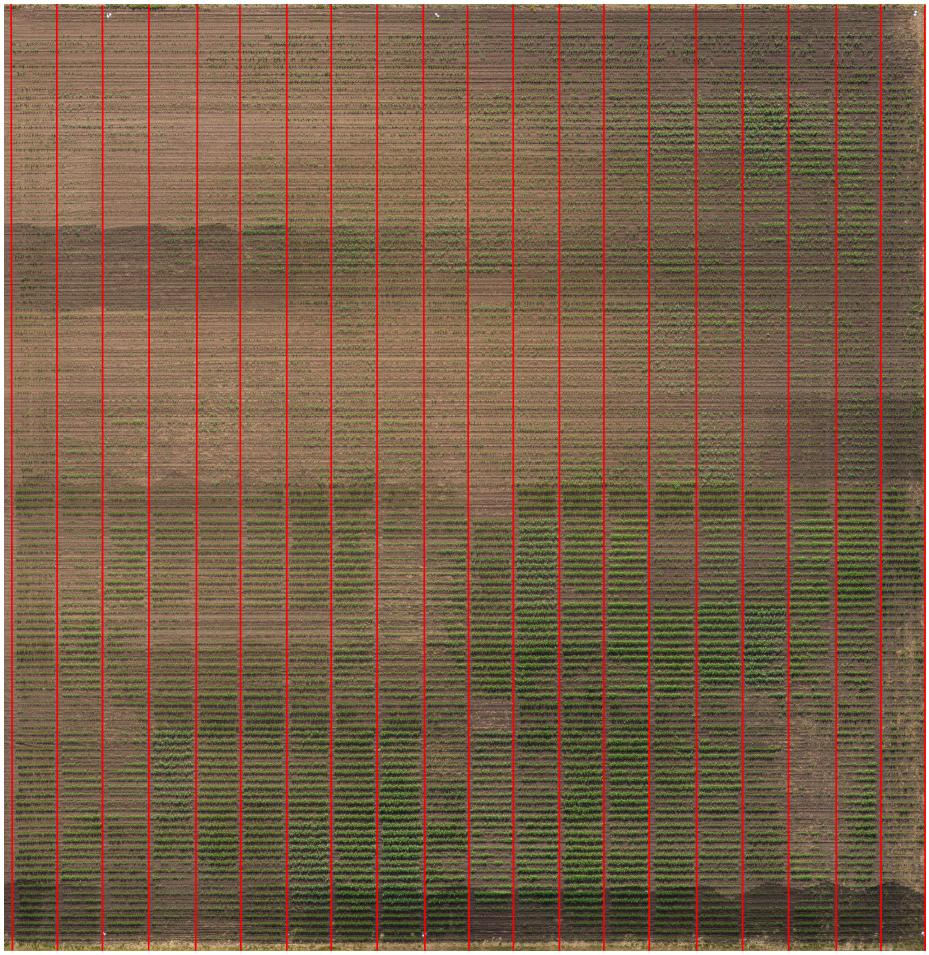


Fig. 2.11.: Vertical lines ($X_n, n = 0, ..., N$) separating ranges are shown in red color.

$$\hat{X}_n = \hat{X}_0 + n\hat{\Delta}X \tag{2.4}$$

$$\hat{X}_0, \hat{\Delta}X = \arg\min_{x_0, \Delta x} \sum_{n=0}^{N-1} p_h(x_0 + n\Delta x). \tag{2.5}$$

The minimization in Equation (2.5) is achieved by brute force. An equally-spaced grid of 100×100 $(X_0, \Delta X)$ candidate points is used. This is feasible because there are only two free variables, thus the grid is of a manageable size. After the best candidate is found, the solution is refined using the Nelder-Mead simplex method [95] to find a local minimum. An example of range-separating lines is shown in Figure 2.11.

Once the vertical lines $X_n, n = 0, ..., N$ are found, the same procedure is repeated within each range separately. The goal is to divide each row into ranges. For each range $g = 0, ..., N-1$, we want to obtain $M+1$ horizontal lines $Y_m^g, m = 0, ..., M$ that separate the range g into M rows. We select only the region of the mask that corresponds to a range g as

$$I_g(x,y) = \begin{cases} 1 & \text{if } I(x,y) = 1 \text{ and } X_g \leq x < X_{g+1} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

For each range g, a vertical profile is obtained as

$$p_v^g(y) = \sum_x I^g(x,y), \tag{2.7}$$

where the sum is horizontal for each image row y of the ROI. In each range g, we estimate the lines that separate field rows as

$$\hat{Y}_m^g = \hat{Y}_0^g + m\hat{\Delta}Y^g \tag{2.8}$$

$$\hat{Y}_0^g, \hat{\Delta}Y^g = \underset{y_0^g, \Delta y^g}{\arg\min} \sum_{m=0}^{M-1} p_v^r(y_0^g + m\Delta y^g). \tag{2.9}$$

Then, the mask that corresponds to the w-th row of all ranges is selected as

$$I_w(x,y) = \begin{cases} I^g(x,y) & \text{if } \exists\ g \mid \hat{Y}_w^g \leq y < \hat{Y}_{w+1}^g \\ 0 & \text{otherwise} \end{cases} \tag{2.10}$$

A new horizontal profile for each row w is obtained as

$$p_h^w(y) = \sum_y I^w(x,y), \tag{2.11}$$

and, again, the same procedure as before is repeated for each row $w = 0, ..., M-1$:

$$\hat{X}_n^w = \hat{X}_0^w + n\hat{\Delta}X^w \tag{2.12}$$

$$\hat{X}_0^w, \hat{\Delta}X^w = \underset{x_0^w, \Delta x^w}{\arg\min} \sum_{n=0}^{N-1} p_h^w(x_0^w + n\Delta x^w). \tag{2.13}$$

A bounding box is constructed around a row segment in the g-th range and w-th row as $\{(X_g^w, Y_w^g), (X_{g+1}^w, Y_{w+1}^g), (X_g^w, Y_{w+1}^g), (X_{g+1}^w, Y_{w+1}^g)\}$. These coordinates constitute the plot boundaries in image coordinates. Figure 2.12 shows the boundaries

around row segments. We can also construct the mask of the row segment in the g-th range and w-th row by combining the mask of rows and the mask of ranges as

$$I_w^g(x, y) = I_w(x, y) \wedge I^g(x, y), \tag{2.14}$$

where "$\wedge$" is the AND operator.

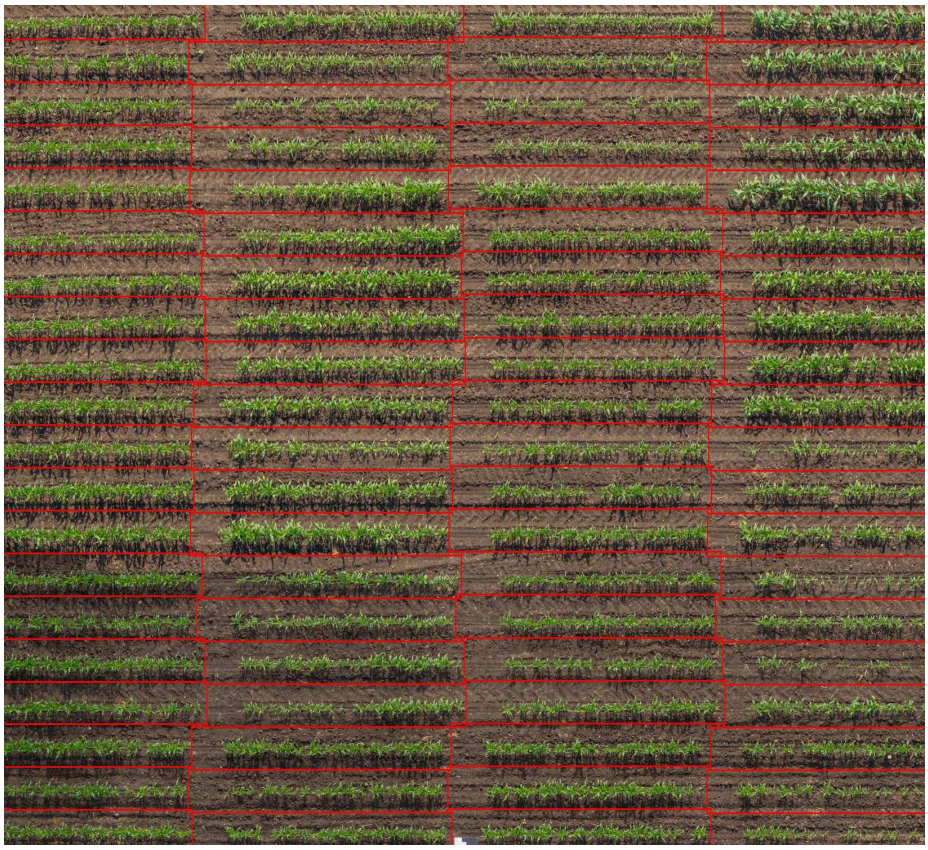


Fig. 2.12.: Plot boundaries obtained using the vertices $(X_g^w, Y_w^g), (X_{g+1}^w, Y_{w+1}^g), (X_g^w, Y_{w+1}^g)$, and (X_{g+1}^w, Y_{w+1}^g) of each row segment.

As the last step, we refine the plot boundaries. We can see in Figure 2.12 that there is a significant amount of space in the are between ranges (along the same row). This is because the mask is $I(x, y) = 0$ in this space (because there is no plant material), making the cost function flat. This extra space in the plot bounding boxes may impact phenotypic trait estimates based on area (such as canopy coverage). To prevent this, we "shrink" the plot boundaries horizontally, by moving the vertical

lines of each plot (X_g^w and X_{g+1}^w) towards the center of the plot, until plant material is found. This is described in pseudocode in Algorithm 1.

Algorithm 1 HorizontalShrinking

for $g = 0, ..., N-1$ **do**
  for $w = 0, ..., M-1$ **do**
    while $I_w^g(X_g^w, y) = 0 \quad \forall\ y \in [Y_w^g, Y_{w+1}^g)$ **do**
      $X_g^w \leftarrow X_g^w + 1.$
    while $I_w^g(X_{g+1}^w, y) = 0 \quad \forall\ y \in [Y_w^g, Y_{w+1}^g)$ **do**
      $X_{g+1}^w \leftarrow X_{g+1}^w - 1.$

The result of the horizontal shrinking, and the final estimate of the plot boundaries is shown in Figure 2.13.

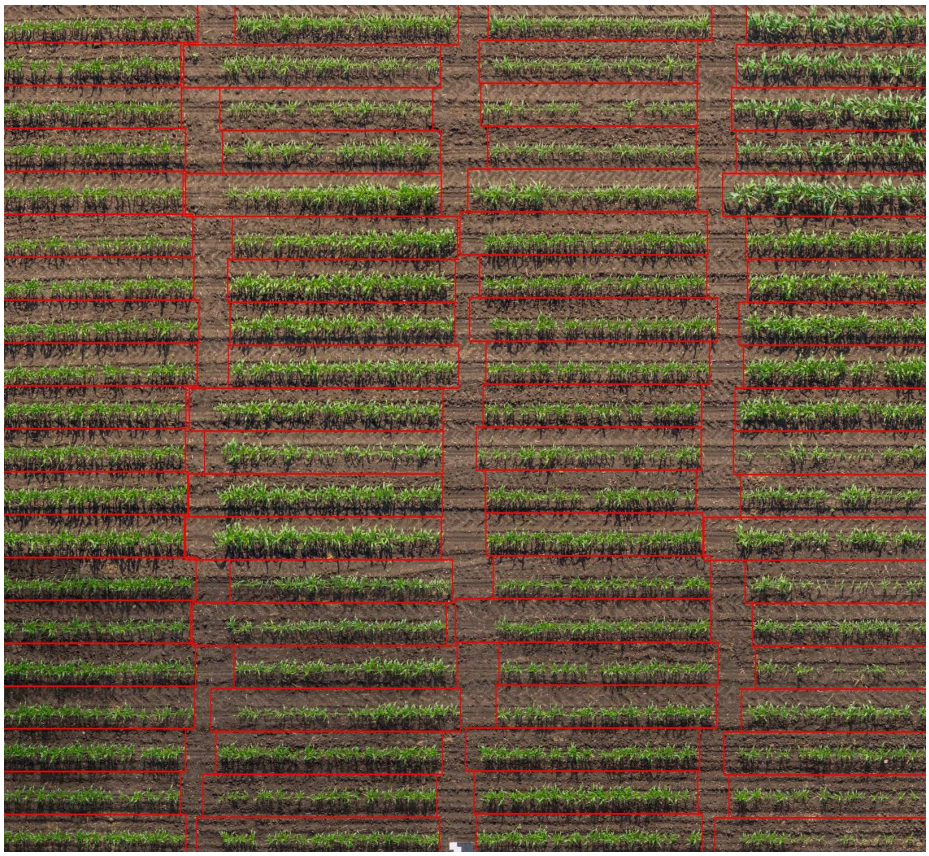


Fig. 2.13.: Plot boundaries after horizontal shrinking.

2.4.1 Non-constant spacing

Equations (2.8), (2.9), (2.12) and (2.13) assume that the rows (ranges) are equally spaced within a range (row). This assumption is valid when the orthophotos are sufficiently well aligned, such as the one in Figure 2.8. However, one needs to relax this condition when processing images such as the one in Figure 2.14(a). This image was generated by extracting the red, green, and blue channel of a hyperspectral camera. This is a line camera, which is very sensitive to perturbations in its motion. The image in Figure 2.14(a) is an extreme case that occured due to a low quality GPS/IMU. If consumer grade GPS/IMUs are used, this is the output that can be achieved without further extensive processing to obtain better geometry. In our case, these "unaligned" images were later post-processed to georeference all the hyperspectral data and extract plots from data where positioning is much more accurate. This example is useful to show the potential of this plot extraction method for irregular plot boundaries appearing for example when the full orthorectification process is not available.

To allow non-constant spacings, one can replace Equations (2.4) and (2.5) with

$$\begin{aligned} \hat{X}_0, ..., \hat{X}_{N-1} = \quad & \underset{x_0,...,x_{N-1}}{\arg\min} \frac{1}{N} \sum_{n=0}^{N-1} p_h(x_n) + \omega \frac{1}{N-2} \sum_{n=0}^{N-2} |\Delta x_n - \overline{\Delta x}|^2 \\ & \text{subject to } \Delta x_n \geq 0 \end{aligned} \tag{2.15}$$

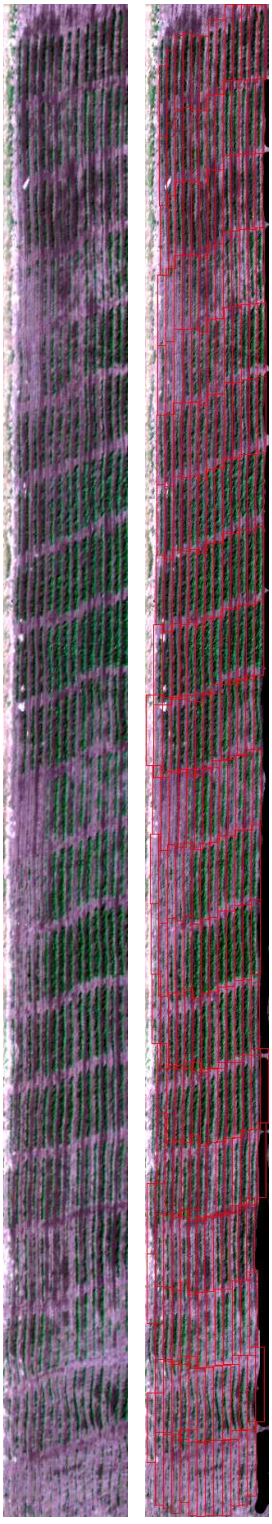
where

$$\overline{\Delta x} = \frac{1}{N-1} \sum_{n=0}^{N-2} \Delta x_n, \tag{2.16}$$

and

$$\Delta x_n = x_{n+1} - x_n. \tag{2.17}$$

The second term in eq. (2.15) is a prior term that corresponds to the variance of the spacing between lines. The higher the weight ω, the more we emphasize on the importance of constant spacing. The parameter ω must be manually selected by the user according to the quality of the orthophoto, as there is no available metric that we can use to cross-validate it.



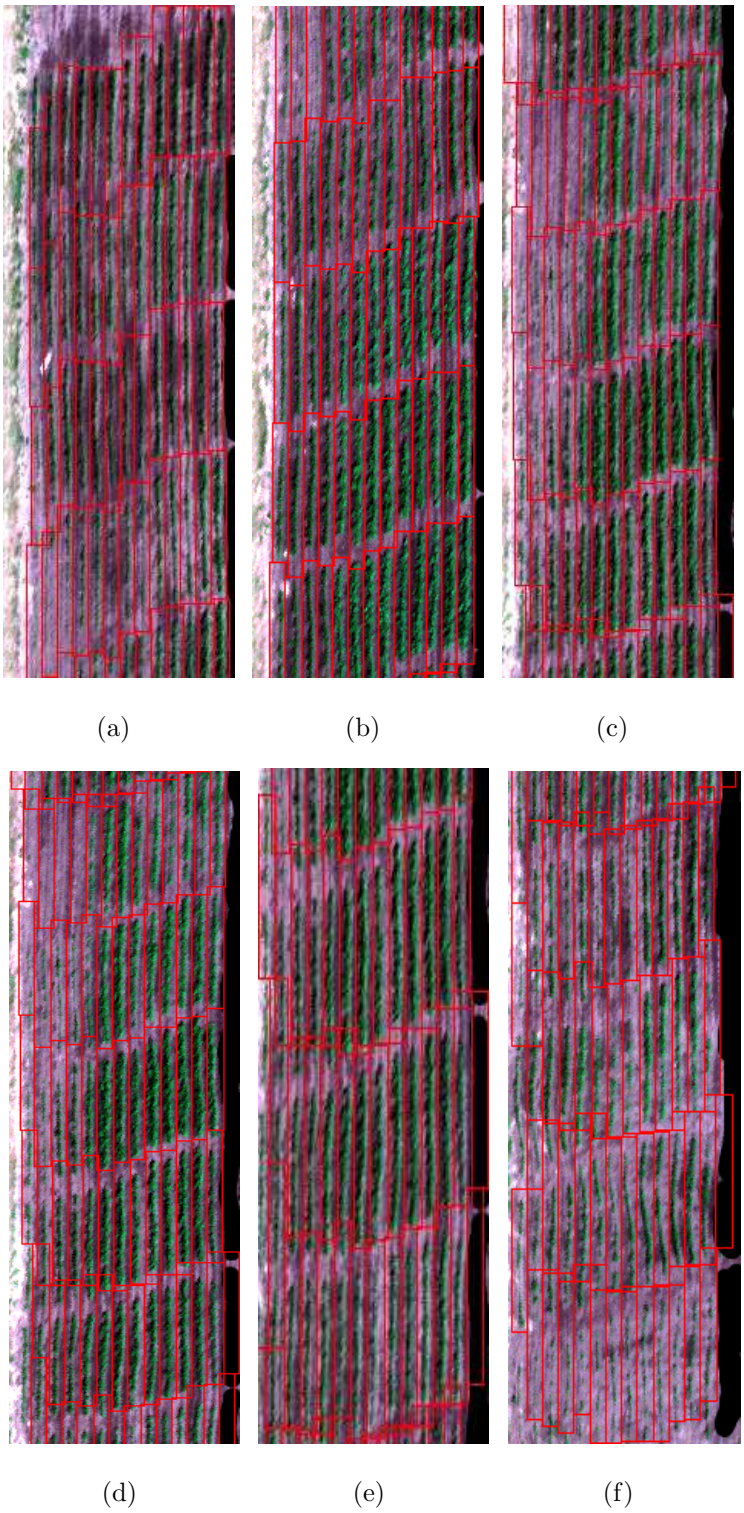
(a) (b)

Fig. 2.14.: (a) Image with irregular boundaries due to a low quality GPS/IMU. (b) Estimated plot boundaries.

Note that the constraint $\Delta x_n \geq 0$ enforces that lines are always consecutive. The optimization in eq. (2.15) now contains $N+1$ free variables instead of 2. Then it is no longer feasible to obtain the solution by brute force, because we would have 100^{N+1} points to evaluate. Also, the derivative is not available, thus we cannot employ techniques such as gradient descent. Instead, we use the Constrained Optimization BY Linear Approximation (COBYLA) method [96] to obtain a local solution to this optimization problem with constraints and without making use of the gradient. As an initial guess, we use the solution assuming constant spacing.

Figure 2.14(b) and Figure 2.15 show the result of this method allowing for non-constant spacing.

This method to extract plots from UAV data does not require training, and the observed average time to process a single image is 0.5 seconds.



(a) (b) (c)

(d) (e) (f)

Fig. 2.15.: Estimated plot boundaries, zoomed sections of Figure 2.14(b).

2.5 Plot Extraction From Ground-Based Platforms

An orthophoto may not always be available, or we may need a higher resolution than that possible in aerial images. For example, one may want to acquire close-up images from a ground-based platform to capture high resolution characteristics of the plants. Figure 2.16 shows two examples of images acquired from our ground-based platform, the Phenorover. Note that these images contain perspective distortion; thus, parallel lines are not imaged as parallel lines, but straight lines are imaged as straight lines [97]. This means that the field rows are no longer parallel in our image, and we cannot use the method described in Section 2.4.

In this section, we describe a method to extract the plots from an image acquired from a ground-based platform. This method is robust against perspective distortions, but not against radial distortions such as lens distortions if it is not corrected first. Figure 2.3 shows an example of an image with lens barrel distortion, where row fields are not straight. In summary, this method has two limitations:

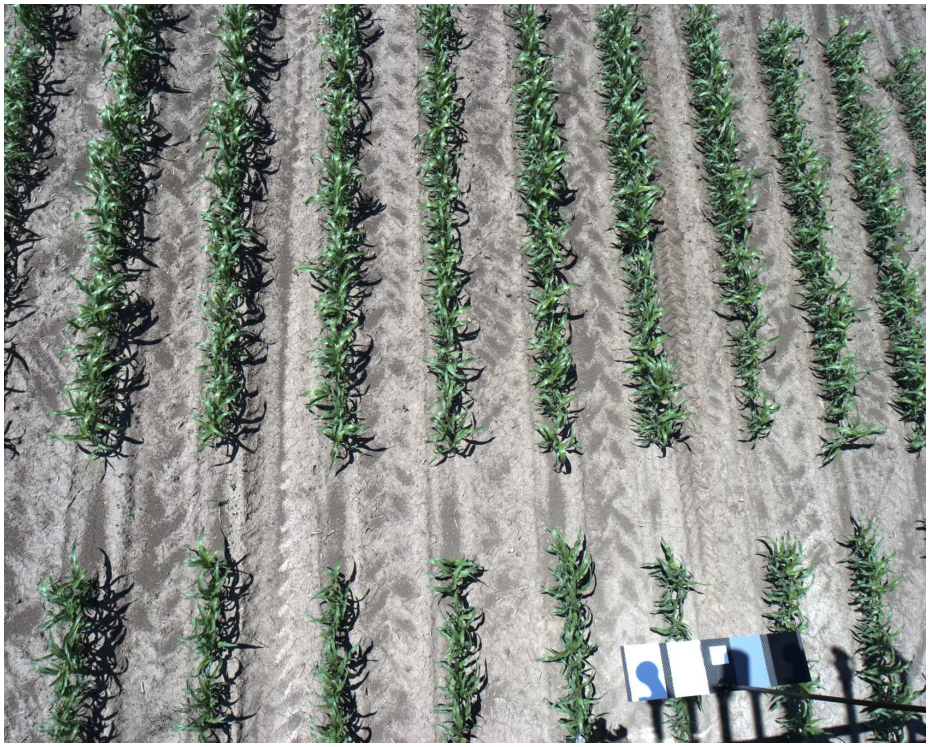
1. The image must not contain radial distortion. One can reduce radial distortions by choosing a camera lens with low lens distortion, or by correcting it as described in Section 2.3. In our datasets, the lens distortion of our RGB cameras is characterized, thus removing this limitation.

2. The method analyzes images separately, thus we need to georeference the images to know the world coordinates of each pixel. Otherwise, this method only segments the field rows, i.e., it assigns to each pixel with plant material to a field row.

We assume that the ground-based platform is moving along the direction of the field rows. Other appropriate names for this method would be “row segmentation” or “plot segmentation”. This row segmentation method works as follows.

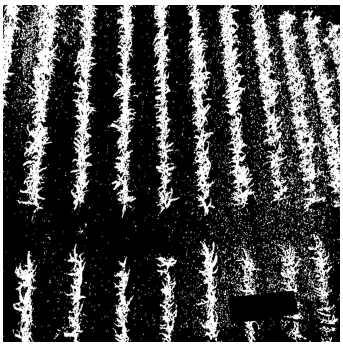


(a) Image acquired on July 13, 2018.

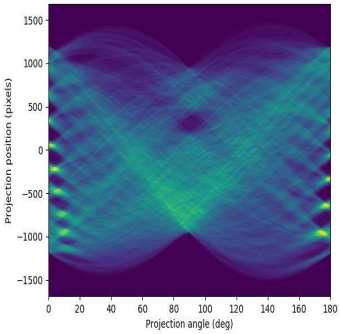


(b) Image acquired on June 6, 2018.

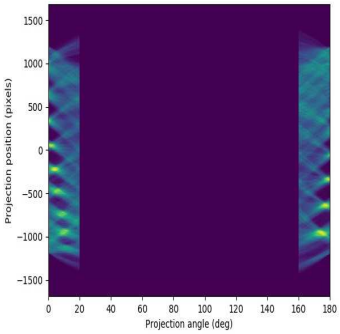
Fig. 2.16.: Two images acquired from our ground-based platform, the PhenoRover, at different dates.



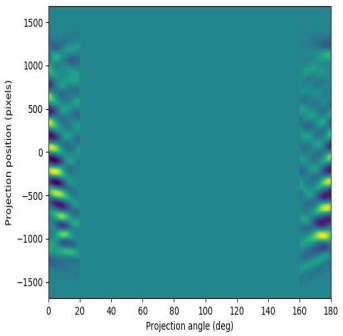
(a) Sorghum mask



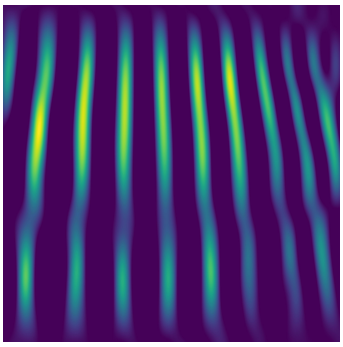
(b) Radon transform (sinogram)



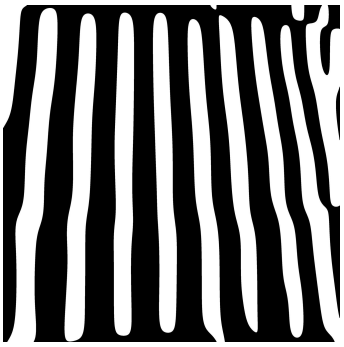
(c) Sinogram after removing non-vertical lines



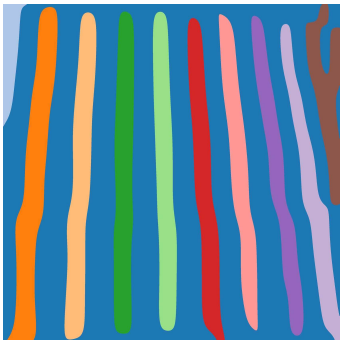
(d) Sinogram after band-pass filtering



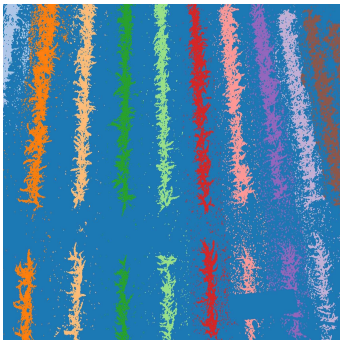
(e) Reconstruction (inverse Radon transform)



(f) Thresholded reconstruction



(g) Connected components



(h) Segmented mask



(i) Segmentation overlaid with original image

Fig. 2.17.: Intermediate steps of the plot extraction using a ground-based image.

Step 1: Vegetation mask

The first step is to obtain a binary mask indicating whether a pixel is plant material or not. This is also known as a vegetation mask. As before, we can use any segmentation technique. In our implementation, we use the method described in Section 2.6. Figure 2.16(b) shows the original and Figure 2.17(a) shows he segmented mask.

Step 2: Radon transform

The second step is to transform the vegetation mask using the Radon transform [98], and denote the Radon transformation as $R(\theta, r)$. The Radon transform is an integral transform that takes line integrals along all possible directions in the image. Figure 2.18 depicts the axes we use for the Radon transform. The origin of the coordinates $(x, y) = (0, 0)$ is in the center of the image. The vector $\hat{n}$ is defined as the normal of the projection line going through the origin. $\hat{n}$ is always defined as "pointing up," i.e, in the direction of the y axis, such that $\hat{n} \cdot y \geq 0$. θ is defined as the angle between $\hat{n}$ and the x axis, i.e.,

$$\theta = \arccos \frac{\hat{n} \cdot y}{|\hat{n}||y|}. \tag{2.18}$$

P is defined as the point in the projection line that is closest to the origin. r is the distance from the origin to P. Because $\hat{n}$ is always defined as "pointing up," it follows that $\theta \in [0, 180)$. To differentiate between two lines with the same θ and distance between the origin and P, we set r to be negative if it is in the lower half of plane.

The Radon transform is an invertible transform, and points in the original space are converted into sinusoids. Straight lines in the original space make these sinusoids intersect in a point in the transformed space. A detailed description of the Radon transform can be found in [98]. In this method, we use the scikit-image v0.14.1 [99] implementation, which uses the center of the image as the origin.

Figure 2.17(b) shows the result of the Radon transform. Note that the non-parallel thick lines that correspond to the field rows are mapped into points of high value close to the edges of the transformed image. If the field rows were completely vertical, these points would have a projection angle of 0° or 180°. In Figure 2.17(a), one can see that the further the field row is from the origin of the image, the more tilted it is.

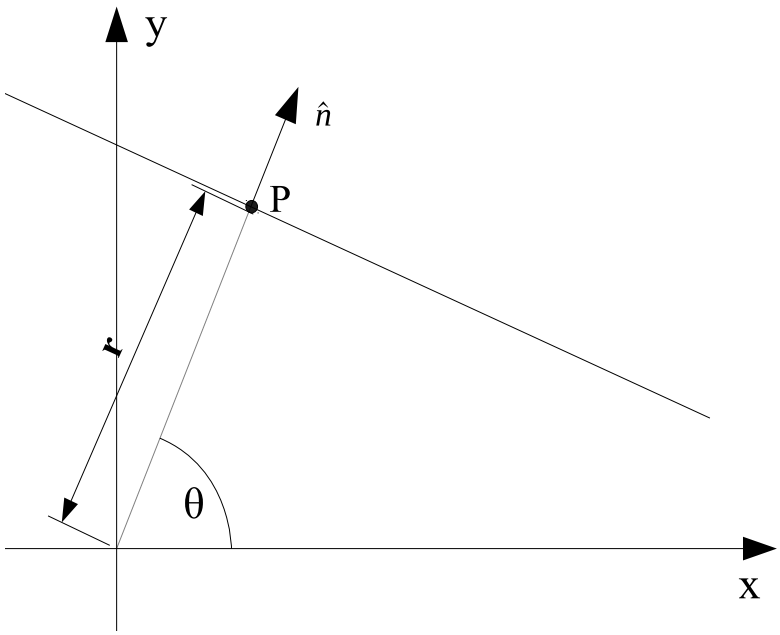


Fig. 2.18.: Diagram of the Radon transform.

Equivalently, we can see in Figure 2.17(b) that the higher $|r|$ is, the closest θ is to $90°$.

Step 3: Filtering in the Radon space

As the ground-based platform is moving along the field rows, we know that the field rows will be somewhat vertical. Thus, we zero out all the areas of the Radon transform that are not vertical enough, i.e.,

$$R(\theta, r) \leftarrow 0 \quad \forall\ \theta \in [20°, 180 - 60°]. \tag{2.19}$$

Figure 2.17(c) shows this intermediate result.

Then, we want to highlight the prominent points in he Radon transform. These points are approximately periodic, according to the spacing between the field rows in the original image. We can see that areas in the Radon transform where there is high frequency are of no interest, because the sinusoids are not intersecting coherently. Also, flat areas (low frequency) do not correspond to any pattern in the original

image. To highlight the prominent points, we use a band-pass the filter, in particular the Laplacian of Gaussians. The result is shown in Figure 2.17(d).

Step 4: Inverse Radon transform

Then, the inverse Radon transform is computed. This is also called backward transformation. The reconstructed image is shown in Figure 2.17(e). Note that patterns in the sorghum mask that do not contribute to (approximately) vertical lines have been removed. In particular, there is no longer a spacing between the ranges, and the object occluding the plants has been removed. Only long, wide lines that correspond to field rows are preserved. Row segments that are very close to the edges of the image are distorted because there are not enough mask points to form a cluster of straight lines.

Step 5: Separating field rows

We select the pixels that have a high likelihood to be very close to a field row by using a threshold of 10% of the maximum value in the reconstruction. The thresholded reconstruction is shown in Figure 2.17(f).

Then the field rows are separated by using connected components. In Figure 2.17(g), one can see each component labeled with a different color.

Step 6: Plant material segmentation

Finally, the pixels in the vegetation mask are labeled with the same label as the closest pixel in the thresholded reconstruction. In other words, we use the k-nearest neighbors algorithm, with $k = 1$, to classify the pixels in the original mask. The labeled mask of plant material is shown in Figure 2.17(h), and Figure2.17(i) shows the original image combined with the labeled mask using alpha-blending.

This method to extract row segments using ground-based imagery does not require training, and the observed average time to process a single image is about 25 seconds.

2.6 Canopy Coverage

The canopy coverage of a plant population is defined as the percentage of land that the vertical projection of the plants covers [100]. Denote as A the rectangular area where the plants are planted. This usually corresponds to the area of a plot. When looking at the plants vertically from the sky, and if the canopy is not completely covered, one will see some soil between the plants. Denote the area of the visible soil as S. The canopy coverage is then defined as

$$C = \frac{A - S}{A} \tag{2.20}$$

Canopy coverage is 100% when the plants cover all space available ($S = 0$). Figure 2.19 shows an example of plants with low canopy coverage, and Figure 2.20 an example with high canopy coverage.



Fig. 2.19.: Example of low canopy coverage. Drone image acquired on June 4th, 2018, at 20 meters of altitude.

Canopy coverage is different, but is related to other traits of interest in agronomy, plant breeding, and forestry, such as canopy closure, and LAI [100, 101]. In Fig-



Fig. 2.20.: Example of high canopy coverage. Drone image acquired on July 2nd, 2018, at 20 meters of altitude.

ure 2.21, the difference between canopy coverage and canopy closure is clearly shown. As described in its definition, the lines when measuring canopy coverage must be vertical projections. Note that when measuring canopy coverage from aerial imagery, we can consider these nadir-looking lines as each pixel in the orthophoto.

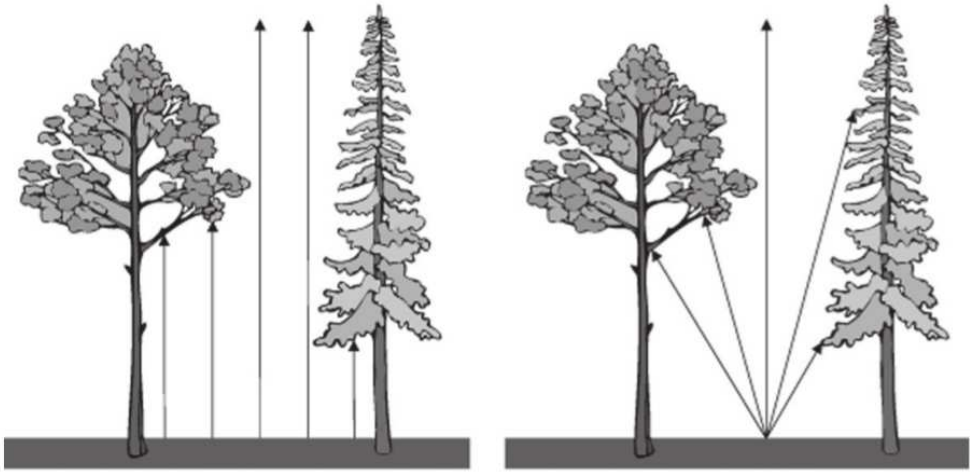


Fig. 2.21.: Left: Canopy coverage. Right: Canopy closure. In this work, we focus on canopy coverage because it can be estimated from nadir-looking aerial cameras. Image source: [101].

