VIDEO PROCESSING FOR AGRICULTURAL APPLICATIONS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

He Liu

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF DISSERTATION APPROVAL

ACKNOWLEDGMENTS

First and foremost I would like to express my sincere gratitude to my advisor Professor Amy R. Reibman for her support of my study and research, for her patience, motivation and knowledge. Her guidance helped me finish my research work and paper writing.

I would like to say thank you to the rest of my committee, Professor James V. Krogmeier, Professor Jan P. Allebach, and Professor Mary Comer, for their insightful comments and suggestions.

I would also like to express my appreciation to the Open AG Technology and Systems (OATS) center for the support of both my projects. It is a great pleasure to work my college Yang Wang in the farming machinery project, and Professor Jacquelyn P. Boerman offers us so much help in the dairy cow project.

I am also thankful to my fellow colleges in my lab, Biao Ma, Chen Bai and Chengzhang Zhong for their help of my work. In addition, I will always cherish the memorable time we spent in the lab including the junior members Haoyu Chen, Jonathan(Shengtai) Ju, and Praneet Singh.

Finally I would like to thank my family for their financial and spiritual support, especially for the years that I am aboard.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

SAE	Society of Automobile Engineers
FPT	Fixed Position Trajectory
CCM	Color Co-occurrence Matrix
CBIR	Content-Based Image Retrieval
LBP	Local Binary Pattern
GLCM	Gray Level Co-occurrence Matrix
CNN	Convolutional Neural Network
STIP	Space-Time Interest Points
IDT	Improved Dense Trajectory
HOG	Histogram of Oriented Gradients
HOF	Histogram of Optical Flow
MBH	Motion Boundary Histograms
HMG	Histograms of Motion Gradients
VLAD	Vector of Locally Aggregated Descriptors
SVM	Support Vector Machine
PCA	Principal Component Analysis
GMM	Gaussian Mixture Model
DVR	Digital Video Recorder
CAN	Controller Area Network
GPS	Global Positioning System
UAV	Unmanned Aerial Vehicle
OVS	Video Object Segmentation
POE	Power Over Ethernet
RF	Random Forest

IOU	Intersection Over Union
GMM	Gaussian Mixture Model
PCC	Pearson Correlation Coefficient
VCP	Valid Cow Percentage
TC	Temporal Consistency
RMSE	Root Mean Square Error
MAPE	Mean Absolute Percentage Error
EP	Error Percentile

ABSTRACT

Liu, He Ph.D., Purdue University, May 2020. Video Processing for Agricultural Applications . Major Professor: Amy R. Reibman.

Cameras are widely used as sensors for a variety of engineering applications. In a typical video-based application, spatial segmentation is a fundamental step which provides the spatial positions of different targets for further analysis. In this thesis, we focus on videos analytics applied to the agricultural industry and describe several video segmentation methods in the context of two practical projects: autonomous farming vehicles and analysis of dairy cow health. In the autonomous farming vehicle project, we propose three spatial segmentation methods based on traditional video features to isolate the regions of the video frame where critical information appears. Two applications that apply the segmentation method are presented: farming activity classification and header-height control for a combine harvester. In the project on cow health, we propose a cow structural model based on the keypoints of joints from a side-view cow video. A detection system is developed using deep learning techniques to automatically extract the structural model from the videos. Based on this model, we also present a preliminary application which estimates the cow's weight based on video information.

1. INTRODUCTION

This thesis introduces two applications in agriculture: video processing for farming machinery and animal health monitoring using visual data. This chapter introduces the related background of both the video processing applications presented in later chapters. In Section 1.1, we first describe the general video applications in agricultural industry, and then talk about the different levels of spatial segmentation problems applied in various tasks. Next, we introduce the video applications for large farming machinery, and their unique challenges in Section 1.2 and Section 1.3. After that, some animal-related visual applications from the literature are introduced in Section 1.4. Finally, we review the previous technical researches about the video and image processing in Section 1.6.

1.1 General Background

1.1.1 Video applications and agriculture

Agriculture industry covers a wide range of subareas, including cultivating soil, raising crops, feeding live stocks, and so on. Recently, new engineering technologies, such as sensing and robotics, have been applied to challenges in the agriculture [1] industry. For example, the traditional farming activities [1] such as planting and harvesting also benefits from these technologies by using advanced farming machines (vehicles), including tractors and combine harvesters. Applications have been designed to improve different aspects of the farming machines including autonomous control. Complete autonomous systems are still difficult to build at current stage, but it is possible to develop systems to help and assist parts of the control process for these machines.

[1]In this thesis, farming activities represent the crop raising with large farming machines.

Among different sensors, videos can provide rich visual information at relatively low cost. Like human eyes, cameras can be applied in different scenarios to monitor a place to sense the environment. Effectively and robustly processing the video data can replace the human label in many applications. Focusing on agriculture, in later chapters, we present two projects which are designed for the purpose of replacing humans. The target of the first application introduced in Chapter 3 autonomous control of farming vehicles. Different sensors including cameras, are placed on the farming machine to collect data from the environment, and all the signals are processed and merged to make final decisions that controls the machine. Another application presented in Chapter 4 is using cameras to monitor diary cow behaviors. We capture the walking positions and the poses of cows and analyze their health conditions, like weight estimation or lameness detection.

1.1.2 Spatial segmentation in video applications

Generally, the video-based systems require a spatial segmentation process for most of the practical applications. However, their segmentation targets are not the same, and different segmentation problems are proposed at different levels. From the literature, semantic segmentation, object detection, and instance segmentation are considered as different levels of spatial segmentation problems. Object detection problem tries to identify the objects in the image, then it classifies each object to a certain category and locates them with bounding boxes. Semantic segmentation labels every pixel in the image frame to a known category. Instance segmentation detects more detailed information: it not only detects all possible objects from the image at pixel level, but also separates the objects into different categories.

The segmentation targets in the two agriculture projects presented in this thesis are also different. In the farming vehicle automation project, the system needs to select the region of interest from the video frames instead of detecting objects. This is more like a semantic segmentation problem, but the challenges are unique in this project.

For example, the image orientation varies from video to video, which makes their frame structures different. In addition, the video content largely depends on the farming activity and the type of machine. As a result, identifying the specific regions under these challenges is the target for this project.

Animal (dairy cows) health project has a different segmentation target. The goal is to analyze the health conditions of every animal, which requires a detailed instance segmentation of all the cow objects. However, extracting the binary masks for every cow object is not enough for health analysis. For example, cow lameness analysis requires visual cues such as the movement of the cow's body parts like legs and hooves. As a result, the segmentation target in this project includes more detailed information such as the location of the cow's joints, which is a keypoint detection problem in spatial domain. Later in Chapter 4, we propose a model to detect both the keypoints and the mask of the cows. A related application about cow-weight estimation is also presented based on the extracted information.

1.2 Videos and autonomous vehicles

A normal autonomous system requires sensors to collect signals from the surrounding environment. Compared to traditional sensors such as GPS and RADAR used in farming machines, cameras can provide a large amount of visual data efficiently, and the data can be easily interpreted for human analysis. With the help of image processing methods and video analytics, cameras have been applied in automation systems for farming vehicles [2–5].

However, farming machines are more complex than automobiles. In addition to steering and speed control, machines like a tractor or a combine harvester have a tool, or attachment, which has interaction between the crops and the field. This interaction needs to be manually controlled by farmers and requires domain knowledge to be automated. Each farming activity requires a distinct attachment with its own requirements for controlling the interaction.

Figure 1.1 shows three different farming activities and their attachments: corn chopping, wheat harvesting, and tillage. The color, shape, and motion of the attachments are all distinctive, which make them effective cues to separate one activity from another. Automating the interaction between the attachment and the field requires identifying the activity and isolating the location of the attachment in the image. For example, when harvesting, the height of the attachment (called the header) should be adjusted based on the condition of the approaching field.

In addition, processing videos in practical farming vehicle applications is also challenging. Different applications have distinct requirements which determine where the camera should be located, and their processing techniques are not the same. In [6], cameras that are placed near the auger of a combine harvester are used to automate the unloading process, while in [7] dash cameras are mounted inside the cockpit of farming vehicles to capture the front view of the operator. But one common challenge for cameras is that the captured images normally include some unrelated areas which are not useful for further analysis [8]. As a result, the most fundamental step for practical image analysis is to identify which region of the image is useful for the application. Unlike many video-based applications whose goal is to detect objects, instead we need to locate distinct regions of the image for further analysis.

1.2.1 Car and farming vehicle automation

This section first introduces the general background of autonomous cars and automation applications on farming vehicles. Then we discuss the importance of video analytic in autonomous systems.

Autonomous driving is one of the most popular applications in the engineering industry. One objective of autonomous driving in an urban environment is to drive safely towards a target position and avoid the obstacles in the road [9]. To detect potential obstacles, different sensors such as LIDAR, RADAR, cameras, and GPS are applied to sense the surrounding environment [8]. Such multi-sensor signals are merged with sensor fusion algorithms to make correct decisions. However, sensor fusion algorithms are an important issue in such methods because different sensors have different reliability under various environment. For example, RADAR is difficult to detect the object lateral position [8] and cameras may fail in foggy environment. The purpose of applying different sensors is to maximize the reliability of detecting obstacles on the road [10].

Comparing to autonomous cars, automation on farming vehicles has different focuses. First, the task of detecting obstacles is at different importance levels between car automation and farming vehicle automation. Autonomous farming vehicle requires basic self-driving capability in the field. But unlike urban environment, the farming vehicles are driving on the widely-opened field, and there are few human or other moving objects. The possible obstacles in the farm are easy to observe, such as big rocks and large tree branches. And in real situation, it is less likely to find such obstacles in the cultivated field, and we can assume the vehicle is able drive in any desired route. In this situation, previous applications on steering automation control mainly focus on other purposes, such as alignment with field edges [3], or designing efficient harvesting route [11].

The second difference between farming vehicles and cars is that farming vehicles normally have secondary controls besides steering. Cars are designed to carry people to reach difference places, but farming vehicles are designed to perform the corresponding farming activities. Different farming vehicles need their own corresponding operations. For example, for a corn chopper, the operator controls the front cutter wisely to harvest the corn field with minimum oil consumption; sprayer needs to control the spraying arm and optimize the usage of fertilizer on different fields. To develop the autonomous farming vehicles, such secondary controls should be the focus.

Compared to autonomous cars, the third difference of farming vehicle automation is their expected targets. Society of Automobile Engineers (SAE) designs five different levels of autonomous cars, with level 5 referring to the fully un-man autonomous cars [12]. Most studies in the autonomous car targets to design level 2 to 4, which means when the algorithm fails, human can take over the control of vehicles. However, in farming vehicles, there is no such automation levels and the target are not clearly defined. The general goal is to design un-man-controlled farming vehicles, but there are few clearly-defined targets for specific operation on some types of farming machine. Because the farming operations need domain knowledge and different operations have their own challenges, and the difficulties of the operations are hard to measure. In addition, the risk of designing autonomous farming vehicles is lower because it is less likely to cause collision damage in the field comparing to an urban environment. This also affect the target of designing automation systems in farming machines.

1.2.2 Videos in automation applications

Camera is one of the widely-used sensors in autonomous vehicles, because it can provide fast and accurate information about surroundings. It is also cost-effective and can detect information that other sensors do not, such as analyzing the traffic signs and traffic lights [13]. Without using other sensors, the visual information is capable enough to support autonomous driving systems. [14] designed an end-to-end driving system for steering wheel control and the system can detect the lanes and control the speed of the vehicle. [15] develops a video-based automation application that can deal with complicated urban environment.

In agriculture industry, camera is also useful for automation applications. [2] shows a machine control system need to use vision-based guidance for monitoring nearby field conditions. In their work, several cameras are placed at the front of a tractor, and they are used to detect the crop positions in front of the field, which is designed for steering control. Apart from driving assistance, the cameras can be set on other positions, for example the end of auger on a combine harvester to monitor the number of crops in the grain container [6].

Table 1.1.: Comparison between different farming activities

Processing steps	Activity classification	Reel control	Corn row alignment (not implemented)
Segmentation	Header region	Field region	Edges of row
Feature sampling	Fixed Position Trajectory	Spatial squares	Spatial lines
Feature types	HOG, Motions	Image Textures	Texture, color
Decision system	One-vs-all SVM	Crop presence classifier	Edge detection + line fitting

1.2.3 Farming automation application comparison

There is some commonality of developing different video-based autonomous farming applications. Table 1.1 summaries some common procedures of some farming video processing applications. We compare three different applications in the table: activity classification, reel control and corn row alignment. The activity classification and reel control are studied and explained in later chapters. The target of corn row alignment is detecting the edges of the corn so that the chopper can drive in alignment with the corn rows. All these applications are trying to find cues from the domain knowledge and train a system to decide. All three activities share some similar processing techniques. For example, all three applications include a spatial segmentation step and their purposes are highlighting the important spatial regions based on domain knowledge. In addition, all applications include an image or video feature extraction process to represent the cues for making decisions.

Although the applications apply the similar processing steps, their methods have different focuses. For example, activity classification uses more motion features between different farming categories, but both reel control and row alignment focus on image-based features, like texture and color. Beside their methods, the targets for each process are not the same among three different applications. For example, we only need rough segmentation results in activity classification because segmentation is only applied to provide the weights of features. But in corn row alignment application, the edges of the corn row need to be segmented accurately so that the chopper does not miss any crops. Therefore, it is reasonable to develop a basic general segmentation method for all, but the criteria to measure the segmentation performance needs to be defined differently in individual applications.

1.3 Challenges in farming video processing

This section introduces the challenges in the farming video processing process. We first talk about the difficulties in farming video capturing, and then introduce the general challenges when processing these videos.

1.3.1 Video collection challenges

For the purpose of monitoring the surroundings of the vehicle, we install cameras inside the cockpit in combine harvesters or tractors, and we call the captured videos farming videos or farming machinery videos. However, the video collection process is not trivial. In general, it is time-consuming to collect videos in an outdoor environment for everyday farming activities [16]. To the best of our knowledge, there are no embedded camera systems available for farming vehicles that can transfer videos directly for analysis, and there are few public farming datasets available. A multi-sensor dataset is published from [17] to detect obstacles on the farm, but their video data only lasts for two hours.

In addition, our target is to build a general segmentation system for different farming activities, and we need to collect various types of videos from different farming events. This is because the training-based systems have poor performance on data they have not been seen before, and poor data variation causes troubles [18]. However, in practice, the timing of when most of the farming activities occur depends on factors such as human labor and weather, and most of these tasks are finished within a short period of time. This means that the farming videos must be collected within a limited time period, for example within several hours a day for a few days a year. Such videos contain the same repeating actions under the same environment for a long time, and special situations like field anomalies or machine breakdown rarely occur.

In summary, the video capturing environment makes it difficult to build one general segmentation method which works for all farming applications with limited computational power. However, it is possible to build a system with a quick learning process using limited data, so that it can be easily adapted to different farming applications. Our proposed segmentation method in Section 3.3.3 is designed based on this idea. It is trained on low-level hand-crafted features with limited data and provides robust segmentation on videos from different farming activities.

1.3.2 Processing challenges

Classifying these farming activities using dashcam videos poses three challenges. The first challenge is the unconstrained capturing environment. With a working machine in the field, operators could place the camera at different positions to get different views, to monitor various aspects of the activity. Such different camera positions can cause significant image structure variations. In addition, the plants in the field have very different colors and shapes, depending on the weather and the season. In addition, there are no rigid objects to track in the field, which means it is hard to find and match robust feature points.

Figure 1.1 provides some examples of images from our farming videos. Each row shows images from the same category but with different capturing angles or colors. The first two images captured from two different fields during chopping corn silage. The second row shows two tillage images with two different camera angles. The last row presents two harvesting images with different camera positions. The left column images are chosen from the training set, and the right column images are from the testing set. We show in Section 3.4 that image-based features are inadequate alone to effectively classify our categories. Therefore, in later sections, we focus on video attributes and consider both spatial and motion information.