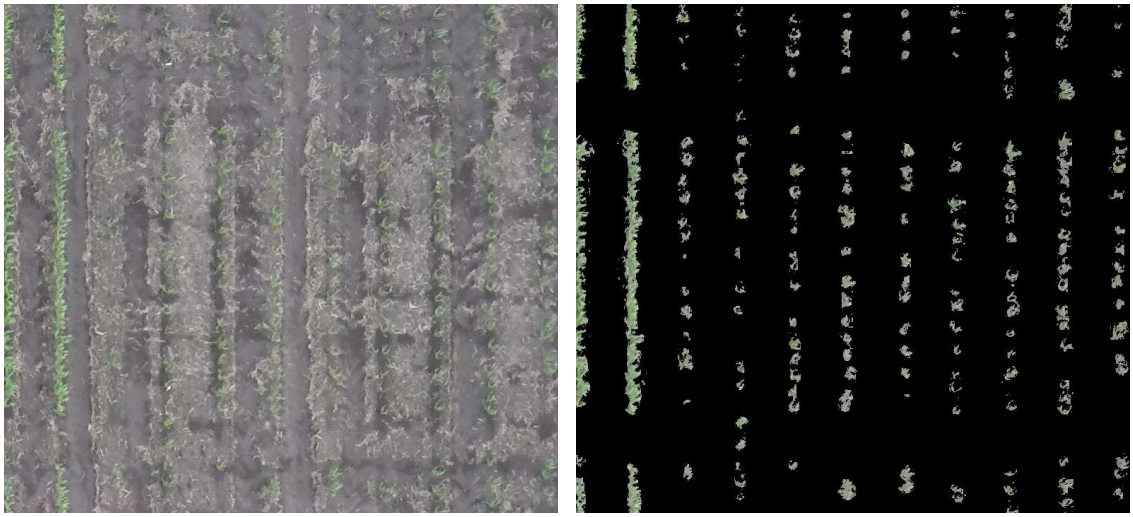
2.7 Leaf Counting

As we can see from the images in Figures 2.3 and 2.4, discerning between leaves is challenging in this dataset. In this section, we propose a method to estimate leaf count without individually segmenting each leaf.

First, the distortion-free image or orthomosaic is converted from RGB to HSV color space. The method for the color transformation from RGB to HSV can be found in [102]. From the image in the HSV color space, a segmentation mask Y is generated. Each pixel Y_m in this mask is obtained as:

$$Y_m = \begin{cases} 1 & \text{if } \tau_1 \leq H_m \leq \tau_2 \text{ and } (\tau_3 \leq S_m \text{ or } \tau_4 \leq V_m) \\ 0 & \text{otherwise,} \end{cases} \tag{2.21}$$

where H_m, S_m, and V_m are the hue, saturation, and value of the pixel m. $m = 0, ..., M-1$. M is the number of pixels in the image. The thresholds τ_1, τ_2, τ_3, and τ_4 are determined experimentally. τ_1 and τ_2 select the characteristic green color of the leaves. τ_3 and τ_4 prevent misclassifying some soil pixels as leaves. An example of the segmentation result is shown in Figure 2.22.



(a) (b)

Fig. 2.22.: (a) Section of an orthorectified mosaic. (b) Segmentation mask.

Then, the number of pixels classified as sorghum leaves is determined as

$$\alpha = \sum_{m=0}^{M-1} Y_m. \tag{2.22}$$

This pixelwise segmentation exploits the strong color difference between the sorghum leaves, the soil, and the panicles. Leaves are generally green or yellow at the plant's senescence. Soil is usually very poorly saturated, usually appearing in the images as brown or gray. Panicles, in contrast, can have different colors. They are usually brown, but they can also be red, white, or yellow [36].

Finally, we want to estimate the number of leaves, denoted as λ, from α. In order to do this, we assume that the number of leaves and the number segmented pixels are linearly related as

$$\lambda = \frac{\alpha}{\rho}. \tag{2.23}$$

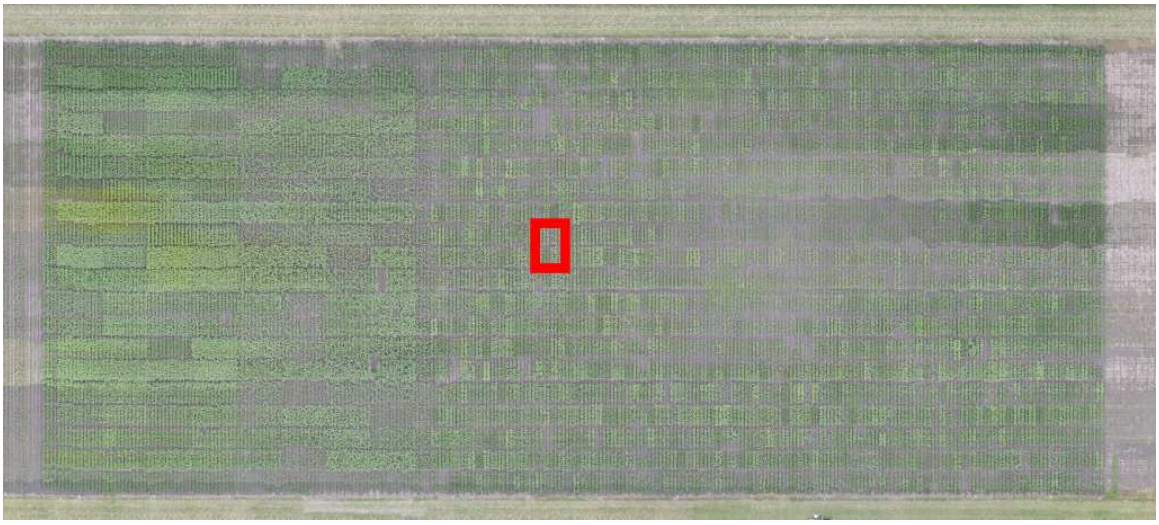
This assumes that all leaves have approximately the same area. ρ is the number of pixels per leaf.

In order to calibrate ρ, a small region of the image is selected. The number of leaves in this region is manually counted and denoted as λ_0. The number of pixels

classified as sorghum is denoted as α_0. Finally, ρ is estimated by $\rho = \frac{\alpha_0}{\lambda_0}$, and the final leaf count can be obtained with Equation 2.23.

We used our best judgment to obtain the ground truth of the leaf count from the images themselves. From this ground truth, we observed that the relationship between the number of leaves and the number of sorghum pixels is approximately linear. To show this, a small section of a orthorectified mosaic from July 15, 2015 is cropped as shown in Figure 2.23(a). This section is shown in Figure 2.23(b). Each of the 16 sorghum subrows is ground truthed. Each subrow contains the same sorghum phenotype. The obtained ground truth is shown in Figure 2.23(c) next to each subrow and plotted in Figure 2.24.

As all pixels contribute in the same manner to the leaf count, this method requires that all leaves are at the same distance from the camera. This condition is fulfilled when using the orthorectified mosaic. Also, only leaves that are visible can be counted.



(a)



(b)



(c)

Fig. 2.23.: Ground truth of a region of the orthorectified mosaic generated from the imagery collected on July 15, 2015. The leaf count of a total of 16 sorghum subrows is ground truthed.

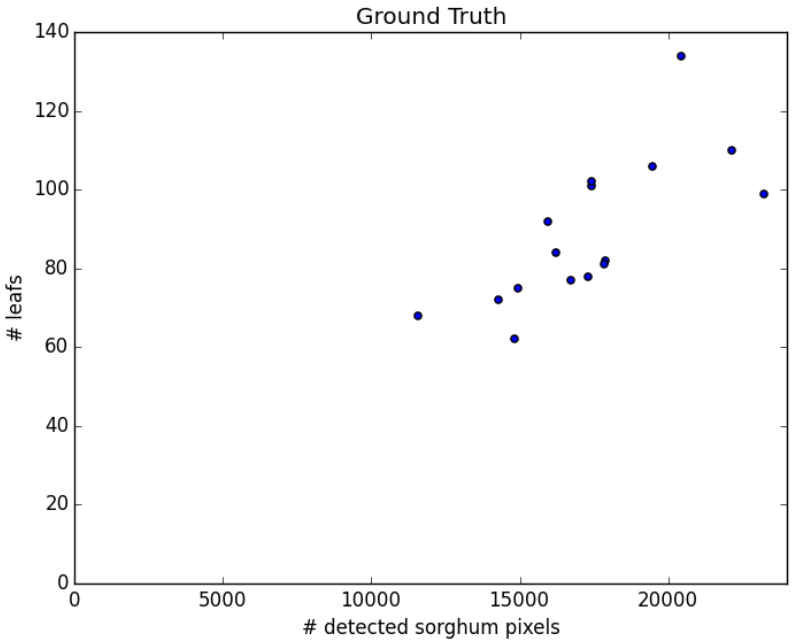


Fig. 2.24.: Relation between leaf count and number of leaf pixels seems linear.

2.7.1 Experimental Results

The image of Figure 2.23(b) is analyzed as follows. The ground truth of one of the 16 subrows (yellow rectangles) is used as training data to calibrate ρ. The resulting value of ρ is used to estimate the leaf count of the remaining 15 subrows. In other words, 6.25% of the data is used for training, and 93.75% for testing. This is repeated, using a different subrow every time. We compute the accuracy of each estimate as

$$100\left(1-\frac{|\lambda - GT|}{GT}\right) \tag{2.24}$$

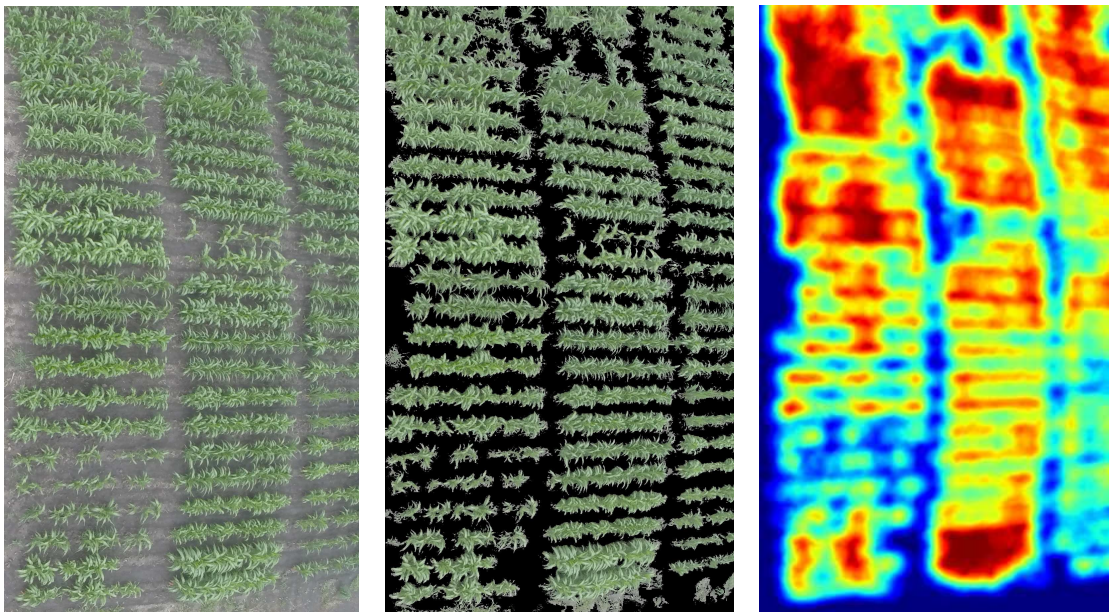
where λ is the estimated leaf count, and GT is the ground truth. Figure 2.25 shows the accuracy over testing data when using each of the rows. The average of the 16 estimates is 87.7 %.



Fig. 2.25.: Accuracies using each subrow as training data for the estimation of ρ.

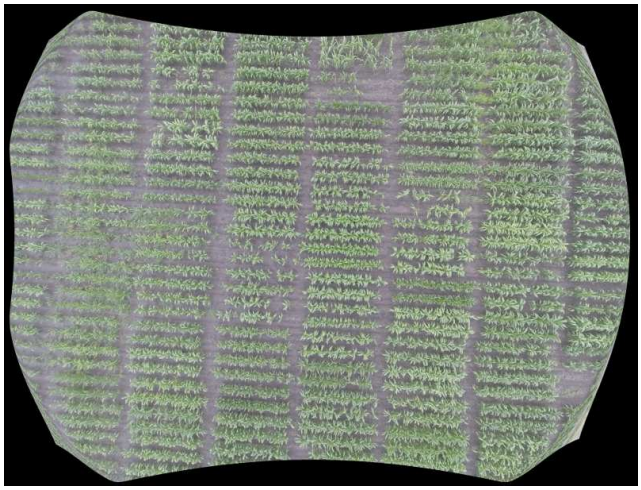
This method is also applied to perspective images, distortion-free images, and orthorectified mosaics from June 15, June 26, July 06, and July 15, 2015. Figure 2.26 shows the leaf segmentation mask of a perspective image taken on July 15, 2015. The local density of leaf pixels is also shown as a heat map. In this heat map, the value of a given pixel is the number of leaf pixels in the neighborhood of that pixel. The size of the neighborhood is set to 41×41 by manually selecting one of the plants. From this heat map we can easily visualize which regions of the image

have higher density of plant leaves. Blue corresponds to the minimum value of leaf density (0), and red corresponds to the maximum ($41 \cdot 41 = 1681$). Figure 2.27 shows the leaf segmentation mask of a distortion-free image taken on July 15, 2015. Figure 2.28 shows the leaf segmentation mask of an orthorectified mosaic. The perspective images used to generate the mosaic were taken on June 24, 2015.

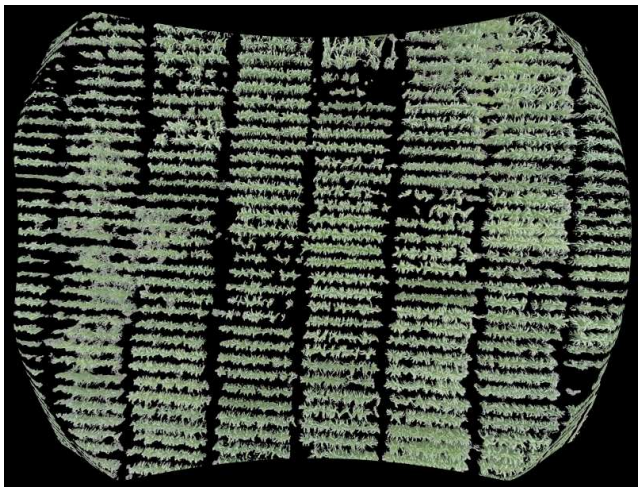


(a) (b) (c)

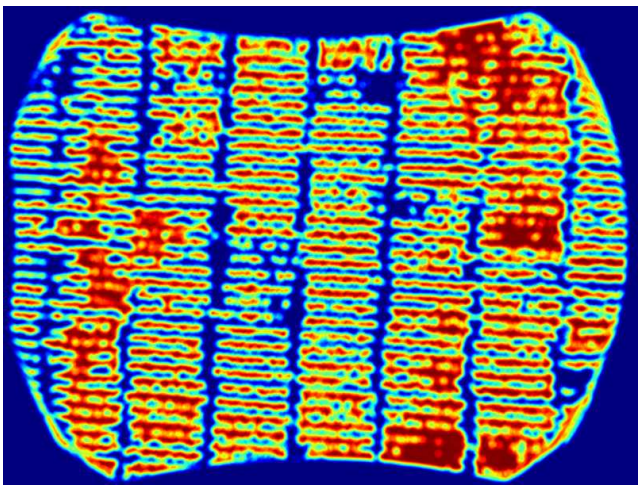
Fig. 2.26.: (a) Region of a perspective image. (b) Leaf segmentation mask using the color-based method. (c) Heat map of local leaf density.



(a)

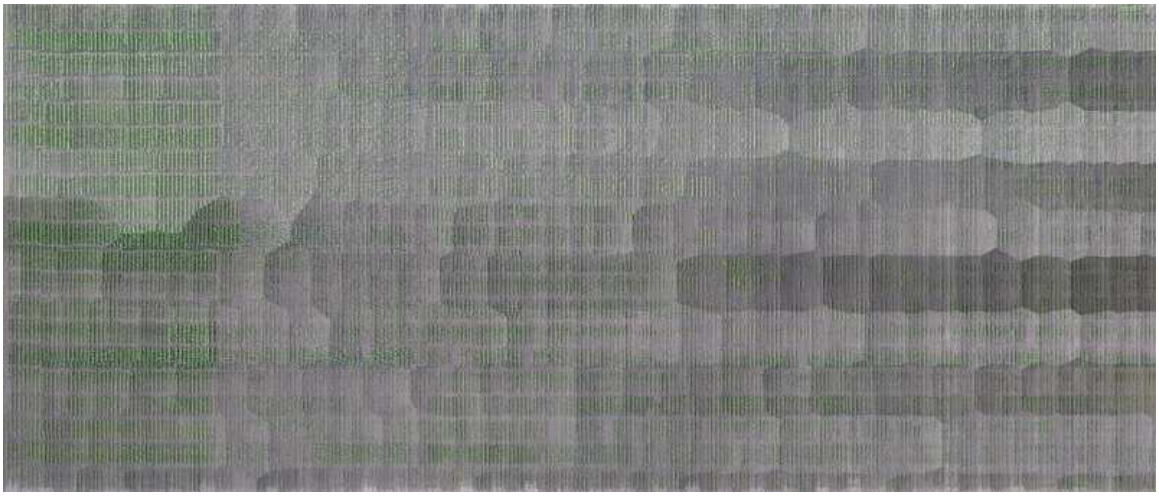


(b)



(c)

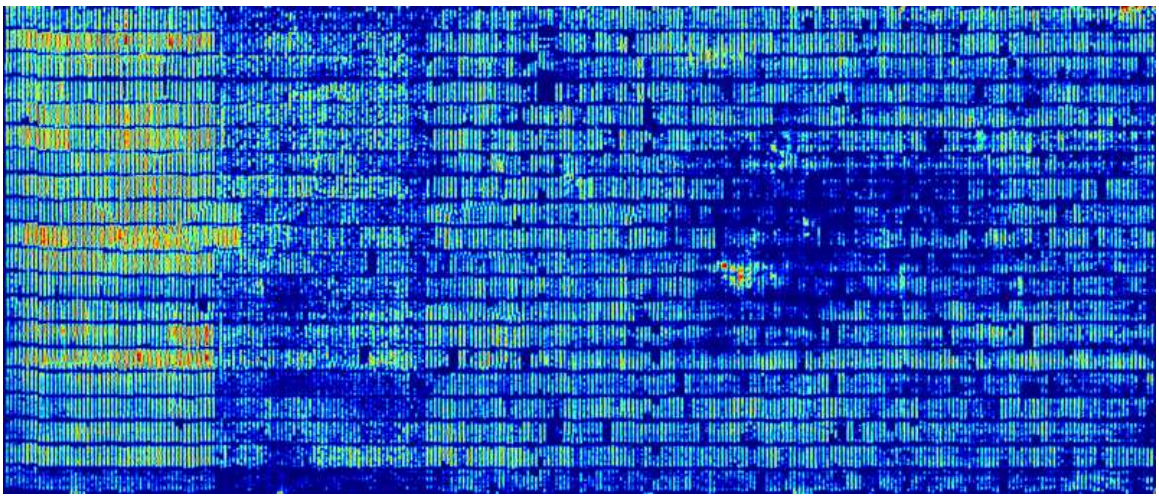
Fig. 2.27.: (a) Distortion-free image. (b) Leaf segmentation mask using the color-based method. (c) Heat map of local leaf density.



(a)



(b)

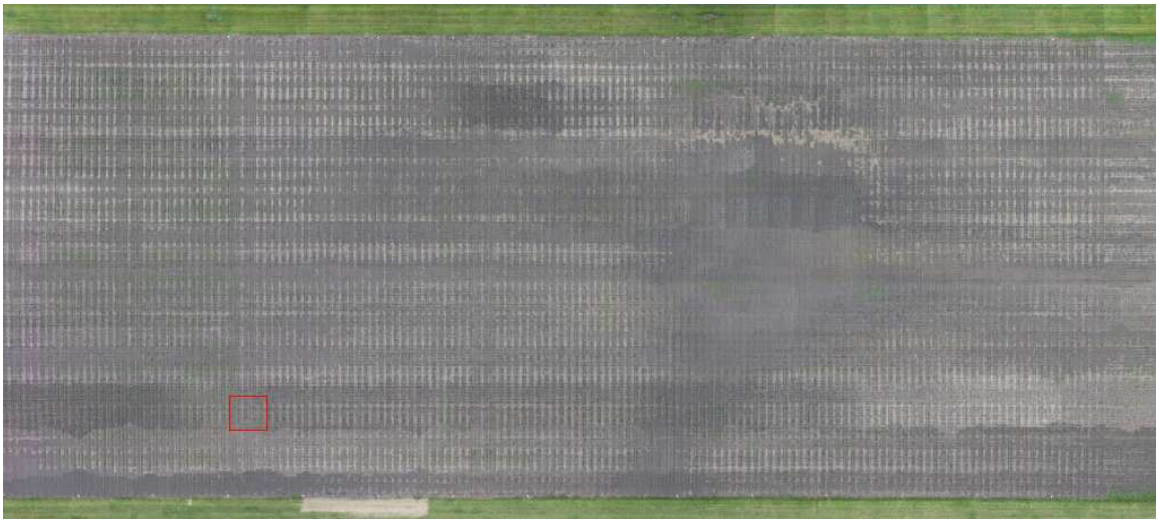


(c)

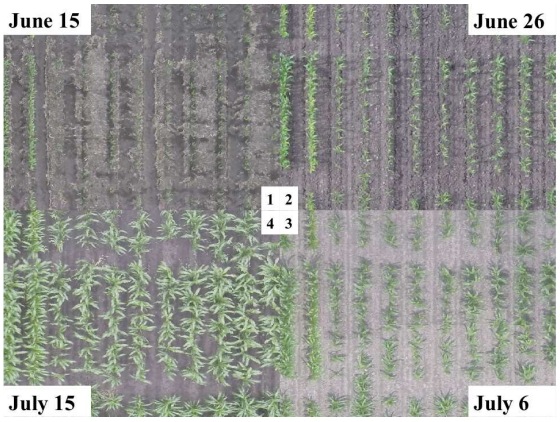
Fig. 2.28.: (a) Orthorectified mosaic. (b) Segmented leaves. (c) Heat map of local leaf density.

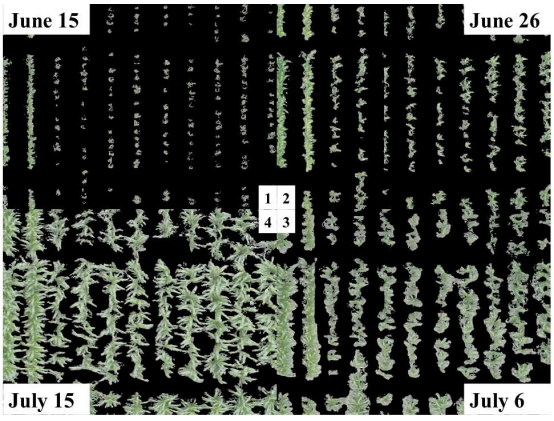
Also, the evolution of the number of leaves in a small region of the field is tracked throughout time. We selected the same exact region of the field (shown in Figure 2.29(a)) for the dates of June 15, June 26, July 06, and July 15, 2015. This is possible because the orthorectification process guarantees that the same pixel location in every orthomosaic corresponds to the same location in the real world. Figure 2.29(b) shows the same region for the 4 different flight dates. The number of leaves for each date is estimated and plotted in Figure 2.29(d).

The thresholds we used for all these images were experimentally chosen as $\tau_1 = 30$, $\tau_2 = 79$, $\tau_3 = 30$, and $\tau_4 = 163$. We did not see high sensitivity in these parameters, and we could mantain these values for all the dates we analyzed. These thresholds are Hue, Saturation, and Value (HSV) values in the range $[0, 0, 0]$ to $[180, 255, 255]$ for 8 bit images.

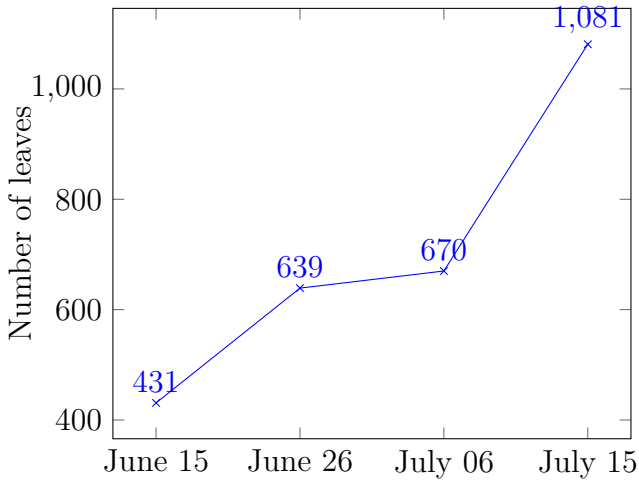


(a)





(b) (c)



(d)

Fig. 2.29.: (a) Selected region of the field. (b) Same exact region of the field on 4 different flight dates. (c) Segmented leaves in the same region on the 4 different dates. (d) Time evolution of the number of leaves.

3. PHENOTYPING: PLANT LOCALIZATION

3.1 Introduction

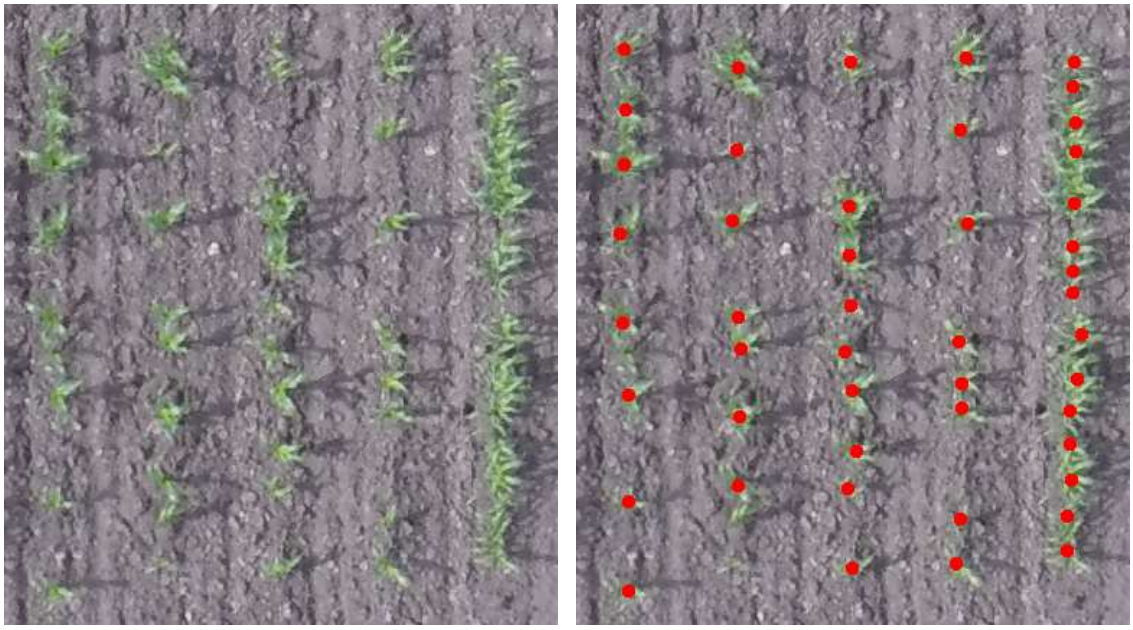
In this chapter, we describe multiple methods to estimate the location of plants in a field. We define the location of a plant as the coordinate where the stalks intercepts the ground. More precisely, we also call this "plant center."

A single sorghum plant may contain more than one tiller. A tiller is a side shoot that grows after the initial, main shoot has grown from the seed [103]. Ideally, one should distinguish between the secondary tillers of a plant and its main stem. For example, corn (maize) usually has a single stem, making this distinction unnecessary, but in sorghum it is more common to have multiple tillers. Because of this, one should use the term "stem center." However in practice, distinguishing a tiller from the main stem is a very challenging task, specially from remote imagery.

Figure 3.1 shows ground truthed locations of sorghum plants. The image is a section of the orthomosaic from June 26, 2015.

Locating plant centers is a critical first step in estimating plant-level phenotypic traits. One must first identify individual plants to assign phenotypic trait estimates (such as leaf count) to a particular plant. Also, locating and counting are often seen as two sides of the same coin [104]. Thus plant counting can be seen as a side task of plant location. Other plot-level traits such as plant height or intra-row spacing also require knowledge of the location of the plant beforehand.

In this chapter, we describe methods to locate plants in UAV imagery. Assuming images are georeferenced, once we know the plant location in the image, we can compute the real-world coordinates of plant centers.



(a) (b)

Fig. 3.1.: (a) Section of the orthorectified mosaic from June 26, 2015. (b) Ground truthed location of each sorghum plant.

3.2 Overview Of Previous Work

Generic object detectors. One could consider plant localization as a particular case of the generic object detection task. In such task, the goal is to estimate the location of all the objects in an image. Then we could use a generic object detector to estimate plant locations. Because of this, we also briefly review the literature of generic object detectors. Recent advances in machine learning, and in particular deep learning [105, 106], have increased the accuracy of localization tasks such as object or keypoint detection. In Fast R-CNN [107], regions in the image that potentially correspond to an object are called "region proposals". Region proposals are generated by classical methods such as selective search [108]. Although activations of the network are shared between region proposals, Fast R-CNN cannot be trained end-to-end. Faster R-CNN [107,109] introduces a Region Proposal Network (RPNs) that allow for

end-to-end training of models. Mask R-CNN [110] extends Faster R-CNN by adding a branch to predict a segmentation mask indicating which pixel belongs to which object. However, this branch runs in parallel with the already existing branch for bounding box recognition. Mask R-CNN could be used to estimate human pose keypoints by indicating in the segmentation mask the presence of the keypoint. However, the loss function in Mask R-CNN is applied location by location, making the keypoint detection highly sensitive to alignment of the segmentation mask. The Single Shot MultiBox Detector (SSD) [111] also estimates fixed-sized bounding boxes and scores that indicate the presence of objects. The described methods require either ground truthed bounding boxes for training, or require to know beforehand the maximum possible number of objects in the image. In [112], they make the observation that generic object detectors such as Faster R-CNN and SSD perform very poorly for small objects.

Counting and locating objects. Counting the number of objects in an image is not a trivial task. A common approach is to estimate a density function whose integral corresponds to the object count [113]. In [114], Shao et al. proposed two methods for locating objects in images. One method first locates and then counts, and the other method first counts.

Two of the datasets that we will use to evaluate our proposed methods consist of heads of people and pupil centers. Because of this, we also briefly review the literature of crowd counting and pupil tracking. Locating and counting people is necessary for applications such as crowd monitoring for surveillance systems, surveys for new businesses, and emergency management [113, 115]. There exist multiple studies in the literature that describe methods to detect and track where people in videos [116, 117]. Some of these detection methods use bounding boxes around each human as ground truth. When the overlap between people is high, such as in sports events or agglomerations in public transport stations, acquiring bounding boxes for each person in a crowd can be labor intensive and imprecise. More modern approaches that estimate a density map avoid the need of bounding boxes. In these approaches,

the label of the density map is constructed from the labels of the people's heads. This is done by centering Gaussians at the center of each head. Zhang et al. [118] estimate the density map by using a Multi-column CNN (MCNN) that learns features at different scales. In [119], Sam et al. train multiple independent CNNs to predict the density map at different crowd densities. Then an additional CNN is used to classify the density of the crowd scene and the appropriate CNN is used. Huang et al. [120] propose to incorporate information about the body structure to reformulate the crowd counting as a multi-task problem. Other works such as Zhang et al. [121] make use of auxiliary information such as the ground truthed perspective map. Methods for pupil tracking and precision agriculture are usually domain-specific. Accurate pupil tracking is required for a wide range of applications, from commercial applications such as video games [122], driving [123, 124] or microsurgery [125]. This means that the center of the pupil must be resolved in images obtained in real-world illumination conditions [126]. In remote precision agriculture, agronomists can use plant traits such as plant spacing to predict future crop yield [7–10, 127], and plant scientists to breed new plant varieties [3, 6]. In [128], Aich et al.count wheat plants by first segmenting plant regions and then counting the number of plants in each segmented patch.

Hausdorff distance. The Hausdorff distance is a measure of the distance between two sets of points [129]. Modifications of the Hausdorff distance [130] have been used for a variety of tasks, including character recognition [131], face recognition [132] and scene matching [132]. In [133], Schutze et al. use the average Hausdorff distance as the metric to evaluate solutions of a multi-objective optimization. Elkhiyari et al. [134] use multiple variants of the Hausdorff distance to compare features extracted by a CNN according for the task of face recognition. In [135], Fan et al. employ the Chamfer and Earth Mover's distance for 3D object reconstruction by estimating the location of a fixed number of points. The Hausdorff distance is also a common metric in the medical imaging community for the evaluation of segmentation boundaries [136–139].

3.3 A Loss Function For Object Localization With Deep Learning

In this section, we propose a method based on Deep Learning to simultaneously locate and count objects in an image. Such objects can be plants, but they can be any type of object. We propose a novel loss function used in the training of a CNN. Our proposed loss function is a modification of the average Hausdorff distance. Our method does not require the use of bounding boxes in the training stage, and does not require to know the maximum number of objects when designing the network architecture. We name our method *Weighted Hausdorff Distance* (WHD).

For simplicity, we describe our method only for a single class of objects, although it can trivially be extended to multiple object classes. Our method is object-agnostic, thus this description does not include any information about the object characteristics. Our approach maps input images to a set of coordinates, and we validate it with diverse types of objects. Figure 3.2 shows an example of object localization with three different object types.

3.3.1 The Average Hausdorff Distance

Our work is based on the Hausdorff distance which we review here briefly. Consider two unordered non-empty sets of points X and Y and a distance metric $d(x, y)$ between two points $x \in X$ and $y \in Y$. The function $d(\cdot, \cdot)$ could be any metric, e.g, the Euclidean distance. The sets X and Y may have different number of points. Let $\Omega \subset \mathbb{R}^2$ be the space of all possible points. In its general form, the Hausdorff distance between $X \subset \Omega$ and $Y \subset \Omega$ is defined as

$$d_{\mathrm{H}}(X, Y) = \max \left\{ \sup_{x \in X} \inf_{y \in Y} d(x, y), \sup_{y \in Y} \inf_{x \in X} d(x, y) \right\}. \tag{3.1}$$

When considering a discretized and bounded Ω, such as all the possible pixel coordinates in an image, the suprema and infima are achievable and become maxima and minima, respectively. This bounds the Hausdorff distance as

$$d(X, Y) \leq d_{max} = \max_{x \in \Omega, y \in \Omega} d(x, y), \tag{3.2}$$



Fig. 3.2.: Example of object localization with three different object types: human heads, pupil centers, and plant centers. The bottom row shows red crosses are on top of each object location. For more details, see the caption of Figure 3.5.

which corresponds to the diagonal of the image when using the Euclidean distance. As shown in [129], the Hausdorff distance is a metric. Thus $\forall X, Y, Z \subset \Omega$ we have the following properties:

$$d_H(X,Y) \geq 0 \tag{3.3a}$$

$$d_H(X,Y) = 0 \iff X = Y \tag{3.3b}$$

$$d_H(X,Y) = d_H(Y,X) \tag{3.3c}$$

$$d_H(X,Y) \leq d_H(X,Z) + d_H(Z,Y) \tag{3.3d}$$

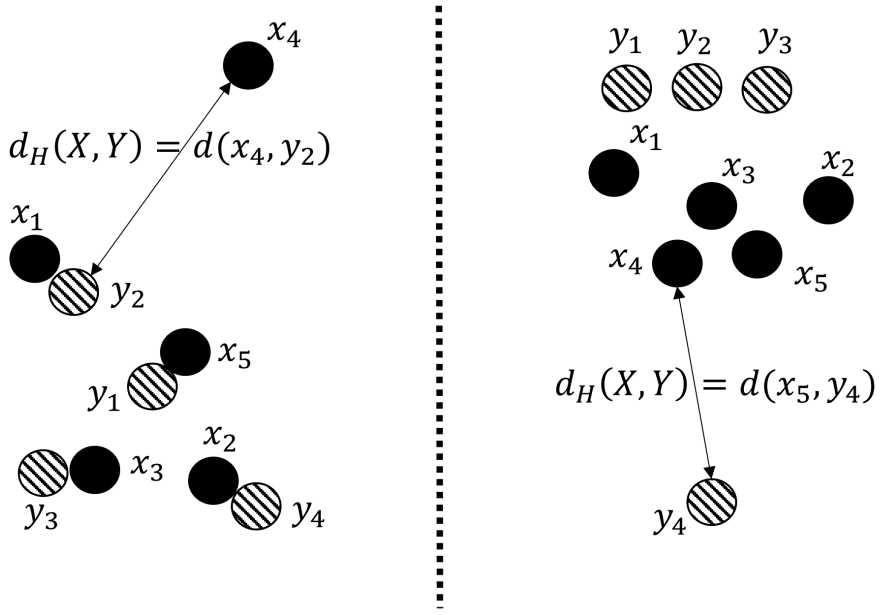


Fig. 3.3.: Illustration of the Hausdorff distance with two different configurations of point sets $X = \{x_1, ..., x_5\}$ (solid dots) and $Y = \{y_1, ..., y_4\}$ (dashed dots). Despite the differences in configuration, their Hausdorff distance are equal because the worst outlier is the same.

Equation (3.3b) follows from X and Y being closed, because in our task the pixel coordinate space Ω is discretized. These properties are very desirable when designing a function to measure how similar X and Y are [140].

A shortcoming of the Hausdorff function is that it is very sensitive to outliers [133, 136]. Figure 3.3 shows an illustration of the Hausdorff distance for two sets of finite points with one outlier. To avoid this, the average Hausdorff distance is more commonly used:

$$d_{\mathrm{AH}}(X,Y) = \frac{1}{|X|}\sum_{x\in X}\min_{y\in Y} d(x,y) + \frac{1}{|Y|}\sum_{y\in Y}\min_{x\in X} d(x,y), \tag{3.4}$$

where $|X|$ and $|Y|$ are the number of points in X and Y, respectively. Note that properties (3.3a), (3.3b) and (3.3c) are still true, but (3.3d) is no longer true. Also, the average Hausdorff distance is differentiable with respect to any point of X or Y.

Let Y contain the ground truth pixel coordinates, and X be our estimation. Ideally, we would like to use $d_{\mathrm{AH}}(X, Y)$ as the loss function during the training of our convolutional neural network (CNN). We find two limitations when incorporating the average Hausdorff distance as a loss function. First, CNNs with linear layers implicitly determine the estimated number of points $|X|$ as the size of the last layer. This is a drawback because the actual number of points depends on the content of the image itself. Second, FCNs such as U-Net [141] can indicate the presence of an object center with a higher activation in the output layer, but they do not return the pixel coordinates. In order to learn with backpropagation, the loss function must be differentiable with respect to the network output.

3.3.2 The Weighted Hausdorff Distance

To overcome these two limitations, we modify the average Hausdorff distance as follows:

$$\begin{aligned} d_{\mathrm{WH}}(p, Y) = & \frac{1}{\mathcal{S} + \epsilon} \sum_{x \in \Omega} p_x \min_{y \in Y} d(x, y) + \\ & \frac{1}{|Y|} \sum_{y \in Y} \underset{x \in \Omega}{M_\alpha} \left[\, p_x d(x, y) + (1 - p_x) d_{max} \, \right], \end{aligned} \tag{3.5}$$

where

$$\mathcal{S} = \sum_{x \in \Omega} p_x, \tag{3.6}$$

$$\underset{a \in A}{M_\alpha} \left[f(a) \right] = \left(\frac{1}{|A|} \sum_{a \in A} f^\alpha(a) \right)^{\frac{1}{\alpha}}, \tag{3.7}$$

is the generalized mean, and ϵ is set to 10^{-6}. We call $d_{\mathrm{WH}}(p, Y)$ the weighted Hausdorff distance (WHD). $p_x \in [0, 1]$ is the single-valued output of the network at pixel coordinate x. The last activation of the network can be bounded between zero and one by using a sigmoid non-linearity. Note that p does not need to be normalized, i.e., $\sum_{x \in \Omega} p_x = 1$ is not necessary. Note that the generalized mean $M_\alpha \left[\cdot \right]$ corresponds

to the minimum function when $\alpha = -\infty$. We justify the modifications applied to Equation (3.4) to obtain Equation (3.5) as follows:

1. The ϵ in the denominator of the first term provides numerical stability when $p_x \approx 0 \; \forall x \in \Omega$.

2. When $p_x = \{0, 1\}$, $\alpha = -\infty$, and $\epsilon = 0$, the weighted Hausdorff distance becomes the average Hausdorff distance. We can interpret this as the network indicating with complete certainty where the object centers are. As $d_{\mathrm{WH}}(p, Y) \geq 0$, the global minimum ($d_{\mathrm{WH}}(p, Y) = 0$) corresponds to $p_x = 1$ if $x \in Y$ and 0 otherwise.

3. In the first term, we multiply by p_x to penalize high activations in areas of the image where there is no ground truth point y nearby. In other words, the loss function penalizes estimated points that should not be there.

4. In the second term, by using the expression $f(\cdot) := p_x d(x, y) + (1 - p_x) d_{max}$ we enforce that

   (a) If $p_{x_0} \approx 1$, then $f(\cdot) \approx d(x_0, y)$. This means the point x_0 will contribute to the loss as in the AHD (Equation (3.4)).

   (b) If $p_{x_0} \approx 0$, $x_0 \neq y$, then $f(\cdot) \approx d_{max}$. Then the point x_0 will not contribute to the loss because the "minimum" $M_{\alpha, x \in \Omega}[\,\cdot\,]$ will ignore x_0 as $f(\cdot) \leq d_{max}$. If another point x_1 closer to y with $p_{x_1} > 0$ exists, x_1 will be "selected" instead by $M_{\alpha}[\,\cdot\,]$. Otherwise $M_{\alpha, x \in \Omega}[\,\cdot\,]$ will be high. This means that low activations around ground truth points will be penalized.

   Note that $f(\cdot)$ is not the only expression that would enforce these two constraints ($f|_{p_x=1} = d(x, y)$ and $f|_{p_x=0} = d_{max}$). We chose a linear function because of its simplicity and numerical stability.

Both terms in the WHD are necessary. If the first term is removed, then the trivial solution is $p_x = 1 \quad \forall x \in \Omega$. If the second term is removed, then the trivial

solution is $p_x = 0 \quad \forall x \in \Omega$. Ideally, the parameter $\alpha \to -\infty$ so that $M_\alpha(\cdot) = ||\cdot||_{-\infty}$ becomes the minimum operator [142]. However, this would make the second term flat with respect to the output of the network. For a given y, changes in p_{x_0} in a point x_0 that is far from y would be ignored by $M_{-\infty}(\cdot)$, if there is another point x_1 with high activation and closer to y. In practice, this makes training difficult because the minimum is not a smooth function with respect to its inputs. Thus, we approximate the minimum with the generalized mean $M_\alpha(\cdot)$, with $\alpha < 0$. The more negative α is, the more similar to the AHD the WHD becomes, at the expense of becoming less smooth. In our experiments, we use $\alpha = -1$.

If the input image needs to be resized to be fed into the network, we can normalize the WHD to account for this distortion. Denote the original image size as $(S_o^{(1)}, S_o^{(2)})$ and the resized image size as $(S_r^{(1)}, S_r^{(2)})$. In Equation (3.5), we compute distances in the original pixel space by replacing $d(x, y)$ with $d(\mathbf{S}x, \mathbf{S}y)$, where $x, y \in \Omega$ and

$$\mathbf{S} = \begin{pmatrix} S_o^{(1)}/S_r^{(1)} & 0 \\ 0 & S_o^{(2)}/S_r^{(2)} \end{pmatrix}. \tag{3.8}$$

3.3.3 Advantage Over Pixelwise Losses

A naive alternative is to use a one-hot map as label, defined as in Equation (3.9).

$$l_i = \begin{cases} 1 & \text{if } i \in Y \\ 0 & \text{otherwise,} \end{cases} \tag{3.9}$$

where i is a 2D index of the pixel coordinates of the image. Then we can use a pixelwise loss such as the L^2 norm, defined as $L^2(l, p) = \sum_{\forall x \in \Omega} |p_x - l_x|^2$. The issue with this approach (and any pixelwise loss) is that it is not informative of how close two points $x \in \Omega$ and $y \in Y$ are unless $x = y$. In other words, it is flat for the vast majority of the pixels, making training unfeasible. By contrast, the WHD in Equation (3.5) will decrease the closer x is to y, making the loss function informative outside of the global minimum.

Another alternative is to construct the labels by centering Gaussians at each true point as in Equation (3.10).

$$l_i = \sum_{y \in Y} \mathcal{N}(y, \sigma)(i). \tag{3.10}$$

Then we train our network using a pixelwise loss such as the L^2 as well. However the parameter σ needs to be carefully tuned. If it is too large and objects are too close to each other, the Gaussians will merge. If σ is too small, the label will tend to be a pointwise label such as in Equation (3.9), and we will run into the problems mentioned in the previous paragraph.

3.3.4 CNN Architecture

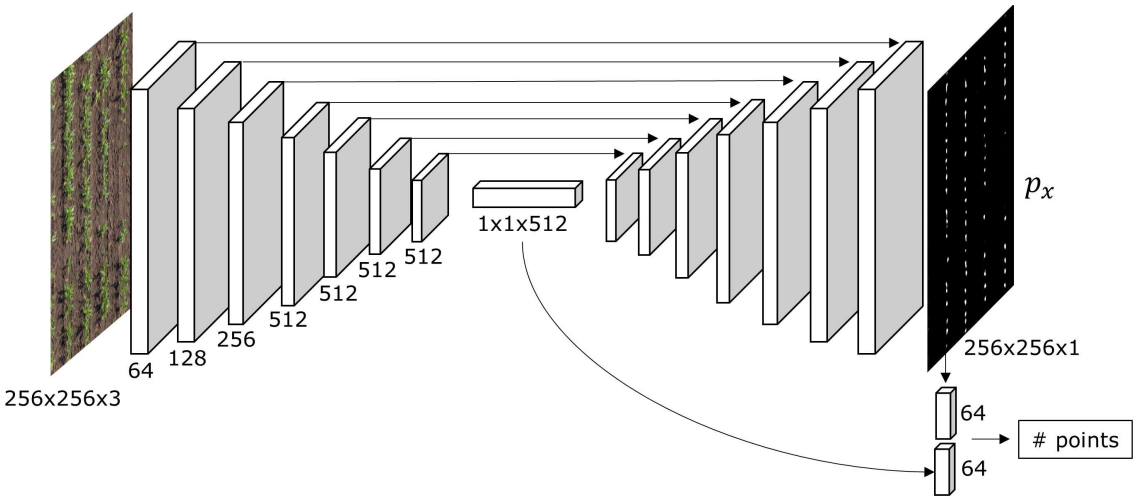


Fig. 3.4.: The FCN architecture used for object localization, minimally adapted from the U-Net [141] architecture. We add a small fully-connected layer that combines the deepest features and the estimated probability map to regress the number of points.

In this section, we describe the architecture of the fully convolutional network (FCN) we use, and how we estimate the final object locations. We want to emphasize that the network design is not a meaningful contribution of this work, thus we have not made any attempt to optimize it because this is not the intention of this thesis. Our main contribution is the use of the weighted Hausdorff distance as the loss function. We adopt the U-Net architecture [141] and modify it minimally for this task. Networks similar to U-Net have been proven to be capable of accurately mapping the input image into an output image, when trained in a conditional adversarial network setting [143] or when using a carefully tuned loss function [141]. Figure 3.4 shows the hourglass design of U-Net. The residuals connections between each layer in the encoder and its symmetric layer in the decoder are not shown for simplicity.

This FCN has two well differentiated blocks. The first block follows the typical architecture of a CNN. It consists of the repeated application of two 3×3 convolutions

(with padding 1), each followed by a batch normalization operation and a Rectified Linear Unit (ReLU). After the ReLU, we apply a 2×2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels, starting with 64 channels and using 512 channels for the last 5 layers.

The second block consists of repeated applications of the following elements: a bilinear upsampling, a concatenation with the feature map from the downsampling block, and two 3×3 convolutions, each followed by a batch normalization and a ReLU. The final layer is a convolution layer that maps to the single-channel output of the network, p.

To estimate the number of objects in the image, we add a branch that combines the information from the deepest level features and also from the estimated probability map. This branch combines both features (the $1 \times 1 \times 512$ feature vector and the 256×256 probability map) into a hidden layer, and uses the 128-dimensional feature vector to output a single number. We then apply a ReLU to ensure the output is positive, and round it to the closest integer to obtain our final estimate of the number of objects, $\hat{C}$.

Although we use this particular network architecture, any other architecture could be used. The only requirement is that the output images of the network must be of the same size as the input image. The choice of a FCN arises from the natural interpretation of its output as the weights (p_x) in the WHD (Equation (3.5)). In previous works [134,135], variants of the average Haussdorf distance were successfully used with non-FCN networks that estimate the point set directly. However, in those cases the size of the estimated set is fixed by the size of the last layer of the network. To locate an unknown number of objects, the network must be able to estimate a variable number of object locations. Thus, we could envision the WHD also being used in non-FCN networks as long as the output of the network is used as the weights in Equation (3.5).

3.3.5 Location Estimation

The training loss we use to train the network is a combination of Equation (3.5) and a smooth L_1 loss for the regression of the object count. The final training loss is

$$\begin{aligned}\mathcal{L}(p, Y) = d_{\mathrm{WH}}(p, Y) =& \frac{1}{\mathcal{S}+\epsilon} \sum_{x \in \Omega} p_x \min_{y \in Y} d(x, y) \\ &+ \frac{1}{|Y|} \sum_{y \in Y} \underset{x \in \Omega}{M_\alpha} [\, p_x d(x, y) + (1 - p_x) d_{max} \,] \\ &+ \mathcal{L}_{\mathrm{reg}}(C - \hat{C}(p)),\end{aligned} \tag{3.11}$$

where Y is the set containing the ground truth coordinates of the objects in the image, p is the output of the network, $C = |Y|$, and $\hat{C}(p)$ is the estimated number of objects. $\mathcal{L}_{\mathrm{reg}}(\cdot)$ is the regression term, for which we use the smooth L_1 or Huber loss [144], defined as

$$\mathcal{L}_{\mathrm{reg}}(t) = \begin{cases} 0.5t^2, & \text{for} |t| < 1 \\ |t| - 0.5, & \text{for} |t| \geq 1. \end{cases} \tag{3.12}$$

This loss is known to be robust to outliers when the regression error is high, and at the same time is differentiable at the origin. Automatic differentiation of Equation (3.11) with respect to the network weights is possible with deep learning frameworks such as PyTorch or TensorFlow.

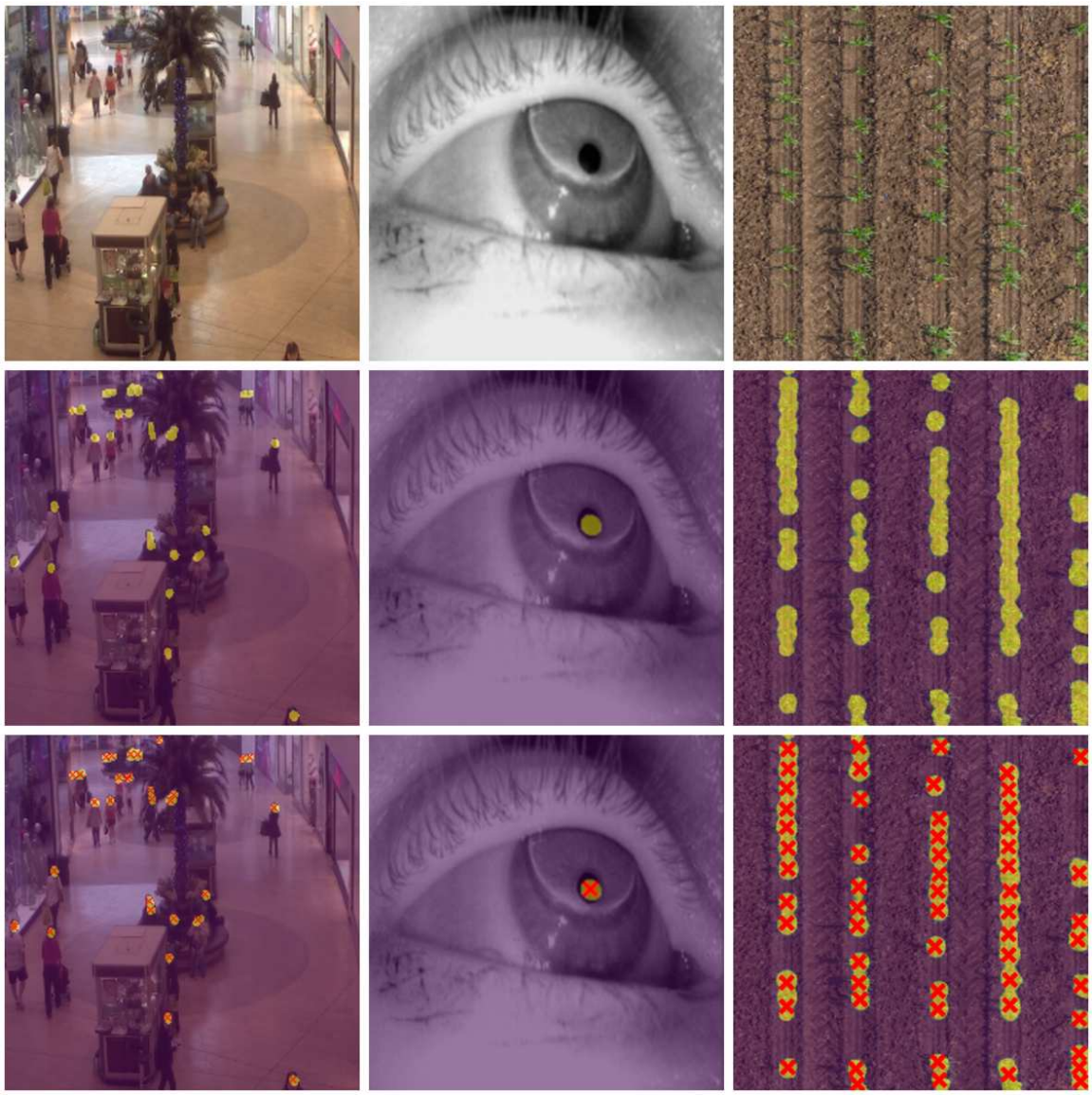


Fig. 3.5.: First row: Input image. Second row: Output of the network (p in the text) in the viridis colormap [145] It can be considered a saliency map of object locations. The output is overlaid onto the input image using alpha-blending [146]. Third row: The estimated object locations are marked with a red cross.

The network outputs a saliency map $p \in [0, 1]$, the confidence that there is an object at pixel x. Figure 3.5 shows p in the second row. During evaluation, our ultimate goal is to obtain $\hat{Y}$, i. e., the estimate of all object locations. In order to convert p into $\hat{Y}$, we threshold p to obtain the pixels $T = \{x \in \Omega \mid p_x > \tau\}$. We can use three different methods to decide which τ to use:

1. A constant τ for all images.

2. Otsu thresholding [84].

3. Beta mixture model-based thresholding (BMM).

Both Otsu and Beta Mixture Model (BMM) thresholding find an adaptive τ different for every image. BMM thresholding consists on fitting a mixture of two Beta distributions to the values of p. A Beta probability density distribution is defined as

$$\text{Beta}(\alpha, \beta)(p) = \frac{p^{\alpha-1}(1-p)^{\beta-1}}{B(\alpha, \beta)}, \tag{3.13}$$

with $p \in [0, 1]$, and

$$\text{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}. \tag{3.14}$$

A Beta mixture with two components is then defined as

$$BMM(\alpha, \beta)(p) = \omega\text{Beta}(\alpha_1, \beta_1)(p) + (1-\omega)\text{Beta}(\alpha_2, \beta_2)(p). \tag{3.15}$$

Fitting a BMM means finding the optimal values for $\alpha_1, \beta_1, \alpha_2, \beta_2$, and ω. These five parameters could be estimated using Expectation Maximization (EM). However, as described in [147], fitting a Beta distribution with EM can be unstable. Instead, we use the algorithm described in [147]. After the parameters have been estimated, we take mean value of the distribution with highest mean as τ, i.e.:

$$\tau_{\text{BMM}} = \max\left\{\frac{\alpha_1}{\alpha_1+\beta_1}, \frac{\alpha_2}{\alpha_2+\beta_2}\right\}. \tag{3.16}$$

After thresholding the saliency map, we fit a Gaussian mixture model to the points T. This is done using the EM [148] algorithm and the estimated number of plants $\hat{C}$.

The means of the fitted Gaussians are considered the final estimate $\hat{Y}$. The third row of Figure 3.5 shows the estimated object locations with red crosses. Note that even if the map produced by the FCN is of good quality, i.e., there is a cluster on each object location, EM may not yield the correct object locations if $|\hat{C} - C| > 0.5$. An example can be observed in the third column of Figure 3.5, where a single head is erroneously estimated as two heads.

3.3.6 Experimental Results

We evaluate our method to locate objects with the WHD using three datasets.

The first dataset consists of 2,000 images acquired from a surveillance camera in a shopping mall. It contains annotated locations of the heads of the crowd. This dataset is publicly available at `http://personal.ie.cuhk.edu.hk/~ccloy/downloads_mall_dataset.html` [149]. The images were randomly split with 80%, 10% and 10% of the images assigned to the training, validation and testing sets, respectively.

The second dataset is presented in [126] with the roman letter V and publicly available at `http://www.ti.uni-tuebingen.de/Pupil-detection.1827.0.html`. It contains 2,135 images with a single eye, and the goal is to detect the center of the pupil. It was randomly split with 80%, 10% and 10% of the images for training, validation, and testing, respectively.

The third dataset consists of aerial images of a crop field taken from a UAV flying at an altitude of 40 m. The images were stitched together to generate a $6,000 \times 12,000$ orthoimage of 0.75 cm/pixel resolution shown in Figure 3.6. The location of the center of all plants in this image was ground truthed, resulting in a total of 15,208 unique plant centers. This mosaic image was split, and the left 80% area was used for training, the middle 10% for validation, and the right 10% for testing. Within each region, random image crops were generated. These random crops have a uniformly distributed height and width between 100 and 600 pixels. We extracted



Fig. 3.6.: An orthophoto of a sorghum field with 15,208 plants. These original images were collected on June 13, 2016. The red region was used for training, the region in green for validation, and the region in blue for testing.

50,000 random image crops in the training region, $5,000$ in the validation region, and $5,000$ in the testing region. Note that some of these crops may highly overlap.

All the images were resized to 256×256 because that is the minimum size our architecture allows. The ground truthed object locations were also scaled accordingly. As for data augmentation, we only use random horizontal flip. For the plant dataset, we also flipped the images vertically. We set $\alpha = -1$ in Equation (3.7). We have also experimented with $\alpha = -2$ with no apparent improvement, but we did not attempt to find an optimal value. We retrain the network for every dataset, i.e., we do not use pretrained weights. For the mall and plant dataset, we used a batch size of 32 and Adam optimizer [150, 151] with a learning rate of 10^{-4} and momentum of 0.9. For the pupil dataset, we reduced the size of the network by removing the five central layers, we used a batch size of 64, and stochastic gradient descent with a learning rate of 10^{-3} and momentum of 0.9. At the end of each epoch, we evaluate the average Haussdorf distance (AHD) in Equation (3.4) over the entire validation set, and keep the epoch at which the AHD on validation is lowest.

As metrics, we report Precision, Recall, F-score, AHD, Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percent Error (MAPE), defined as

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{C}_i - C_i|, \tag{3.17}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left|\hat{C}_i - C_i\right|^2} \tag{3.18}$$

$$\text{MAPE} = 100\frac{1}{N}\sum_{\substack{i=1\\C_i\neq 0}}^{N}\frac{\left|\hat{C}_i - C_i\right|}{C_i}, \tag{3.19}$$

where N is the number of images in the dataset, C_i is the true number of objects in the i-th image, and $\hat{C}_i$ is our estimate.

A true positive is counted if an estimated location is at most at distance r from a ground truth point. A false positive is counted if an estimated location does not have any ground truth point at a distance at most r. A false negative is counted if a true location does have any estimated location at a distance at most r. Precision is the proportion of our estimated points that are close enough to a true point. Recall is the proportion of the true points that we are able to detect. The F-score is the harmonic mean of precision and recall. We are aware that we can achieve a precision and recall of 100% even if we estimate more than one object location per ground truth point. This would not be an ideal localization. To take this into account, we also report metrics (MAE, RMSE and MAPE) that indicate if the number of objects is incorrect. The AHD can be interpreted as the average location error in pixels.

Figure 3.9 shows the F-score as a function of r. Note that r is only an evaluation parameter. It is not needed during training or testing. MAE, RMSE, and MAPE are shown in Table 3.1. Note that we are using the same architecture for all tasks, except for the pupil dataset, where we removed intermediate layers. Also, in the case of the pupil detection, we know that there is always one object in the image. Thus, regression is not necessary and we can remove the regression term in eq. (3.11) and fix $\hat{C}_i = C_i = 1\ \forall i$.

A naive alternative approach to object localization would be to use generic object detectors such as Faster R-CNN [109]. One can train these detectors by constructing bounding boxes with fixed size centered at each labeled point. Then the center of each bounding box can be taken as the estimated location. We used bounding boxes of size 20×20 (the approximate average head and pupil size), anchor sizes of 16×16 and 32×32 and an intersection over union threshold of 0.4. The threshold we used for the softmax scores was 0.5, because it minimizes the AHD over the validation set. We used the VGG-16 architecture [152] and trained it using stochastic gradient descent with learning rate of 10^{-3} and momentum of 0.9. For the pupil dataset, we always selected the bounding box with the highest score. We experimentally observed that Faster R-CNN struggles with detecting very small objects that are very close to each other. Table 3.2, Table 3.3, and Table 3.4 show the results of Faster R-CNN on the mall, pupil, and plant datasets, respectively. Note that the mall and plant datasets, with many small and highly overlapping objects, are the most challenging datasets for Faster R-CNN. This behaviour is consistent with the observations in [112], where, all generic object detectors perform very poorly, with Faster R-CNN yielding a mean Average Precision (mAP) of 5% in the best case.

We also experimented using mean shift [153] instead of Gaussian mixtures (GM) to detect the local maxima. However, mean shift is prone to detect multiple local maxima, and GMs are more robust against outliers. In our experiments, we observed that precision and recall were substantially worse than using GM. More importantly, using Mean Shift slowed down validation an order of magnitude. The average time for the Mean Shift algorithm to run on one of our images was 12 seconds, while fitting GM using expectation maximization took around 0.5 seconds, when using the scikit-learn implementations [154].

We also investigated the effect of the parameter τ, and the three methods to select it presented in Section 3.3.5. One may think that this parameter could be a trade-off between some metrics, and that it should be cross-validated. In practice, we observed that τ does not balance precision and recall, thus a precision-recall curve is

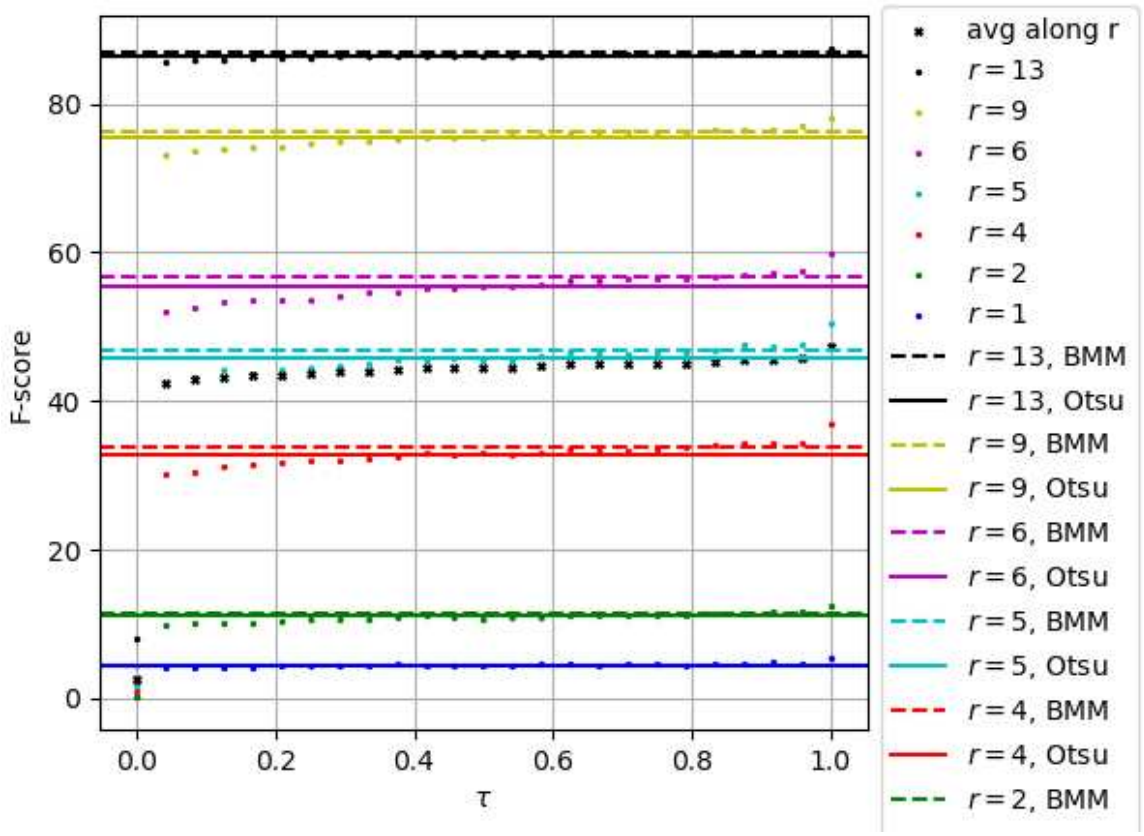


Fig. 3.7.: Effect on the F-score of the threshold τ

not meaningful. Instead, we plot the F-score as a function of r in Figure 3.9. Also, cross-validating τ would imply fixing an "optimal" value for all images. Figure 3.7 shows that we can do better with adaptive thresholding methods (Otsu or BMM). Note that BMM thresholding (dashed lines) always outperforms Otsu (solid lines), and most of fixed τ. To justify the appropriateness of the BMM method, note that in Figure 3.5 most of the values in the estimated map are very high or very low. This makes a Beta distribution a better fit than a Normal distribution (as used in Otsu's method) to model p_x. Figure 3.8 shows the fitted BMM and a kernel density estimation of the values of τ adaptively selected by the BMM method.

Lastly, as our method locates and counts objects simultaneously, it could be used as a counting technique. We also evaluated our technique in the task of crowd counting using the ShanghaiTech Part B dataset presented in [118], and achieve a MAE

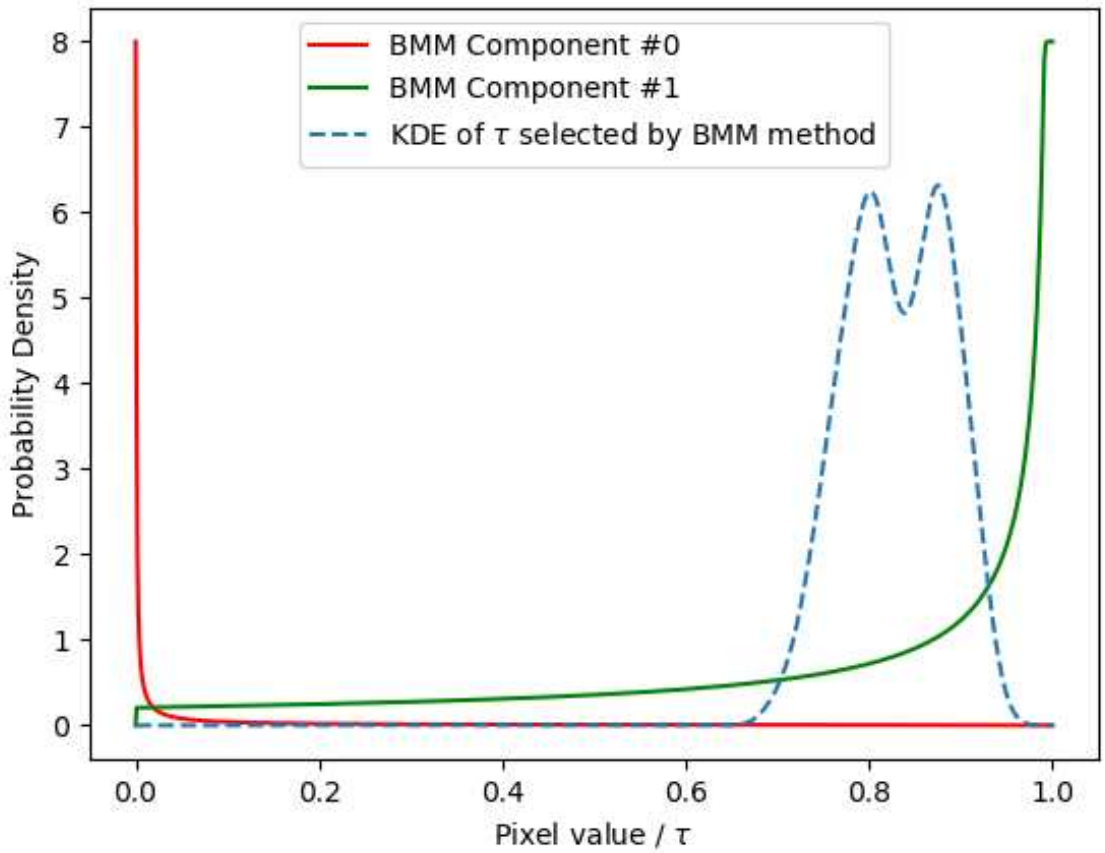


Fig. 3.8.: Beta mixture model fitted on the values of p_x, and the thresholds τ used by the BMM method.

of 19.9. Even though we do not outperform state of the art methods that are specifically fine-tuned for crowd counting [155], we can achieve comparable results with our generic method. We expect future improvements such as architectural changes or using transfer learning to further increase the performance.

Our method takes approximately 0.2 seconds per image to estimate all object locations in an image, and around three days for training, using three NVIDIA Titan Xp cards. In comparison, Faster-RCNN required about two weeks of training to achieve the accuracy mentioned above for the plant dataset.