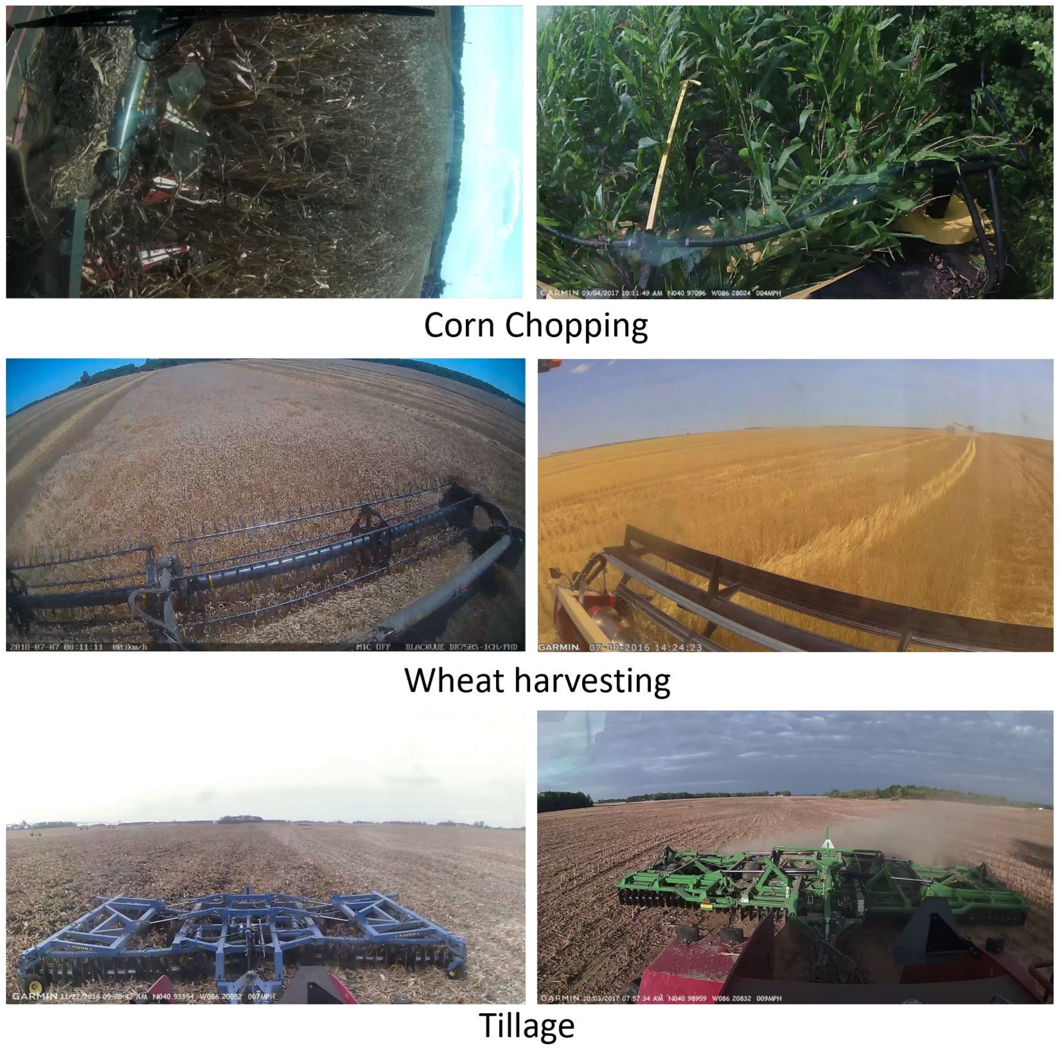


Figure 1.1.: Some farming video frames from vehicle mounted cameras: chopping corns (top), tillage (middle), harvesting (bottom). Images on the left are in the training data; images on the right are in the test data. Notice that these videos are captured with different camera positions and various angles.

The second challenge is the variety of very different video motions. Dashcam videos normally have three types of motion conditions: very low or no motion, object motion and camera motion. In many instances, motions can be very useful to distinguish activity categories, but not always. Camera movements can dominate motion features, for example when vehicle is slowing down. If these motions are used as features, all other activities which have a similar slowing down movement might be incorrectly classified.

In this case, camera motion becomes a problem in the feature extraction process. Furthermore, some videos can be static because the machine may stop anytime in the field and no motions are included in those periods.

The third challenge is the limited number of videos. To our knowledge, currently there are no large-scale farming video datasets available in either the classification or the farming community. In addition, the duration of a commercial cropping season is relatively long, and normally it takes a year to collect data from planting to harvesting. Because a farming season is relatively long, which means most farming activities take place only once a year, it is hard to collect a large amount of data in a short time period.

All three challenges are considered in our system development. As mentioned before, our goal is to segment regions instead of some specific objects. Summarizing the possible features from farming machinery videos, the structure and object in each region can change due to vehicle movement, and the image distortion also challenges the segmentation because of the outdoor environment [7]. Outdoor illumination changes quickly and influences the color of every region in the video. Some regions could be blocked by shadows and window glare due to the direction of movement and sunshine. Apart from color, the outdoor regions also suffer from noise and blur distortion, which are caused by the dust in the field or foggy weather.

Comparing to color and spatial structure, motion information is robust to outdoor illumination changes, but the variety of motion from a farming vehicle is also challenging. For normal vehicle-mounted cameras, two types of motions are summarized in [19]: camera-induced motion and independent motion (also called object motion in [7]). But both motion types are more complicated in farming videos compared to those in the videos from automobiles. Camera-induced motions are generated by the moving cameras, and farming vehicles are very shaky when moving in the field, which causes noises when analyzing the motions. Independent motions are caused by movements other than the camera, and most of these motions on farming videos come from the interaction between the machine and the crops. These motions are difficult to model and challenge the segmentation process.

Both the image distortion and complicated motions are considered in our proposed segmentation method. In section 3.1, We extract hand-crafted features of both color and motion information and train a classifier to segment images. Our classifier is effective given the practical challenges in farming videos.

1.4 Videos processing for animals

This section introduces the background of the animal-related video applications. Dairy cow is the target animal we focus in this thesis. We first introduce the motivations for this project in the thesis, and then review some animal-related video applications.

1.4.1 Dairy cow health problems

Monitoring animal health is a critical component of livestock farming, because healthy animals are more productive. In practices, the health monitoring is often performed visually, because animal appearance and behavior are key indicators of health changes. For example, trained farmers can analyze a dairy cow's health condition based on visual appearance [20], and can detect potential illnesses such as lameness [21]. However, time and labor limitations preclude a human routinely watching for these changes, especially in practical farms which house a large number of cows. Thus, there is increasing interest in substituting automated video analytics for human observations. Indeed, video analytic techniques have been applied to different industries including animal agriculture. With the help of the latest computer vision and image processing algorithms, visual animal biometrics has become an emerging research topic [22]. By applying video analytics methods, it is possible to develop a camera system that automatically detects the cow's health condition with a low cost.

1.4.2 Related applications for cows

There are some previous researches on cow-related visual applications and most of them focus on lameness detection. However, their processing techniques are developed for a specially-designed environment where the captured images are clear enough to process. Normally their detection targets are limited to a specific region instead of the entire cow body structure. For example, methods like [23, 24] only detect the cow's back curvature while other applications such as [25, 26] only track the trajectories of the legs and hooves. As a result, these methods are not general enough to provide a complete body structure. Apart from lameness, more researchers focus on cow identification problem with visual data [27, 28]. Their target is to extract cow features such as the traditional image features [29] or CNN-based features [30], to distinguish different cow identities. Their videos are collected from various devices, such as surveillance cameras in a pen [27] or even from Unmanned Aerial Vehicle (UAV) [28]. But all these methods need a fundamental step that detects and locates the cows within the image or videos.

1.5 Contributions

This section summarizes the major contributions in this thesis. There are two practical agriculture applications proposed: the video applications on farming machinery automation and video analytics on dairy cow health. In the farming machinery project, we define three spatial regions of the videos that are captured from the cockpit of the farming vehicles, and proposed three different spatial segmentation methods to detect every region. All three methods apply hand-crafted features based on color and motion information, which provide robust results under constraints such as limited data and computational power. One of the methods is based on training classifier to segment the image into all three regions, which provides a solution for this semantic segmentation problem. Two related applications are also introduced for farming machine automation. In the farming video classification application, we apply the idea to select the most distinguishable features for classification and propose a two-branch classification pipeline.

This farming video classification uses the idea to extract features based on the segmentation results. Another application is the header height control for combine harvesters. In this application, we proposed a crop presence classifier which estimates the amount of crops in the front field region.

In the dairy cow project, we mainly solve a keypoint structure detection problem for side-view cows. A cow structural model is proposed based on the keypoint joints, which represents the cow's spatial location and its walking pose. We also developed the cow structure model detection system based on two CNNs and post processing. This detection system is robust to the rough capturing environment with bad illuminations and poor video qualities, which can also detect multiple cow objects together in a frame. The corresponding evaluation metrics for the cow structure model is also proposed which includes two unsupervised metrics which work without ground truth labels. In addition to the model, we also implement a practical application which automatically evaluates the cow's weight purely based on the surveillance cameras. Two camera videos are applied to extract visual features, and the features are further fit into regression models for weight prediction. The result shows our prediction error is better than some recent work which applies complicated hardware systems.

Another contribution in this thesis is the video logging systems. Since both projects are practical applications, video data collection is the fundamental step for further video analytics. In farming vehicle application, an ISOBlue HD system is developed to gather information from the farming machines, including videos, GPS, CANBUS data, etc. This system can be automatically switched on when is machine starts to work and switched off when the machine shuts down. In the dairy cow project, different camera devices are experimented and the currently version is based on the IP surveillance cameras. A controlling module is designed for the camera controls and it only records videos at required time periods. Both two proposed system are deployed into practical applications and all the videos processed in the experiments are captured based on these video logging systems.

1.6 Literature review

This section reviews the typical research topics in video processing and introduce some previous practical video applications.

1.6.1 Image & video spatial segmentation

Image spatial segmentation is important because it allows us to distinguish the relevant visual cues from the proper area. Color-based graph cut methods such as [31–33] are popular for still image segmentation. But as we shown later that color features are not robust under different illuminations. [34] improves the graph cut method by adding temporal connection to the initial graph, which better separates the front object and the background. But as they stated, sensitivity to illumination changes is still a limitation of their method. Also as mentioned in [5], the color information is not useful for segmenting outdoor farming videos.

Compared with still images, videos contain motion that can be informative for spatial segmentation. The Video Object Segmentation (VOS) problem [35] is described as segmenting the front moving object from the background scene. In [36], motion information is applied. This method segments out foreground region by assuming the foreground only has large motion. The graph cut method is applied with video temporal information in [34] to segment the foreground object, and [37] considers video segmentation as a voting scheme from a region similarity graph.

More recent methods apply Convolutional Neural Networks (CNN) to segment objects from the image; for example [38] uses region proposals and [39] applies fully-connected networks. Recently, CNN methods also become popular for solving VOS problems [40, 41]. However, these methods require large training datasets such as [42, 43], and most of the popular methods are trained and tested with these public datasets that do not contain relevant images related to agriculture or farming. In addition, most of these datasets include videos with clearly-labelled moving objects in the foreground which is developed for object detection purposes. But for videos captured from farming vehicles, the foreground is not well defined, and it is hard to apply these methods to our segmentation problem. Training a large image segmentation network for farming applications requires a large number of images and ground truth labels, and it is difficult to gather so much data.

For practical applications, obtaining ground truth labels is time-consuming, so new methods are being developed to handle the label limitations. Semi-supervised methods train both labelled and unlabeled data together, while weakly-supervised methods use image-level labels on pixel-level segmentation tasks [44]. Saliency maps are used as supervised knowledge for segmentation in [45], and [46] defines image constraints to characterize the weakly-supervised labels. These methods solve the problem of either lack of labels or low-quality labels, but they do not consider the computational load. As mentioned in Section 1, computational power is limited for practical farming video systems. As a result, although the previous work tackles some practical problems such as labelling, they are not suitable for farming applications.

1.6.2 Object detection and keypoints detection

Object detection is one subset of spatial segmentation problem. Traditional Video Object Segmentation (VOS) methods such as [36, 47, 48] detect objects using motion information from video sequences. With the development of Convolutional Neural Network (CNN), new learning-based methods achieve much better results. Methods such as Mask R-CNN [49], DeepLab [50], and You Only Look Once (YOLO) [51], are widely applied to solve the problem of image semantic segmentation, which requires detection and classification of the objects in an image. CNNs are also applied for video object segmentation. One Shot Video Object Segmentation (OSVOS) [41] does an online fine-tuning process which is trained on one frame of the video sequence and applied to all the other frames in the sequence. This method is further extended with image-based detection methods for semantic guidance [52]. Other methods such as [53–55] apply different models using either temporal information or memory for object detection. There are also some popular public VOS datasets available for benchmarks, such as the YouTube VOS [56] and DAVIS dataset [42]. However, all these methods generate bounding boxes or pixel-level masks to represent detected objects, but the structural information of the object is not identified. Additional processing would be required to extract further detailed information from these masks.

Apart from spatial segmentation, there are also research focusing on object structural information such as keypoint detection and pose estimation. Benefiting from the public human pose datasets such as MPII [57] and COCO human skeleton [58], advanced methods are developed for human skeleton detection. DeepPose [59] first applies CNN for human body parts detection based on images, and the stacked hourglass network [60] extends it to detect humans at multiple spatial scales. To solve multi-human detection, the relationship between human joints are considered. ArtTrack [61] generates a simplified human body-part model; OpenPose [62,63] and Deepcut [64] use part affinity fields to model the joint relationships. However, all these methods are designed by incorporating different levels of knowledge about the human body, and they are not easily altered or fine-tuned for other objects like cows.

Recently, new methods such as the DeepLabCut [65] toolbox, LEAP [66] and DeepFly3D [67] extend keypoint detection to animals. One advantage of these methods is that they provide a means for users to define body parts; this allows the algorithm to adapt to different animal structures. The DeepLabCut toolbox also provides simple access to fine-tune the networks, and it can achieve promising results with a small amount of training data. However, there are two major limitations of these methods. First, they only support the labeling of one object per frame, and they do not work with multiple objects. This limits their usefulness in many situations. Second, they are designed for video sequences that have been captured under laboratory conditions, with clean background and clear illuminations. In later sections, our experiment shows that the DeepLabCut [65] method does not work well on our cow videos.

1.6.3 Image & video classifications

Image classification has been studied for years. Feature selection is critical when performing image classifications. Spatial image features including color and texture, and robust feature points, are effective for classification problems. For dash camera videos in the farming environment, it is hard to find rich and consistent interest points in the field [68]. The color features such as Color Co-occurrence Matrix (CCM) [69] and Content-Based Image Retrieval (CBIR) [70] are also not effective due to lighting issues. Texture features are more robust. The Local Binary Pattern (LBP) [71] is a commonly used rotation invariant texture feature. It computes the histogram of the gray-level differences between the center pixel and its neighbors. The Gray Level Co-occurrence Matrix (GLCM) feature [72] is used to capture textures in block-based regions. This texture feature collects the histogram of all possible gray pixel level pairs and computes the variation statistics of the histogram. Apart from traditional image features, recently, Convolutional Neural Networks (CNN) [73] show much better performance. They train networks on different classification purposes and apply these trained networks as feature extractors. But we show below that image based spatial features are not sufficient for farming activity classification. Also it is not possible to perform CNN classification method on crop presence classification.

In video feature extraction, there are many sampling methods to select interesting feature positions, such as the cuboid detector [74], Space-Time Interest Points (STIP) and dense trajectories (DT) [75]. STIP is designed to select points from a 3D volume by computing the gradient matrix of each position and thresholding the trace and determinant. Trajectory-based feature extraction methods were developed for studying on human action recognition, and each motion trajectory contains a series of feature points over time. In [75], interest points are traced based on Kanade Lucas Tomasi (KLT) trackers or dense optical flow and then merged as trajectories. But not all the features generated by such methods are useful, so methods have been developed to reject inefficient positions. Page rank and visual similarity graphs were designed in [76] to prune static features by finding regions of interest. In [77], the writer applies the MPEG motion estimation as descriptors to efficient features from predicted motion regions. Then feature descriptors are extracted along the neighbor pixels through every trajectory. In [78], the Improved Dense Trajectory (IDT) method is proposed with a trajectory sampling method that incorporates human detection and homography estimation. Detection of humans is applied to remove feature points that may confuse the homography estimation, that is used to reject those trajectories that are caused by camera motion. The human detection is removing feature points for homography estimation and homography is used to reject the trajectories caused by camera motions.

In video classification problems, local feature descriptors are used to encode information in 3D video volumes. Some widely-used descriptors are Histogram of Oriented Gradients (HOG) [79], Histogram of Optical Flow (HOF) [80], Motion Boundary Histograms (MBH) [81], and 3D HOG [82]. These descriptors are widely used in activity recognition and classification. HOG and HOF are directly extracted from video frames and dense optical flow, but feature descriptors like MBH and Histograms of Motion Gradients (HMG) [83] are computed based on either spatial gradient or temporal gradient of optical flow. Such gradient-based descriptors are more robust to camera motions.

Farming applications require a system that can automatically classify videos into different categories or contexts. Traditional image and video classification algorithms extract features from the data and train classifiers with ground truth labels. Their performance largely depends on where the features are sampled and the corresponding feature descriptors. Among different feature sampling methods, key points such as SIFT [84], SURF [85] and ORB [86] are most widely used. In videos, features are sampled from 3D points such as Space-Time Interest Points [87] or 3D cuboids [74]. A trajectory-based [78] feature sampling method can be more effective to capture the motions in video streams.

Feature descriptors are generated based on the sampled positions. Popular image feature descriptors include color features such as Color Co-occurrence Matrix (CCM) [69] and Content-Based Image Retrieval (CBIR) [70], and texture features such as Local

Binary Pattern (LBP) [71], the Gray Level Co-occurrence Matrix (GLCM) [72], and Histogram of Gradient (HOG) [79]. Video feature descriptors such as Histogram of Optical Flow (HOF) [80] and Motion Boundary Histograms (MBH) [81] are mainly extracted from 3D volumes.

Recently, CNNs are widely applied to image and video classification, such as the PlacesCNN [88] for image scene classification, and [89] and [90] for video classification. However, most of the learning-based classification algorithms are not trained on farming-related data, and it is hard to find large-scale public datasets of farming images or videos. Moreover, these methods require significant computational power which is hard to applied in practical farming vehicles.

1.6.4 Classification models

Video classification methods include standard methods and approaches using CNN. Standard classification methods normally include three steps: feature extraction, feature encoding and classifier training, such as [76, 78, 91–94]. Widely-used encoding methods include Fisher encoding [95] and Vector of Locally Aggregated Descriptors (VLAD) [96], and the one-vs-rest Support Vector Machine (SVM) is one of the most commonly-used classifiers. Recently, more CNN-based approaches have appeared, such as the two-stream architecture [97] and [89] with its fovea stream and context stream. The two-stream architecture is proposed with two separated nets with one for image frames and another for motion optical flows. [89] trained CNN with fovea stream and context stream and tried on the UCF-101 dataset [98].

CNNs are powerful models that have been applied to solve a variety of tasks, such as image spatial segmentation, video classification, and even a large complicated autonomous system [14]. Agricultural applications [99, 100] also apply CNNs for various purposes such as plant detection. However, our proposed method is not related to CNNs. First, the hardware constraint limits the computational power of this application, so it is not suitable to use a CNN. More importantly, the visual differences between our target regions can be easily characterized using simple classifiers. Therefore, there is no reason to apply more powerful methods, and the experiment in Section 3.1 shows that our hand-crafted features achieve better segmentation results. Also, using models like CNNs are more likely to cause over-fitting problems, especially when the training data is not enough. In Section 3.1, we train a light-weighted classifier with simple video features which can be easily incorporated into practical farming applications.

1.6.5 Related public datasets

There are many video classification datasets available online. Human action recognition is a widely-studied topic and some related public datasets are UCF-101 [98], HMDB [101] HMDB51 [102], Hollywood [103] and Sport-1M [89]. The Sport-1M dataset is the largest; it includes one million YouTube video clips with 487 different sports categories. However, there are no farming-related outdoor activity video datasets available. In the image classification field, there are some datasets for scene classification purposes. The SUN dataset [104] includes 899 different categories including indoor, urban and natural scenes. The Places dataset [73] is a much larger scene-centric dataset, and the Places-CNN [73] is trained on this for scene classification. Both datasets provide some farming-related categories, but their categories are not specific enough to separate our video frames into different farming activities.

Apart from image and video processing topics, we also introduce some previous works on autonomous vehicles. Autonomous driving systems such as [15] and [14] apply video-based approaches for steering control. For farming vehicles, video has mostly been applied for vehicle alignment automation. [2] develops a video-based algorithm to detect the lateral cutting edges on corn chopping. [105] applies motion analysis to adjust the tractor steering in the corn field. In addition to the corn field, the Hough transform is used in [106] to estimate the best chopping route. Besides videos, [107] reviews autonomous farming applications for tractors including navigation and steering control. But for studies related to the header of combine such as [108–110], they only focus on keeping the header at a constant height relative to the ground. They do not predict when the header height should be adjusted.

2. VIDEO CAPTURING SYSTEM

Gathering video data from practical agricultural applications is critical. This chapter introduces the camera capturing systems used for two major projects: the farming vehicle automation and the dairy cow health analysis. We start by introducing several versions of our developed video logging system on farming machines in Section 2.1, followed by a general processing procedure for a video-based autonomous applications in Section 2.2. Next, we present the camera capturing system installed in the dairy farm in Section 2.3.

2.1 Camera system on farming vehicles

Video provides large amount visual information for automation video system. However, there are no camera systems which are specifically designed to capture videos in agriculture applications. In previous years, we used normal dash cameras to capture videos and transmit data with normal SD cards. This is a slow and inefficient process. Therefore, we develop new video capturing systems installed inside the cockpit of farming vehicles. In this section, we first talk about the limitation of video capturing process and then introduce two developed new video capturing systems.

2.1.1 Limitation on previous video capturing process

All the video data used in this work are collected by our research group and to the best of our knowledge, there are very few public video datasets captured from commercial farming vehicles. We are working with a farmer who owns a farm with corn, wheat and soybeans. In the farm, the commonly used machines include tractor, combine harvester, corn chopper, sprayer and planter. The normal farming activities include planting, spraying, tillage, harvesting and bailing. We target to capture all these farming activities happened in the farm. Figures in appendix A show the images of different farming activities we collected from the farm.

The purpose of placing a camera in a farming vehicle is to monitor the surrounding environment. And the captured videos provide enough information to develop automation applications. However, we need to consider where we should place our camera inside the cockpit, in other words, what should be captured by the camera. From the operator's point of view, the front field is the most critical region to monitor. The front view provides the most direct information to make decisions on steering control or header control. But one single dash camera can only capture a part of the front region even with wide capturing angle. Apart from the front view, it is also important to observe both the left side and right side of the vehicle. For example, on a harvesting machine, the operator should synchronize the harvester vehicle with the truck on the side, for accurately dumping the crops into the truck. Besides, there are other regions that need to be monitored from the cockpit, such as the control display. But we are limited by the number of cameras and their powering systems. To select the most useful regions from the cockpit, one possible solution is to capture what the operator sees during the operation process.

Most of the cameras we used in the farming vehicles are dash cameras, including Garmin Dashcam 10, Garmin Dashcam 45, Mobius ActionCam, and the latest IP camera Ubiquiti G3 bullet. Some cameras have displays which are easier to control the capturing angle. We also use first-person cameras to capture the views of the operator. However, unlike dash cameras which can capture videos with connected power supply, first-person cameras are limited by their battery capacity. The latest nose camera we have, the IVUE Rincon, has battery capacity which only supports two-hour capturing time. All these dash cameras and nose cameras are capable of capturing 1080p videos at 30-frame per second, but each camera has a limited amount of storage.

The current video collecting process is complicated. Because of the physical distance of the farm, we are not able to adjust the camera system constantly. In this case, we ask farmers to mount the dash cameras inside the cockpit of the vehicles and they return the cameras back to us to copy the videos once a week. There are several limitations in this process. Firstly, the farmer needs to mount and control the cameras every time, and the videos are captured at different angles. Secondly, the video collecting efficiency is very low because of the limitation of camera storage.

2.1.2 Wireless video capturing system

Our first proposed video capturing system is designed for video logging using WIFI transmission. In the system, we use three dash cameras and one nose camera. We mount two dash cameras on the front window, with one pointing to the left side of the vehicle and another to the right. Another dash camera is mounted on the back widow, and it is pointing inside of the cockpit for capturing the operator's actions. We also ask the operator to wear a nose camera to capture what he or she sees during the operation. All the cameras have built-in WIFI modules, which means that they can broadcast their own local wireless network. A laptop is also placed in the vehicle to automatically download the videos that captured from the four cameras.

The detailed system processing procedures are introduced as follows. Firstly, all dash cameras and the laptop will be turned on automatically when vehicle engine starts, and each camera will broadcast its own WIFI network. Then a Python script starts to run automatically when the laptop is turned on. The script controls the system to connect the wireless networks generated by every camera. When the laptop and a camera is connected through WIFI, the script will search the camera storage and download the latest captured videos. The script switches the wireless connections repeatedly between all cameras and downloads the videos from every camera. When the dash camera storage is full, the camera will overwrite the previous captured videos which are already stored on the laptop. Therefore, we can capture the videos from all four cameras, and these videos can be synchronized by their captured time.

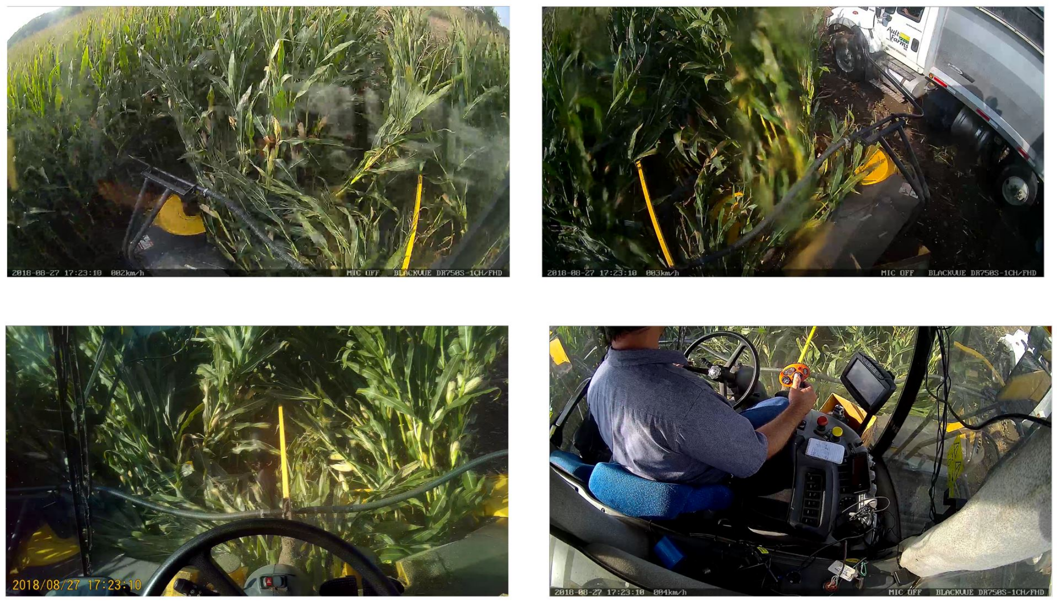


Figure 2.1.: Different video angles from the camera system installed in a corn chopper. The upper two images show the sample images from the two dash cameras on the left and right side. The bottom left image is captured by the nose camera. The bottom right camera is placed at the back window to capture the actions of the operator.

Figure 2.1 shows some example frames captured by four different cameras in a corn chopper. All four images are captured at the same time. The top two cameras are pointing to the left and right side of the chopper, and we can see there is a truck of the right side, which collects the harvested corns. There are some common regions between the left and right images, for example the straight yellow bar in the middle, which can be used to analyze the 3D stereo information. The bottom left image is selected from the first-person video captured by the nose camera. The first-person video sequence has very large global motions. At this moment shown in Figure 2.1, the operator is looking at the bottom, and the image frame also captures the yellow bar which is also shown on the two dash cameras. Then the viewing angle of the operator can also be estimated based on the common object. The bottom right image is captured by the third dash camera. This camera is capturing the inside cockpit and we can observe the motion on the joystick controlled by the operator.

Here we explain some details and rationales of this system design. Firstly, the goal of the system is to capture every possible angle from the cockpit, but we only apply three dash cameras to capture the important regions, because the number of dash cameras is limited by the power supply. The cameras are charged from the cigarette lighter in the vehicle and we also need to charge the laptop at the same time. There is an upper limit for power consumption, so we only choose three most critical regions to capture. To improve this, we can replace the laptop by some smaller processing systems with lower power consumption. Secondly, we placed two cameras at front because we want to capture the wide angle of the operator's views. The images from two side cameras provide the spatial positions for the front region by applying stereo analysis. Thirdly, the position of the dash camera that points inside the cockpit is not fixed. The goal of this camera is to capture how the operator controls the machine, including the operator's hand position, steering control, and button pressing. But in different machines, the control handle or joystick are not at the same position. Adjusting this dash camera to the proper position is necessary.

This design solves some video collecting issues we encountered in the previous collecting process. In this new design, farmers are not intervened by the manually control of the video capturing system. The dash cameras start to capture once the engine is started and videos will be transmitted automatically. In addition, the cameras are mounted at the same position without removing, so the capturing angles can be fixed. Furthermore, the storage is largely increased since the laptop storage is extendable comparing to fixed camera storage.

2.1.3 IP camera capturing system (ISOBlueHD)

We also design and build another video logging system which also works on the farming vehicles. Unlike the wireless video logging system, this one applies Internet Protocol (IP) cameras which powered by Ethernet cables. Figure 2.2 shows the system pipeline and figure 2.3 presents the actual developed box. This system is controlled by an ISOBlue board [111], which is programmed to log data from different sensors from the vehicle, including Controller Area Network (CAN bus), GPS, and videos. The board is directly connected to the machine battery and controls on and off for all the sensors. As mentioned before, the cameras are powered through Ethernet through a Power Over Ethernet (POE) switch, and the switch is controlled with a relay board performs like an automatic on and off button. During operation, the board send a signal to the relay board to turn on all the switch and all the cameras. Then we run FFMPEG software on the board to record videos from all cameras simultaneously, and all the videos are saved directly to a hard drive connected to the board. While the system needs to be shut down, the board will send a signal to turn off the relay board, together with the switch and all the cameras.

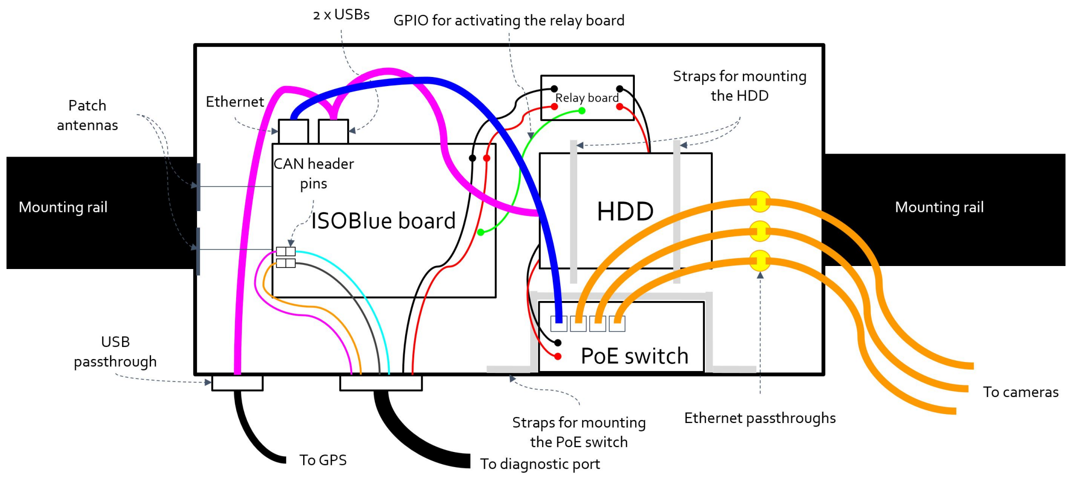


Figure 2.2.: The system of the IP camera capturing system.