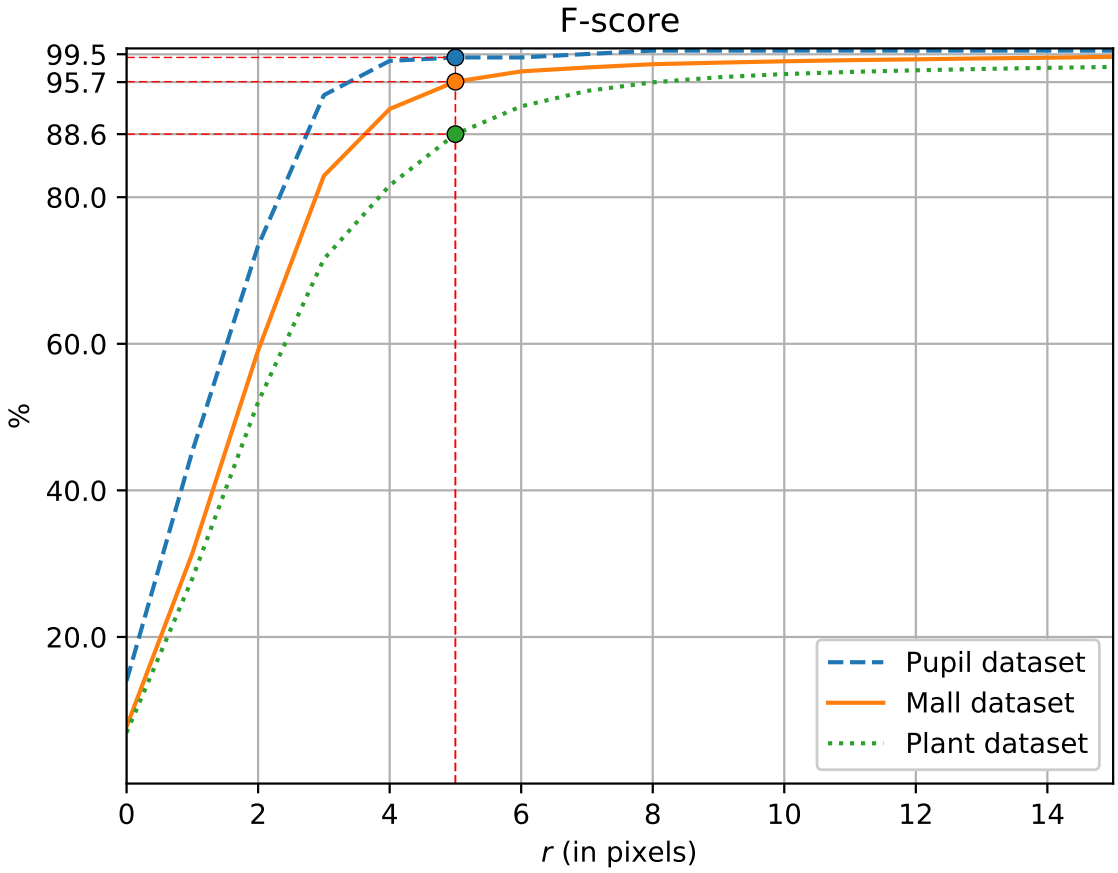


Fig. 3.9.: The F-score as a function of r for the three datasets. r is the maximum distance between a ground truth and an estimated object location to consider a correct or missing detection. The higher r the easier it is to achieve a higher precision and recall.

Table 3.1.: Results of our method for object localization, using $r = 5$. Metrics are defined in Equations (3.4) and (3.17) to (3.19). Regression metrics for the pupil dataset are not shown because there is always a single pupil ($\hat{C} = C = 1$). Figure 3.9 shows the F-score for other r values.

Metric	**Mall dataset**	**Pupil dataset**	**Plant dataset**	**Average**
Precision	95.2%	99.5%	88.1%	94.4%
Recall	96.2%	99.5%	89.2%	95.0%
F-score	95.7%	99.5%	88.6%	94.6%
AHD	4.5 px	2.5 px	7.1 px	4.7 px
MAE	1.4	-	1.9	1.7
RMSE	1.8	-	2.7	2.3
MAPE	4.4%	-	4.2%	4.3 %

Table 3.2.: Head location results using the mall dataset, using $r = 5$.

Metric	**Faster-RCNN**	**Ours**
Precision	81.1%	**95.2 %**
Recall	76.7%	**96.2 %**
F-score	78.8 %	**95.7 %**
AHD	7.6 px	**4.5 px**
MAE	4.7	**1.4**
RMSE	5.6	**1.8**
MAPE	14.8%	**4.4 %**

Table 3.3.: Pupil center localization results, using $r = 5$. Precision and recall are equal because there is only one estimated and one true object.

Method	**Precision**	**Recall**	**AHD**
Swirski [156]	77 %	77 %	-
ExCuSe [126]	77 %	77 %	-
Faster-RCNN	99.5 %	99.5 %	2.7 px
Ours	**99.5** %	**99.5** %	**2.5 px**

Table 3.4.: Plant location results using the plant dataset, using $r = 5$.

Metric	**Faster-RCNN**	**Ours**
Precision	86.6 %	**88.1 %**
Recall	78.3 %	**89.2 %**
F-score	82.2 %	**88.6 %**
AHD	9.0 px	**7.1 px**
MAE	9.4	**1.9**
RMSE	13.4	**2.7**
MAPE	17.7 %	**4.2 %**

3.4 Counting With Deep Learning

In occasions, plant scientists, plant breeders, or agronomists may already know the plant count of certain plots in the field, and not know the exact location of each plant. Ground truthing plant locations implies using software tools for labeling the UAV imagery, or manually surveying the plants in the field.

In this section, we describe a method to count the number of plants in an image, without any notion of localization. We do not need to know the location of the plants. We use regression to count the number of plants and do not approach the problem as a classification problem by consider the plant count as the class number.

Our method is different from [104, 114, 157] in that we do not consider the plant count as the class number. Typically, the output layer of a neural network consists of a set of neurons whose activations are interpreted as the probabilities that the image belongs to each class. For counting, one can assign these classes to the number of plants in the image. This would impose an implicit maximum number of plants in the image, as the number of neurons in this output layer. This approach would correspond to a classification problem. Denote the number of neurons in the last layer as C_{max}, and an index over these neurons as $x = 1, ..., C_{max}$. In a classification task, a common loss function is the cross entropy:

$$H(q, p) = -\sum_{x=1}^{C_{max}} p(x) \log q(x), \tag{3.20}$$

where $q(x)$ is the activation at the x-th neuron of the last layer, and $p(x)$ is the label. If we construct $p(x)$ using one-hot encoding, i.e,

$$p(x) = \begin{cases} 1 & \text{if } x = C_i \\ 0 & \text{otherwise,} \end{cases} \tag{3.21}$$

where C_i is the label (plant count) of the i-th image, then the loss function only takes into account a single neuron (the C_i-th neuron) and becomes

$$H(q, p) = -\log q(C_i). \tag{3.22}$$



Fig. 3.10.: Three examples of row segments at different growth stages. Top: An image from June 13. Center: The same row segment on June 21, 2016. Bottom: An image from June 21, 2016, whose label is noisy. It is hard to ground truth the plant count from the UAV image.

This implies that during training using an image with C plants, we would ignore all activations except for $x = C$. One can interpret $p(x)$ as the certainty the network has that an image belongs to the class x. Equations (3.21) and (3.22) are then appropriate when classes are independent. However, the classes in our dataset are not completely independent. Consider the following example, where an image contains 2 plants. If our estimate is 3, the penalty for the network should be lower than if the estimate was 5. This is aggravated in our dataset because there is considerable label noise. When plants are clustered, the label may be more than one count incorrect from the real value (see Figure 3.10).

To address these problems, we employ the $\mathcal{L}_p$ norm as the loss function to be minimized, defined as

$$\mathcal{L}_p(\hat{x}, C_i) = |\hat{x} - C_i|^p, \tag{3.23}$$

where $\hat{x}$ is our plant count estimate. For the particular case of $p = 2$, the $\mathcal{L}_p$ norm becomes the Mean Squared Error (MSE) loss. When $p = 1$, this error can be interpreted how many plants our estimate will be off of the real value.

3.4.1 Modifications To The CNN Architecture

We tested various CNN architectures (AlexNet [158], Inception-v2 [159], Inception-v3 [160], and Inception-v4 [161]). In all of them, we replace the last layer with a single neuron without non-linearity. The activation of this neuron corresponds to the pre-softmax activation of the output layer. The value of the activation may be non-integer and occasionally negative. Because of this, we take the absolute value and round it to the closest integer. The final value is our plant count estimate denoted as $\hat{x}$. In this way, the task of plant counting is posed as a regression problem instead of as a classification problem. Unlike with the cross entropy, when using the $\mathcal{L}_p$ norm, the further an estimate is from the label, the more it is penalized.

Many neural network architectures require square images for training, e.g, in AlexNet and Inception-v2, the input images are 224×224 and 299×299, respectively. However, the images of our dataset are not rectangular, but 546×103. To overcome this limitation, we slightly modify the last layers of the networks. Our modifications consist only on removing one pooling layer. In AlexNet [158], our input image size would make the size of the feature maps after the last max pooling to be 15×2. As a 2 pixels-wide feature map may be too narrow to capture horizontal features, we remove the last max pooling layer. Thus, the feature map before the first fully-connected layer becomes of size 32×5. In Inception-v2, Inception-v3, and Inception-v4, after the last convolutional layer we obtain a feature map of size 18×4, 15×1, and 15×1, respectively. These feature maps cannot be used with the 8×8 average pooling layer. We remove this last average pooling layer and we use the feature map directly as input to the fully-connected layer. Figure 3.11 shows the last part of Inception-v3 and Inception-v4, with our modifications.

3.4.2 Experimental Results

In our experiments, we used a dataset of 2,048 images such as the ones in Figure 3.10. These images are crops of row segments of a sorghum field obtained using

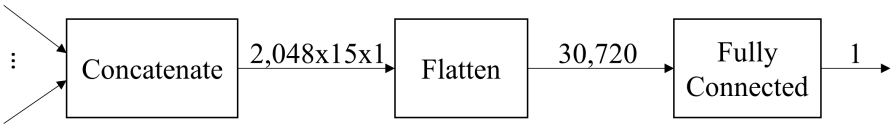


Fig. 3.11.: Modified part of Inception-v3 and Inception-v4. Note that we have removed the last max pooling. The activation of the last layer is the estimate the number of plants.

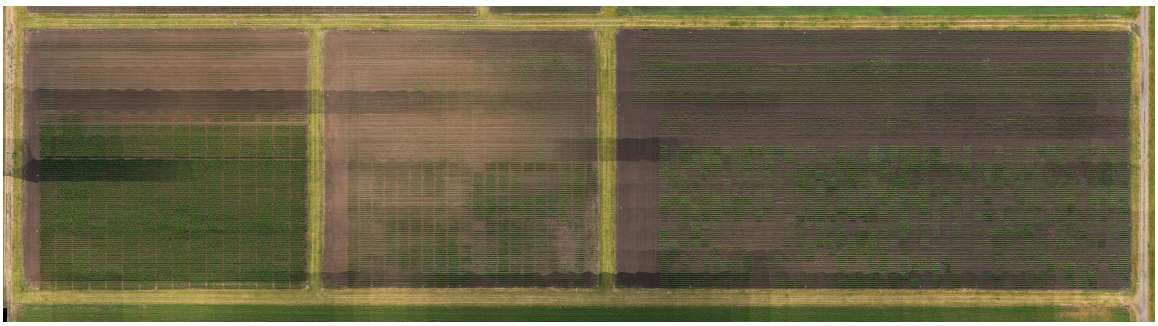


Fig. 3.12.: Orthophoto of a sorghum field acquired on June 21, 2016. Of the three panels, only the central panel is used for analysis.

the plot extraction method described in Section 2.4. The plot boundaries were modified to preserve the center and have the average size of 546×146 pixels. These images are the crops of the orthophoto corresponding to these modified boundaries. The orthophoto used is shown in Figure 3.12. Only the top half of the central panel is used because it is the only area that can be labeled. In other areas of the field the plants overlap too much to count them from the images.

The dataset is randomly split such that 80% of the images are used for training, 10% for validation, and 10% for testing. This partition results into 1,983 images for training, 250 for validation, and 249 for testing.

A limitation of our dataset is the small size of the training set. One way to address this problem is to make use of data augmentation techniques [162]. Linear and nonlinear transforms are used to create "new" training images without changing the label (plant count). We use four types of data augmentation: vertical flip, horizontal

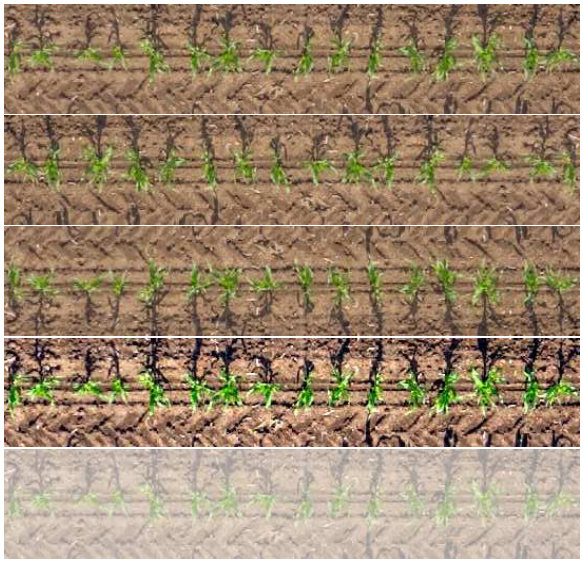


Fig. 3.13.: Data augmentation techniques used in our study. From top to bottom: (1) Original image. (2) Horizontal flip. (3) Vertical flip. (4) Contrast change. (5) Brightness change.

flip, brightness change, and contrast change. The vertical and horizontal flips are applied with a probability of 0.5 each. The brightness change consists on adding the same random value to the three 255-valued channels of the input image. This random value is uniformly distributed in the range of $[-5, 5)$. The contrast change consisted on scaling all three channels of the normalized image. The scaling factor is uniformly distributed in the range $[0.1, 10)$. The data augmentation techniques are used in random order during training. Figure 3.13 shows the result of using these augmentations to one of the images of the dataset.

For evaluation purposes, the metric we use is the Mean Absolute Percent Error (MAPE):

$$\text{MAPE} = 100\frac{\overline{|\hat{x} - C|}}{C}, \tag{3.24}$$

where $\overline{(\cdot)}$ indicates the average along all the images in the testing set.

Firstly, we evaluated the effect of p in Equation (3.23). We use AlexNet for the evaluation of p because it is smaller than the Inception architectures, thus it trains

the fastest. Table 3.5 shows the Mean Absolute Percent Error (MAPE) for different values of p, and we can see we obtain the lowest MAPE with $p = 1$. As the MAPE metric and the $\mathcal{L}_1$ norm only differ in a constant $\left(\frac{100}{C}\right)$, it is reasonable that training with $p = 1$ yields the lowest error.

Table 3.5.: Impact of p on the MAPE when using AlexNet and data augmentation.

p	**Testing MAPE (w/o data augmentation)**
1	**7.9 %**
1.5	8.5 %
1.8	8.4 %
2	8.2 %

After setting $p = 1$, we evaluated the performance of various neural network architectures, including Alexnet, Inception-v2, Inception-v3, and Inception-v4. Table 3.7 shows the error on the testing set. We obtain the lowest MAPE of 6.7% with Inception-v3. A MAPE of 6.7%, if the true label is $C = 14$, means that our estimate is in average less than one plant off the real value. Also, note from Table 3.7 that the error is consistently reduced when using data augmentation. Data augmentation accounts for a reduction of the error between 0.4% and 1%. Inception-v4, the CNN with the most number of parameters, produces a higher error than any other architecture. This may indicate overfitting and the need to increase the dataset size, to use more data augmentation techniques, or to investigate more aggressive regularization techniques.

We used implementations of the neural networks that use the TensorFlow framework [163]. We did not use pre-trained model parameters. In other words, we trained

Table 3.7.: Performance of different CNN architectures on the testing dataset (with $p = 1$).

Architecture	Testing MAPE (w/o data augmentation)	Testing MAPE (w/ data augmentation)
AlexNet	8.3 %	7.9 %
Inception-v2	8.2 %	6.7 %
Inception-v3	**7.1 %**	**6.7 %**
Inception-v4	12.4 %	11.4 %

the networks from scratch. The optimization technique used was Adam [150] with exponential decay rates of $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and $\epsilon = 1e^{-8}$. When using AlexNet and Inception-v2, the learning rate was set to $\alpha = 1e^{-4}$, and when using Inception-v3 and Inception-v4, the learning rate was set to $\alpha = 1e^{-5}$. We used a batch size of 96. The training was run for 50,000 iterations when training AlexNet and Inception-v2, and for 500,000 iterations when training Inception-v3 and Inception-v4. The iteration at which the model returned the lowest validation MAPE was selected. The reported testing errors correspond to the testing error of the model at such iteration. This occurred after 12 hours of training for the Inception-v3 case. The machine used for computation was an Intel i7-5930K and one GeForce GTX Titan X.

3.5 Plant Location Model Using A Statistical Model

In this section, we describe a statistical model to estimate the position of each sorghum plant. The location of each plant is defined as the pixel coordinates where the stalk intersects the ground plane, and can be used to automatically obtain the

intra-row and inter-row spacing. A row of plants is defined as all the plants that are aligned together. Inter-row spacing is defined as the distance between rows. Intra-row spacing is defined as the distance between plants within the same row.

The number of plants in an image is denoted as the constant P and assumed to be known a priori. The positions of the plants are modeled as a random vector X, i.e.,

$$X = [X_0, X_1, ..., X_{P-1}], \tag{3.25}$$

where X_p, $p = 0, ..., P-1$, contains the (i, j) coordinates of the p-th plant:

$$X_p = \begin{bmatrix} X_{p,i} \\ X_{p,j} \end{bmatrix}. \tag{3.26}$$

Our goal is to estimate X from Y, where is Y is the color-based segmentation mask (Section 2.7) obtained with Equation 2.21.

The 2D coordinates of the pixel m are denoted as $K(m)$. A vector Z is constructed as

$$Z = [Z_0, Z_1, ..., Z_{N-1}], \tag{3.27}$$

where each element $Z_n = K(n)$, $n = 0, ..., N-1$, is included if $Y_n = 1$. N is the number of pixels classified as leaf pixels. Notice that $N \leq M$. Hence, Z contains the coordinates of the sorghum pixels.

The plant p that is closest to the pixel n is denoted as $C(n)$. The Euclidean distance from the pixel n to the plant $C(n)$ is denoted as D_n and is computed as

$$\begin{aligned} D_n &= \|K(n) - K(C(n))\|_2 \\ &= \underset{p=0,1,...,P-1}{\arg\min} \|K(n) - X_p\|_2 . \end{aligned} \tag{3.28}$$

Figure 3.14 depicts the location X_p of one sorghum plant, and the distance D_n to a leaf pixel n.

D_n is modeled as a random variable with exponential conditional probability density with mean and standard deviation σ. Therefore the probability density function for a leaf pixel n at distance $D_n = d_n$ from $C(n)$ is

$$p_{D_n}(d_n) = \frac{1}{\sigma} e^{-\frac{d_n}{\sigma}}. \tag{3.29}$$

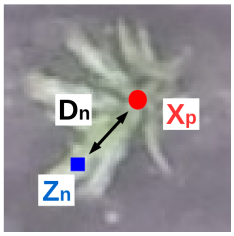


Fig. 3.14.: A single sorghum plant. The distance from pixel n (with coordinates Z_n) to X_p (the coordinates of Sorghum plant p) is D_n, obtained by Equation 3.28.

σ can be interpreted as the average radius of a plant.

Then, the conditional distribution of a single point Z_n only depends on its closest plant:

$$\begin{aligned} p_{Z_n|X}(z_n|X) &= p_{Z_n|X_{C(n)}}(z_n|X_{C(n)}) \\ &= \frac{1}{\sigma} e^{-\frac{d_n}{\sigma}}. \end{aligned} \tag{3.30}$$

From our assumptions above, we have conditional independence between each Z_n (given X). Then, the joint conditional density of Z can be factorized as

$$\begin{aligned} p_{Z|X}(z|X) &= \prod_{n=1}^{N} p_{Z_n|X}(z_n|X) \\ &= \prod_{n=1}^{N} \frac{1}{\sigma} e^{-\frac{d_n}{\sigma}} \\ &= \left(\prod_{n=1}^{N} \frac{1}{\sigma} \right) \left(\prod_{n=1}^{N} e^{-\frac{1}{\sigma} d_n} \right) \\ &= \frac{1}{\sigma^N} e^{-\frac{1}{\sigma} \sum_{n=1}^{N} d_n}. \end{aligned} \tag{3.31}$$

This model assumes that the leaf distribution does not have any direction preference, i.e., the leaves grow uniformly in all directions. In some situations, however, the plant is tilted, and the stalk is not completely at the center of the plant.

Since we are using an orthorectified mosaic, the crop field follows certain structure in the image. The plants in the image are very much aligned in rows as they are in the field. We make use of this information to introduce a prior distribution for X.

The conditional probability density of the position of one plant X_p given the remaining plants $(X_0, ..., X_{p-1}, X_{p+1}, ..., X_{P-1})$ is assumed normal:

$$p_{X_p|X_{q\neq p}}(x_p|X_{q\neq p}) = \frac{1}{2\pi|R_p|^{1/2}} \exp\left(-\frac{1}{2}\left\|x_p - \mu_p\right\|^2_{R_p^{-1}}\right), \tag{3.32}$$

where μ_p are the coordinates of the vertical and horizontal plant lines where X_p is a member, and

$$R_p = \begin{bmatrix} \sigma^2_{p,i} & 0 \\ 0 & \sigma^2_{p,j} \end{bmatrix} \tag{3.33}$$

is the covariance matrix of the positions of the plants that are aligned with the plant p, either vertically or horizontally. $\sigma^2_{p,i}$ and $\sigma^2_{p,j}$ are the vertical and horizontal standard deviations of X_p (see Figure 3.15). $\sigma^2_{p,j}$ is typically very low because of the alignment of the planter at planting time. R_p is a diagonal matrix when the field rows are aligned with the image axis in the orthorectified image.

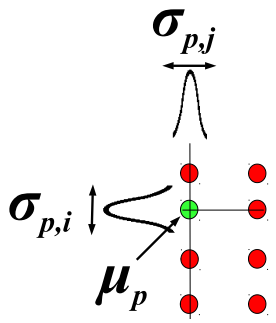


Fig. 3.15.: Vertical and horizontal alignments of the plants in the field when viewed from the top. Red dots are the plants whose position is known. The green dot is the plant whose prior position we are developing.

From Equation 3.31, we can obtain the Maximum A Posteriori (MAP) estimate of X as

$$\begin{aligned}
\hat{X}(Z) &= \arg\max_x p_{X|Z}(x|Z) = \\
&= \arg\max_x \frac{p_{Z|X}(Z|x)p_X(x)}{p_Z(Z)} \\
&= \arg\max_x \ \ln p_{Z|X}(Z|x) + \ln p_X(x) - \ln p_Z(Z) \\
&= \arg\min_x \ \left(-\ln p_{Z|X}(Z|x) - \ln p_X(x)\right) \\
&= \arg\min_x \ \left(-N\ln\sigma + \frac{1}{\sigma}\sum_{n=1}^{N} d_n - \ln p_X(x)\right) \\
&= \arg\min_x \ \left(\frac{1}{\sigma}\sum_{n=1}^{N} d_n - \ln p_X(x)\right).
\end{aligned} \tag{3.34}$$

Obtaining a closed form for $p_X(x)$ involves dependencies between plant positions because the plant positions are not mutually independent. Instead, using Equation 3.32, we iteratively obtain the MAP estimate of each plant position X_p separately:

$$\begin{aligned}
\hat{X}_p(Z, X_{q\neq p}) &= \arg\max_{x_p} p_{X_p|Z,X_{q\neq p}}(x_p|Z, X_{q\neq p}) \\
&= \arg\max_{x_p} \frac{p_{Z|X_p,X_{q\neq p}}(Z|x_p, X_{q\neq p})p_{X_p|X_{q\neq p}}(x_p|X_{q\neq p})}{p_{Z|X_{q\neq p}}(Z|X_{q\neq p})} \\
&= \arg\max_{x_p} p_{Z|X}(Z|X)p_{X_p|X_{q\neq p}}(x_p|X_{q\neq p}) \\
&= \arg\min_{x_p} \ \left(-\ln p_{Z|X}(Z|x) - \ln p_{X_p|X_{q\neq p}}(x_p|X_{q\neq p})\right) \\
&= \arg\min_{x_p} \ \left(-N\ln\sigma + \frac{1}{\sigma}\sum_{n=1}^{N} d_n + \ln(2\pi) + \frac{1}{2}\ln|R_p| + \frac{1}{2}\left\|x_p - \mu_p\right\|^2_{R_p^{-1}}\right) \\
&= \arg\min_{x_p} \ \left(\frac{1}{\sigma}\sum_{n=1}^{N} d_n + \frac{1}{2}\left\|x_p - \mu_p\right\|^2_{R_p^{-1}}\right).
\end{aligned} \tag{3.35}$$

So far, we have assumed that σ is a known parameter. We estimate σ after every Iterative Coordinate Descent (ICD) [164] iteration using the Maximum Likelihood Estimator (MLE):

$$\begin{aligned}\hat{\sigma}(Z,X) &= \arg\max_{\sigma} p_{Z|X,\sigma}(Z|X,\sigma) = ... \\ &= \frac{1}{N}\sum_{n=0}^{N-1} d_n.\end{aligned} \tag{3.36}$$

Particular cases

For the special case in which the prior term is not used, the estimate $\hat{X}_p(Z, X_{q\neq p})$ in Equation 3.35 is reduced to

$$\hat{X}_p(Z, X_{q\neq p}) = \arg\min_{x_p} \sum_{n=1}^{N} d_n. \tag{3.37}$$

This corresponds to the cost function of the k-means clustering technique. In this case, $\hat{X}_p(Z, X_{q\neq p})$ is usually obtained as in Equation 3.38, which is the average of the points in the cluster formed by plant p.

$$\hat{X}_p(Z, X_{q\neq p}) = \frac{\sum_{n=1}^{N} h(x_p|Z_n) Z_n}{\sum_{n=1}^{N} h(x_p|Z_n)}, \tag{3.38}$$

where

$$h(x_p|Z_n) = \begin{cases} 1 & \text{if } p = C(n) \\ 0 & \text{otherwise} \end{cases} \tag{3.39}$$

is the membership function that indicates whether the pixel n corresponds to plant x_p or not.

Another special case occurs when the prior distribution about the intra-row spacing prior is not used. When $\sigma_{p,i} \to \infty$, Equation 3.35 becomes

$$\lim_{\sigma_{p,i}\to\infty} \hat{X}_p(Z) = \arg\min_{x_p} \left(\frac{1}{\sigma} \sum_{n=1}^{N} d_n + \frac{1}{2} \left(\frac{x_{p,j} - \mu_{p,j}}{\sigma_{p,j}} \right)^2 \right). \tag{3.40}$$

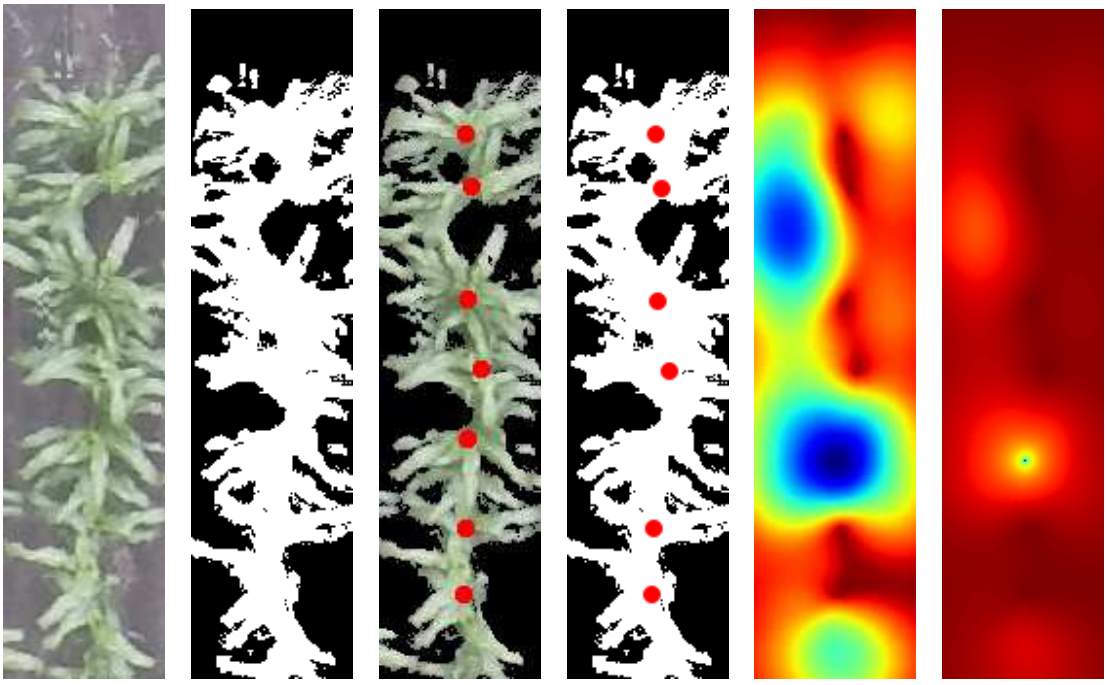
3.5.1 Experimental Results

The cost function to be minimized in Equation 3.37,

$$\phi(x_p) = \phi\left(\begin{bmatrix} x_{p,i} \\ x_{p,j} \end{bmatrix}\right) = \sum_{n=1}^{N} d_n \tag{3.41}$$

is plotted in Figure 3.16. Figure 3.16(a) shows a section of an orthorectified mosaic containing 7 sorghum plants. Figure 3.16(b) shows the color-based segmentation mask obtained as explained in Section 2.7. Figure 3.16(c) shows the ground truth of all the plant locations of the image. Figure 3.16(d) shows that, for this iteration, we fix 6 plant positions and estimate the location of the remaining plant. Figure 3.16(e) show the cost function at this iteration. Figure 3.16(f) emphasizes the location of the global minimum by showing it in gamma scale. The coordinate with lowest cost is selected as the location for this plant at this iteration. Note that the function is non-convex.

We use a very conservative minimization technique similar to gradient descent [164]. We cannot directly employ gradient descent because the cost function is not differentiable. Our minimization technique works as follows. At any given iteration, the cost function is evaluated in the surroundings of the current candidate position of a sorghum plant. A 8-pixel neighborhood is used as the surroundings of a pixel. The plant is moved to the pixel in this neighborhood that minimizes the cost function until we reach a local minimum.



(a) (b) (c) (d) (e) (f)

Fig. 3.16.: (a) Image containing 7 sorghum plants. (b) Segmentation mask Y, obtained as explained in Section 2.7. (c) Red dots are the ground truth of the plant locations. (d) All plant locations are assumed to be known except for one, which we are estimating. (e) Cost function 3.41. (f) Cost function in gamma scale to highlight the global minimum.

4. AN API AND WEB PLATFORM FOR PHENOTYPING

4.1 An API For Phenotypic Data And Metadata

Phenotypic data that can be collected in a crop field is very diverse. It can include phenotypic traits of a plot (such as canopy coverage), or phenotypic traits of individual plants (such as the length of each leaf). It can also include geometric information, such as the physical boundaries of the plot or the panel. It is also important that it is accompanied by metadata that makes the other data usable, such as the units of the measurements, the name of the panel, or plot numbers.

It is important to standardize these data so that multiple software tools can unambiguously exchange information. Not only tools, but also experts of very diverse fields such as machine learning, computer vision, remote sensing, agronomy, and plant science can understand, parse, modify, and store the data in a consistent way.

Simple file formats such as Comma-Separated Values (CSV) are not appropriate for these complex data, because we need multiple nested levels of information. We need to represent information of a plant inside a plot, which is inside a panel, which is inside a crop field, etc.

There also exist multiple Geographic Information System (GIS) standards that are often used in remote sensing, such as GeoTIFF, GeoJSON, or Shapefile. GeoTIFF only includes geometric information, and we want to also include information that is not linked to any specific TIFF image file. None of the formats that we are aware of are designed to store both geometry information and phenotypic traits estimated from remote sensing.

It is also required for a usable format to include metadata to structure the phenotypic data in a self-contained manner. For example, information about the crop field layout or identifiers of the breeding experiments are helpful to unambiguously

reference the phenotypic data. Because being usable and explicit is more important than being efficient, we prefer text-based formats over binary formats. This avoids imposing limits on data types, and allows the user to view and modify data without specialized software.

We have designed an Application Programming Interface (API) to be as generic as possible, that can represent phenotypic information, the layout of crop fields, and various types of metadata.

4.1.1 XML specification

Here we describe the API for phenotypic data, geometric information, and metadata. The file format we use to describe it is Extensible Markup Language (XML). However, this API should be considered an abstract specification of nested fields, and it could also be expressed in any other markup language such as JavaScript Object Notation (JSON).

The software implementing the methods described in this thesis to estimate phenotypic traits, read its inputs and write its outputs using this file format.

Listing 4.1 shows an example of an XML file that follows our API specification. Table 4.1 and Table 4.2 describe all the XML elements and attribute and the information they contain. For detailed definitions about agronomical terms describing a crop field structure, see Section 2.1.

Listing 4.1: Example of our API specification in XML format

```
<fields apiversion="0.4.0">
  <field>
    <name>F54</name>
    <orthophoto>
        <filename>20170609_F41_GNSS_INS_1CM.jpg</filename>
        <md5sum>8948b7b9d61279cf61b431513322a5c3</md5sum>
        <resolution>
            <x units="meters">0.01</x>
            <y units="meters">0.01</y>
        </resolution>
        <origin_coordinates>
```

```
        <latlong>
            <latitude units="degrees">40.470715</latitude>
            <longitude units="degrees">-86.991450</longitude>
        </latlong>
    </origin_coordinates>
</orthophoto>
<groundtruthed_by>
  <plot_boundaries>Jieqiong Zhao (zhao413)</plot_boundaries>
</groundtruthed_by>
<estimated_by>
  <plant_locations>automatic tool (run by Javier Ribera)</plant_locations>
</estimated_by>
<plant_locations>
  [[4230, 51546], [4171, 51846], [4172, 52076], [4174, 51922]]
</plant_locations>
<panels>
    <panel>
        <name>GxE</name>
        <plots>
            <plot>
                <orthophoto_chop_filename>
                    20160621_F54_GxE_Ortho_0.75cm_range_001_row_001_5szgohkl3bonbdjp1zg4.jpg
                </orthophoto_chop_filename>
                <location>
                    <top units="pixels" wrt="orthophoto">16736</top>
                    <left units="pixels" wrt="orthophoto">1659</left>
                    <bottom units="pixels" wrt="orthophoto">17280</bottom>
                    <right units="pixels" wrt="orthophoto">1755</right>
                </location>
                <plot_number>border</plot_number>
                <range_number>1</range_number>
                <row_number>84</row_number>
                <subrow_grid_location>
                    <x units="rows">1</x>
                    <y units="ranges">1</y>
                </subrow_grid_location>
                <leaf_count>296</leaf_count>
                <canopy_coverage>0.883</canopy_coverage>
                <plants>
                    <plant>
                        <location>
                            <y units="pixels" wrt="orthophoto">4230</y>
                            <y units="pixels" wrt="plot">36</y>
                            <y units="cm" wrt="plot">36.0</y>
```

```
                        <y units="cm" wrt="orthophoto">4230.0</y>
                        <x units="pixels" wrt="orthophoto">51546</x>
                        <x units="pixels" wrt="plot">44</x>
                        <x units="cm" wrt="plot">44.0</x>
                        <x units="cm" wrt="orthophoto">51546.0</x>
                    </location>
                    <leaf_count>13</leaf_count>
                </plant>
            </plants>
        </plot>
    </plots>
  </panel>
 </panels>
</field>
</fields>
```

Table 4.1.: Description of the elements of our API.

XML Element	**Description**
<field>	Crop field. This is the largest notion of area of land for agronomical purposes.
<panel>	Crop panel. A smaller area of land inside a crop field.
<name>	Name of the parent element as a human-readable string.
<groundtruthed_by>	Name of who ground truthed which trait.
<estimated_by>	Name of which software was used to estimate which trait.
<orthophoto>	Orthorectified image of the parent element. The orthophoto can be of a field or a panel.
<filename>	Name of the image file of about the parent element. This file name should be unique.

continues on next page...

<md5sum>	MD5 checksum [165] of the file, used to unequivocally reference to the file.
<resolution>	Size of a pixel in the orthophoto.
<origin_coordinates>	Coordinates of the origin of the orthophoto (used to georeference the pixels). The origin is defined as the top left coordinate of the orthophoto image.
<latlong>	Latitude-Longitude coordinates.
<latitude>	Latitude coordinate (north-south angular position with respect to the Equator).
<longitude>	Longitude coordinate (east-west angular position with respect to the Greenwich meridian).
<plot>	Field single-row plot, as defined in Section 2.1.
<range_number>	Range number of this plot.
<row_number>	Row number of this plot.
<plot_number>	Identifier of a plot (multiple row segments can have the same plot number).
<subrow_grid_location>	The location in the grid formed by the row segments of the panel that contains this plot. y indicates the range from top to bottom and x indicates the row from left to right.
<canopy_coverage>	Canopy coverage, as described in Section 2.6.
<leaf_count>	Number of leaves in the plot (if the elment is inside a plot) or in the plant (if inside a plant).

continues on next page...

<plant>	A single plant.
<location>	Location of the parent element.
<top>	If the location is a rectangle, “top” is the “y” coordinate of the top border. The “y” axis points to the bottom.
<bottom>	If the location is a rectangle, “bottom” is the “y” coordinate of the bottom border. The “y” axis points to the bottom.
<left>	If the location is a rectangle, “left” is the “x” coordinate of the left border. The “x” axis points to the bottom.
<right>	If the location is a rectangle, “right” is the “x” coordinate of the right border. The “x” axis points to the bottom.

Table 4.2.: Description of the attributes of our API. Attributes give more information to understand the content of an element.

XML Attribute	Description
apiversion	Version of the API as (x.y.z), where x corresponds to a major release version, y to a minor version (breaking backward compatibility), and z to an addendum (backwards compatible within the same minor version). The most current version and the one described in this thesis is v0.4.0.
units	Units of the measurement described in a particular element. This attribute should only be in elements that correspond to measurements (e.g., phenotypic traits).
wrt	"With Respect To". The "reference element" that an element is using. For example, if the element contains coordinates, this can be the origin of the coordinate system. Example: "plot" or "orthophoto".

4.2 A Web Platform For Phenotyping[1]

The methods described in this dissertation are implemented in the form of Python packages that can be executed from the command-line. However, in a production environment plant breeders and agronomists want to phenotype the plants of their own crop field. Processing massive amounts of data in a local machines is not a convenient or scalable approach because these end-users may not have enough processing power.

In this section, we describe DIBPS, the *Distributed Image-Based Phenotyping System*, an easy-to-use web platform running on the cloud, where users can estimate phenotypic traits using their own RGB imagery. Only a modern web browser is

[1]This web platform was developed as part of a joint work with Yuhao Chen.

needed to upload the users' data, select the traits that want to be estimated, and download the results.

DIBPS is implemented using Django, (**You need to put some reference cites in here for Django** a framework for web development using Python, and has the following features:

- Login with a personal user account, isolating the data of different users. A screenshot is shown in Figure 4.1.
- If a user is an administrator, create new users, and assign users administration permissions.
- Upload data, including images and XML files following the API described in Section 4.1.1. A screenshot is shown in Figure 4.2.
- Analyze uploaded data. A screenshot is shown in Figure 4.3. The available plant phenotyping methods are:
  - Plot Extraction from UAV. This corresponds to the method described in Section 2.4.
  - Plant localization. This corresponds to the method described in Section 3.3.
  - Leaf segmentation. This method is not described because it is not a contribution of this thesis. It was designed and implemented by Yuhao Chen.
- Download the results. A screenshot is shown in Figure 4.5.
- Visualize the extracted plots by drawing bounding boxes around each row segment in the orthophoto. This is available in the form of an additional tool.

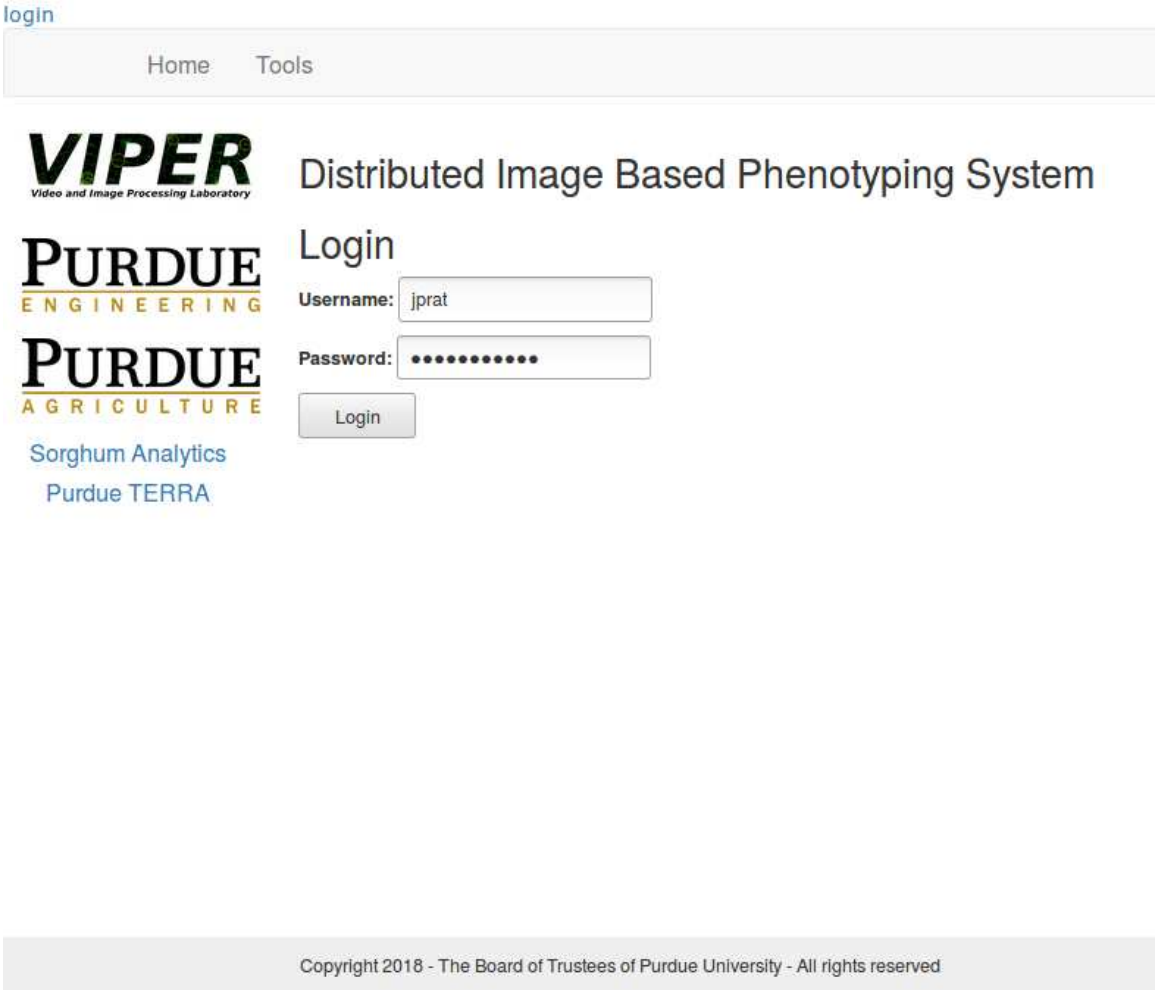


Fig. 4.1.: Page to login to Distributed Image-Based Phenotyping System (DIBPS). Each user has his own private account.

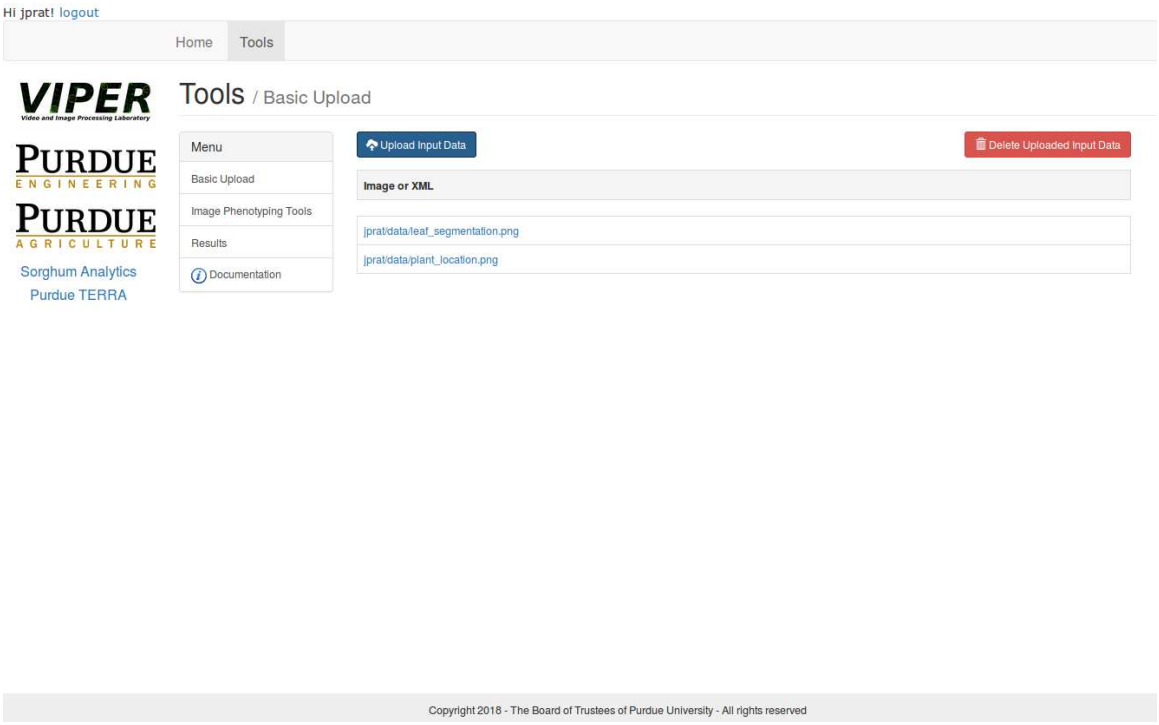


Fig. 4.2.: Page of DIBPS to upload user data.

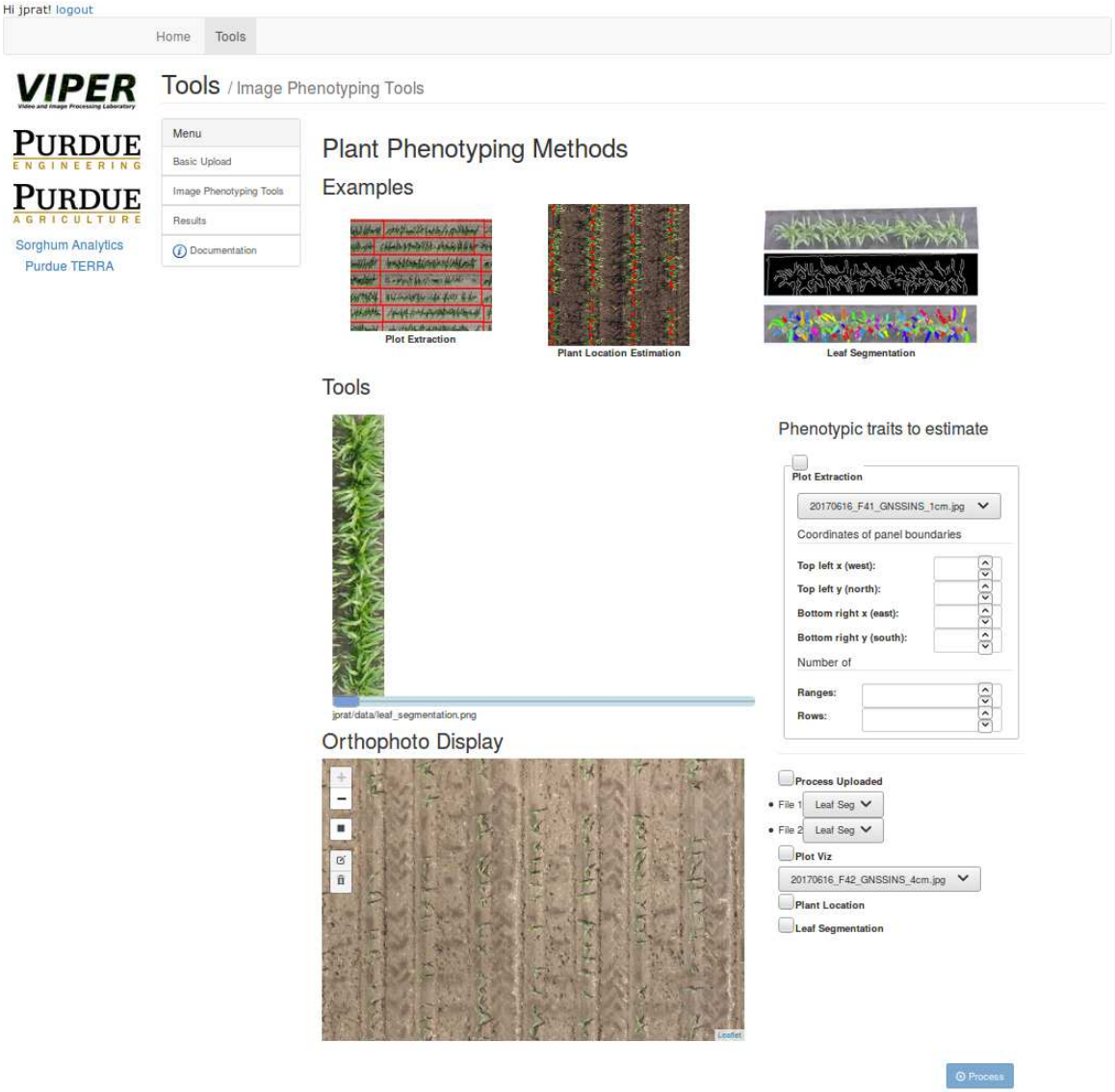


Fig. 4.3.: Page of DIBPS to process user data. The available plant phenotyping methods are plot extraction, plant localization, and leaf segmentation.

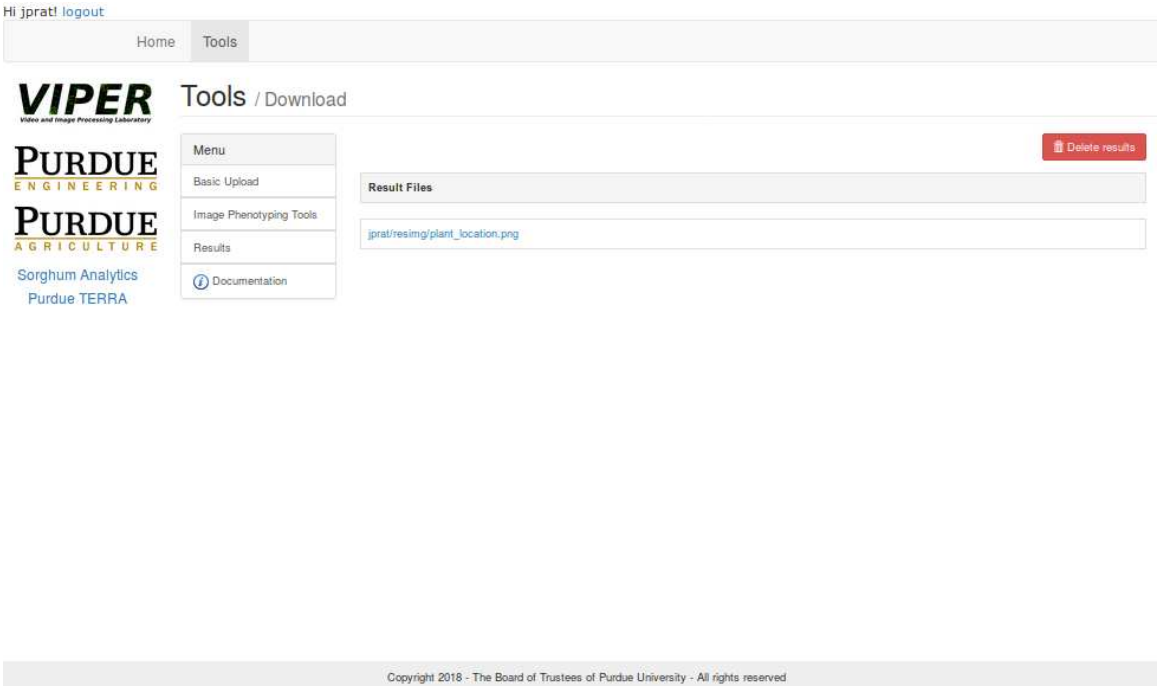


Fig. 4.4.: Page of DIBPS to download the results after the processing has finished.

The plot extraction tool requires the user to provide some parameters. The user provides them to DIBPS using the user interface shown in Figure 4.5. These parameters are:

- The coordinates of the panel of interest. This can be provided by typing the top-left and bottom-right coordinates in a text box, or by drawing a rectangle using the "Orthophoto Display" section.
- Number of rows and ranges in the panel. These correspond to N and M, respectively, in Section 2.4.

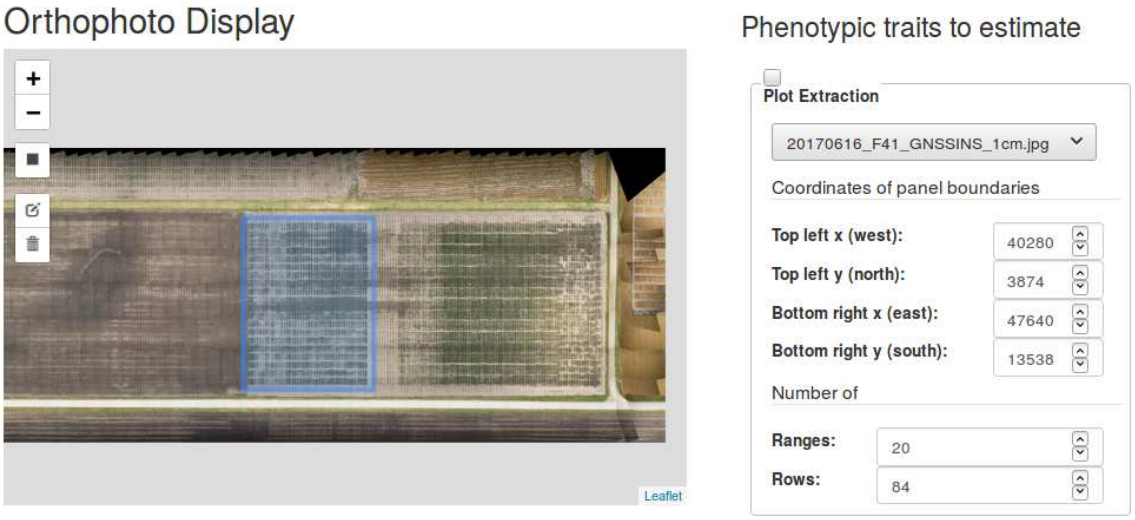


Fig. 4.5.: Parameters that the user needs to provide to the plot extraction tool in DIBPS.

5. CROWDSOURCING TO ENHANCE CROWDFLOW ESTIMATION

5.1 Crowd Flow Estimation

In this section, we describe the automatic method to which we incorporate crowdsourcing. This method is not a contribution of this thesis. It was originally conceived in [57] by Satyam Srivastava, Ka Ki Ng and Edward J. Delp. However, the details of the method are crucial to understand the incorporation of crowdsourcing. The only contribution is Section 5.1.3, which is a small improvement to this method.

In Section 5.1.1, they key ideas of the method are explained. Section 5.1.2 describes how to obtain the foreground pixel count, which is used in the final estimation. Section 5.1.4 describes the process to estimate the crowdedness. Finally, in Section 5.1.5, we explain the procedure to calibrate the parameters of the method with training data.

5.1.1 Key Ideas and Overall Scheme

The final goal is to estimate the number of people that have crossed a desired region of the image in a given time, i. e. the crowd flow.

This method follows an indirect approach. This is, it relates characteristics of the crowd to low level features. In this case, our low level features are the number of foreground pixels and texture features. The assumption is that the number of people present is proportional to the number of foreground pixels.

The region of the space where people crossing will be counted is called “Tripwire” and it must be manually specified by the user.

We compute the crowd flow estimation proportionally to the accumulation of foreground pixels in the Tripwire:

$$\upsilon_N = \frac{\tilde{S}_N}{C}, \tag{5.1}$$

where υ_N represents the number of people that have crossed the Tripwire up to the frame number N. S_N is the accumulated foreground pixel count in that period. The details on how to obtain S_N will be discussed in Section 5.1.2. C is the scaling factor used to scale the number of foreground pixels to the number of people crossing. It represents the number of pixels that every person shows. Crowdedness is related to occlusions, and more occlusions mean less pixels can be seen per person. Because of this reason, the lineal proportion holds true as long as the crowdedness remains constant. Thus, the value of C depends on the level of crowdedness of the scene. How to obtain the scaling factor is explained later in Section 5.1.4.

In addition to the Tripwire, the operator must also select another region called ROI used for the crowdedness estimation. An example of a Tripwire and a ROI over a surveillance video is depicted in Figure 5.1.

An overall scheme of the whole method is shown in Figure 5.2. For each frame, the lower branch computes the foreground pixel count inside the Tripwire and the upper branch estimates the scaling factor from the density level of the ROI. The lower branch is detailed in Section 5.1.2, and the upper branch in Section 5.1.4.

5.1.2 Foreground Pixel Count

In Figure 5.3, the process to obtain the foreground pixel count from the Tripwire is schematized.

Let the image of an arbitrary frame with frame number n be defined as

$$F_n = \{f_n(i,j) | i = 0, 1, ..., W-1 \text{ and } j = 0, 1, ..., H-1\}, \tag{5.2}$$



Fig. 5.1.: Example of a ROI and a Tripwire drawn over a frame of a surveillance video. ROI is in blue, and Tripwire in red. Source: [57]

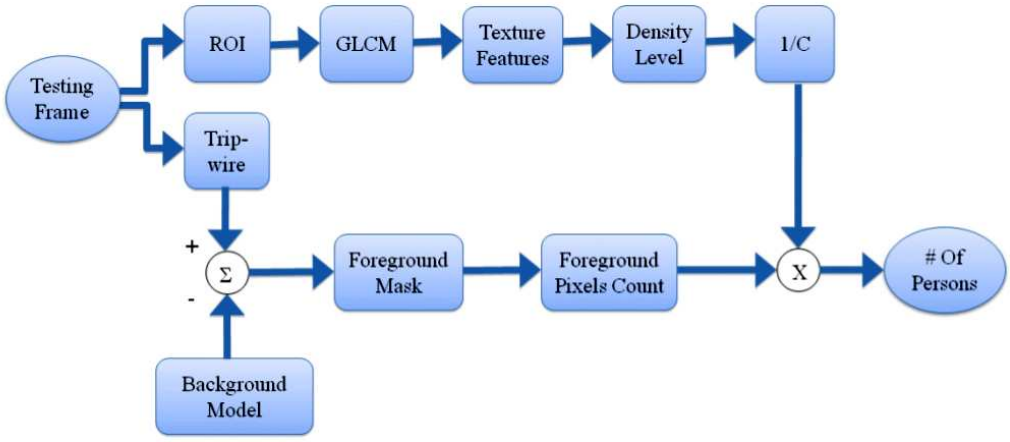


Fig. 5.2.: Overall scheme of the automatic crowd flow estimation method.

where $f_n(i,j)$ is the value of the pixel (that may be a 3-D RGB value) at row coordinate i and row column j. W and H are the width and height of the frame, respectively. Then, we can represent the foreground mask of F_n as an indicator function I_n:

$$I_n(i,j) = \begin{cases} 1 & : (i,j) \in \text{foreground segmentation} \\ 0 & : (i,j) \notin \text{foreground segmentation.} \end{cases} \tag{5.3}$$

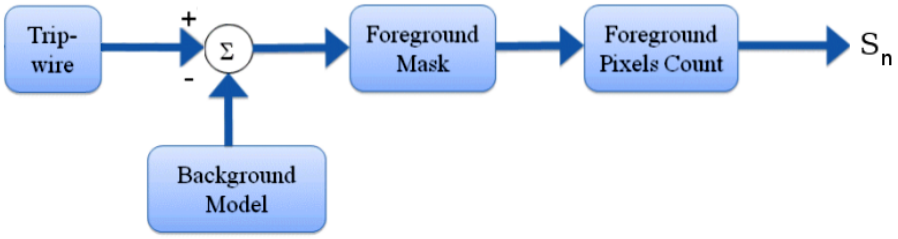


Fig. 5.3.: Scheme of the foreground pixel count.

Also, let the Tripwire be represented as the set of pixels $\Re = \{(i,j)|(i,j) \in$ Tripwire$\}$. An example of the foreground mask of the Tripwire is shown in Figure 5.4.

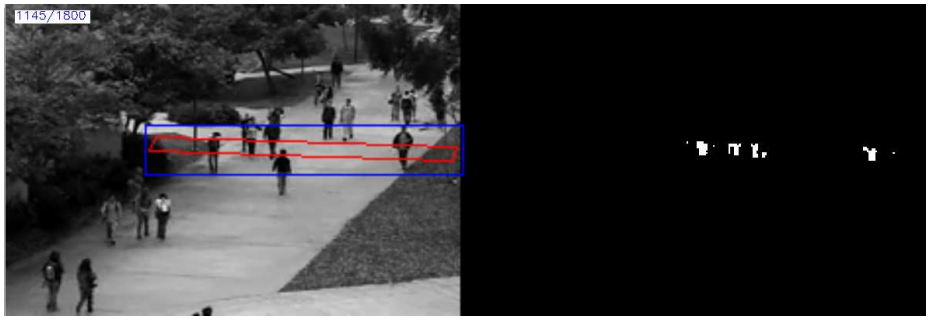


Fig. 5.4.: An arbitrary frame, on the left, and the foreground mask of the Tripwire on the right.

To obtain the foreground mask I_n, we apply the Background Subtraction (BS) from [166–168] that makes use of Mixture Of Gaussians (MOG). We use the implementation from OpenCV 2.4.9 [169].

There is only need to obtain the foreground mask of the Tripwire. Note that the computational cost of performing Background Subtration in a small region is much smaller than in the whole frame.

The foreground pixel count S_n of a frame F_n is then determined as how many pixels belong to the foreground segmentation, i. e,

$$S_n = \sum_{(i,j)\epsilon\Re} I_n(i,j). \tag{5.4}$$

The foreground pixel accumulation S_N up to the frame number N is the accumulation of foreground pixels in that period, i. e,

$$S_N = \sum_{n=0}^{N} S_n = \sum_{n=0}^{N} \sum_{(i,j)\epsilon\Re} I_n(i,j). \tag{5.5}$$

However, Equation (5.5) deals with all frames in the period the same way, regardless of their level of crowdedness. To overcome this limitation, we will take into consideration the variance in the level of crowdedness later in Section 5.1.4.

In addition, due to perspective distortions, the blob size of an object in the foreground mask varies according to the distance from the camera. Unfortunately, if Equation (5.5) is directly used, all pedestrians will be assumed to be at the same distance from the camera. A weighting function is introduced to consider perspective in the foreground mask.

5.1.3 Weighting Function

We suggest to incorporate some weights in (5.5) to account for the effect of perspective distortions. We incorporate a weighting function $\omega(i,j)$ that weights every pixel (i,j) of the foreground mask $I(i,j)$ as in Equation (5.6). The result of using a weighting function results in a weighted foreground count accumulation $\tilde{S}_N$:

$$\tilde{S}_N = \sum_{n=0}^{N} \sum_{(i,j)\epsilon\Re} I_n(i,j) \cdot \omega(i,j). \tag{5.6}$$

The goal is to make the value of $\omega(i,j)$ higher as the object at pixel (i,j) is further from the camera. The proposed method to compute $\omega(i,j)$ works as follows:

First, the user is asked to draw a quadrilateral which corresponds to a rectangle on the floor in the real world. This quadrilateral is defined by its four sides L_1, L_2,

L_c and L_f as in Figure 5.5. The user is also asked to indicate the closer and further sides, L_c and L_f, respectively.

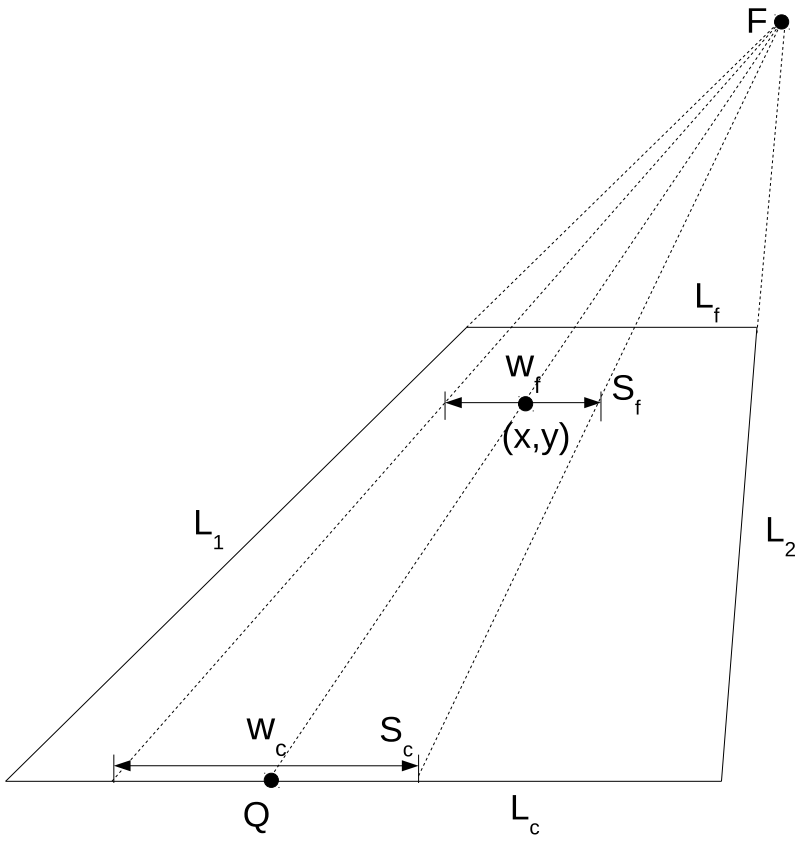


Fig. 5.5.: Weighting scheme illustration. The further a pixel is, the smaller the segment W_f becomes and the higher its weight results.

Due to perspective, lines L_1 and L_2, which are parallel in the real world, appear to intersect at the vanishing point F. Second, for each point (i, j) inside the tripwire $\Re$, a line parallel to L_1 and L_2 in the real world is defined as the line that passes through (i, j) and the vanishing point F. This line intersects with L_c at the point Q. Third, we define a segment S_c centered at Q and laying on L_c. This segment S_c will always have a predefined length W_c. Forth, this segment is projected towards the vanishing point, F, until it reaches the original point (i, j), resulting in a segment called S_f

centered at (i, j) and parallel to S_c, L_f and L_c. This segment S_f has length W_f. As a result of perspective, the resulting length W_f is not equal to W_c: it is shorter because it is further from the camera.

Therefore, the weighting function is defined as:

$$\omega(i, j) = \frac{W_c}{W_f}. \tag{5.7}$$

$\omega(i, j)$ will not depend on the value of W_c, because W_f will be proportional to W_c. Because of this, the length W_c can be fixed with the same value for all points (i, j), e.g, $W_c = 1$.

The resulting weighting function does not depend on n, so it can be computed only once before the crowd flow estimation starts.

The implementation of this method is done using by homogeneous coordinates [97], as it leads to simpler equations in perspective geometry.

5.1.4 Crowdedness Estimation

Once the weighted foreground pixels count $\tilde{S}_N$ has been computed, it must be scaled properly to get the final people count. As shown in Equation (5.1), the proportionality factor is $1/C$. In order to estimate C, i.e, the number of pixels every person shows, the level of crowdedness must be estimated. In this thesis, we refer to level of crowdedness, crowdedness and level of occlusion as synonyms. Given that more crowdedness results in more occlusion between people and less pixels visible per person, a higher level of crowdedness should correspond to a lower value of C.

This method uses the relation between crowdedness and texture described in [55, 56]. Figure 5.6 suggests that the texture of an image is related to its crowdedness. A sparse scene, with a low level of crowdedness, presents a fine texture. In contrast, a crowded scene, with a high level of crowdedness, presents a coarse texture. Hence, we deduce the level of crowdedness of every frame using texture properties. An extensive study of texture in image processing can be found in [170].

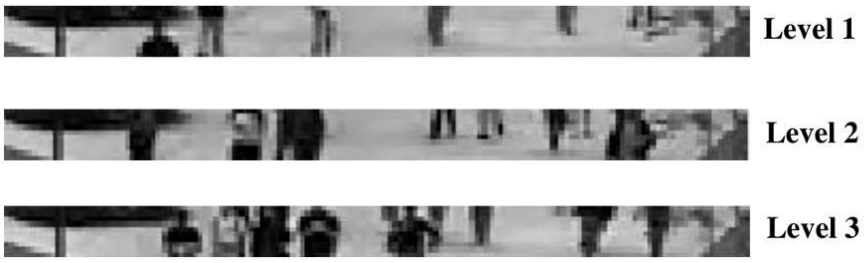


Fig. 5.6.: A sparse scene, with a low level of crowdedness, presents a fine texture. In contrast, a crowded scene, with a high level of crowdedness, presents a coarse texture.

In this thesis, we characterize the texture of an image by means of a Gray Level Co-occurrence Matrix (GLCM) [170]. The GLCM matrix models a texture by characterizing the probability that a pixel with a given gray level is adjacent to another specific gray level. It consists on an $S \times S$ matrix Pij that contains the probability of "jumping" from gray level i to j when the image is scanned in a given direction, where $i, j = 0, 1, ..., G-1$, and G is the number of quantized gray tones of the input image. Generally, the GLCM technique analyzes pixels separated by distance d, but in this method we employ adjacent pixels, i.e, we fix $d = 1$. Figure 5.7 shows an example of GLCM calculations of a very simple image.

The GLCM is normalized by dividing all the elements by the number of elements such that all elements sum up to one, i. e, $\sum_{i=1}^{G} \sum_{j=1}^{G} P_{ij} = 1$.

We obtain the level of crowdedness in the Tripwire for each frame. The user must also manually select a ROI, a rectangular region of the image, and it must surround the Tripwire. This will be the region where the crowdedness will be estimated.

This method follows the scheme in Figure 5.8 to characterize the texture of every frame. From the ROI of a given frame, 4 GLCM matrix are created. Each matrix is computed by considering different directions for the adjacent pixels: right (0^o), top-right (45^o), top (90^o), and top-left (135^o). Then for each matrix, we extract 4 scalars features: energy (Equation (5.8)), entropy (Equation(5.11)), homogeneity (Equation (5.10)) and contrast (Equation (5.9)). With these 16 scalar features, a

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

(a) 4x4 image consisting of 4 gray levels

$$P_H = \begin{pmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

(b) GLCM of 5.7(a) scanned in horizontal ($\theta = 0^{\circ}$)

$$P_V = \begin{pmatrix} 6 & 0 & 2 & 0 \\ 0 & 4 & 2 & 0 \\ 2 & 2 & 2 & 2 \\ 0 & 0 & 2 & 0 \end{pmatrix}$$

(c) GLCM of 5.7(a) scanned in vertical ($\theta = 90^{\circ}$)

$$P_{LD} = \begin{pmatrix} 2 & 1 & 3 & 0 \\ 1 & 2 & 1 & 0 \\ 3 & 1 & 0 & 2 \\ 0 & 0 & 2 & 0 \end{pmatrix}$$

(d) GLCM of 5.7(a) scanned in diagonal ($\theta = 135^{\circ}$)

$$P_{RD} = \begin{pmatrix} 4 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 \\ 0 & 2 & 4 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

(e) GLCM of 5.7(a) scanned in diagonal ($\theta = 45^{\circ}$)

Fig. 5.7.: GLCM calculations ($d = 1$).