One advantage of this system is that all the collected data can be easily synchronized. Synchronization between all sensors is important if we want to merge different sensor data together, and this board uses the system time stamp to label all the signals. For example, all the videos are captured with a maximum length of 10 minutes, and the name of the video includes the system time on the board. Based on the video starting time and the frame time index, the frames can be synchronized with GPS and CAN data,

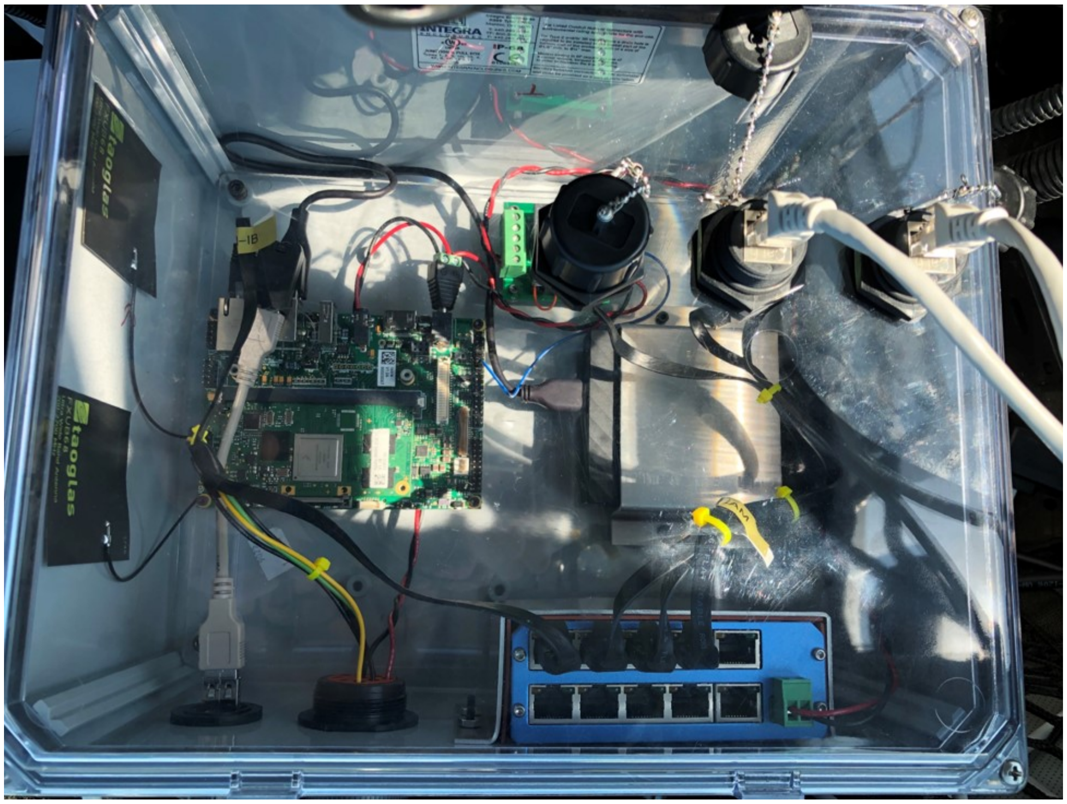


Figure 2.3.: The actual ISOBlue HD system box.

which provides multiple information of the machine. The merged data can be used for higher-level event and activity detection for the farming vehicle.

2.1.4 Total video collection

The wireless video capturing system was applied in 2018, and it captured more than 200 hours of data, which is more than the sum of videos we captured before the that years. The videos from the past several years are mostly captured by a single dash camera. The videos are collected by the multi-camera system and these videos are synchronized by the captured time. In total, we have collected around more than 300 hours of videos data across different farming activities, which is around 2 TB of storage in total. Figures in Appendix A show some example images we collected on different farming activities.

Due to the large number of videos, the pre-processing of these raw videos needs a large amount of work. For example, not all the captured videos are useful, in this case some videos are captured when the vehicle is not working, or when the camera accidentally falls from the mounted position. Selecting the informative videos is normally the first step. In addition, the raw video sequences have different lengths, ranging from 1 minute to 15 minutes depending on the camera types and settings, which means editing the raw videos is necessary for further processing.

The IP camera system (ISOBlueHD) was deployed in the wheat harvesting season in July 2019. In total, we captured around 100 hours of videos together with the GPS and CAN bus data available. These videos mainly cover left and right side of the vehicle, and part of the human operator. The synchronized data are further processed to detect abnormal events happened on a combine harvester, such as unusual reversing, and harvesting on wheat which is blown down by wind. These detection results are beyond the scope of this thesis.

2.1.5 Other possible applications

Beside farming vehicles, our first proposed wireless capturing system can be applied to many other applications. For example, our system can be applied in the surveillance system. Old fashioned surveillance cameras are limited by the number of ports to the Digital Video Recorder (DVR) and its storage. In addition, the placement of surveillance cameras is limited by the wires. There are some latest wireless camera systems, but they are not designed for research purposes and downloading the video data from their systems is complicated. For commercial surveillance camera systems, we are limited to control the camera settings or capture quality. In our proposed system, we can control every camera in the system and the video transmission process. In addition, the videos can be directly downloaded to the processing server which is used to do further video analysis. As a result, it is possible to apply real-time analysis based on this wireless camera system.

2.2 General procedures of developing a video-based autonomous application

The target of the automation system is to learn from the operator, and the system can make the same actions as the operators do after training. There are some common steps for designing an automation system based on video processing.

(a) Define the target autonomous farming activity.

(b) Search cues/rules to make decision.

(c) If learning from operator, also record the operator's actions. If learning from rules, label the input information.

(d) Extract information from source video data.

(e) Video data analysis to generate cues.

(f) Train the automation system to make decision.

Clearly defining an automation problem is critical and we also need to know the rules why the operator performs such operation. The cameras are used to monitor the cues and the captured videos can be further processed as indicators to make decisions. For example, for header control of a combine harvester, the rule is that when there are no crops in front, the header should be lifted. So we use the dash cameras to capture the front field region and analyze the field.

The key processes in the general steps are the video data analysis and automation system training. Cameras can capture a large amount of information, but normally the cues need to be extracted by image processing techniques such as segmentation and feature extraction. Extracting useful and effective cues from the video data is the key for training the automation system.

We use the current multi-camera systems above as an example. Figure 2.4 presents the data analysis process and the automation system training process. As shown in the figure, the boxes at the bottom show the flow of an operator that is driving a machine. The operator takes the input of visual information and performs the corresponding actions. We assume the performed actions from the operator are correct. Our target is to

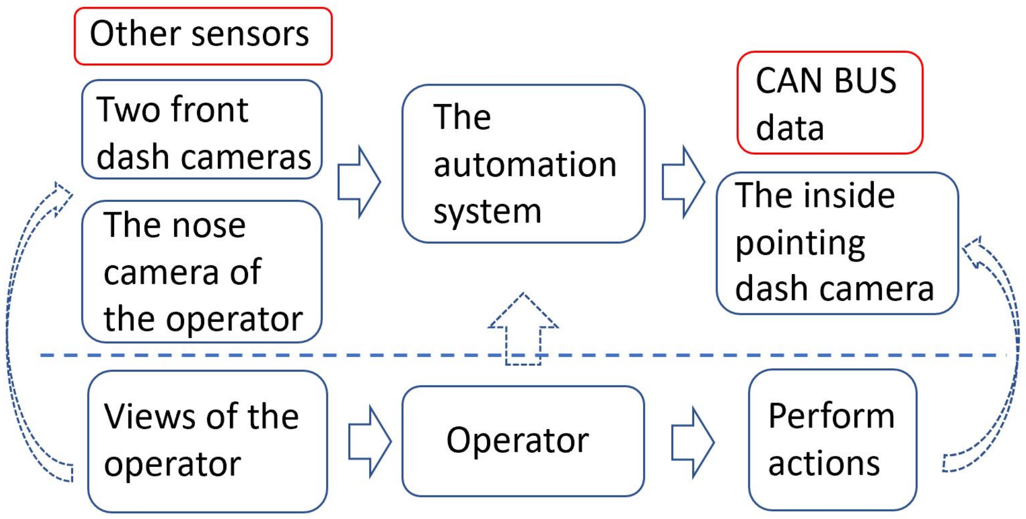


Figure 2.4.: The flow chart of an autonomous farming vehicle application based on the current multi-camera system.

learn the automation system in the upper half of the flow chart. The camera system described above is designed to provide us the videos. The videos captured from the front of the vehicle can be considered as the input of the automation system. The video captured from the nose camera highlights the region that is currently observed by the operator. This tells the analysis system which spatial region should be focused. Apart from the input, the actions of operators are also captured, which can be assumed as the output of the system. With both input and output data, we can train the automation system and produce the same action as the operator does. To extend the video-based system, adding other sensors to the input helps with the training process, for example the GPS data. In addition, the Controller Area Network (CAN) bus data are useful at the output side because it presents the actual situations of the machine.

2.3 Camera system in dairy farm

This section introduces the camera capturing system applied in the dairy cow farm. We first describe the camera system and then explain the designed camera positions and their capturing target.

2.3.1 The camera system

Unlike the camera system on autonomous farming vehicles, this system is applied as the surveillance camera system with fixed power supply. This system applies the IP cameras and a POE switch for video capturing. Figure 2.5 shows an deployed camera system applied in the cow farm. All the videos are captured using FFMPEG software running on the board, which directly records the videos from the cameras simultaneously. There are four cameras applied in the system, and all the recorded videos are directly saved to a hard disk connected to the board. This system is always on since it is connected to the fixed power supply, but the cameras are turned on during the time when the cows appear in the camera.

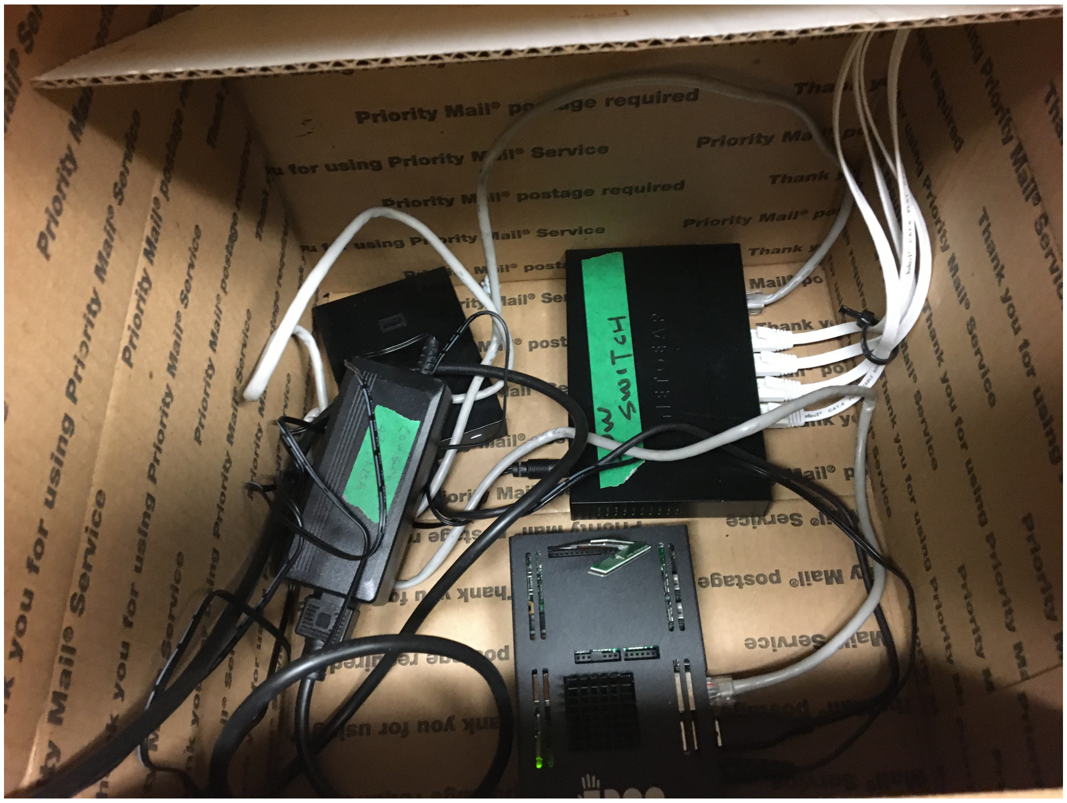


Figure 2.5.: The deployed cow camera system box.

The commercial dairy farm we deployed our system has a central walking path where the cows are getting milked twice a day. The central path is bounded by fences which only allow one cow to walk through at a time. Based on the capturing target, we installed four surveillance cameras at four different angles to capture every angle of the walking path. Figure 2.6, 2.7, 2.8 show some sample frames from four different angle based on two different camera types. Figure 2.6 is captured based on a low quality Digital Video Recorder (DVR), which provides very low quality videos. The other two examples are captured using the IP cameras, which show better quality videos with higher resolution and wider viewing angle.

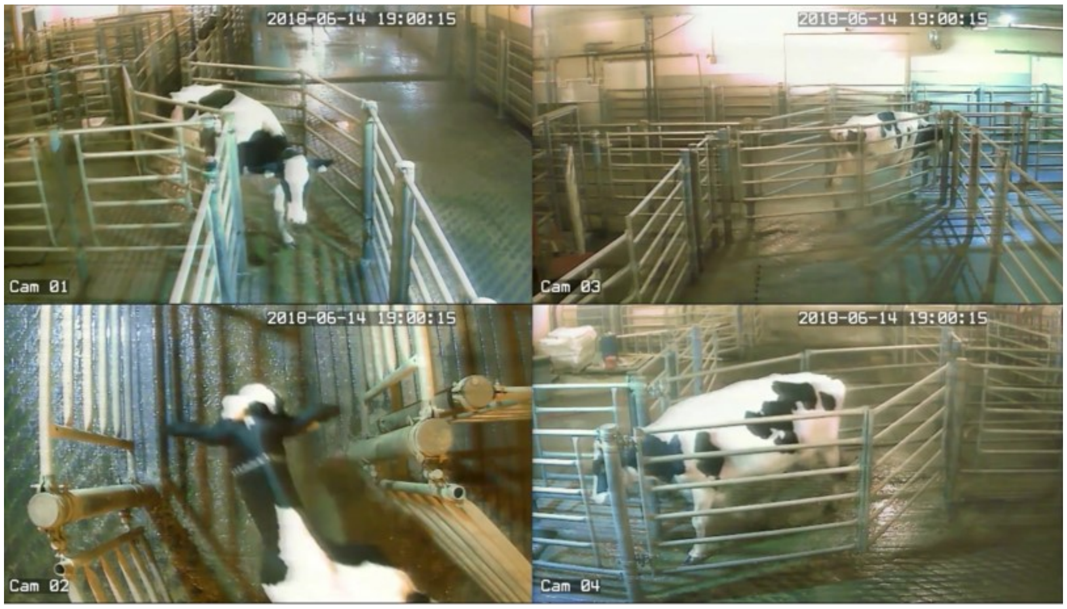


Figure 2.6.: Example frames 1 of the cow cameras from four angles.