16-D texture feature vector, t_n, is assembled to represent the texture of the ROI in the frame number n.

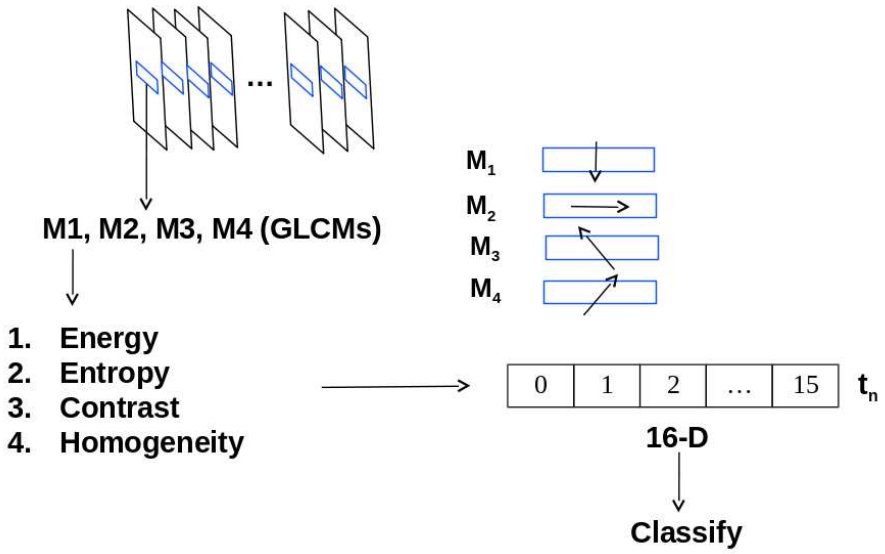


Fig. 5.8.: Scheme to obtain the texture feature vector.

"Energy" is defined to be the square root of the sum of squared elements in the GLCM and is maximum for a constant image. Contrast is a measure of the intensity contrast between a pixel and its neighbors over the whole image. Contrast is minimum for a constant image. Homogeneity measures the closeness of the distribution of elements in the GLCM to the diagonal of the matrix and is 1 for a diagonal GLCM. Entropy is a statistical measure of randomness.

$$Energy(P) = \sqrt{\sum_{i,j} p_{ij}^2} \tag{5.8}$$

$$Contrast(P) = \sum_{i,j} p_{ij}(i-j)^2 \tag{5.9}$$

$$Homogeneity(P) = \sum_{i,j} \frac{p_{ij}}{1+(i-j)^2} \tag{5.10}$$

$$Entropy(P) = -\sum_{i,j} p_{ij} \log p_{ij} \tag{5.11}$$

Once we have the texture feature vector t_n of the ROI, we have to classify it into one of the L levels of crowdedness. As a result of the training process, explained later in detail in Section 5.1.5, every level of crowdedness is represented by a texture feature vector, called "reference" vector. Each reference feature vector is associated with its corresponding scaling factor C_l, $l = 0, 1, ..., L-1$. This is a classification problem in which we classify a 16-D vector into one of the levels of crowdedness represented by the L reference vectors. The closest reference vector is used for classification, this is a K-Nearest Neighbors (KNN) classifier in which K = 1. Figure 5.9 shows a 2-D representation of the feature space.

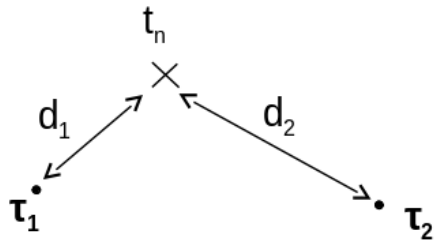


Fig. 5.9.: A 2-D representation of the 16-D feature space. t_n is the texture feature vector of the ROI and τ_l, $l = 0, 1, ..., L-1$ are the reference feature vectors of the L levels of crowdedness. The closest τ_l to t_n is taken and its corresponding scaling factor is used.

The level of crowdedness of the frame n, l_n is estimated as the same level of the closest reference texture feature, as shown in Equation (5.12):

$$l_n = \arg \min_l d(t_n, \tau_l). \tag{5.12}$$

The Euclidean distance function $d(\cdot, \cdot)$ is used to compare feature vectors. As the range of the possible values of each feature might differ much, this may lead to the domination of one component in the distance measure. Because of this, we normalize the components of the feature vectors to approximately [0, 1].

When the level of crowdedness is determined, the second step is to use the associated scaling factor C_n for this level of crowdedness to estimate the number of people, as shown in Equation (5.13), which is a combination of Equations (5.1) and (5.4):

$$\upsilon_N = \sum_{n=1}^{N} \frac{\sum_{x,y \epsilon \Re} I_n(x,y) \cdot \omega(x,y)}{C_n}. \tag{5.13}$$

5.1.5 Training Data

This method requires a preliminary training stage to be performed by an expert, e.g, the operator of the system. The accuracy of the result is highly dependent on the quality of the training data and the exactitude of the training stage output.

Let L be the number of crowdedness levels specified by the operator for a video. As stated earlier, the scaling factor C relating the number of foreground pixels to the number of people crossing the Tripwire region is dependent on the level of crowdedness. The training process aims to train the classifier to classify the 16-D texture feature vectors into one of the levels of crowdedness. The first output of the training stage is L texture feature vectors $(\tau_0, \tau_1, ..., \tau_{L-1})$ representing L levels of crowdedness. Each vector is used as a reference vector for the corresponding level of crowdedness. From now on we shall call these L texture feature vectors as reference vectors. During the training stage, the operator is asked to count the number of people crossing the Tripwire region during a short period of time and C_l is determined accordingly. $C_l, l \in 0, 1, ..., L-1$, is the second output of the training stage.

The training process is performed as follows: first, the system operator is asked to mark the Tripwire region and the ROI surrounding it. The perspective weighting function is determined according to Section 5.1.3. Figure 5.1 shows an example from our test dataset [171] including the Tripwire region and the ROI surrounding it. Next, M video frames are chosen randomly from the training video segment. For each frame, the operator is asked to classify each video frame into one of the L levels of crowdedness. τ_l's are estimated as the average of the training vectors for each corresponding level. To find the scaling factor C_l for each level, we find the longest

stable period for each level of crowdedness. A stable period is a set of consecutive frames with the same level of crowdedness. L video segments corresponding to each level l are shown during training and the operator has to count the number of persons crossing the tripwire region, υ_l. The scaling factor C_l for each level of crowdedness is determined by computing the accumulation of foreground pixels in each video segment, $\tilde{S}_l$, and inverting Equation (5.1), which results in Equation (5.14):

$$C_l = \frac{\tilde{S}_l}{\upsilon_l}. \tag{5.14}$$

We developed a visual interface to guide the operator in the training data stage. This user interface is described in Subsection 5.3.

5.2 Uncertainty Characterization and Crowdsourcing

The classification step and the resulting scaling factor of the current ROI is critical for the performance of the method. Shadows, distortions or sudden or gradual environmental changes may affect the texture and a misclassification of the texture feature vector, resulting in a biased crowd flow estimation.

To overcome this, the performance of the classification is enhanced by using crowdsourcing. We aim to reduce the uncertainty in the classification by asking the "o-crowd", the crowdsourcing members, to assist the automatic method.

5.2.1 Uncertainty of The Classifier

For a frame n, the distance between the texture feature vector t_n and the nearest reference vector, d_1, might be very comparable to the distance between t_n and the second nearest neighbor, d_2. In this case, the classifier will choose the level of crowdedness corresponding to the nearest reference vector even if the difference between the two distances $(d_2 - d_1)$ is very small.

Let t be the texture feature vector, or data point, of a frame, and let $\tau_l, l = 1, 2, 3, ..., L$ be the reference texture feature vectors, or reference points. $\tau_l, l =$

$1, 2, 3, ..., L$ are sorted such that τ_1 is the nearest reference point to the data point t, while τ_L is the furthest. The distance from t to τ_l is denoted by d_l. Accordingly, we have $d_l \leq d_k, \forall l \leq k$. d_1 might be very comparable to d_2. We propose three different methods to quantify the uncertainty of the classification of t.

The first characterization of uncertainty consists of the ratio of the distances to the two nearest reference points as shown in Equation (5.15). The second characterization, expressed in Equation (5.16), is a generalization of the first one and uses the ratio to all of the reference points. The third characterization, in Equation (5.17), makes use of the distance d_b to the border that separates d_1 and d_2. By definition, the three characterizations are bounded between 0 and 1. In Equation 5.17, $d_{1,2}$ is the distance bewtween τ_1 and τ_2.

$$\mu_1 = \frac{d_1}{d_2} \tag{5.15}$$

$$\mu_2 = \sum_{i=2}^{L-1} \frac{d_1}{d_i} \tag{5.16}$$

$$\mu_3 = \frac{1}{1 + \frac{d_b}{d_{1,2}}} \tag{5.17}$$

5.2.2 Use of Crowdsourcing

For each frame n, the uncertainty of its classification μ_n is watched. If μ_n exceeds a predefined threshold, α, (Equation (5.18)) the data point t_n is considered uncertain and is referred to the o-crowd for labeling. The o-crowd is asked to classify the uncertain frame number n.

$$\mu > \alpha \tag{5.18}$$

α represents the maximum uncertainty allowed in the classifier, and must be between 0 and 1. For example, when using the first uncertainty type, $\alpha = 0.5$ requires the feedback of the o-crowd whenever the distance between t_n and the second closest reference is less than twice the distance to the closest reference vector. α relates to

how often we reach out the o-crowd. We call α the "crowdsourcing parameter." A lower value of α implies a lower utilization of the o-crowd, whereas a higher value makes the o-crowd to be asked more often. For instance, if $\alpha = 0$, the o-crowd will be referred at every frame, and if $\alpha = 1$ the o-crowd will never be referred.

This threshold defines a "certainty area" around the reference vectors. When a data point falls inside the certainty area, it is considered "certain." A smaller certainty area implies a lower probability for a testing data point falling inside a certainty area. The higher α, the bigger these certainty areas are. Figures 5.10 and 5.11 illustrate a 2-D representation of the certainty areas around two reference vectors. Feature vectors outside all certainty areas are automatically referred to the o-crowd. The video frame corresponding to this t_n is shown to the o-crowd and they are asked to estimate its level of crowdedness and the number of people in the crowd.

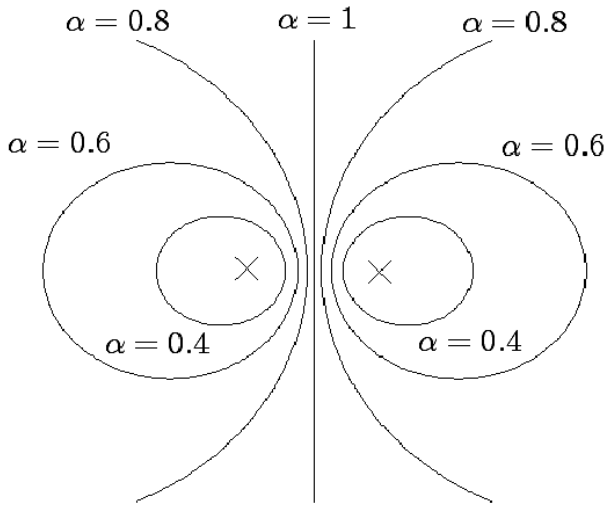


Fig. 5.10.: Uncertainty characterization #1. Certainty areas around two reference points corresponding to four values of α. They correspond to hyperspheres, whose size increases with α.

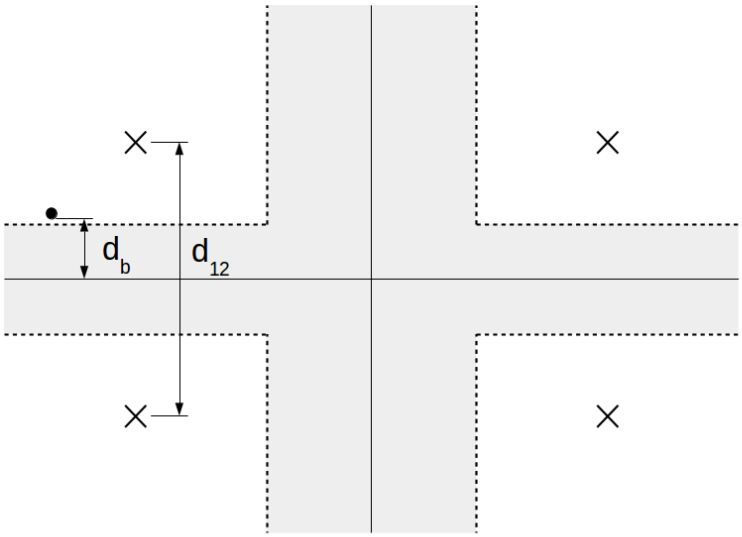


Fig. 5.11.: Uncertainty characterization #3. The two crosses are two reference vectors. The dot is the data point to be classified. The uncertainty area (in gray) is a region in the margin around the border in between two references. The width of this region is relative to the separation of the two references and α.

5.2.3 Crowdsourcing Task

When the o-crowd is asked to assist the classifier, we create a "crowdsourcing task". In Amazon MTurk [67], these are called HITs, Human Intelligence Tasks, but in this thesis we will refer to them more generally as crowdsourcing tasks. The members of the o-crowd are requested to solve it.

In our case, the task consists of classifying the uncertain frame and counting people crossing the Tripwire. Figure 5.12 shows an example of crowdsourcing task. First, the o-crowd member is asked to classify the ROI of the uncertain frame into one of the L levels of crowdedness. Second, the number of people crossing the Tripwire in a short video segment around the uncertain frame must be provided.

The video shown to the o-crowd is assumed to contain a constant level of crowdedness. Thus, the length of the video must be short enough so the crowdedness does not change. However, very short videos imply counting fractions of bodies, which

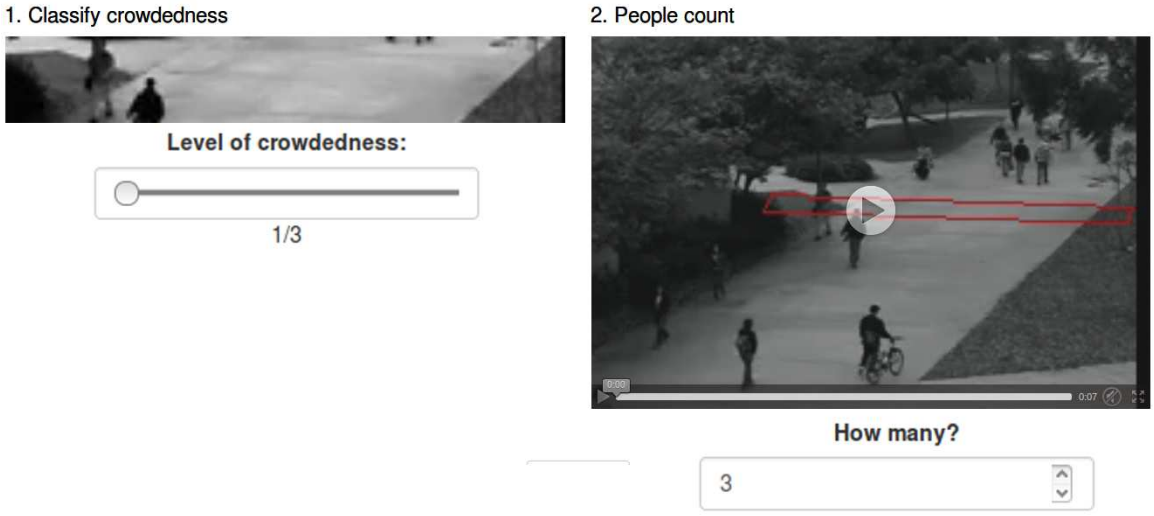


Fig. 5.12.: The o-crowd is asked to classify the ROI into one of the L levels of crowdedness, and also to count the number of people crossing the Tripwire.

is notably hard for a human observer. The length of the video segment was fixed empirically to 200 frames. 200 frames in the UCSD dataset [171], which runs at 30 fps, corresponds to 6.6 s.

With this information, a new scaling factor can be computed the same way as in the training stage as in Equation (5.14).

$$C = \frac{\tilde{S}}{v} \tag{5.19}$$

where v is the people count provided by the o-crowd and $\tilde{S}$ is the accumulation of foreground pixels in the video segment of the crowdsourcing task. Now the data point t_n has a new scaling factor and can be used in the crowd flow estimation.

In addition, when using crowdsourcing, we must make sure not to ask the o-crowd for redundant information. This is particularly important because consecutive video frames are likely to have similar feature vectors. Therefore, we store all the feature vectors for which the o-crowd has provided feedback as new reference vectors, as well as their recently computed scaling factors. Implicit certainty areas are created around the new reference vectors. Hence, future feature vectors in next frames will

not be referred to the o-crowd, as they will likely fall into the certain area of the new reference vector. Also, future classifications will perform better by taking into consideration this new reference vector.

However, not all reference feature vectors are created equal. There is an important difference between the feature vectors created during the training process and new feature vectors labeled by the o-crowd. The vectors generated during training process have been averaged from many frames, and the training was performed by an expert operator. In contrast, the new feature vectors have been labeled by an inexpert o-crowd using a single frame. Thus, the original reference vectors are more trustworthy than the newly incorporated reference vectors. To take this into account, certainty areas surrounding o-crowd reference vectors are scaled down. This is done by reducing α by a factor of 0.9 only for reference vectors produced by crowdsourcing.

In conclusion, we make use of crowdsourcing to help the automatic method whenever the classifier analyzes a frame that is expected to lead to a poor estimation. The information provided by the o-crowd is incorporated to the classifier in an online basis, thus making the method learn from crowdsourcing.

5.2.4 Results Aggregation

When we reach out to the o-crowd for labeling, we may obtain different answers for the same crowdsourcing task. This is because o-crowd members may have different abilities and personal biases. Thus, we propose two different approaches to aggregate the crowdsourcing task results.

The first approach averages all of the o-crowd answers while the second approach calculates the average of the subset with the smallest variance. Let Q be the number of members of the o-crowd who have solved the same crowdsourcing task. We have Q labels to aggregate. Let $\tilde{S}^{(q)}$ be the people count provided by o-crowd member number q, $q = 1, ..., Q$. The first approach would aggregate crowdsourcing tasks as Equation (5.21), and the second approach would aggregate them using Equation (5.21).

$$\tilde{S} = \frac{1}{Q} \sum_{q=1}^{Q} \tilde{S}^{(q)} \tag{5.20}$$

$$\tilde{S} = \frac{1}{|\Omega|} \sum_{q \in \Omega} \tilde{S}^{(q)} \tag{5.21}$$

where

$$\Omega = \arg\min_{\Omega_s} \sum_{q \in \Omega_s} \left(\tilde{S}^{(q)} - \frac{1}{|\Omega_s|} \sum_{q \in \Omega_s} \tilde{S}^{(q)} \right)^2 \tag{5.22}$$

is the subset of the labels containing $|\Omega_s| = K$ labels with lowest variance, $s = 1, ..., \binom{Q}{K}$.

5.3 Web Platform

In this section, we describe a web-based tool that allows fine control of the crowd flow estimation. It is also used by the o-crowd members to solve crowdsourcing tasks.

5.3.1 Control by The Operator

There can be many different videos being analyzed at the same time in the server. The tab shown in Figure 5.13 lets the operator monitor every process which is running a crowd flow estimation in the server. Each process can analyze only one video file or one real-time stream. For each process, the operator can monitor:

- The Process Identifier (PID) of the UNIX process
- The video being analyzed. When clicked, it is displayed on the screen
- The progress of the analysis, as the frame number being analyzed
- The number of people that have been counted crossing the Tripwire so far
- The last time instant when we have information about this process

- Whether this process has emitted a crowdsourcing task and it has not been solved yet
- How many crowdsourcing tasks have been emitted
- The value of the crowdsourcing parameter α
- The name (or label) used to reference the training data that this process is using
- The number of reference feature vectors that the classifier contains
- Whether the "time machine" option is enabled or not
- Whether the process has reached the end of the video or not

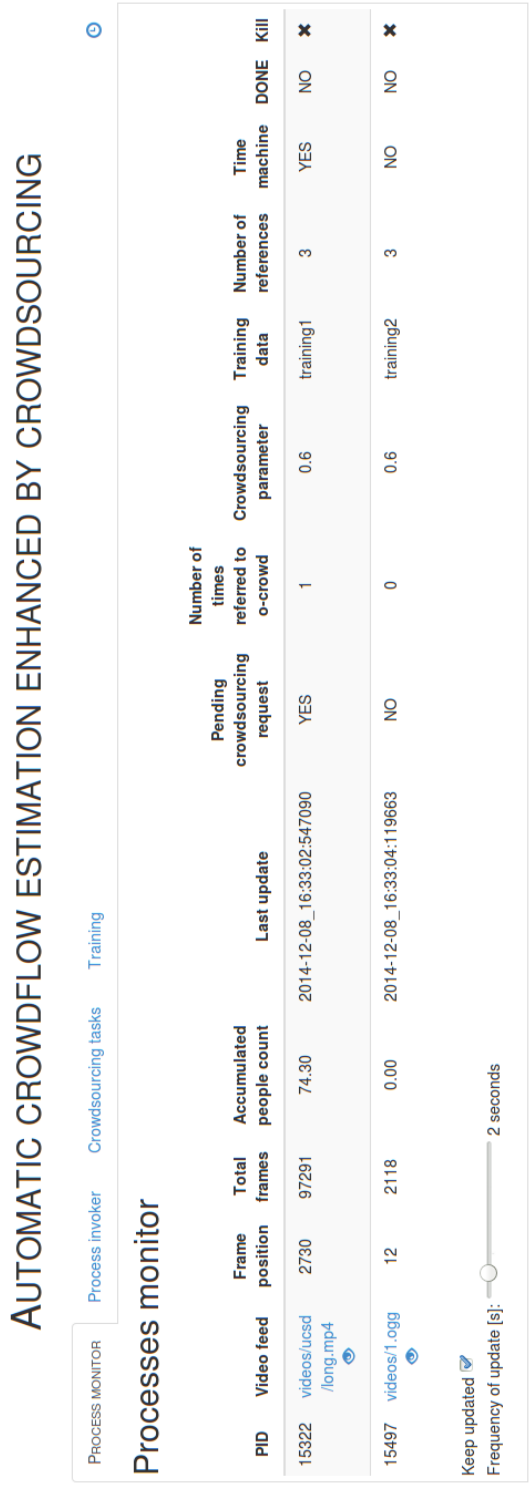


Fig. 5.13.: Web platform's processes monitor. It lets the operator watch and stop processes running a crowd flow estimations in the server.

The time machine option makes the crowd flow estimation process store the frame number at the instant when a crowdsourcing request is created. Then, whenever this task is solved, it goes back to that frame number and keeps analyzing the video since that time instant. This rewinding allows to use the new information provided to method by the o-crowd to improve the estimation as much as it can, as by default the estimation does not stop because a crowdsourcing task has been emitted. When a real-time stream is being analyzed, we disable the option to activate this feature.

The operator can stop the analysis of a specific process using the "×" button. This kills the UNIX process running in the server. Finally, the slider in the lower part sets how often this information is updated from the server.

In the second tab, the operator can invoke new processes in the server to analyze a video file or a real-time stream. Figures 5.14 and 5.15 show this screen. If the operator has selected a video file, he must write the path to the video file. If he has selected a real time video stream, a list of incoming streams will be shown and one must be selected.

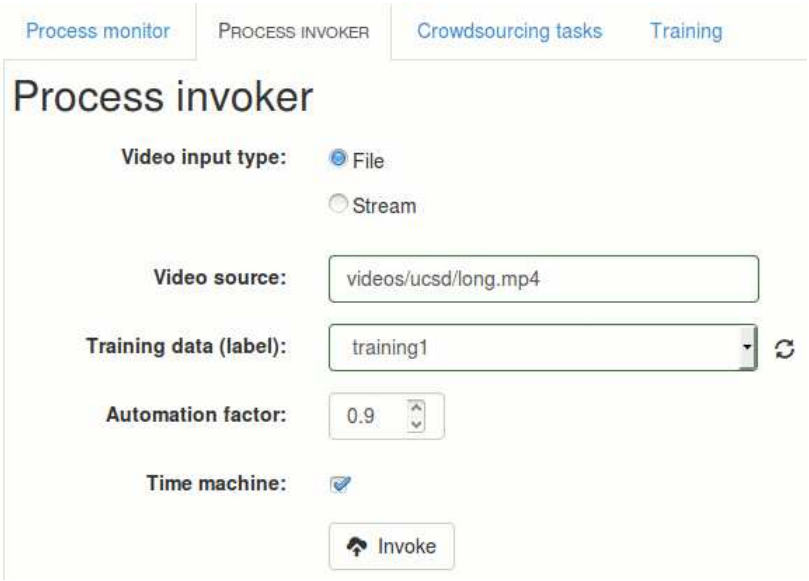


Fig. 5.14.: Web platform's processes invoker. A video file is selected as video input type.

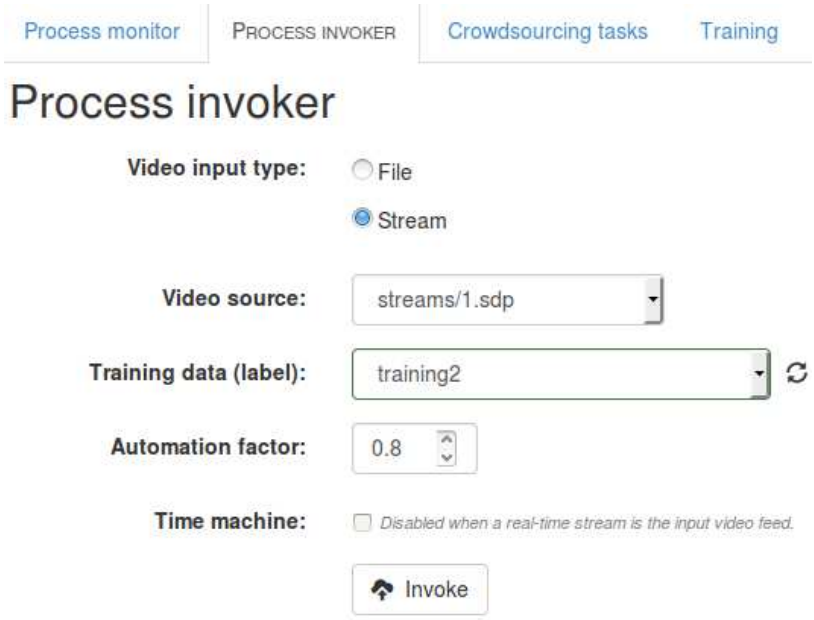


Fig. 5.15.: Web platform: processes invoker. A real-time video stream is selected as video input type.

5.3.2 Crowdsourcing Tasks

The o-crowd members are only allowed to access the screen shown in Figure 5.16. The members of the o-crowd can select one of the pending crowdsourcing tasks, i.e, a crowdsourcing task that has not been solved yet. A crowdsourcing task contains the PID of the process that generated it and the time when the task was generated.

Once the desired task is selected, the questions to the o-crowd will appear on the bottom of the page, as shown in Figure 5.16. The o-crowd member can move the slider to indicate the level of crowdedness of the uncertain frame and type how many people are crossing the Tripwire in the video segment. Once all answers are provided, the button "Solve Task" sends the solved crowdsourcing task to the server so it can be incorporated to the process that generated it.

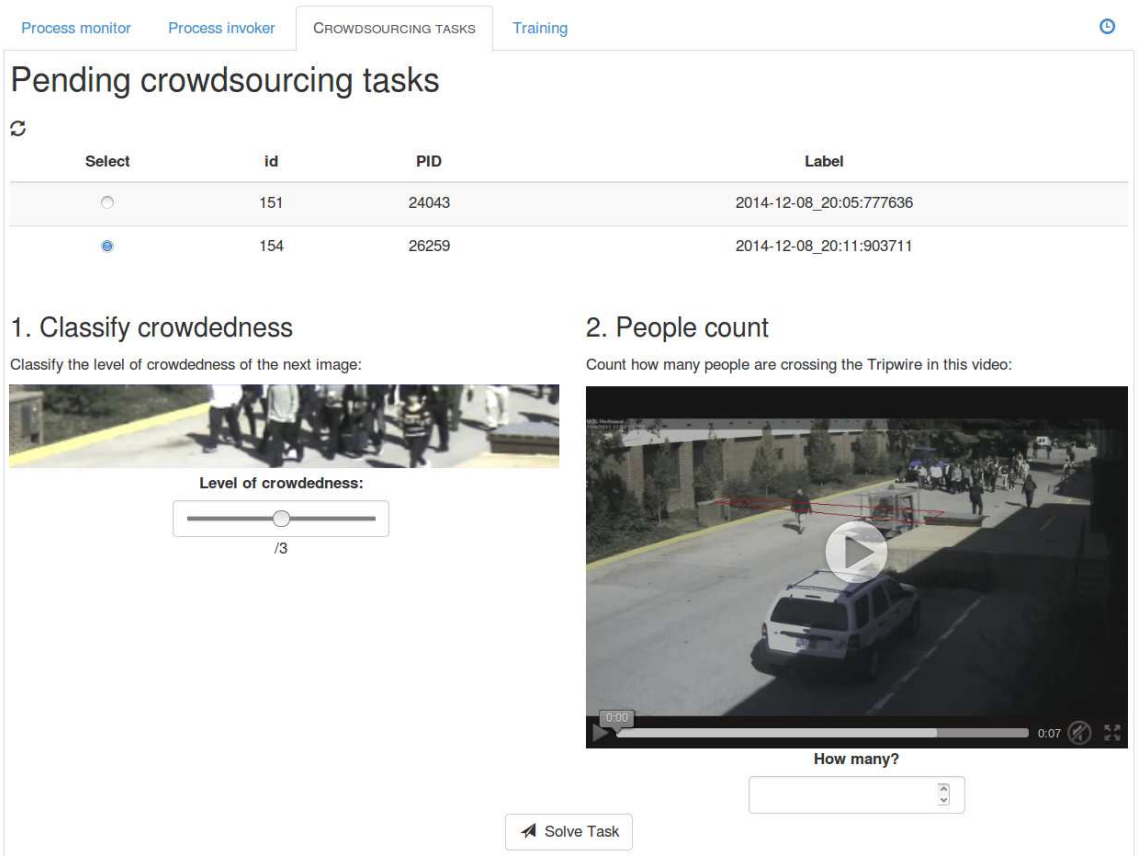


Fig. 5.16.: Web platform: processes invoker. Web platform: crowdsourcing tasks. The o-crowd can select and solve a pending crowdsourcing task.

5.3.3 Training

In this tab, an expert can create new training data. At the top of Figure 5.17, the operator can also check all the available training data and their parameters.

The process to create new training data consists of the following steps. First, as shown at the bottom of Figure 5.17, the operator must type the video path. Second, as shown in Figure 5.18, the operator must give a unique name for the future training data. Third, as shown in Figure 5.19, the Tripwire must be provided. The Graphical User Interface (GUI) allows to draw a Tripwire on the surveillance video by clicking and dragging its vertices. Forth, as shown in Figure 5.20, the ROI must be provided. The GUI allows to draw a ROI on the surveillance video by clicking and dragging

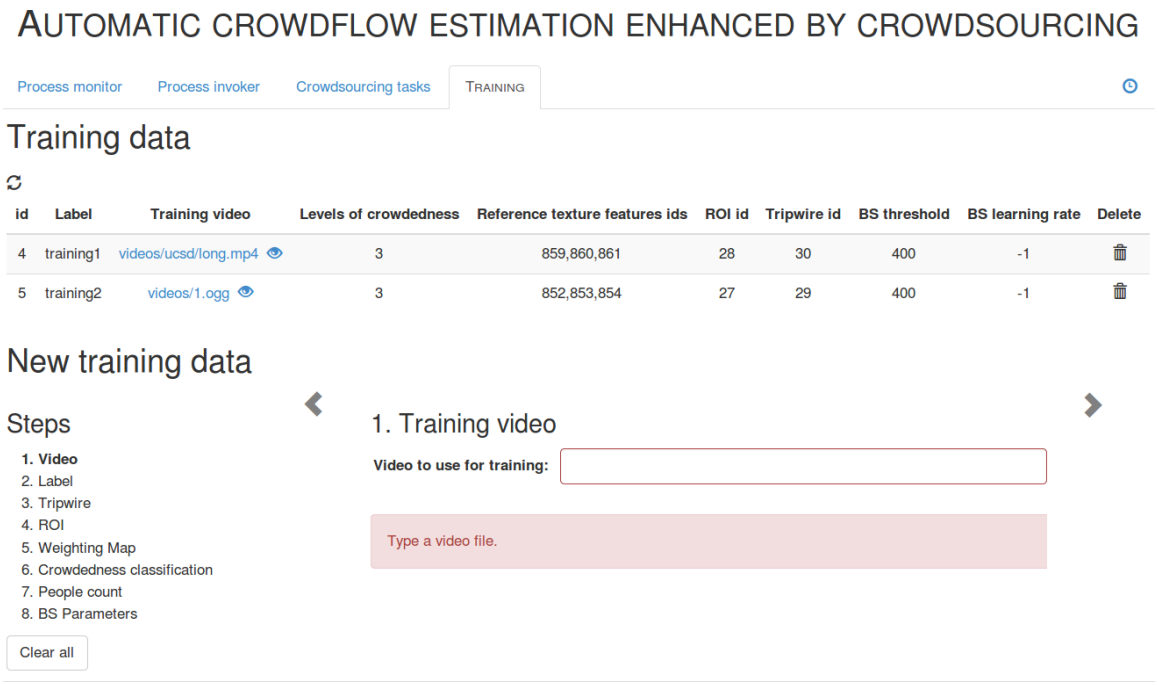


Fig. 5.17.: Web platform: training. The top section shows the available training data and the lower section allows to create new training data.

its vertices. The fifth step corresponds to the weighting scheme explained in Section 5.1.3. The GUI for this step was disabled because no significant improvements were noticed. Sixth, as shown in Figure 5.21, the operator is asked to classify the level of crowdedness of some random snapshots focused on the ROI. The number of levels of crowdedness, as well as the number of random snapshots to be asked for classification, is configurable. Seventh, as shown in Figure 5.22, the backend extracts and returns as many video segments as levels of crowdedness. These video segments contain a constant level of crowdedness in all their frames. The Tripwire is drawn on all the videos. The user is asked to provide the people count of each of the videos. Lastly, Figure 5.23 shows the final step, where the values of the parameters of the Background Subtraction can be specified. The button labeled “Assemble new training data” will send all the information collected from the user to the backend. After this, new

training data will be computed in the server, and a new entry will appear in the table at the top of the screen shown in Figure 5.17.

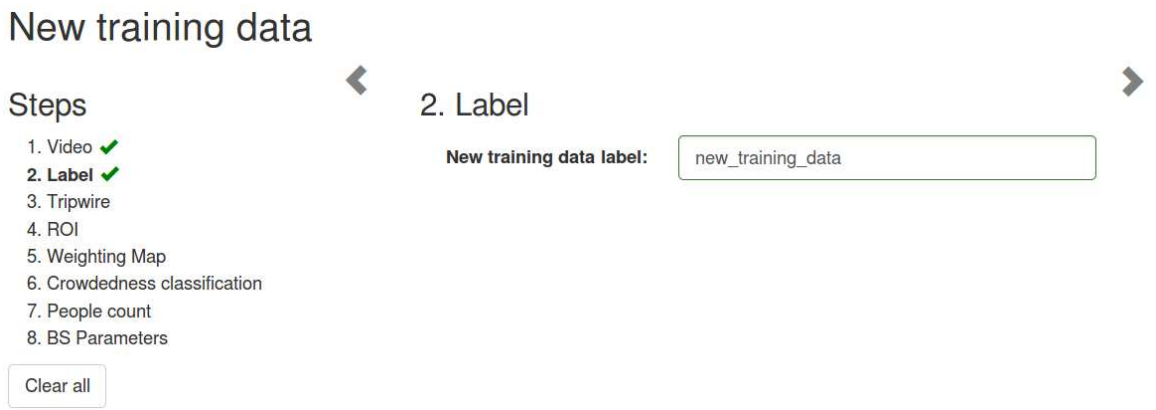


Fig. 5.18.: Web platform: new training data, step 2. The label of the future training data can be chosen.

However, before sending the definitive training data to the server, it is convenient to check the effect of the BS parameters on the foreground segmentation. The button "Preview" of Figure 5.23 shows a preview of the video as can be seen in Figure 5.24. The foreground segmentation of the training video is computed in the server with the provided parameters and the resulting video is sent back to the client browser. This way, the user can fine-tune the BS parameters to achieve the desired foreground segmentation.

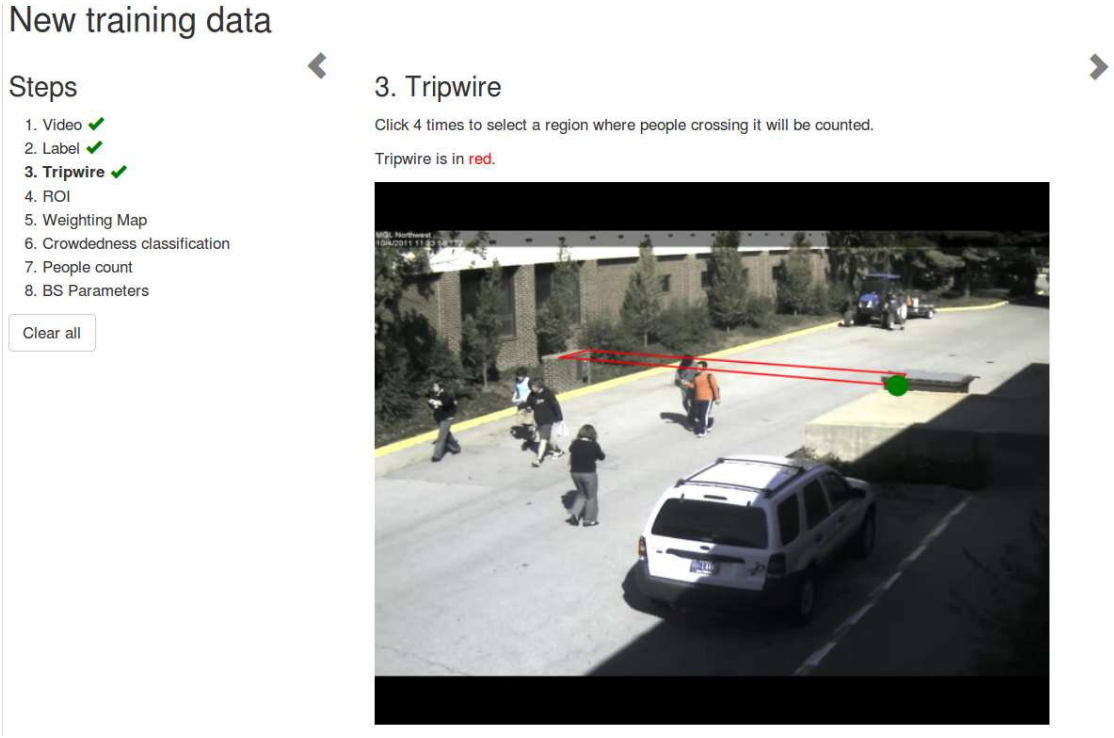


Fig. 5.19.: Web platform: new training data, step 3. The Tripwire must be drawn over the surveillance video.

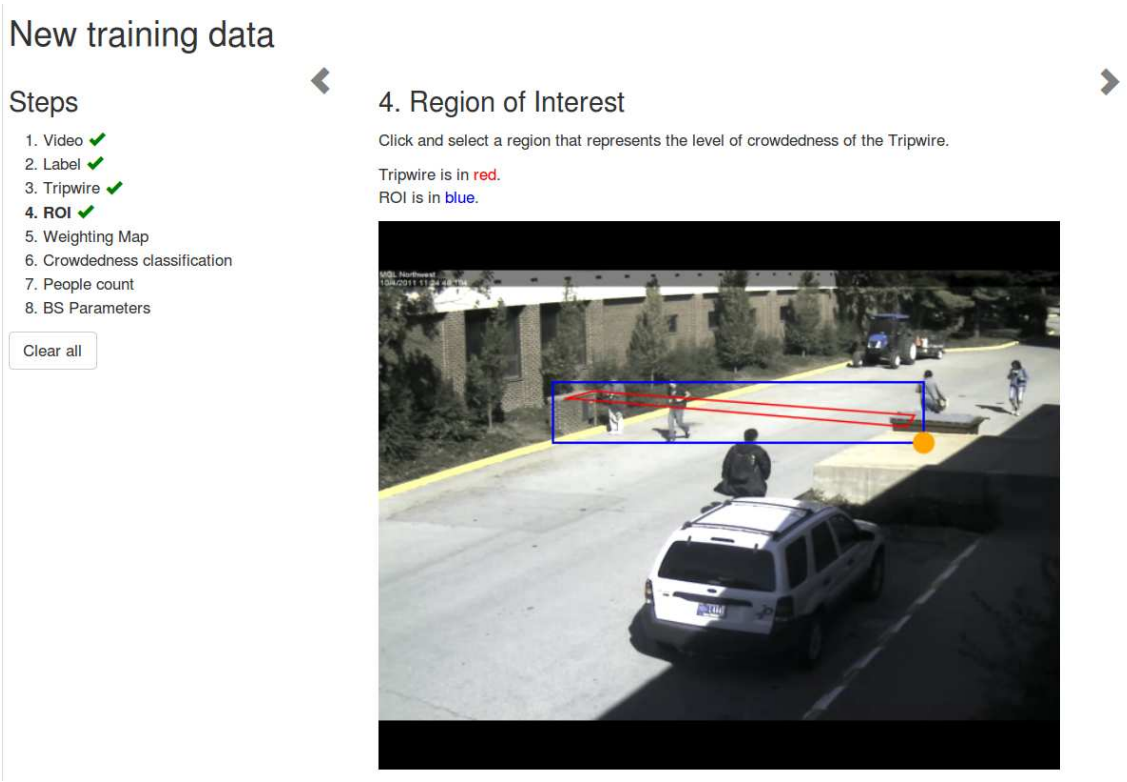


Fig. 5.20.: Web platform: new training data, step 4. The ROI must be drawn over the surveillance video.



Fig. 5.21.: Web platform: new training data, step 6. The random snapshots must be classified into one of the levels of crowdedness.

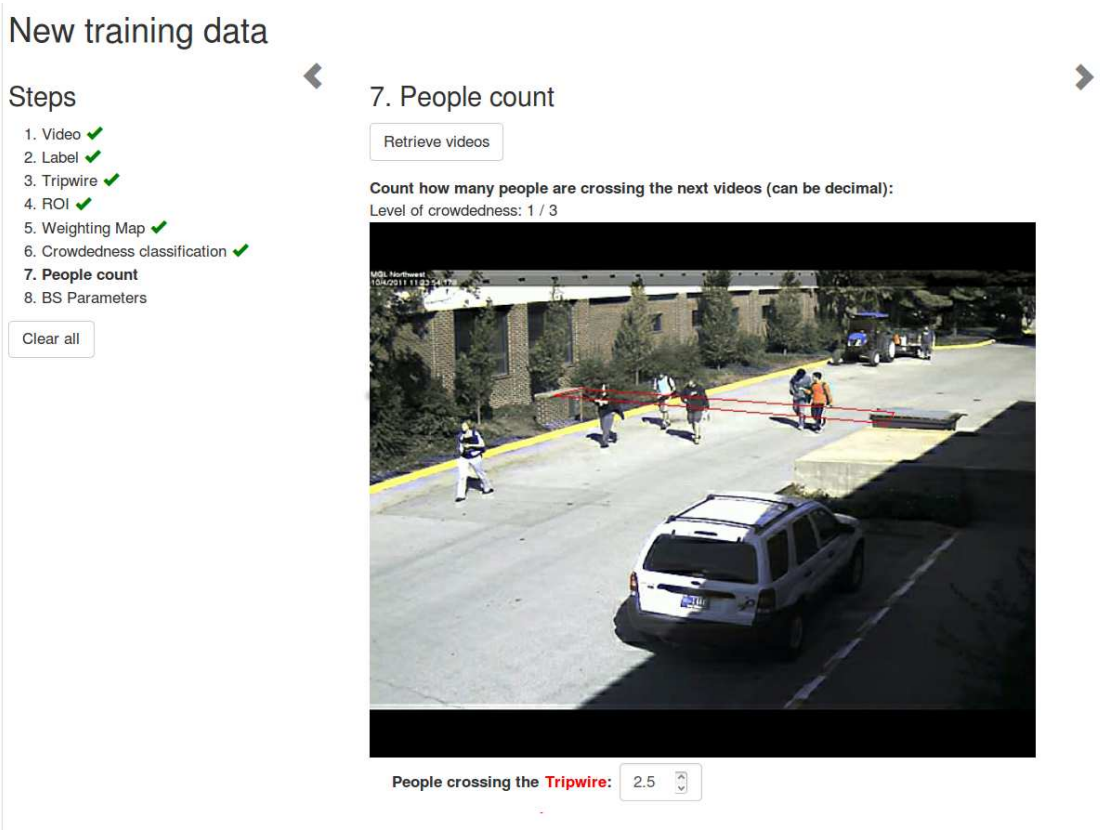


Fig. 5.22.: Web platform: new training data, step 7. The people count for the video segment corresponding to each level of crowdedness must be provided.

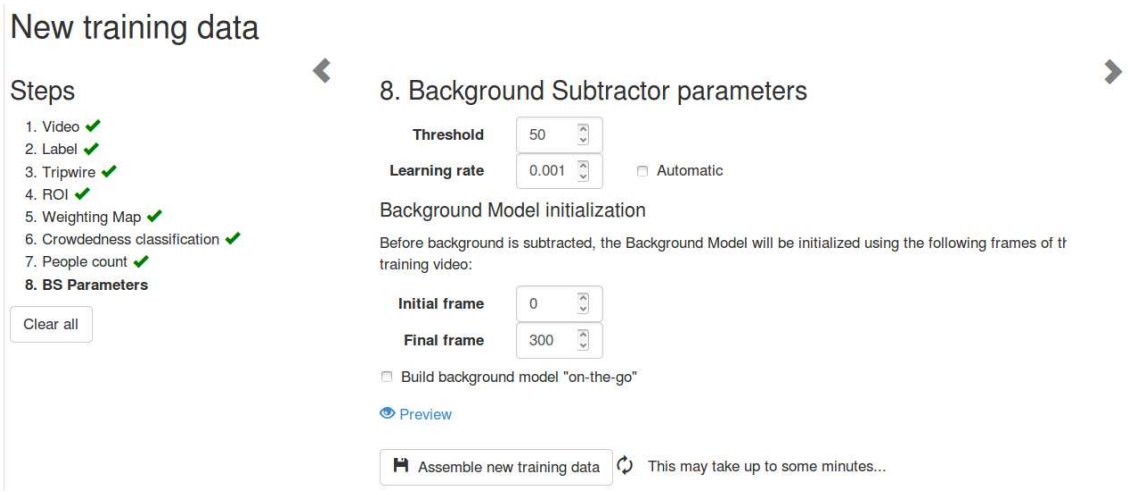


Fig. 5.23.: Web platform: new training data, step 8. The parameters of the Background Subtraction can be configured in this screen.

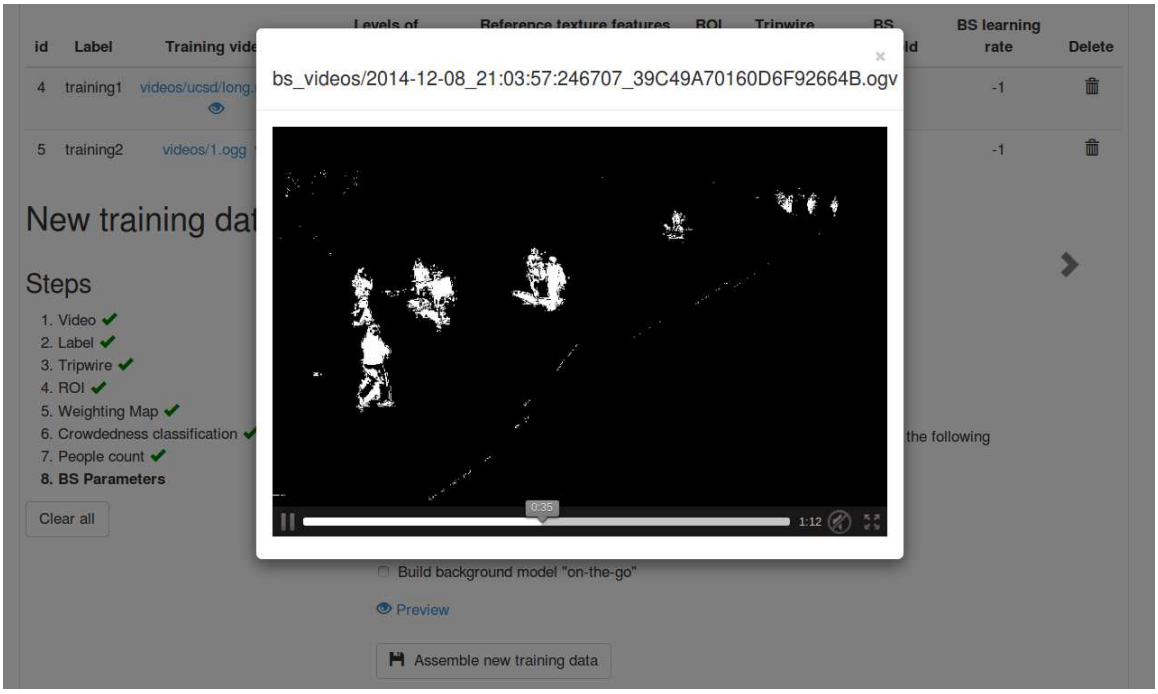


Fig. 5.24.: Web platform: new training data, BS preview. The effect of the BS parameters on the foreground segmentation can be tested using this preview feature.

5.4 Experimental Results

For our experiments, we employed a publicly available surveillance video dataset, the UCSD pedestrian dataset [171]. It consists of a 54 minutes long video taken from a stationary camera at the resolution of 238×158. This dataset is suitable for testing purposes since it is long enough (54 minutes) and contains different crowd density levels.



Fig. 5.25.: A frame of the University of California, San Diego pedestrian dataset

We performed two sets of experiments. In the first experiment, we evaluated the incorporation of crowdsourcing into the crowd flow estimation method. In this case, the o-crowd consisted on two expert members that jointly solved the crowdsourcing tasks. In the first experiment, the uncertainty characterization used is always the characterization #1 (Equation (5.15)). In the second experiment, we compared the three proposed characterization of the uncertainty, and the aggregation criteria.

In the first experiment, the video is segmented in the following way: the first 30 minutes are used solely for training, and the remaining minutes are used to create 5 consecutive clips of 4 minutes length. These 5 video segments are used for testing purposes. In the second experiment, we only used the first 8 minutes of video. We segmented this 8 minutes into 6 clips, the first clip is used solely for training and it

is 3 minutes long. The remaining 5 minutes are segmented into 5 clips of 1 minute duration each. The clip segmentation for the second experiment is depicted in Figure 5.26.

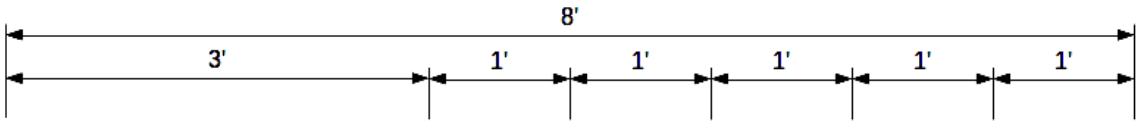


Fig. 5.26.: Video segments used in the second experiment. The first 3 minutes of the UCSD dataset are used solely for training. The testing is performed only using the following 5 minutes, segmented into 1 minute long clips.

The degradation of the testing segments consisted of H.264 compression using ffmpeg [172] and a boost in the contrast of the video frames. In the first experiment, the Constant Rate Factor (CRF) is set to 33 and the contrast increased to 1.68 using the eq2 filter. In the second experiment, the CRF is set to 35 and the contrast increased to 2.68. As a result, the size of the degraded video files is reduced to a approximately 40% in both cases.

We measured the crowd flow error rate and the utilization of the o-crowd in every segment. The error rate is defined as

$$100\frac{|\upsilon_e - \upsilon_{GT}|}{\upsilon_{GT}} \tag{5.23}$$

where υ_{GT} is the ground truth, and υ_e is the estimated crowd flow. The utilization of the o-crowd is how many labeling tasks have been requested to the o-crowd, which corresponds to the number of texture feature vectors that have appeared outside the certainty areas.

We evaluate the effect of the quality degradations on the error rate when using the automatic method with no crowdsourcing ($\alpha = 1$). The plot in figure 5.27 shows that the degradation of the video quality increases the error rate for almost all the video segments, for the setup of the first experiment. Although the video segment

number #2 is not affected much, in average, the error rate increases from 9.2% to 34.4% when distortions are introduced.

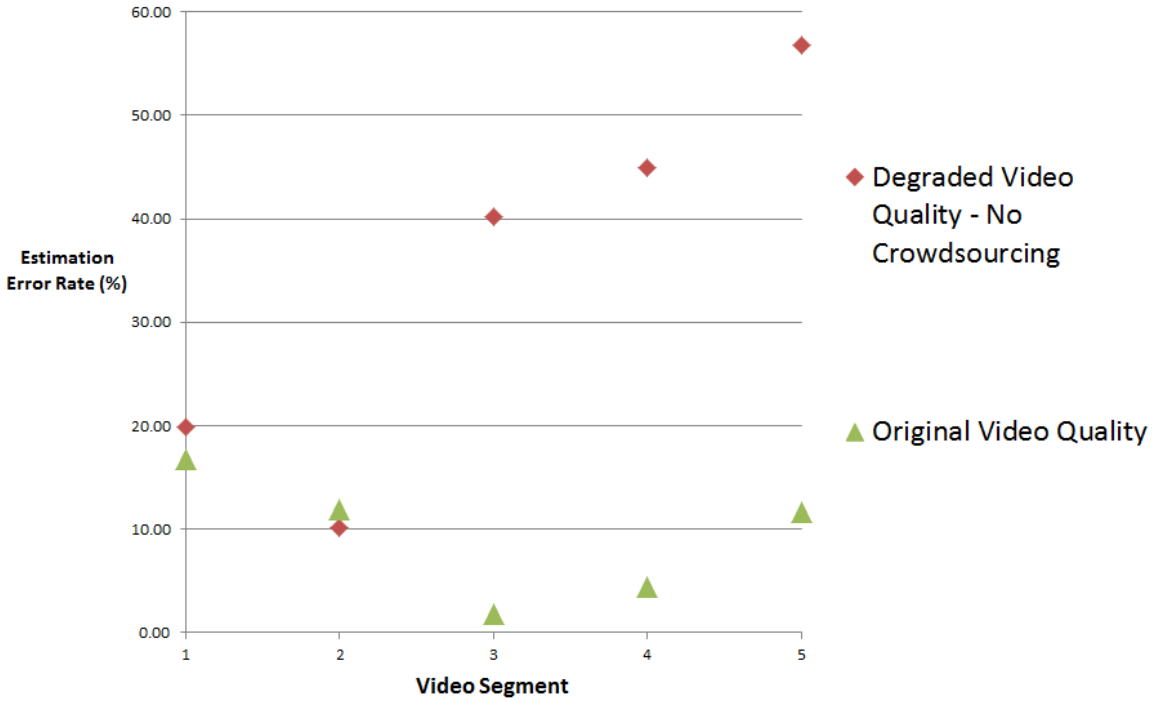


Fig. 5.27.: In the first experiment, the average error rate increases from 9.2% to 34.4% when distortions are introduced and no crowdsourcing is employed.

In the setup of the second experiment, the results are very similar. Table 5.1 shows that the average error rate increases by a factor of 4 when the video quality is degraded.

Now, we incorporate crowdsourcing to the automatic method to adapt it to video degradations. We use two different values for the crowdsourcing parameter, $\alpha = 0.5$ and 0.6. Table 5.3 shows the reduction of the average error rate when crowdsourcing is incorporated in the first experiment. The average error rate decreases appreciably when crowdsourcing is incorporated.

Furthermore, when using crowdsourcing, we would like the method to learn from the o-crowd input. Then, one would expect the utilization of the o-crowd to decrease as we progress in time. In other words, the number of crowdsourcing tasks should

Table 5.1.: Average error rates before and after applying video degradations. The error rate increases from 8 to 32% when distortions are introduced and no crowdsourcing is employed. The setup is the second experiment.

	α	Average Error Rate
Original quality	1	8.1%
Degraded quality	1	31.7%

Table 5.3.: Results of the crowdsourcing incorporation in the first experiment. The average error rate appreciably decreases when crowdsourcing is incorporated, and a lower α results in lower average error rate.

α	Average Error Rate
1 (no crowdsourcing)	34.4%
0.6	18.0%
0.5	14.3%

decrease throughout the video segments. Figure 5.28 displays the number of emitted crowdsourcing tasks per video segment in the first experiment. Note the decrease in the utilization of the o-crowd as the video progresses in time. We can also observe this decrease in the results of the second experiment shown in Figure 5.29.

In Table 5.5, we compare the resulting error rate when two different uncertainty characterizations are used. The same value of α may trigger the crowdsourcing a different number of times, depending on the type of uncertainty characterization. To make a fair comparison, we chose different values for α for each approach such that

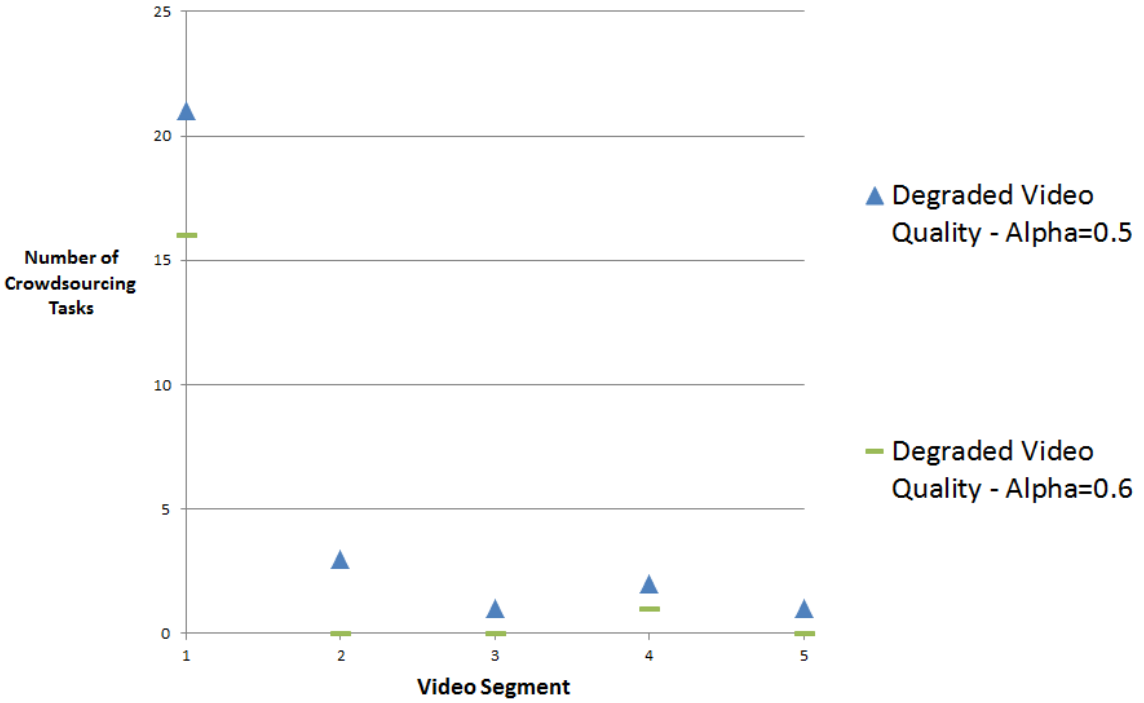


Fig. 5.28.: Evolution of the o-crowd utilization in the first experiment. Note that as the video progresses in time, the o-crowd is reached less often.

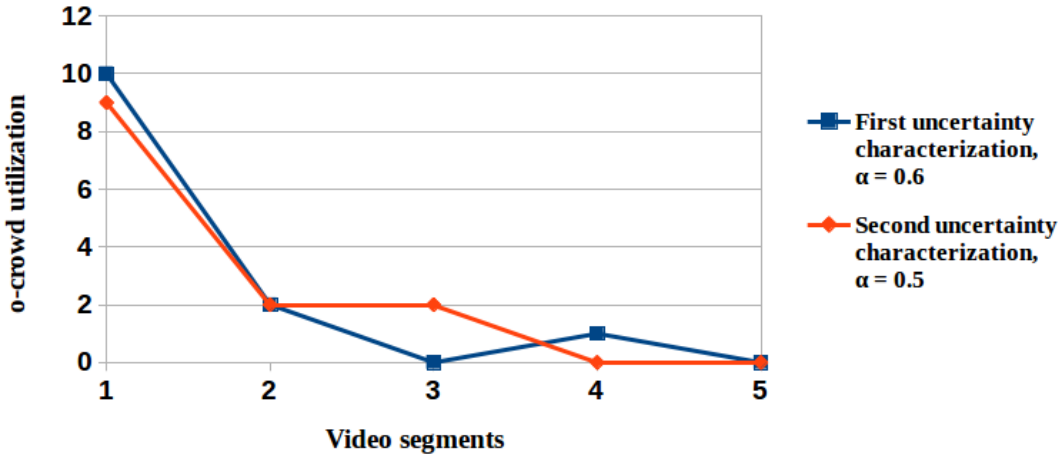


Fig. 5.29.: Evolution of the o-crowd utilization in the second experiment. Note that as the video progresses in time, the o-crowd is reached less often.

the resulting total utilization is similar. The results in Table 5.5 are computed by using a single expert o-crowd member. The second type of characterization achieves a lower error rate. The results of the third characterization are not included in Table 5.5. We experimentally observed that, even after the degradation, all the data points never fell close to the border, which implies a low value for μ_3. For example, in the first segment, the value of μ_3 is 0.2, which implies that the distance to the border d_b is 4 times the distance between the 2 closest reference vectors, $d_{1,2}$. As a result, we conclude that the estimation in the degraded video of this dataset cannot be enhanced with the third type of uncertainty characterization.

Table 5.5.: The error rate when using the second uncertainty characterization is lower than using the first characterization, given the same amount of labeling necessary. The experiment set up is the one of the second experiment.

	Uncertainty Characterization	α	**Total Utilization**	**Average Error Rate**
Degraded quality	μ_1	0.6	13	15.2%
Degraded quality	μ_2	0.5	13	12.5%

Lastly, we compare the two proposed approaches to aggregate crowdsourcing results. We use 5 o-crowd members, the first uncertainty characterization ($\mu_1 = 0.6$), and the setup of the second experiment. The results are shown in Table 5.7. The first row shows the results when all the answers of the 5 o-crowd members are averaged. The second row shows the results when the average of the subset ($K = 3$) with the smallest variance is calculated. We observe that only using the 3 labels with lowest variance yields to a decrease in the average error rate of 6.5%.

Table 5.7.: By aggregating only the $K = 3$ o-crowd members with the most similar labels, we reduce the error rate.

Aggregation of K o-crowd Members	**Total Utilization**	**Average Error Rate**
$K = 5$	13	26%
$K = 3$	13	19.5%

The UCSD dataset does not include ground truth information. Therefore, we used our best judgment to provide the ground truth data. However, the numbers might be "0.5" person form the true values due to the fact that some people are crossing the tripwire at the beginning or the end of each clip.

The parameter values are experimentally set. The threshold used in the foreground segmentation is empirically set to 400 in the first experiment and to 50 in the second experiment. This threshold represents the distance from a pixel value to the background model to decide whether it belongs to the foreground or the background. The learning rate represents how fast the background model is updated. In the first experiment, it is set to 0.01. In the second experiment, the learning rate is left to be decided and updated automatically by the moving object detection presented in [167, 168] and implemented in OpenCV 2.4.9 [169]. The Gray Level Co-occurrence Matrix (GLCM) matrix is computed using the scikit-image library [99] and using 8 gray levels. The number of random snapshots of the ROI used to compute the reference texture vectors during the training stage is $M = 50$. The selected number of crowdedness levels is $L = 3$.

6. SUMMARY AND FUTURE WORK

6.1 Summary

In this thesis, we developed new methods for plant phenotyping from UAV and ground-based platforms, object localization, and reduction of uncertainty in classification methods using crowdsourcing. The main contributions of this work are:

- Plot Extraction

  We address the problem of extracting sections of an image that belong to different field plots. This is known as "plot extraction" and enables further phenotypic analyses. For example, one can use the extracted plots to estimate the canopy coverage or leaf count of each plot separately. We describe two methods for plot extraction. One method extracts plots from an orthophoto of UAV images, by minimizing an energy function that finds straight prarallel lines between the rows of plants. The other method extracts row segments from the "PhenoRover", acquiring images from the top of the crop canopy. This method uses the Radon Transform to find the most dominant almost-parallel lines in the non-rectified image.

- Leaf count and canopy coverage estimation

  We describe a technique to segment plant material, and estimate canopy coverage and leaf count at the plot level. We use this technique to estimate the leaf count of a crop field without individually segmenting each leaf. This method assumes that leaves have approximately the same area. We evaluate this technique with perspective, distortion-free, and orthorectified images of sorghum plants, achieving an average accuracy of 87.7 % using ground truth provided by manual leaf count from the images.

- Plant Counting and Location

  We investigate the problem of counting and locating plants from UAV imagery. Counting and locating are usually considered two sides of the same coin. We propose methods to count and locate plants as separate tasks, and a method to simultaneously locate and count generic objects. A statistical model allows to estimate of the location of each plant, by making use of prior information that accounts for the alignment of the plants in the field. We count the number of plants in a plot by using a regression loss and a CNN. We also design a novel loss function, which we call *Weighted Hausdorff Distance*, and employ it to locate sorghum plants and estimate intra-row plant spacing.

- Crowdsourcing

  We incorporate crowdsourcing to improve the accuracy of a crowd flow estimation method. Diverse characterizations of the uncertainty of a classifier are proposed and evaluated, as well as different criteria to aggregate the labels provided by heterogeneous crowdsourcing labelers. The method uses the crowdsourced label to reduce the error rate, and retrains a classifier to reduce how often to ask the crowdsourcing crowd. Our experimental evaluation using a publicly available dataset suggests that crowdsourcing reduces the error rate, and that the utilization of the o-crowd is reduced with time.

- Web phenotyping system, and online crowdsourcing tool

  We develop an online platform that allows plant scientists and agronomists to make use of our phenotyping tools and estimate plant traits from their own data. We also develop an online system to reach the crowdsourcing labelers. Crowdsourcing members can annotate unlabeled data for which the automatic method has low confidence.

6.2 Future Work

We have demonstrated the efficacy of the proposed methods. Nonetheless, our work can be extended and improved in the following ways:

- Plot extraction

  We described two methods to extract plots. One uses UAV imagery and the other uses images acquired from a ground-based platform. The definition of a plot boundary as a rectangle surrounding the row segment is a limitation. To mitigate the dependency on well aligned orthophotos, a more flexible definition, where boundaries are not straight, may be needed. This would allow for row segments to "wiggle," showing as curved lines in the image, due for example to a low-end GPS/IMU. Also, if the orthophoto does not contain an entire panel and some row segments are not shown, our assumption of constant M and N will be incorrect. We could do a hyper-parameter search over M and N, around the approximate values that the known field structure provides.

- Supervised methods for plant material segmentation

  We proposed a simple but effective technique to segment plant material, that we then use to estimate leaf count and canopy coverage. According to visual inspection, this technique is very precise, but it requires thresholds to be finely tuned, and they may depend on the lighting conditions. One could manually label plant material from the imagery, and use it as training data to automatically estimate these thresholds. Also other features besides color, such as texture, might be helpful in the segmentation. As this is usually the first step for other methods, it should be kept fast, and easily tunable by the user.

- Plant Localization

  We have proposed methods to locate and count plants in the field, and any type of object in images. Neural network architectures are constantly being improved. Our CNN-based counting method, and the WHD method could

potentially be improved simply by replacing the architectures with more modern models. One could evaluate a variety of architectures on our datasets, or use their performance on ImageNet as an initial guidance to select a better model. The output layer of the network could be composed of more than one feature map, and train the network using the average of the losses for each feature map. This would allow one to locate and count objects of multiple type, without the need to retrain the network.

- Crowdsourcing

  We have proposed multiple characterizations of the uncertainty of classification methods, and how to aggregate crowdsourcing labels to reduce this uncertainty. As the suggested crowdsourcing method is generic and it applies to any feature vector, it could be evaluated using other supervised classifiers. The automatic crowdflow estimation is only one of the possible applications. Although hundreds of human labelers would be required, a more in-depth study of the trade-off between accuracy and frequency to reach the o-crowd would help the operator when setting the parameter α.