For each figure presented above, we call the top left angle front-view, the bottom left angle top-view, and the right two views side-views. Each viewing angle captures a different side of the cow and each view has its best capturing timing. The target information we want to extract and process from each angle varies, and each angle encounters a different level of challenges. In addition, the best processing timing also varies among four different angles. As a result, in further processing, each viewing angle is processed individually, and the corresponding methods are presented in later chapters.

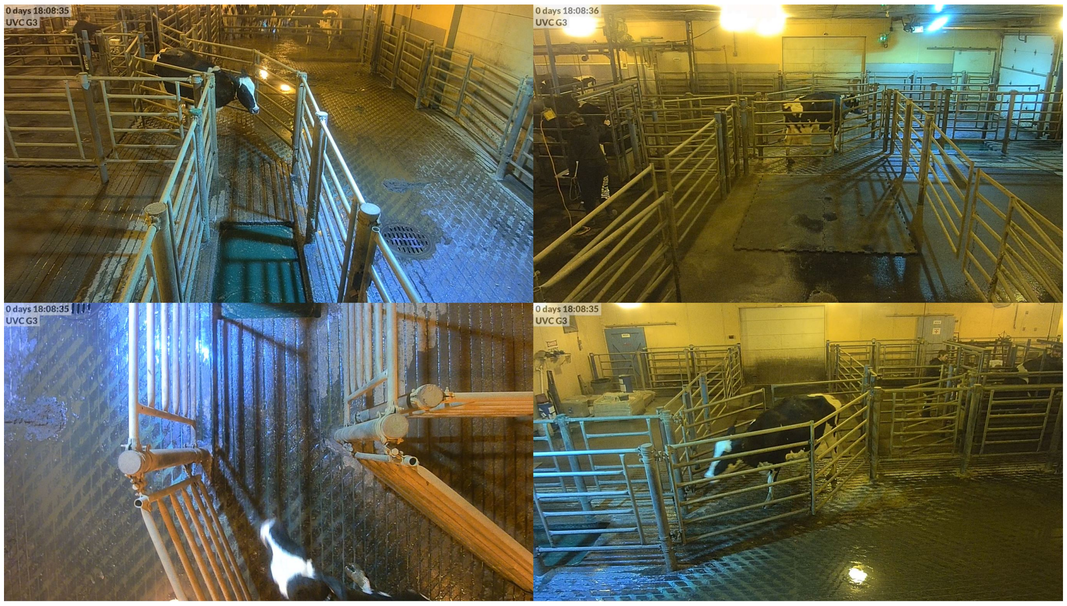


Figure 2.7.: Example frames 2 of the cow cameras from four angles.

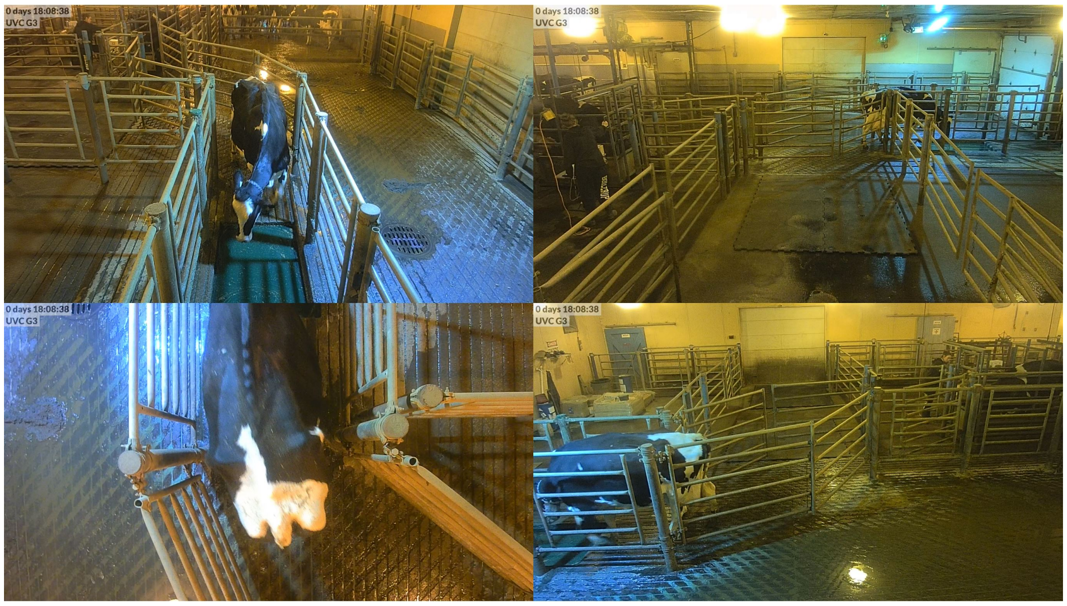


Figure 2.8.: Example frames 3 of the cow cameras from four angles.

2.3.2 Cow video collection

Two major camera systems are applied in the dairy cow farm, and all the videos are captured during the milking process. In total, we collected more than 400 hours DVR videos and more than 300 hours IP camera videos. Among the IP camera videos, there are around 100 hours videos which are captured during the weighing process. This is a periodic operation in the dairy farm, which normally happens once a month. During this process, the farmers do a health check on every cow individually, including measuring weight and ultrasound examination. The cow's health results are recorded by the cow IDs and saved in our dataset. We collected four different weighting sections, and each section includes around 110 to 130 cows. Currently, the records include more than 170 cow identities from all four different days collected in total.

3. APPLICATIONS ON FARMING VIDEOS

This chapter introduces the spatial segmentation process applied in large farming machines. We first present three different versions of our proposed spatial segmentation method for farming machines in Section 3.1. Next, we introduce two applications which apply the spatial segmentation as a part of their systems. The first application in Section 3.3 is related to farming video classification, and the spatial segmentation module helps to extract the most distinguishable features which largely improves the classification accuracy and feature efficiency. The second application in Section 3.4 is related to the header control of a combine harvester. The segmentation process locates the front field region which sets the region of interest for further processing modules.

3.1 Spatial segmentation of farming videos

This section introduces the background and our provided methods for spatial segmentation in farming machines. Section 3.1.1 talks about the background and motivation of spatial segmentation in the farming scenario. Next, three proposed spatial segmentation methods are explained in detail.

3.1.1 Background

Video-based applications require to know the region of interests in order to perform further processes. The goal of spatial segmentation here is to identify every region captured in all the farming videos. Our video data are collected from multiple farms in the US between 2016 and 2018. For the purpose of control automation, the cameras are placed by farmers inside the cockpit of large agriculture vehicles to capture a view similar to that observed by a human operator. Notice that we refer to these captured videos

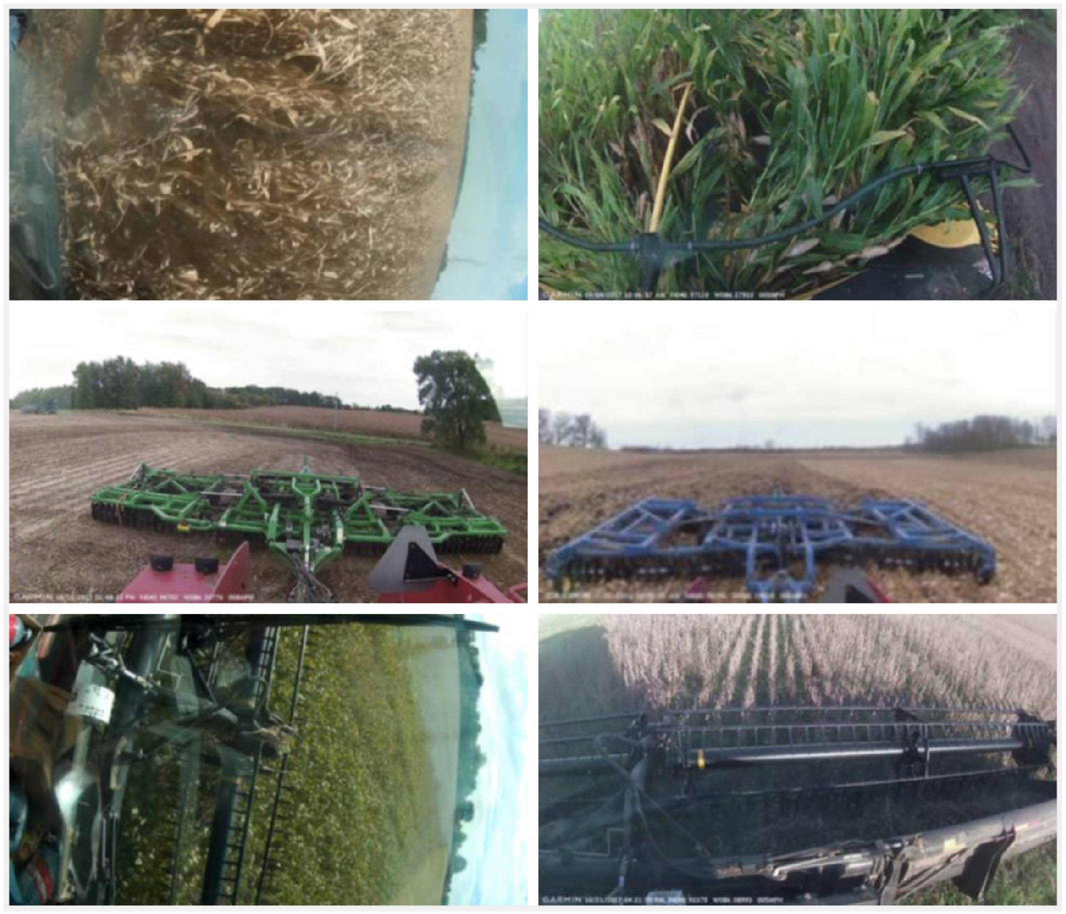


Figure 3.1.: Sample video frames captured by dash cameras mounted on different farming machines: chopping corn, harvesting, and tillage.

as farming videos for the rest of this paper. Typical dashboard cameras (dash cams) are used for capture because they can be easily mounted on the windows of the farming machines, and the cameras are always pointed towards either the front side or back side of the vehicles based on the farming activities. Figure 3.1 shows some example frames.

In our farming videos, there are normally three common regions captured from the cockpit. The first important region is the header or attachment region, which is normally connected to the front side. Different farming activities use various types of attachments. For example, the first row of Figure 3.1 shows two different attachments used for corn chopping, although the attachment is blocked by the corn on the left image. The second important region is the upcoming field in front of the vehicle. There are many types of fields and each has a unique appearance. Our collection mainly covers three categories: soybean, wheat and corn. But for the same type of field, the color and shape can be different; see for example, the two types of wheat fields in the second row of Figure 3.1. The third region includes all other parts such as sky or faraway objects that have no large motion. These low-motion regions are crucial in multi-camera systems because the feature points in this region can be used to connect different cameras. As a result, our target is to segment the farming videos into these three regions.

This farming video segmentation problem is different than the typical video segmentation such as [34, 35, 40, 41]. First, there are no particular objects to detect in farming videos, and our target is instead to divide the frame into different regions. In object detection, the appearance of the target (like shape and color), do not dramatically change overtime. But for region detection, the content inside the region is not determined, and it also changes across time. Second, since the cameras are manually mounted by farmers, the capturing angles and scene structures of the videos are not controllable, shown in Figure 3.1. This requires the segmentation method to be robust to all these challenges caused during capturing; for example, it should be able to quickly adapt to new viewing angles. Third, there are many practical constraints like time and hardware. The segmentation should process videos as quickly as possible because it needs to save time for further analysis, especially for real-time applications. It is also not practical to install powerful machines on farming vehicles, which limits the computational power of the method. Currently, we have not seen any related applications which address all the issues listed above. But the methods provided in the next three subsections are all targeting to separate either the field or the header region in the frame.

Spatial segmentation in agriculture videos is challenging. As mentioned in Section 1.3, the unconstrained capturing environment, various video motions, and the limited amount of data are three general problems for processing videos of different agricultural activities [68]. Color-based segmentation methods rely heavily on lighting conditions in the outdoor environment. As a result, they are not effective when shadows or window reflections are present. Motion information is more robust than color in visual agricultural field analysis [68]. For a video captured on a slowly moving vehicle, the spatial structure of frames remains the same in a short time period. Instead of color, [68] uses video motion to select spatial regions of farming videos. For a video captured on a straight moving vehicle, the field region should have the most consistent forward motion. The motions generated from other regions are not consistent across time, especially in the combine harvester video, where the header regions generate chaotic motions and occupy a large area of the frame.

Next, we explain three segmentation methods for farming video segmentation. These methods are developed to locate the three regions we defined: object motion region, camera motion region, and the low motion region.

3.1.2 Method 1: the Active map method

This first segmentation method we proposed, which is also covered in [7], is targeted to locate the header region which contains the object motion. As discussed, object motion is the most critical motion feature for activity separation. Camera motions produce very strong features, yet they are not unique to a particular activity. As a result, the goal is to separate the potential object motion regions from the camera motions. In [78], a homography is estimated for consecutive frames to compensate for the camera motion. But in our agricultural videos, there are no robust feature points for homography estimation because there are no rigid objects in the field. The MBH feature can cancel camera motion by computing a spatial gradient on optical flow, but it also removes useful motions and only leaves motion boundaries. Therefore, in our method, we compute gradient from both the spatial and temporal domain of the optical flow. Then the regions are selected if they have large gradient in either the temporal or spatial domain in a volume of video.

Formally, for a video volume V with frame number T, the dense optical flow is estimated for every frame noted as OF_t where $t \in \{1, 2, ...T\}$. Then we convolve each OF_t with horizontal and vertical gradient kernels to obtain OF_DX_t and OF_DY_t. The temporal derivative OF_DT_t is computed as the difference between consecutive dense optical flows. Note that the shape of OF, OF_DX, OF_DY and OF_DT are all the same as V. With three derivative flow maps, the active pixel positions are selected using:

$$\begin{aligned} Active\_map = OR\big((&OF\_DX > TH_{dx}),\\ &(OF\_DY > TH_{dy}),\\ &(OF\_DT > TH_{dt})\big) \end{aligned} \tag{3.1}$$

where TH_{dx}, TH_{dy} and TH_{dt} are threshold parameters and OR is logical operator OR. The $Active\_map$ is a 3D binary volume. Then object-motion regions are selected by summing $Active\_map$ over the time axis, resulting in a 2D map. Those positions $p(x, y)$ in the 2D map that have values that are larger than $T/2$ are selected as object-motion regions. As a result, the optical flow analysis provides a spatial mask which is used for all T frames in the volume. Finally, a spatial sampling step is added to pick positions between some minimum distance. This prevents large overlapping feature vectors if two sampled positions happen to be too close.

Figure 3.2 shows an example of the sampling result of a harvesting video. The upper two images are the original frame and the computed optical flow map, and the lower two are the sampled results from IDT [78] and our method. Notice that, in this video, the combine is driving forward and in the upper right optical flow figure, the flows above the hand-drawn red line are caused by the forward motions. These camera motions are not necessarily related to the harvesting category, but the rotation motions below the red line are distinctly object motions. The bottom left IDT method selects forward motions as features and chooses features from the sky region. This method only selects the object motion at the bottom region, which is more accurate and therefore will be more informative.