REFERENCES

REFERENCES

[1] S. K. Panguluri and A. A. Kumar, *Phenotyping for Plant Breeding*. New York, NY: Springer New York, 2013. https://doi.org/10.1007/978-1-4614-8320-5

[2] J. W. White, P. Andrade-Sanchez, M. A. Gore, K. F. Bronson, T. A. Coffelt, M. M. Conley, K. A. Feldmann, A. N. French, J. T. Heun, D. J. Hunsaker, M. A. Jenks, B. A. Kimball, R. L. Roth, R. J. Strand, K. R. Thorp, G. W. Wall, and G. Wang, "Field-based phenomics for plant genetics research," *Field Crops Research*, vol. 133, pp. 101 – 112, July 2012. https://doi.org/10.1016/j.fcr.2012.04.003

[3] J. L. Araus and J. E. Cairns, "Field high-throughput phenotyping: the new crop breeding frontier," *Trends in Plant Science*, vol. 19, no. 1, pp. 52 – 61, 2014. https://doi.org/10.1016/j.tplants.2013.09.008

[4] A. A. Gitelson, A. Viña, T. J. Arkebauer, D. C. Rundquist, G. Keydan, and B. Leavitt, "Remote estimation of leaf area index and green leaf biomass in maize canopies," *Geophysical Research Letters*, vol. 30, no. 5, p. 1248, March 2003. https://doi.org/10.1029/2002GL016450

[5] M. R. Golzarian, R. A. Frick, K. Rajendran, B. Berger, S. Roy, M. Tester, and D. S. Lun, "Accurate inference of shoot biomass from high-throughput images of cereal plants," *Plant Methods*, vol. 7, no. 1, pp. 1–11, February 2011. https://doi.org/10.1186/1746-4811-7-2

[6] E. H. Neilson, A. M. Edwards, C. K. Blomstedt, B. Berger, B. L. Møller, and R. M. Gleadow, "Utilization of a high-throughput shoot imaging system to examine the dynamic phenotypic responses of a C_4 cereal crop plant to nitrogen and water deficiency over time," *Journal of Experimental Botany*, vol. 66, no. 7, pp. 1817–1832, February 2015. https://doi.org/10.1093/jxb/eru526

[7] J. H. M. Thornley, "Crop yield and planting density," *Annals of Botany*, vol. 52, no. 2, pp. 257–259, August 1983. https://doi.org/10.1093/oxfordjournals.aob.a086571

[8] I. Tokatlidis and S. D. Koutroubas, "A review of maize hybrids' dependence on high plant populations and its implications for crop yield stability," *Field Crops Research*, vol. 88, no. 2, pp. 103–114, August 2004. https://doi.org/10.1016/j.fcr.2003.11.013

[9] D. E. Farnham, "Row spacing, plant density, and hybrid effects on corn grain yield and moisture," *Agronomy Journal*, vol. 93, pp. 1049–1053, September 2001. https://doi.org/10.2134/agronj2001.9351049x

[10] B. S. Chauhan and D. E. Johnson, "Row spacing and weed control timing affect yield of aerobic rice," *Field Crops Research*, vol. 121, no. 2, pp. 226–231, March 2001. https://doi.org/10.1016/j.fcr.2010.12.008

[11] I. A. Ciampitti, S. T. Murrell, J. J. Camberato, M. Tuinstra, Y. Xia, P. Friedemann, and T. J. Vyn, "Physiological dynamics of maize nitrogen uptake and partitioning in response to plant density and nitrogen stress factors: Ii. reproductive phase," *Crop Science*, pp. 2588–2602, September 2013. https://doi.org/10.2135/cropsci2013.01.0041

[12] A. Masjedi, J. Zhao, A. M. Thompson, K. Yang, J. E. Flatt, M. M. Crawford, D. S. Ebert, M. R. Tuinstra, G. Hammer, and S. Chapman, "Sorghum biomass prediction using uav-based remote sensing data and crop model simulation," pp. 7719–7722, July 2018. https://doi.org/10.1109/IGARSS.2018.8519034

[13] D. Lazár, "Chlorophyll a fluorescence induction," *Biochimica et Biophysica Acta (BBA) - Bioenergetics*, vol. 1412, no. 1, pp. 1 – 28, March 1999. https://doi.org/10.1016/S0005-2728(99)00047-X

[14] D. Lazár, "Parameters of photosynthetic energy partitioning," *Journal of Plant Physiology*, vol. 175, pp. 131 – 147, March 2015. https://doi.org/10.1016/j.jplph.2014.10.021

[15] A. Miyao, Y. Yukimoto, H. Kitano, J. Itoh, M. Maekawa, K. Murata, O. Yatou, Y. Nagato, and H. Hirochika, "A large-scale collection of phenotypic data describing an insertional mutant population to facilitate functional analysis of rice genes," *Plant Molecular Biology*, vol. 63, no. 5, pp. 625–635, March 2007. https://doi.org/10.1007/s11103-006-9118-7

[16] H. R. Bolhar-Nordenkampf, S. P. Long, N. R. Baker, G. Oquist, U. Schreiber, and E. G. Lechner, "Chlorophyll fluorescence as a probe of the photosynthetic competence of leaves in the field: A review of current instrumentation," *Functional Ecology*, vol. 3, no. 4, pp. 497–514, 1989. https://doi.org/10.2307/2389624

[17] Y. Goulas, Z. G. Cerovic, A. Cartelat, and I. Moya, "Dualex: a new instrument for field measurements of epidermal ultraviolet absorbance by chlorophyll fluorescence," *Applied Optics*, vol. 43, no. 23, pp. 4488–4496, August 2004. https://doi.org/10.1364/AO.43.004488

[18] N. Fahlgren, M. A. Gehan, and I. Baxter, "Lights, camera, action: high-throughput plant phenotyping is ready for a close-up," *Current Opinion in Plant Biology*, vol. 24, pp. 93–99, April 2015. https://doi.org/10.1016/j.pbi.2015.02.006

[19] N. M. Boopathi, *Genetic Mapping and Marker Assisted Selection: Basics, Practice and Benefits*. Springer India, 2013. https://doi.org/10.1007/978-81-322-0958-4

[20] D. C. Koboldt, K. Steinberg, D. E. Larson, R. K. Wilson, and E. R. Mardis, "The next-generation sequencing revolution and its impact on genomics," *Cell*, vol. 155, pp. 27–38, 2013. https://doi.org/10.1016/j.cell.2013.09.006

[21] K. Wetterstrand, "DNA sequencing costs: Data from the NHGRI genome sequencing program (GSP)," 2016. https://www.genome.gov/sequencingcostsdata

[22] L. Li, Q. Zhang, and D. Huang, "A review of imaging techniques for plant phenotyping," *Sensors*, vol. 14, no. 11, pp. 20 078–20 111, October 2014. https://doi.org/10.3390/s141120078

[23] S. Sankaran, L. R. Khot, C. Z. Espinoza, S. Jarolmasjed, V. R. Sathuvalli, G. J. Vandemark, P. N. Miklas, A. H. Carter, M. O. Pumphrey, N. R. Knowles, and M. J. Pavek, "Low-altitude, high-resolution aerial imaging systems for row and field crop phenotyping: A review," *European Journal of Agronomy*, vol. 70, pp. 112 – 123, October 2015. https://doi.org/10.1016/j.eja.2015.07.004

[24] A. Singh, B. Ganapathysubramanian, A. K. Singh, and S. Sarkar, "Machine learning for high-throughput stress phenotyping in plants," *Trends in Plant Science*, vol. 21, no. 2, pp. 110–124, 2016. https://doi.org/10.1016/j.tplants.2015.10.015

[25] Z. Zhang, A. Masjedi, J. Zhao, and M. M. Crawford, "Prediction of sorghum biomass based on image based features derived from time series of uav images," *Proceedings of the IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 6154–6157, July 2017. https://doi.org/10.1109/IGARSS.2017.8128413

[26] S. A. Tsaftaris and C. Noutsos, "Plant phenotyping with low cost digital cameras and image analytics," *Information Technologies in Environmental Engineering*, pp. 238–251, April 2009. https://doi.org/10.1007/978-3-540-88351-7_18

[27] X. Wu, "Color quantization by dynamic programming and principal analysis," *ACM Transactions on Graphics*, vol. 11, no. 4, pp. 348–372, October 1992. https://doi.org/10.1145/146443.146475

[28] N. Fahlgren, M. Feldman, M. A. Gehan, C. S. M. S. Wilson, D. W. Bryant, S. T. Hill, C. J. McEntee, S. N. Warnasooriya, I. Kumar, T. Ficor, S. Turnipseed, K. B. Gilbert, T. P. Brutnell, J. C. Carrington, T. C. Mockler, and I. Baxter, "A versatile phenotyping system and analytics platform reveals diverse temporal responses to water availability in setaria," *Molecular Plant*, June 2015. https://doi.org/10.1016/j.molp.2015.06.005

[29] J.-M. Pape and C. Klukas, "3-D histogram-based segmentation and leaf detection for rosette plants," *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pp. 61–74, March 2015, Zurich, Switzerland. https://doi.org/10.1007/978-3-319-16220-1_5

[30] M. Minervini, M. M. Abdelsamea, and S. A. Tsaftaris, "Image-based plant phenotyping with incremental learning and active contours," *Ecological Informatics*, vol. 23, pp. 35–48, September 2014. https://doi.org/10.1016/j.ecoinf.2013.07.004

[31] G. Lobet, X. Draye, and C. Périlleux, "An online database for plant image analysis software tools," *Plant Methods*, vol. 9, no. 38, pp. 1–8, October 2013. https://doi.org/10.1186/1746-4811-9-38

[32] "LemnaTech," URL: https://lemnatec.com/.

[33] "Deepfield Robotics," URL: https://deepfield-robotics.com/.

[34] C. Granier, L. Aguirrezabal, K. Chenu, S. J. Cookson, M. Dauzat, P. Hamard, J. Thioux, G. Rolland, S. Bouchier-Combaud, A. Lebaudy, B. Muller, T. Simonneau, and F. Tardieu, "Phenopsis, an automated platform for reproducible phenotyping of plant responses to soil water deficit in arabidopsis thaliana permitted the identification of an accession with low sensitivity to soil water deficit," *New Phytologist*, vol. 169, no. 3, pp. 623–635, February 2006. https://doi.org/10.1111/j.1469-8137.2005.01609.x

[35] D. Kelly, A. Vatsa, W. Mayham, L. Ngô, A. Thompson, and T. KazicDerek, "An opinion on imaging challenges in phenotyping field crops," *Machine Vision and Applications*, pp. 1–14, December 2015. https://doi.org/10.1007/s00138-015-0728-4

[36] J. D. Plessis, *Sorghum production.* Department of Agriculture of Republic of South Africa, 2003. https://www.nda.agric.za/docs/Infopaks/FieldCrops_Sorghum.pdf

[37] R. K. Bacon, R. P. Cantrell, and J. D. Atxell, "Selection for seedling cold tolerance in grain sorghum," *Crop Science*, vol. 26, no. 5, pp. 900–903, September 1986. https://doi.org/10.2135/cropsci1986.0011183X002600050013x

[38] J. Yu, M. R. Tuinstra, M. M. Claaseen, W. B. Gordon, and M. D. Witt, "Analysis of cold tolerance in sorghum under controlled environment conditions," *Field Crops Research*, vol. 85, no. 1, pp. 21–30, January 2004. https://doi.org/10.1016/S0378-4290(03)00125-4

[39] US Grains Council, "Sorghum handbook: White sorghum, the new food grain," 2005. https://www.agmrc.org/media/cms/Sorghum_Handbook_B5FE1C2B5DBCF.pdf

[40] J. Taylor, "Overview: Importance of sorghum in africa," *Afripro; Workshop on the Proteins of Sorghum and Millets: Enhancing Nutritional and Functional Properties for Africa*, vol. 2, no. 4, April 2003, Pretoria, South Africa.

[41] R. L. Monk, F. R. Miller, and G. G. McBee, "2nd southern biomass energy research conference sorghum improvement for energy production," *Biomass*, vol. 6, pp. 145 – 153, 1984. https://doi.org/10.1016/0144-4565(84)90017-9

[42] A. Almodares and M. R. Hadi, "Production of bioethanol from sweet sorghum: A review," *African Journal of Agricultural Research*, vol. 4, no. 9, pp. 772–780, 2009.

[43] W. L. Rooney, J. Blumenthal, B. Bean, and J. E. Mullet, "Designing sorghum as a dedicated bioenergy feedstock," *Biofuels, Bioproducts and Biorefining*, vol. 1, pp. 147–157, October 2007. https://doi.org/10.1002/bbb.15

[44] D. Wang, S. Bean, J. McLaren, P. Seib, R. Madl, M. Tuinstra, Y. Shi, M. Lenz, X. Wu, and R. Zhao, "Grain sorghum is a viable feedstock for ethanol production," *Journal of Industrial Microbiology & Biotechnology*, vol. 35, no. 5, pp. 313–320, May 2008. https://doi.org/10.1007/s10295-008-0313-1

[45] G. S. Campbell and J. M. Norman, "The description and measurement of plant canopy structure," *Plant canopies: their growth, form and function*. Cambridge University Press Cambridge, 1989, vol. 31, pp. 1–19.

[46] M. Valera and S. Velastin, "Intelligent distributed surveillance systems: a review," *IEE Proceedings of Vision, Image and Signal Processing*, vol. 152, no. 2, pp. 192–204, April 2005. https://doi.org/10.1049/ip-vis:20041147

[47] C. Norris, M. McCahill, and D. Wood, "The growth of CCTV: a global perspective on the international diffusion of video surveillance in publicly accessible space," *Surveillance & Society*, vol. 2, no. 2/3, pp. 110–135, 2004.

[48] C. S. Regazzoni, A. Cavallaro, Y. Wu, J. Konrad, and A. Hampapur, "Video analytics for surveillance: Theory and practice [from the guest editors]," *IEEE Signal Processing Magazine*, vol. 27, no. 5, pp. 16–17, September 2010. https://doi.org/10.1109/MSP.2010.937451

[49] H. Dee and S. Velastin, "How close are we to solving the problem of automated visual surveillance?" *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 329–343, September 2008. https://doi.org/10.1007/s00138-007-0077-z

[50] B. Zhan, D. N. Monekosso, P. R. S. A. Velastin, and L. Xu, "Crowd analysis: a survey," *Machine Vision and Applications*, vol. 19, no. 5-6, pp. 345–357, October 2008. https://doi.org/10.1007/s00138-008-0132-4

[51] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 3, pp. 334–352, August 2004. https://doi.org/10.1109/TSMCC.2004.829274

[52] E. G. T. B. Moeslund, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, vol. 81, no. 3, pp. 231 – 268, March 2001. https://doi.org/10.1006/cviu.2000.0897

[53] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Computing Surveys*, vol. 38, no. 4, December 2006. https://doi.org/10.1145/1177352.1177355

[54] J. H. Yin, S. A. Velastin, and A. C. Davies, "Image processing techniques for crowd density estimation using a reference image," *Recent Developments in Computer Vision; Second Asian Conference on Computer Vision (ACCV)*, pp. 489–498, December 1995, Singapore. https://doi.org/10.1007/3-540-60793-5_102

[55] A. N. Marana, S. A. Velastin, L. F. Costa, and R. A. Lotufo, "Automatic estimation of crowd density using texture," *Safety Science*, vol. 28, no. 3, pp. 165–175, April 1998. https://doi.org/10.1016/S0925-7535(97)00081-7

[56] A. N. Marana and S. A. Velastin and L. F. Costa and R. A. Lotufo, "Estimation of crowd density using image processing," *Proceedings of the IEE Colloquium on Image Processing for Security Applications*, pp. 11/1–11/8, March 1997, London, United Kingdom. https://doi.org/10.1049/ic:19970387

[57] S. Srivastava, K. K. Ng, and E. J. Delp, "Crowd flow estimation using multiple visual features for scenes with changing crowd densities," *Proceedings of the 8th IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pp. 60–65, August 2011, Klagenfurt, Austria. https://doi.org/10.1109/AVSS.2011.6027295

[58] A. B. Chan, Z. J. Liang, and N. Vasconcelos, "Privacy preserving crowd monitoring: Counting people without people models or tracking," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–7, June 2008, Anchorage, AK. https://doi.org/10.1109/CVPR.2008.4587569

[59] D. Kong, D. Gray, and H. Tao, "Counting pedestrians in crowds using viewpoint invariant training." *Proceedings of the British Machine Vision Conference*, pp. 63.1–63.10, September 2005, Oxford, UK. https://doi.org/10.5244/C.19.63

[60] Z. Yu, C. Gong, J. Yang, and L. Bai, "Pedestrian counting based on spatial and temporal analysis," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pp. 2432–2436, October 2014, Paris, France. https://doi.org/10.1109/ICIP.2014.7025492

[61] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 14, no. 6, pp. 1–4, 2006, Dorsey Press.

[62] E. Estellés-Arolas and F. González-Ladrón-De-Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information Science*, vol. 38, no. 2, pp. 189–200, April 2012. https://doi.org/10.1177/0165551512437638

[63] J. Surowiecki, *The Wisdom of Crowds*. Knopf Doubleday Publishing Group, 2005.

[64] A. Doan, R. Ramakrishnan, and A. Y. Halevy, "Crowdsourcing systems on the world-wide web," *Communications of the ACM*, vol. 54, no. 4, pp. 86–96, 2011. https://doi.org/10.1145/1924421.1924442

[65] S. Vijayanarasimhan and K. Grauman, "Large-scale live active learning: Training object detectors with crawled data and crowds," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1449–1456, June 2011, Providence, RI. https://doi.org/10.1007/s11263-014-0721-9

[66] A. Sorokin, D. Berenson, S. Srinivasa, and M. Hebert, "People helping robots helping people: Crowdsourcing for grasping novel objects," *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2117–2122, October 2010, Taipei, Taiwan. https://doi.org/10.1109/IROS.2010.5650464

[67] "Amazon Mechanical turk," URL: https://www.mturk.com.

[68] "Mob4hire," URL: https://www.mob4hire.com.

[69] "Freelancer," URL: https://www.freelancer.com.

[70] "uTest," URL: https://www.utest.com.

[71] D. Brabham, *Crowdsourcing*. The MIT Press, 2013.

[72] N. J. Gadgil, K. Tahboub, D. Kirsh, and E. J. Delp, "A web-based video annotation system for crowdsourcing surveillance videos," *Proceedings of the SPIE/IS&T Electronic Imaging, Imaging and Multimedia Analytics in a Web and Mobile World*, vol. 9027, pp. 90 270A–1–12, March 2014, San Francisco, CA. https://doi.org/10.1117/12.2042440

[73] K. Tahboub, N. Gadgil, J. Ribera, B. Delgado, and E. J. Delp, "An intelligent crowdsourcing system for forensic analysis of surveillance video," *Proceedings of the SPIE/IS&T Electronic Imaging, Video Surveillance and Transportation Imaging Applications*, vol. 9407, pp. 94 070I–94 070I–9, February 2015. https://doi.org/10.1117/12.2077807

[74] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of Artificial Intelligence Research*, vol. 4, pp. 129–145, March 1996. https://doi.org/10.1613/jair.295

[75] Y. Yan, R. Rosales, G. Fung, and J. Dy, "Active learning from crowds," *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 1161–1168, June 2011, Bellevue, WA. https://dl.acm.org/citation.cfm?id=3104482.3104628

[76] Y. Yan, R. Rosales, G. Fung, M. W. Schmidt, G. H. Valadez, L. Bogoni, L. Moy, and J. G. Dy, "Modeling annotator expertise: Learning when everybody knows a bit of something," *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, pp. 932–939, May 2010, Chia Laguna, Italy.

[77] G. Acquaah, *Principles of Plant Genetics and Breeding*. John Wiley & Sons, 2012.

[78] T. N. Carlson and D. A. Ripley, "On the relation between NDVI, fractional vegetation cover, and leaf area index," *Remote sensing of Environment*, vol. 62, no. 3, pp. 241–252, December 1997. https://doi.org/10.1016/S0034-4257(97)00104-1

[79] E. Hamuda, M. Glavin, and E. Jones, "A survey of image processing techniques for plant extraction and segmentation in the field," *Computers and Electronics in Agriculture*, vol. 125, pp. 184 – 199, 2016. https://doi.org/10.1016/j.compag.2016.04.024

[80] P. Lottes, R. Khanna, J. Pfeifer, R. Siegwart, and C. Stachniss, "UAV-based crop and weed classification for smart farming," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3024–3031, May 2017. https://doi.org/10.1109/ICRA.2017.7989347

[81] W. Guo, U. K. Rage, and S. Ninomiya, "Illumination invariant segmentation of vegetation for time series wheat images based on decision tree model," *Computers and Electronics in Agriculture*, vol. 96, pp. 58 – 66, 2013. https://doi.org/10.1016/j.compag.2013.04.010

[82] J. Torres-Sánchez, F. López-Granados, and J. Peña, "An automatic object-based method for optimal thresholding in uav images: Application for vegetation detection in herbaceous crops," *Computers and Electronics in Agriculture*, vol. 114, pp. 43 – 52, 2015. https://doi.org/10.1016/j.compag.2015.03.019

[83] J. M. Peña Barragán, M. Kelly, A. I. d. Castro, and F. López Granados, "Object-based approach for crop row characterization in uav images for site-specific weed management," *Proceedings of the Geographic Object-Based Image Analysis Conference*, pp. 426–430, 2012, Rio de Janeiro, Brazil.

[84] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, January 1979. https://doi.org/10.1109/TSMC.1979.4310076

[85] J. Guerrero, G. Pajares, M. Montalvo, J. Romeo, and M. Guijarro, "Support vector machines for crop/weeds identification in maize fields," *Expert Systems with Applications*, vol. 39, no. 12, pp. 11 149 – 11 155, 2012. https://doi.org/10.1016/j.eswa.2012.03.040

[86] H. Huang, J. Deng, Y. Lan, A. Yang, X. Deng, and L. Zhang, "A fully convolutional network for weed mapping of unmanned aerial vehicle (UAV) imagery," *Public Library Of Science (PLOS) ONE*, vol. 13, no. 4, p. e0196302, April 2018. https://doi.org/10.1371/journal.pone.0196302

[87] A. Milioto, P. Lottes, and C. Stachniss, "Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns," *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2229–2235, May 2018. https://doi.org/10.1109/ICRA.2018.8460962

[88] I. Sa, Z. Chen, M. Popović, R. Khanna, F. Liebisch, J. Nieto, and R. Siegwart, "weednet: Dense semantic weed classification using multispectral images and MAV for smart farming," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 588–595, 2018. https://doi.org/10.1109/LRA.2017.2774979

[89] M. Bah, A. Hafiane, and R. Canals, "Deep learning with unsupervised data labeling for weed detection in line crops in UAV images," *Remote Sensing*, vol. 10, no. 11, p. 1690, October 2018. https://doi.org/10.3390/rs10111690

[90] F. Chianucci, L. Disperati, D. Guzzi, D. Bianchini, V. Nardino, C. Lastri, A. Rindinella, and P. Corona, "Estimation of canopy attributes in beech forests using true colour digital images from a small fixed-wing uav," *International Journal of Applied Earth Observation and Geoinformation*, vol. 47, pp. 60 – 68, May 2016. https://doi.org/10.1016/j.jag.2015.12.005

[91] "Agronomy Center for Research and Education," URL: https://ag.purdue.edu/agry/acre.

[92] J. Ribera, F. He, Y. Chen, A. F. Habib, and E. J. Delp, "Estimating phenotypic traits from UAV based RGB imagery," *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on Data Science for Food, Energy, and Water*, August 2016, San Francisco, CA.

[93] F. He, T. Zhou, W. Xiong, S. M. Hasheminnasab, and A. Habib, "Automated aerial triangulation for uav-based mapping," *Remote Sensing*, vol. 10, no. 12-1952, December 2018. https://doi.org/10.3390/rs10121952

[94] M. Elbahnasawy, T. Shamseldin, R. Ravi, T. Zhou, Y. Lin, A. Masjedi, E. Flatt, M. Crawford, and A. Habib, "Multi-sensor integration onboard a uav-based mobile mapping system for agricultural management," *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3412–3415, July 2018. https://doi.org/10.1109/IGARSS.2018.8517370

[95] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The computer journal*, vol. 7, no. 4, pp. 308–313, January 1965. https://doi.org/10.1093/comjnl/7.4.308

[96] M. J. Powell, "A direct search optimization method that models the objective and constraint functions by linear interpolation," *Advances in optimization and numerical analysis*. Springer, 1994, pp. 51–67. https://doi.org/10.1007/978-94-015-8330-5_4

[97] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[98] S. R. Deans, *The Radon transform and some of its applications*. Dover Publications, 2007, Mineola, NY.

[99] S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Y. and the scikit-image contributors, "scikit-image: Image processing in python," *PeerJ*, p. 2:e453, June 2014. https://doi.org/10.7717/peerj.453

[100] S. Jennings, N. Brown, and D. Sheil, "Assessing forest canopies and understorey illumination: canopy closure, canopy cover and other measures," *Forestry: An International Journal of Forest Research*, vol. 72, no. 1, pp. 59–74, December 1999. https://doi.org/10.1093/forestry/72.1.59

[101] L. Korhonen, K. T. Korhonen, M. Rautiainen, and P. Stenberg, "Estimation of forest canopy cover: a comparison of field measurement techniques," *Silva Fenica*, vol. 40, no. 4, 2006. https://doi.org/10.14214/sf.315

[102] M. K. Agoston, *Computer Graphics and Geometric Modeling*. Springer London, 2005. https://doi.org/10.1007/b138805

[103] A. D. Bell and A. Bryan, *Plant Form: An Illustrated Guide to Flowering Plant Morphology*. Timber Press, September 2008.

[104] S. Seguí, O. Pujol, and J. Vitria, "Learning to count with deep object features," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 90–96, June 2015, Boston, MA. https://doi.org/10.1109/CVPRW.2015.7301276

[105] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, November 2016.

[106] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015. https://doi.org/10.1038/nature14539

[107] R. Girshick, "Fast R-CNN," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, December 2015. https://doi.org/10.1109/ICPR.1994.576361

[108] J. R. R. Uijlings, , K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, September 2013. https://doi.org/10.1007/s11263-013-0620-5

[109] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 36, no. 6, pp. 1137–1149, June 2017. https://doi.org/10.1109/TPAMI.2016.2577031

[110] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *arXiv:1703.06870*, April 2017. arxiv.org/abs/1703.06870

[111] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "SSD: Single shot multibox detector," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 21–37, October 2016, Amsterdam, The Netherlands. https://doi.org/10.1007/978-3-319-46448-0_2

[112] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, Honolulu, HI. https://doi.org/10.1109/CVPR.2015.7298684

[113] V. Lempitsky and A. Zisserman, "Learning to count objects in images," *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pp. 1324–1332, December 2010, Vancouver, Canada.

[114] J. Shao, D. Wang, X. Xue, and Z. Zhang, "Learning to point and count," *arXiv:1512.02326*, December 2015. https://arxiv.org/abs/1512.02326

[115] F. Xiong, X. Shi, and D. Yeung, "Spatiotemporal modeling for crowd counting in videos," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 5151–5159, October 2017, Venice, Italy. https://doi.org/10.1109/ICCV.2017.551

[116] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2008, Anchorage, AK. https://doi.org/10.1109/CVPR.2008.4587583

[117] M. D. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. V. Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1820–1833, 2011. https://doi.org/10.1109/TPAMI.2010.232

[118] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Single-image crowd counting via multi-column convolutional neural network," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 589–597, June 2016, Las Vegas, NV. https://doi.org/10.1109/CVPR.2016.70

[119] D. B. Sam, S. Surya, and R. V. Babu, "Switching convolutional neural network for crowd counting," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4031–4039, July 2017. https://doi.org/10.1109/CVPR.2017.429

[120] S. Huang, X. Li, Z. Zhang, F. Wu, S. Gao, R. Ji, and J. Han, "Body structure aware deep crowd counting," *IEEE Transactions on Image Processing*, vol. 27, no. 3, pp. 1049–1059, March 2018. https://doi.org/10.1109/TIP.2017.2740160

[121] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 833–841, June 2015, Boston, MA. https://doi.org/10.1109/CVPR.2015.7298684

[122] V. Sundstedt, *Gazing at Games: An Introduction to Eye Tracking Control*. San Rafael, CA: Morgan & Claypool Publishers, 2012, vol. 5, no. 1.

[123] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, "Learning from simulated and unsupervised images through adversarial training," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2107–2116, July 2017, Honolulu, HI. https://doi.org/10.1109/CVPR.2017.241

[124] J. Gu, X. Yang, S. De Mello, and J. Kautz, "Dynamic facial analysis: From bayesian filtering to recurrent neural network," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1548–1557, July 2017, Honolulu, HI. https://doi.org/10.1109/CVPR.2017.167

[125] W. Fuhl, T. Santini, C. Reichert, D. Claus, A. Herkommer, H. Bahmani, K. Rifai, S. Wahl, and E. Kasneci, "Non-intrusive practitioner pupil detection for unmodified microscope oculars," *Computers in Biology and Medicine*, vol. 79, pp. 36–44, December 2016. https://doi.org/10.1016/j.compbiomed.2016.10.005

[126] W. Fuhl, T. Kübler, K. Sippel, W. Rosenstiel, and E. Kasneci, "ExCuSe: Robust pupil detection in real-world scenarios," *Proceedings of the International Conference on Computer Analysis of Images and Patterns*, pp. 39–51, September 2015, Valletta, Malta. https://doi.org/10.1007/978-3-319-23192-1_4

[127] R. Sui, B. E. Hartley, J. M. Gibson, C. Yang, J. A. Thomasson, and S. W. Searcy, "High-biomass sorghum yield estimate with aerial imagery," *Journal of Applied Remote Sensing*, vol. 5, p. 053523, 2011. https://doi.org/10.1117/1.3586795

[128] S. Aich, I. Ahmed, I. Obsyannikov, I. Stavness, A. Josuttes, K. Strueby, H. Duddu, C. Pozniak, and S. Shirtliffe, "Deepwheat: Estimating phenotypic traits from crop images with deep learning," *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, March 2018, Stateline, NV.

[129] H. Attouch, R. Lucchetti, and R. J. B. Wets, "The topology of the ρ-Hausdorff distance," *Annali di Matematica Pura ed Applicata*, vol. 160, no. 1, pp. 303–320, December 1991. https://doi.org/10.1007/BF01764131

[130] M.-P. Dubuisson and A. K. Jain, "A modified Hausdorff distance for object matching," *Pattern Recognition*, pp. 566–568, October 1994. https://doi.org/10.1109/ICCV.2015.169

[131] Y. Lu, C. L. Tan, W. Huang, and L. Fan, "An approach to word image matching based on weighted Hausdorff distance," *Proceedings of International Conference on Document Analysis and Recognition*, pp. 921–925, September 2001. https://doi.org/10.1109/ICDAR.2001.953920

[132] K. L. K. Lin and W. Siu, "Spatially eigen-weighted Hausdorff distances for human face recognition," *Pattern Recognition*, vol. 36, no. 8, pp. 1827–1834, August 2003. https://doi.org/10.1016/S0031-3203(03)00011-6

[133] O. Schutze, X. Esquivel, A. Lara, and C. A. C. Coello, "Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 504–522, August 2012. https://doi.org/10.1109/TEVC.2011.2161872

[134] H. E. Khiyari and H. Wechsler, "Age invariant face recognition using convolutional neural networks and set distances," *Journal of Information Security*, vol. 8, no. 3, pp. 174–185, July 2017. https://doi.org/10.4236/jis.2017.83012

[135] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2463–2471, July 2017, Honolulu, HI. https://doi.org/10.1109/CVPR.2017.264

[136] A. A. Taha and A. Hanbury, "Metrics for evaluating 3D medical image segmentation: Analysis, selection, and tool," *BMC Medical Imaging*, vol. 15, no. 1, p. 29, August 2015. https://doi.org/10.1186/s12880-015-0068-x

[137] S. K. Zhou, H. Greenspan, and D. Shen, *Deep Learning for Medical Image Analysis*. London, United Kingdom: Academic Press, 2017.

[138] S. Liao, Y. Gao, A. Oto, and D. Shen, "Representation learning: A unified deep learning framework for automatic prostate mr segmentation," *Proceedings of the Medical Image Computing and Computer-Assisted Intervention*, pp. 254–261, September 2013, Nagoya, Japan. https://doi.org/10.1007/978-3-642-40763-5_32

[139] P. Teikari, M. Santos, C. Poon, and K. Hynynen, "Deep learning convolutional networks for multiphoton microscopy vasculature segmentation," *arXiv:1606.02382*, June 2016. https://arxiv.org/abs/1606.02382

[140] E. M. Arkin, L. P. Chew, D. P. Huttenlocher, K. Kedem, and J. S. Mitchell, "An efficiently computable metric for comparing polygonal shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, March 1991. https://doi.org/10.1109/34.75509

[141] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 234–241, October 2015, Munich, Germany. https://doi.org/10.1007/978-3-319-24574-4_28

[142] C. S. Kubrusly, "Banach spaces L^p," *Essentials of Measure Theory.* Springer, Cham, 2005, p. 83. https://doi.org/10.1007/978-3-319-22506-7_5

[143] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, Honolulu, HI. https://doi.org/10.1109/CVPR.2017.632

[144] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, pp. 73–101, 1964.

[145] J. D. Hunter, "Matplotlib: A 2D graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, June 2007. https://doi.org/10.1109/MCSE.2007.55

[146] T. Porter and T. Duff, "Compositing digital images," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 253–259, July 1984. https://doi.org/10.1145/964965.808606

[147] C. Schröder, "A hybrid parameter estimation algorithm for beta mixtures and applications to methylation state classification," *Algorithms for Molecular Biology*, vol. 12, no. 21, pp. 62–66, August 2017. https://doi.org/10.1186/s13015-017-0112-1

[148] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, November 1996. https://doi.org/10.1109/79.543975

[149] C. C. Loy, K. Chen, S. Gong, and T. Xiang, "Crowd counting and profiling: Methodology and evaluation," *Modeling, Simulation and Visual Analysis of Crowds.* Springer, October 2013, pp. 347–382. https://doi.org/10.1007/978-1-4614-8483-7_14

[150] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the International Conference for Learning Representations (ICLR)*, vol. abs/1412.6980, April 2015, San Diego, CA. https://arxiv.org/abs/1412.6980

[151] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of Adam and beyond," *Proceedings of the International Conference on Learning Representations (ICLR)*, February 2018, Vancouver, Canada. https://openreview.net/forum?id=ryQu7f-RZ

[152] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proceedings of the International Conference on Learning Representations (ICLR)*, May 2015, San Diego, CA. https://arxiv.org/abs/1409.1556

[153] 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, August 2002. https://doi.org/10.1109/34.1000236

[154] 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, October 2011.

[155] Y. Li, X. Zhang, and D. Chen, "CSRNet: Dilated convolutional neural networks for understanding the highly congested scenes," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1091–1100, June 2018, Salt Lake City, UT.

[156] L. Świrski, A. Bulling, and N. Dodgson, "Robust real-time pupil tracking in highly off-axis images," *Proceedings of the Symposium on Eye Tracking Research and Applications*, pp. 173–176, March 2012, Santa Barbara, CA. https://doi.org/10.1145/2168556.2168585

[157] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, "A large contextual dataset for classification, detection and counting of cars with deep learning," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 785–800, October 2016, Amsterdam, The Netherlands. https://doi.org/10.1007/978-3-319-46487-9_48

[158] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *arXiv:1404.5997*, April 2014. https://arxiv.org/abs/1404.5997

[159] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *Proceedings of the International Conference on Machine Learning (ICML)*, vol. 37, pp. 448–456, July 2015, Lille, France. http://proceedings.mlr.press/v37/ioffe15.pdf

[160] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818–2826, June 2016, Las Vegas, NV. https://doi.org/10.1109/CVPR.2016.308

[161] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv:1602.07261*, August 2016. https://arxiv.org/abs/1602.07261

[162] A. Rozantsev, V. Lepetit, and P. Fua, "On rendering synthetic images for training an object detector," *Computer Vision and Image Understanding*, vol. 137, pp. 24–37, August 2015. https://doi.org/10.1016/j.cviu.2014.12.006

[163] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. https://www.tensorflow.org

[164] S. J. Wright, "Coordinate descent algorithms," *Mathematical Programming*, vol. 151, pp. 3–34, June 2015. https://doi.org/10.1007/s10107-015-0892-3

[165] R. Rivest, "The MD5 message-digest algorithm," *RFC 1321*, April 1992. https://doi.org/10.17487/RFC1321

[166] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," *Video-Based Surveillance Systems*, P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni, Eds. Boston, MA: Springer US, 2002, pp. 135–144. https://doi.org/10.1007/978-1-4615-0913-4_11

[167] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," *Proceedings of the International Conference on Pattern Recognition (ICPR)*, vol. 2, pp. 28–31, August 2004. https://doi.org/10.1109/ICPR.2004.1333992

[168] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, May 2006. https://doi.org/10.1016/j.patrec.2005.11.005

[169] "OpenCV," URL: https://www.opencv.org.

[170] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of the IEEE*, vol. 67, no. 5, pp. 786–804, May 1979. https://doi.org/10.1109/PROC.1979.11328

[171] A. B. Chan and N. Vasconcelos, "Modeling, clustering, and segmenting video with mixtures of dynamic textures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 5, pp. 909–926, May 2008. https://doi.org/10.1109/TPAMI.2007.70738

[172] "FFmpeg," URL: https://www.ffmpeg.org.

VITA

VITA

Javier Ribera was born in Barcelona, Catalonia, Spain. In 2014, he received his Bachelor of Science degree in Superior Telecommunications Engineering from the Polytechnic University of Catalonia (Universitat Politècnica de Catalunya), at the Barcelona School Of Telecommunications Engineering (Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona). He was a visitor scholar in the Video and Image Processing (VIPER) laboratory in 2014. While in the graduate program at Purdue, he worked on projects sponsored by the Advanced Research Projects Agency-Energy (ARPA-E), U.S. Department of Energy (DoE), and the U.S. Department of Homeland Security Center of Excellence (VACCINE),

Mr. Ribera has one year of industry experience with Samsung and Telefónica. His current research interests include machine and deep learning, image processing, and computer vision. He is a student member of the IEEE and the IEEE Signal Processing Society. He has been a reviewer of the IEEE Signal Processing Letters.