3.1.3 Method 2: the two-step segmentation

Different from the previous method, this one is designed to find the field region instead of the header region, which is also presented in [5]. Color is useful for separating

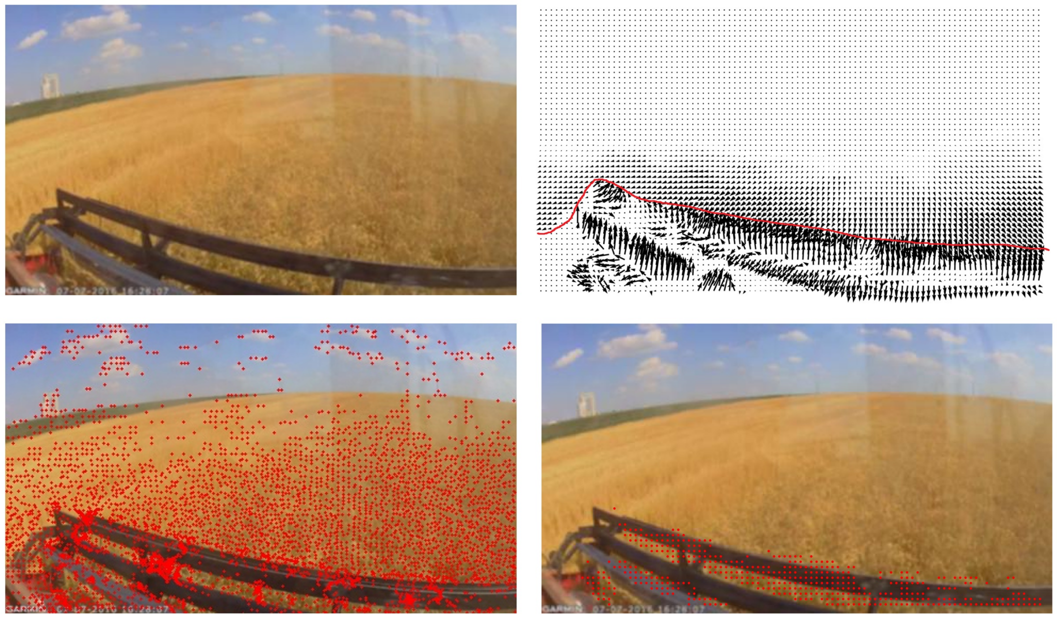


Figure 3.2.: The feature sampling method comparison on a harvesting video. Top left: the original image, top right: the optical flow at this frame, bottom left: the trajectories positions from IDT [78], bottom right: sampled positions using our motion separation method. In the optical flow image, the hand drawn red line partitions the object motions from the camera motions. In two lower images, red points represent the positions of trajectories for feature extraction.

out the field, but it varies in different frames. Inspired by the idea from the previous method, motion analysis could provide a coarse but robust location of the field. Compared to the chaotic rotation in the header region, the motion in the field region is more consistent across time. As a result, we analyze the consistency of motion at every spatial position in the video frame.

The motion consistency measure is based on optical flow analysis. The consistency measure C of a spatial position in a video block is computed by equation (3.2),

$$C = \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}[\frac{du_t}{dt} < \theta] \times \mathbb{1}[\frac{dv_t}{dt} < \theta] \tag{3.2}$$

where T is the length of the video block, (u, v) is the optical flow and θ is minimum motion threshold. This measure is computed based on the assumption that there are no sudden motion changes in a short time period. A small value of the measure represents the case when the motion at this position is rapidly changing, which corresponds to the header region. Large consistency means that this point has low motion changes or no motion at all. An extra motion magnitude thresholding process is added to eliminate the no-motion regions such as the sky.

The above segmentation method only provides a coarse detection of the field, and typically only identifies the closer part of the field. But this incomplete field region helps to pinpoint the specific color distribution of the field in that particular frame. Thus, in the second step, an RGB-based color histogram is generated from the coarsely-segmented field as reference, and we search the rest of field by comparing the local histogram with the reference. Based on the spatial connectivity, we follow a region growth searching method and start the refinement process from the coarsely-segmented mask.

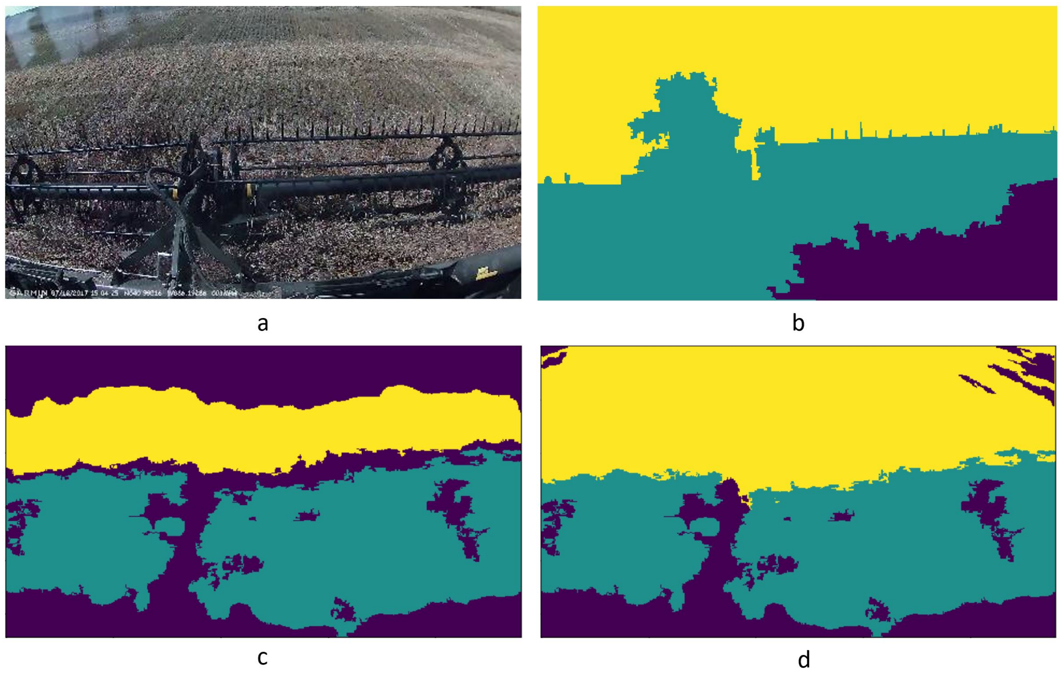


Figure 3.3.: The field segmentation result: yellow region is the target field region, green shows the front reel; a: the original image, b: segmentation result using [34], c: coarse segment result from motion consistency measure, d: segmentation after color refinement.

Figure 3.3 shows an example of the field segmentation result. Notice that Figure 3b using [34] is based purely on color information, but the green region mixes the field and header incorrectly because of the dark illumination. Figure 3c shows the result of coarse segmentation and it can be observed that the faraway field region is not included. After performing color-based refinement shown in Figure 3d, the field region has grown to include the whole field region. Notice the reel region (green) is not the target of this process, so no further refinement is performed.

3.1.4 Method 3: Learning based segmentation

The previous two segmentation methods are applied in two different farming applications, but they both have limitations. First, each method uses hand-crafted features with specifically-tuned threshold parameters, which may not generalize well to other farming applications. Second, each method only detects its desired target regions and ignores all other parts of the frame. However, in the next section, we present a new robust training-based segmentation method that overcomes these two limitations. The new method applies a training-based system which quickly adapts to different applications with a small number of training labels. It also segments all regions at the same time, which makes it applicable to both tasks we studied before. We further show that this proposed method achieves equal or better results compared to our previous methods on both applications.

In order to separate spatial regions of general farming videos, we propose a training-based segmentation method which uses a classifier to separate pixel positions. Every pixel position in the frame is a classification unit and represented by a feature vector. Both color and motion information are used to form the feature vector, but their spatial positions [112] are not included because the region positions of farming videos vary between different capture angles. Note that we still apply color features here because they improve the spatial smoothness of the segmentation as shown later in Figure 3.7. The color features are extracted using the Content-Based Image Retrieval (CBIR) method [70], which is the histogram of the pixels in the HSV color space over a set of pixel values, with 8 bins of illuminance channel and 6 bins of two-color channels. The motion features are hand-crafted with length 14D and are described in detail below. In total, each pixel position has a feature vector of 14D color feature and 14D motion feature.

The motion features are extracted from the magnitudes of the optical flows over a video block. The feature vector includes two parts: a measure of the motion magnitude and the motion temporal consistency. The motion magnitudes are directly characterized by six percentiles ranging from 1% to 90% and the differences between two neighbor percentile values. The motion temporal consistency is summarized by two values: the total sum of motion S_{total} and the sum of Step-By-Step motion S_{sbs}, which are defined in Equation (3.3) and (3.4).

$$S_{total} = \frac{1}{T}\sqrt{\left(\sum_{t=1}^{T} u_t\right)^2 + \left(\sum_{t=1}^{T} v_t\right)^2} \tag{3.3}$$

$$S_{sbs} = \frac{1}{T}\sqrt{\sum_{t=1}^{T}\left(\left(\frac{du_t}{dt}\right)^2 + \left(\frac{dv_t}{dt}\right)^2\right)} \tag{3.4}$$

This total sum of motion S_{total} describes the net movement of a pixel position during a period T, which is similar to the cumulative distance [113] for video blocks. Here the net movement is computed as the sum of optical flows over time, where the periodical rotations are cancelled, and the overall forward motion is preserved. The step-by-step motion sum S_{sbs} records the sum of the motion changes, which is computed as the sum of all flow differences $(du_t/dt, dv_t/dt)$ between neighboring frames. By accumulating the differences, the rotating header region which is ignored in S_{total} can be highlighted. As a result, comparing these two types of motion sums can separate the camera motion and object motion, especially when the camera motions are basically forward or backward. Both consistency values and their difference are normalized and added to the motion feature vector.

Figure 3.4 visualizes these two indicators. The main region in the sum of motion S_{total} identifies the field region and part of the rotating header, and the main region in the step-by-step motion S_{sbs} mainly indicates the rotating header region. Their difference mask shown in the bottom right image highlights the upcoming field region on the left side. Note that there is a smaller and brighter region in middle of the difference mask. It represents the conveyor belt (Figure 3.10), which is also highlighted because the belt motions are also consistent.

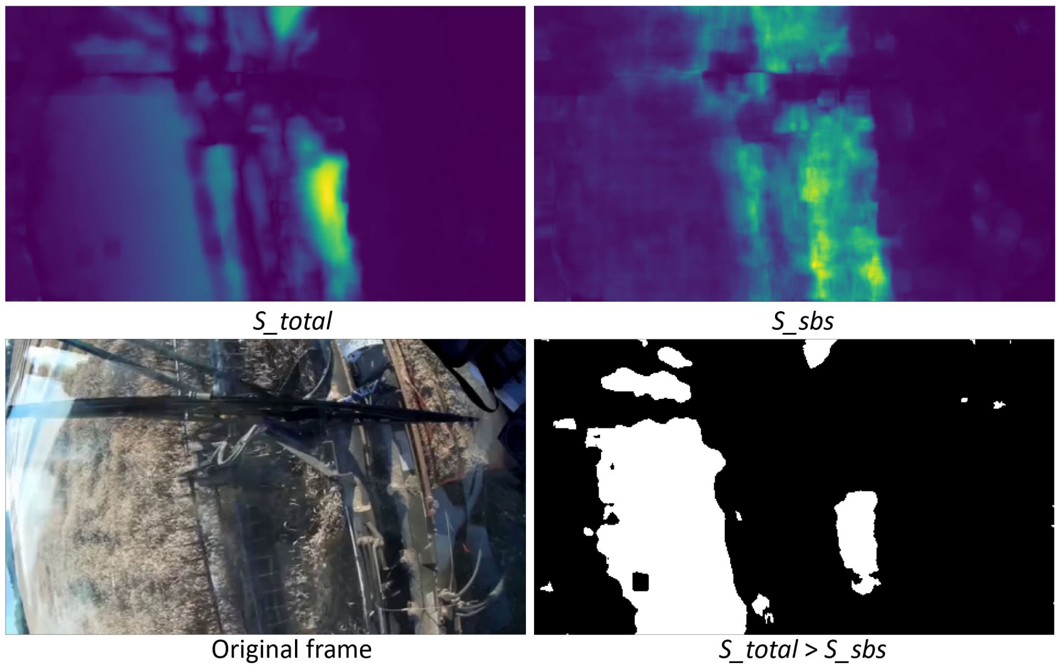


Figure 3.4.: Motion difference measure in farming videos. The white region in the bottom right mask shows where $S_{total} > S_{sbs}$. Notice that the original image is rotated, and the sky is on the left.

Using these features, spatial segmentation can be achieved by classifying all pixels into different regions. Based on the training data, here we use a Random Forest (RF) classifier for two reasons. First, the classification is performed on each pixel position, which means the size of the training data is huge, and a RF classifier is much faster to train than other classifiers such as SVM. Another reason is that our feature vector contains multiple types of information like motions and color, and some features may be more useful. However, most linear classifiers treat all features equally without a preference, while the RF can focus more on the distinguishable parts of the feature vector [114].

3.2 Experiments on spatial segmentation

This section presents the experiments which compare all the segmentation methods mentioned above, including some popular classification methods from the literature. For our training-based segmentation method, we use three different choices of features: color only, motion only and both features concatenated together. Every feature group is individually trained with a RF classifier.

3.2.1 Experiment preparation

This subsection compares different segmentation methods using farming videos and the manually-labelled ground truth. In total, 229 wheat and bean harvesting video clips are prepared in this experiment. These video clips are selected and pre-processed from the video dataset that we collected from the farms. During pre-processing, the raw videos are downsampled to the resolution of 480 × 272 and temporally segmented into video blocks with a length of 30 frames (1 second). The temporal segmentation enables us to assume that the video structure remains the same during a short period of time.

The pre-processed video clips are hand-labelled into the three spatial regions: the upcoming field region, the header (attachment) region, and low motion region (including sky and part of the body of vehicle). Each video is only labelled with one ground truth mask for all frames because the frame structure has little variation. As a result, there is no guarantee that a given pixel position has the same label across the entire second. Because of this, we only label the pixel positions that maintain a single region consistently across time. In other words, not all the pixels in the frame are labeled, but if the mask indicates that a pixel belongs to one category, this position will always belong to that category across the entire clip. As a result, there are some gaps (the black regions in the ground truth label in Figure 3.7) in the labels, and this could influence the calculated segmentation performance as discussed below.

In our experiment, we train three RF classifiers using different features: the color, motion, and both features together. In a real application, the number of ground truth labels for segmentation is limited. So, we choose only one sample video clip from the 229 video clips to be the training data for all three classifiers. Here one clip is sufficient to provide training data for pixel-level classification. In the actual training, 5000 pixel-positions are randomly selected for each segmentation region. Notice all these clips are chosen from several farms, which means the one selected training clip could be similar to some of the testing clips, and we consider this issue when reporting the experiment results.

In this experiment, we compare our proposed methods with 5 different segmentation methods. Since there are no previous segmentation methods which solve a similar problem, here we test our videos on two popular VOS algorithms: Non-Local Consensus Voting (NLCV) [37] and Saliency Aware Video Object Segmentation (SAVOS) [35]. In addition, we compare with our two previous hand-crafted segmentation methods [5, 7]. Note that because the method in [5] is only designed for the header region, in this experiment we extend it to create method [5]* by adding extra thresholds to separate all three regions. Furthermore, we also compare with a CNN-based method Deeplab [50]. Although CNN-based methods are not practical to use for farming application, we still want to explore the potentials of these methods in the future. In this experiment, we fine-tune the last layer of the pre-trained Deeplab [50] network which is denoted by Deeplab [50]*. Here the training data are the same limited ground truth labels used to train the random forest classifiers, so we can obtain a fair comparison. The quantitative comparison results are presented in Table 3.1 and Figure 3.5, and two sets of visual segmentation results are presented in Figure 3.7.

3.2.2 Comparison results

Numerical comparison results

All 228 testing video clips are quantitatively measured with the ground truth masks. The segmentation measures are based on all three regions, together with the overall frame. For each region, the Intersection Over Union (IOU) percentages are computed for all the testing videos, and we report their mean and standard deviation (std) values. Here, the std value is reported because the method is expected to generate robust segmentation on videos from different scenes. Methods with a higher mean percentage and a smaller std value show better and robust performance. We first show the general performance comparison between different methods using a cumulative accuracy curve, and then the detailed numerical measures are explained later.

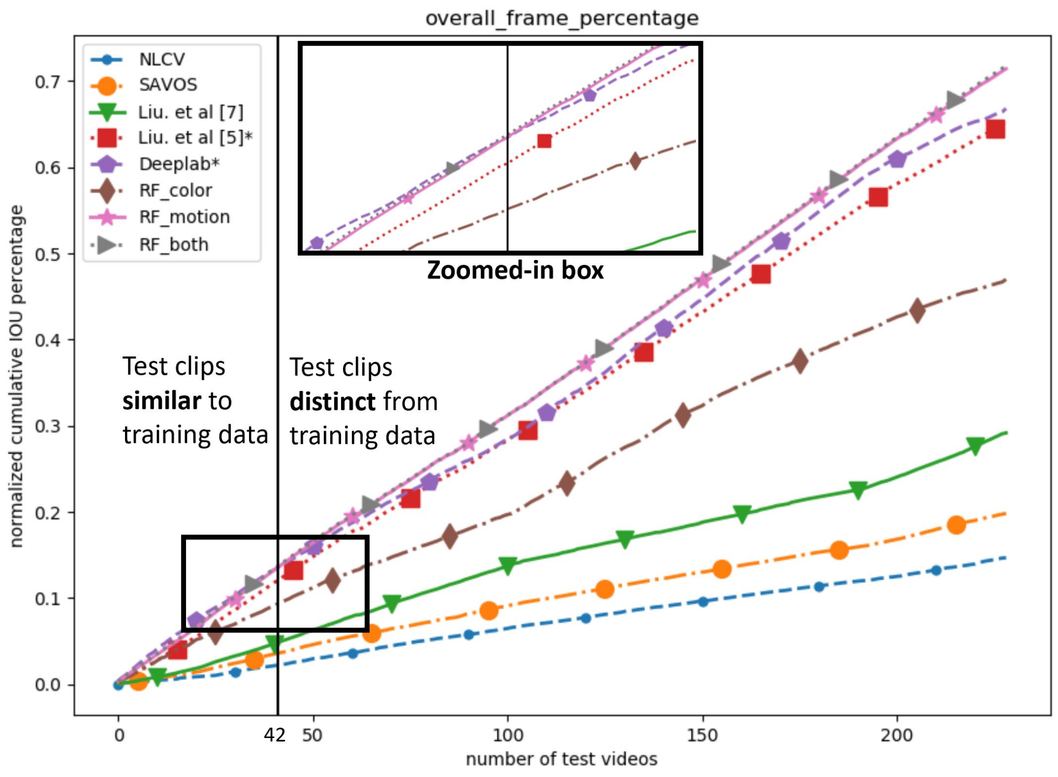


Figure 3.5.: The normalized cumulative IOU percentages between different methods on the overall frame measure. X axis is the number of videos, Y axis is the normalized sum of previous IOU percentages. The vertical line separates the testing clips: the left side shows the clips similar to the training data, and the right side shows testing clips different from the training data. Notice a zoomed-in plot is provided in the black box.

A cumulative IOU percentage plot of the accuracy on the overall frame measure is presented in Figure 3.5. Each line represents the IOU measures of one method, and its detailed mean and std values are shown in the last two rows in Table 3.1. The x axis of Figure 3.5 shows the number of the test videos and y is the normalized sum of all

IOU percentages at this number of testing videos. For example, when x value is 100, the corresponding y value is the sum of all testing videos from 1 to 100. In general, a line that reaches higher y values means this method has better accuracy, and the straightness of a line represents the robustness across different input videos. Notice among all testing videos, there are some clips which are similar to the training data, and we use a black vertical line to separate them: testing clips on the left side (first 42 clips) have similar structure as the training video, while the others do not.

In Figure 3.5, we can see that two proposed random forest methods *RF_motion* (pink) and *RF_both* (gray) are above the other lines and are relatively straighter than the others. Methods like [5]* and Deeplab* are not as robust. Specifically for Deeplab* (purple), applied to the first 42 testing videos, reaches similar performance with other methods, but its performance starts to decrease after the vertical separation line (better viewed in the zoomed-in box) when the testing data have different scene structures.

The detailed numerical measures of all the methods are reported in Table 3.1. Each row shows the IOU of a target region and each column represents a segmentation method. In general, the RF method with both color and motion features achieves better performance than other methods, with both high mean accuracy and stable std values.

Among all the methods, three major comparisons from this table are discussed below. First, considering the mean accuracy, the improved hand-crafted method [5]* using fixed thresholds is slightly better than all training-based methods, but its std value is much larger. This means fixed threshold values can be adjusted to make the average accuracy high, but the performance is not robust when applied to different video inputs. Second, comparing the Deeplab [50]* method with others, we can see it provides reasonable performance for the field region, but fails on other regions. This is mainly because the color of the field is much more similar across different testing videos. In addition, it has a high std, which indicates a lack of robustness, potentially caused by the limited number of training clips. Third, the methods from the last two columns have similar performance: one uses only motion features and the other uses both motion and

Table 3.1.: The segmentation comparison with different methods. The values represent the mean and standard deviation (std) of Intersection Over Union (IOU) percentages. Higher mean percentage and lower std values means a better performance. The bold number indicates the best performance of each row.

Measured regions		NLCV [37]	SAVOS [35]	Liu et al. [7]	Liu et al. [5]*	Deeplab [50]*	RF_ color	RF_ motion	RF_ both
header	mean	0.17	0.379	0.608	**0.617**	0.499	0.375	0.614	0.615
	std	0.126	0.136	0.107	0.133	0.185	0.159	**0.092**	**0.092**
field	mean	N/A	N/A	N/A	**0.646**	0.63	0.278	0.622	0.619
	std	N/A	N/A	N/A	0.28	0.218	0.204	0.209	**0.203**
low_ motion	mean	N/A	N/A	N/A	0.35	0.457	0.313	0.493	**0.511**
	std	N/A	N/A	N/A	0.252	0.289	**0.164**	0.197	0.189
overall_ frame	mean	0.146	0.198	0.291	0.652	0.668	0.649	0.714	**0.717**
	std	0.053	0.051	0.094	0.118	0.161	0.136	0.095	**0.092**

color features. But the std values for the second method is slightly lower. This shows that adding color features has the potential to improve the robustness for segmentation.

Visual comparison

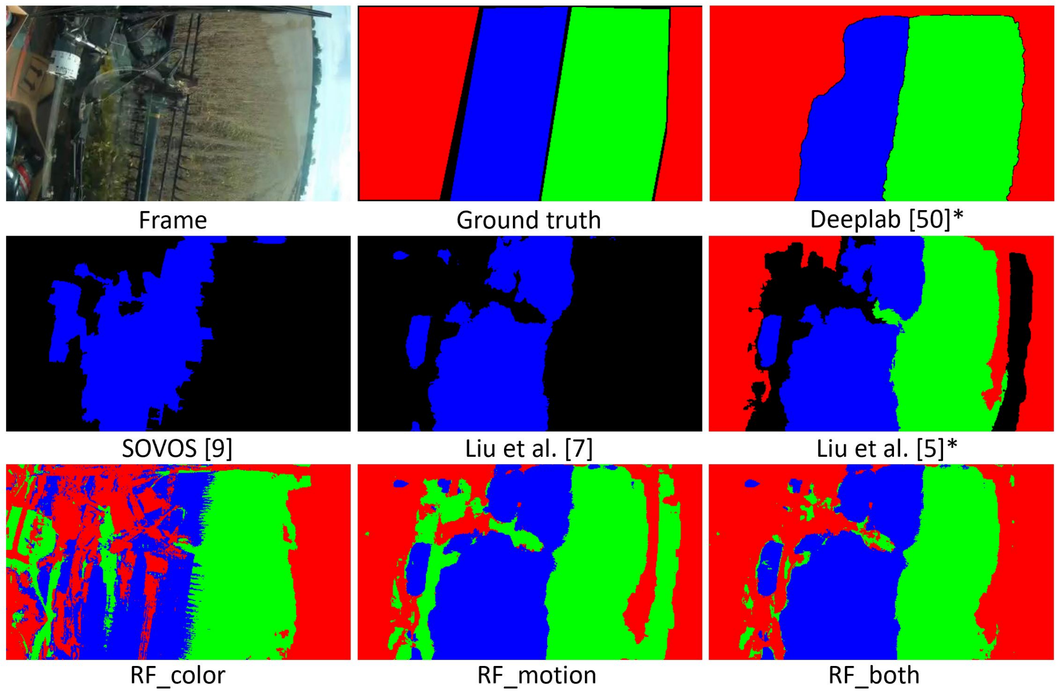


Figure 3.6.: Segmentation results of two examples. Example (a)

In this section, we select two examples to show the visual segmentation results in Figure 3.7, which present some segmentation results from two testing videos. The example clip (a) is captured from the same farming vehicle as the training data, but example (b) is from a different scene. The blue, green, and red regions respectively represent the header (attachment) region, the upcoming field region, and the low-motion region. The black region represents unlabeled gaps in the ground truth or pixels that are not labeled by the algorithm.

In this figure, the SAVOS [35] and [7] only segment the header region, and SAVOS is not accurate enough because of the poor performance of the super-pixel segmentation. Notice the Deeplab [50]* method works well on the left example because this testing

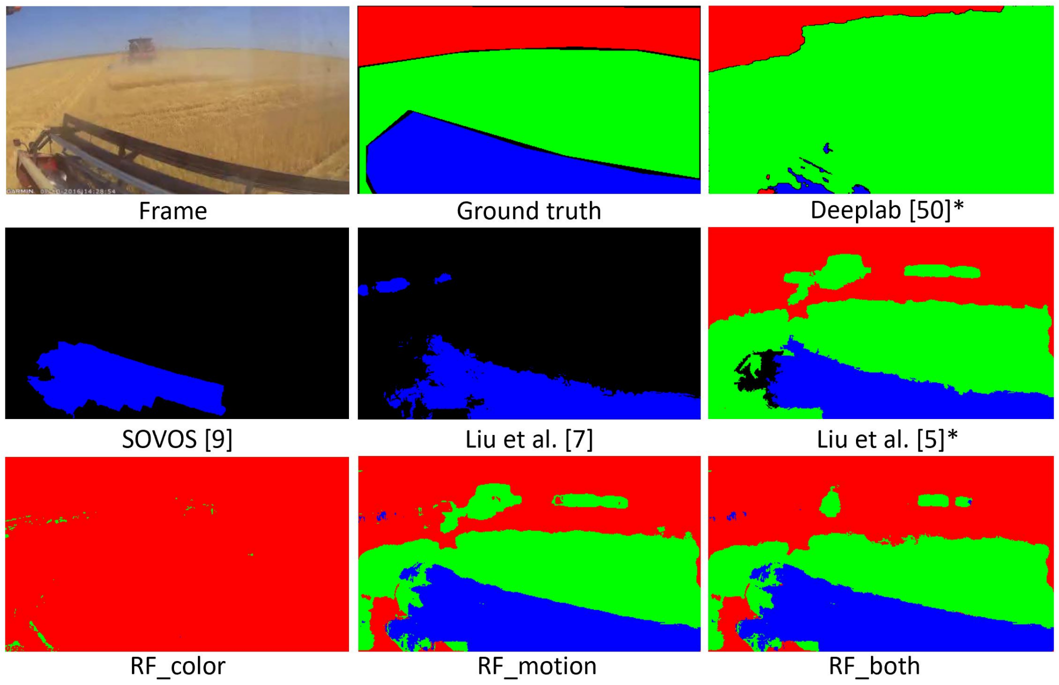


Figure 3.7.: Segmentation results of two examples. Example (b). Two examples have similar structure as the training data, and example (b) is captured from a different farm. The blue region is the header or attachment, green region is the upcoming field region, and red region is low-motion region. The black region is unlabeled.

video is similar to the training data, but it fails on the right one because of the scene difference. This scene change also explains the drop of its curve in Figure 3.5 after video number 40.

Our previous methods [7] and [5] are specifically designed with fixed threshold parameters to locate only the header (blue) region and field (green) region respectively. The improved [5]* can perform multiple-region segmentation based on fixed threshold values. But these fixed values are not robust, so the method misses some regions and has unlabeled black areas for the left video. Considering our proposed RF methods, their performances largely depend on the features they used. Color features over-segment the left clip but fail on the right one because of the color difference. Motion features are more robust in general but adding color features together corrects some miss-classified regions shown in both examples.

3.2.3 Discussion of training-based method

In general, the new training-based segmentation method outperforms all other methods, including two other proposed methods. It has four advantages for farming video segmentation. First, our method is robust to different farming applications. Using color and motion features can achieve segmentation in real-time with machines equipped with normal computational power and storage, such as a single-board computer. Second, with a quick training process, no fixed thresholds are required, and it is capable of handling practical challenges. Third, unlike complicated models such as a CNN, this classifier uses few labels for training, but achieves similar or better segmentation results. In addition, the new classifier segments all three spatial regions in one step, which can be directly applied to applications focusing on different regions. In the next two sections, we introduce two farming applications which use different regions based on the segmentation.

3.3 Application 1: farming video classification

This section introduces one farming application based on the segmentation results. Classifying the farming videos into different categories is important for further processing. However, the domain knowledge helps for better classification, such as the specific regions in farming videos that allow us to select the most distinguishable features. In this section, we first introduce the general classification model which could benefit from the domain knowledge in subsection 3.3.1. Then we explain the motivation and target of this application in subsection 3.3.2, and subsection 3.3.3 shows how to apply the general classification model for farming activities. Next, three experiments are presented. Subsection3.3.4 compares the classification performance of two branches; subsection 3.3.5 compares the improvements using different methods as the second branch, and subsection 3.3.6 compares the feature efficiency in the farming activity classification.

3.3.1 Generalized two-branch classification model

In real applications, video classification methods that use domain knowledge typically perform better than methods that are designed for a general scenario [115, 116]. Based on the domain knowledge, the specific targeted methods can be designed to extract the unique domain features that better distinguish the different categories. Such specifically-designed systems have been developed in different industries, such as sports [117], transportation [118], and medical applications [119]. However, these unique characteristics are not always applicable to all input data, and there are always exceptions. For example, the object motions of farming videos are useful to recognize farming activities, but some raw videos may be static or have little object motion. As a result, classifying videos with only the specific domain features is not enough.

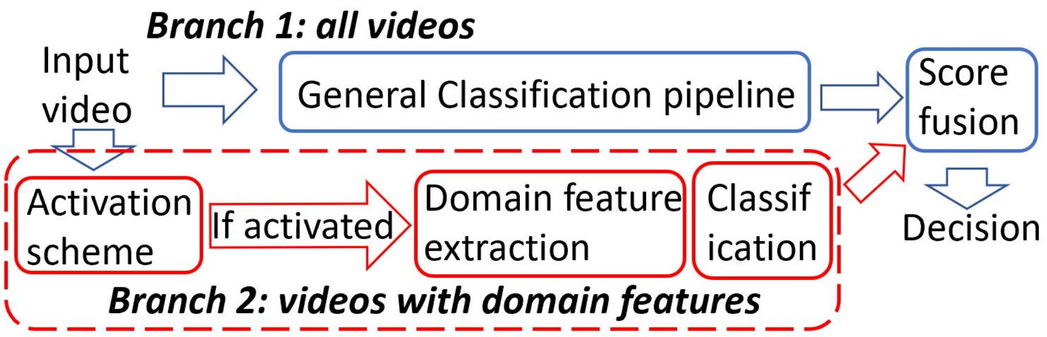


Figure 3.8.: The generalized two-branch video classification pipeline. Note that when applied to farming video classification, the activation scheme is based on video segmentation.

In [7], we proposed a two-branch classification pipeline, which applies two specific feature sampling strategies on different videos. In this paper, we expand this framework to a generalized two-branch classification pipeline, which is shown in Figure 3.8. The upper branch processes all input videos, but unlike our previous design, this branch can incorporate any general classification method. The second branch (shown in the red dashed box) selects those videos with unique domain-related features and processes them with specific methods. There is an activation scheme of the second branch to decide whether the input video has characteristics that will lead to better classification with the specifically-designed classifier (Branch 2). This scheme can be determined by different indicators based on domain knowledge, such as amount of object motion in farming video classification. Finally, the results from the general and specific classifiers are merged to improve the overall accuracy. Different score fusion methods can be applied to form the final decision. Depending on the reliability of the classifiers, simple fusion methods such as direct averaging or weighted sum can be effective. It is also possible to train a classifier that learns the weights between multiple branches [120].

3.3.2 Motivation of farming videos classification

Towards the goal of automating farm vehicles, cameras like dashcam, are placed on big agricultural machines to sense the environment. There are different types of large moving machines working on the farm, such as tractors and combines. However, operating farming machines is not easy since operators need to do multi tasks at the same time, such as controlling of the vehicle, controlling of any attachments, and detecting anything anomalous. These tasks depend on their exact farming activity, like planting, spraying, harvesting and tillage. For example, harvesting needs to monitor the on the reel speed and orientation, but for tillage it is more critical to minimize field overlap and avoid trenching. To learn from the videos captured on the farming vehicles, one important step is to separate videos into different farming activities. But farmers cannot document the activities as they happen, because they must focus on getting their work done in a narrow time window.

Currently, farming videos are collected by farmers when they are operating on machines. This video gathering process is limited by many issues, such as the camera storage. And it also needs farmers to transfer the videos to a server manually. To solve this problem, we developed a wireless video-collecting system that can transmit videos using wireless network without human control and store the videos on a secure server. But as mentioned above, all these videos are captured from different activities. Manually labelling many videos into different farming activities is time-consuming. Therefore, one important pre-processing step before analyzing the data, is to classify the farming videos into different farming activities.

In farming video, motion information is more effective to distinguish different farming activities. Thus, extracting the motion information is the key to apply this prior knowledge, which better classifies the farming videos. However, the motion conditions of farming videos are chaotic. In the literature, many of the public activity classification datasets are captured from static cameras [121], where all motion features are generated by the foreground objects such as human motion. However, the dash camera videos captured from farming vehicles have strong camera motions which are not effective for distinguishing among farming activities. Many of the camera motion estimation methods apply feature-point matching to cancel the global motions [78, 122]. But in farming videos, there are few robust feature points available.

To solve the classification problem, we apply the general two-branch video classification framework to deal with the unpredictable video motions from dashcams. In this framework, the videos are separated into two subsets using the proposed segmentation method from subsection 3.1.2. One subset of videos has camera motions only or no motion at all, while another set has motions generated by the activity or object motion. To extract positions from two subsets of video separately, we use dense sampling [123] and proposed the concept of Fixed Position Trajectories (FPT). Cuboid methods [74] are applied to build feature descriptors for both subsets, but the cuboid designs are different. In our designs, features for videos containing object motions are extracted more densely in the temporal domain, while the features for camera-motion or low-motion videos are extracted more densely from the spatial domain. By applying different sampling and extracting strategies on different videos, the generated features are more efficient and informative, which makes this framework more adaptive to videos with various motion situations.

3.3.3 Apply two-branch model on farming videos

As explained above from Section 1, farming videos have strong domain features. The most effective cues to distinguish different farming activities are in the attachment region. The attachment interacts with the vehicle and the field in a way that is unique for each activity. As a result, the motion information from the attachment region is most effective, and our proposed segmentation method allows us to sample these distinguishable features for the video classification task. Next we explain the classification process using two-branch network in detail.

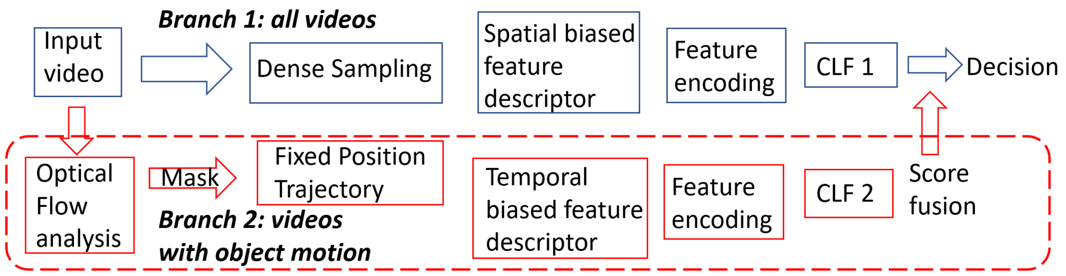


Figure 3.9.: Flow chart of the classification framework. Branch 1 (upper row) is primary and works on all videos, branch 2 (in red box) only works on object motion videos.

As mentioned above, this two-branch framework is inspired by the motions of farming video. Our video clips normally have three types of motions: static, camera motions and object motion. Static videos have no motion at all or are slowly shifting or shaking. Motion-based feature extraction methods such as Improved Dense Trajectories (IDT) are not effective on these static regions. The second type is camera motion. The motion in these scenes are generated only by the camera movement or the vehicle movement. This occurs when the vehicle is driving, and the outside objects are moving in the reverse direction. Camera motions are useful in detecting vehicle motion, but not effective for classifying farming activities. The third type of videos contains object motions that are generated by the objects themselves. These motions are robust to camera movements, and yet capture the movement pattern of the activity. Therefore, they are the most critical features.

Simply applying the same extraction methods on all types of videos cannot provide efficient features. So we generate the two different classification branches as shown in Figure 3.9, where both branches (upper and lower rows) include a complete classification pipeline. While they have different feature extraction steps, the two classification branches both have these classification steps: Principal Component Analysis (PCA) feature dimension reduction, Fisher encoding and one-vs-rest SVM training. In this framework, the upper path is primary, so all videos are processed by the first branch. The optical flow analysis is performed on the input video and used to determine if it contains enough object motion. This analysis performs spatial segmentation and produces spatial-temporal masks that indicate the object-motion regions. If no object-motion regions are found, the second (lower red) branch is disabled and the result of the first branch is final. If the video has enough object motions, the second branch is activated to process that video. Then both branches are enabled, and the final decision is determined by a fusion process. Ideally the fusion process should be a model trained using training videos as was done in [123] and [120]. Unlike [120] that trains an SVM classifier to merge results from different classifiers, we simply add the scores from two classifiers and choose the category with the highest score. Because the number of our training videos is limited, we avoid extra training steps that may cause overtraining.

The feature extraction process includes both a feature sampling step and a descriptor generation step. In the system shown in Figure 3.9, the two branches extract features differently. Branch 1 uses dense sampling with spatial-focused descriptors and Branch 2 uses FPT and temporal-focused descriptors.

Branch 1 is designed for classifying all video clips, including slow motion videos. Thus, its features should focus more on spatial information than temporal motion. We use the dense sampling method and include every position in the spatial frame into the feature vector. As shown in [92], the cuboid method has better performance than other methods to generate spatial-temporal local feature descriptors. Each input volume is first temporally cut into large temporal chunks with length T and each chunk is spatially segmented into a fixed-sized $M \times M \times T$ cuboid. Within each cuboid, different types of features are extracted in smaller blocks, and then concatenated to form a descriptor. The shape of a feature vector per video is fixed and determined by the volume shape only.

Branch 2 is designed only for object motion clips. Its features are extracted more densely in the temporal domain to capture motion information. In this part, the pre-computed object motion mask is applied, and features are only extracted around positions in the mask. To describe motion features, a trajectory is better than individual points like STIP, because a motion should be traced as a series of points. But for dash-cam videos, the spatial structures of consecutive frames are normally fixed across time. Inspired by optical flow stacking from [97], we extract features at same positions through time which we call a Fixed Position Trajectory (FPT). Unlike trajectories in [78], FPTs have no drifting problems and they can use long temporal windows. The length of FPTs need not be fixed, but here we set them to a fixed length to maintain dimensions for every feature descriptor. The neighborhood region to compute the descriptor is generated with the selected positions at center and lasts T frames. The blocks inside each neighborhood region have fewer frames (a smaller value of t) and this enables the descriptor to preserve more motion information.

3.3.4 Classification performance experiment

All videos are collected from dashcams that are mounted on combines and tractors at local farms during the last two years. All the raw captured videos are uniformly sampled into short 5-second (150-frame) clips every 30 seconds. Each short clip is down-sampled to 480×272 and only the middle 120 frames are used for feature extraction. These short videos are labelled with three basic farming categories: wheat harvesting, corn chopping and tillage. In total, we randomly select 2400 short clips from all different camera angles with 800 clips for each category. Within each category, 500 are for training and 300 are for testing. The training and testing videos are separated based on the camera angles. For example in Figure 3.1, the training frames on the left have very different angles and structures from the testing frames on the right.

Note that the focus of the second branches is on the feature sampling procedure. Therefore, the rest of the experimental design follows that in [120] to process extracted features from all different methods. In the implementation, we use the optical flow from [124], because this method provides accurate dense flow although it is time consuming. We implement the Fisher encoding method based on [125] while the PCA and one-vs-rest SVM classifier are both from the Sklearn package [126]. Since our features have much smaller shape, the PCA process is not performed for our system.

In this experiment, we compare our two-classification framework with other algorithms that have feature sampling strategies, such as IDT [78]. Since the IDT method uses HOG, HOF and MBH, we apply these three features in our strategy for a fair comparison. Recall that the proposed system has two classification branches; Branch 1 is for all video clips and Branch 2 is only for clips which have object motions. In Branch 1, we design $32 \times 32 \times 30$ cuboids and $16 \times 16 \times 10$ blocks inside each cuboid. The cuboid temporal length is set to be one second (30 frames), since motion information is not the focus for Branch 1. Based on this setting, each descriptor has the same feature dimension (396) as the feature from IDT, with 8 bins for HOG and MBH and 9 bins for HOF. In Branch 2, all videos have strong motions; and features are extracted from positions that are computed from the optical flow analysis process shown in Figure 2. Then the neighboring cuboid size is smaller both temporally and spatially. The cuboid size is set to be $16 \times 16 \times 30$ and block size is $8 \times 8 \times 6$. A smaller duration for each block helps to capture more motion information. With this setting, each feature descriptor has a dimension of 660. The fusion stage adds the scores from the two classifiers and selects the most probable category.

In addition, an image-based scene classification method Places-CNN [73] is compared. We select frames from both training and testing videos at one frame per second (5 images per video). Inspired by [89], we treat the CNN as a feature extractor and fine-tune the top layer. The AlexNet model with pre-trained PLACE dataset is the underlying model. The outputs of the last fully connected layer (FC7) are collected and these 4096-dimensional features are input to a one-vs-rest SVM classifier. In total, 7500 im-

Table 3.2.: Feature sampling strategy comparison. Numbers in parentheses are averaged values. Note *Places-CNN is fine-tuned with our data.

	Method	Feature per video $N \times D$	Accuracy
Object motion videos	Proposed B1	480×396	0.808
	IDT [78]	$(78773.67) \times 396$	0.822
	Proposed B2	$(1596.17) \times 660$	**0.857**
All videos	*Places-CNN [73]	5×4096	0.696
	Uijlings et al. [123]	2090×594	0.755
	IDT [78]	$(46989.83) \times 396$	0.751
	Proposed B1	480×396	0.792
	B1 + B2	N/A	**0.840**

ages selected from the trained videos are used to train the classifier and 4500 images are tested.

Two groups of videos are compared for different sampling methods. By optical flow estimation, 478 out of 900 testing clips have significant object motions. So we first compare the performance on 478 videos with large object motions and then compare all 900 videos.

The results are shown in Table 3.2. The first group (upper 3 rows) considers only 478 object-motion videos and we compare our two branches B1 and B2, together with the IDT method. Among all three methods, the dense sampling method (Branch 1) has the worst performance since no feature sampling methods are used. Both IDT and our Branch 2 apply motion-based sampling strategies, and Branch 2 has better accuracy. Apart from accuracy, the feature shapes of each video from the three methods are very different. IDT has the most feature descriptors N, because it extracts more trajectories (see Figure 3.2) spatially.

The second group (lower 5 rows) considers the entire group of videos, which includes videos with all motion types. Notice that image-based method Places-CNN has lower performance than the motion-based methods. Among motion-based methods, the accuracy of the IDT method decreases relative to the first test because this method is not accurate for videos that have only small motions or purely camera motions. Our Branch 1 applies a dense extraction method with cuboid, that are uniformly sampled from every position in the video volume. The method in [123] is also densely sampled, but it has much smaller cuboids and blocks, which makes both feature number N and dimension D larger than our design. But considering the performance, our output accuracy is slightly higher, which means denser sampling might not always provide better accuracy. The last row demonstrates that our whole framework outperforms other methods. This result directly applies results from Branch 1 for videos without object motion, while for object motion videos, it combines the probabilities from two classifiers.

3.3.5 Branch 2 improvement comparison

This section presents a farming activity classification experiment that demonstrates that adding Branch 2 improves the overall performance of the classification system. Due to the limited farming data, only three farming activities are classified: tillage, corn chopping and wheat/bean harvesting. All the video clips used in this experiment have a fixed length of 5 seconds, and they are randomly selected from the raw farming videos. Each clip is manually labeled with a farming activity label. To measure the robustness of the system, we separate all the video clips into training and testing groups based on the time they were captured. The training clips are all selected from 2016 and 2017, and the testing clips are from other days during 2017 and 2018. For each activity, we select 500 clips for training and 500 clips for testing, which equals to 1500 training clips and 1500 testing clips in total.

In the experiment, we choose different general classifiers as the baseline (Branch 1 in Figure 3.9) and compare them with the overall accuracy after adding Branch 2. Four general video classification systems are tested as Branch 1: the Improved Dense Trajectory (IDT) [78], the dense-extracted features [123], the first branch classifier from [7], and a video-based 3D Convolutional Neural Network (C3D) model [90]. We implement the dense cuboid method [123] and our previous method [7]. The original implementation of IDT feature extraction [78] is directly used. For the CNN method [90], we apply the pre-trained C3D network and select the last layer from the network as the feature vector. So the farming videos are input to the network and the output features are used to train a random forest classifier. Note that this pre-trained model uses the ResNet architecture [127] that is trained on Kinetics human action dataset [128].

Next we describe the implementation details of Branch 2. Four segmentation methods from Section 3.1 are selected to activate Branch 2: the fine-tuned Deeplab [50]*, our previous method [7], the improved [5]*, and the new proposed *RF_motion*. These segmentation methods produce spatial masks, but the rest of the classification processes are the same. We extract video features including HOG, HOF and MBH from the masks using FPT. The Fisher Vector (FV) from [125] is used to encode the feature vector and a random forest classifier is trained as the classifier for this branch. The final decision is made by averaging the scores from both branches.

Table 3.3.: The comparison of video classification methods. Note the right four columns show the results by adding Branch 2, and each column uses different segmentation method. Here C3D [90]* on the bottom row is used as feature extractor.

General Classifiers	Branch 1 only	Overall result adding Branch 2			
		[50]*	[7]	[5]*	RF_ motion
IDT [78]	0.591	0.661	0.618	0.619	0.754
Uijlings et al. [123]	0.772	0.798	0.815	0.812	0.833
Liu et al. [7]	0.768	0.792	0.80	0.819	0.834
C3D [90]*	0.818	0.869	0.826	0.830	**0.872**

The classification results are shown in Table 3.3. In this table, the second column shows the classification results with Branch 1 only, and the right four columns present the overall accuracy after adding the second branch with different segmentation methods. Comparing these columns, it can be observed that the performance of all four general classifiers are improved after adding the second branch. The overall best performance is achieved by the C3D [90] method as a general classifier plus the cuboid-based method with *RF_motion* segmentation.

Comparing four segmentation methods for feature sampling in the second branch, we can see the proposed *RF_motion* method has the best performance. Note that each segmentation result activates a different subset of videos to be processed by the Branch 2 classifier. So we do not report accuracy for Branch 2 alone because it does not operate in isolation. But inaccurate segmentation can cause unnecessary computation such as the Deeplab [50]* method, which activates Branch 2 for almost all testing videos. It increases both time and computational requirements for this method relative to the other segmentation methods.

Comparing the four general classifiers in Table 3.3, the improvements provided by the second branch vary. This is mainly due to the feature sampling and extraction strategies used in the two branches. The features from Branch 1 are extracted from the whole frame, while features in Branch 2 are hand-crafted to concentrate more on object motions. However, the similarities between the feature-sampling strategies of two branches limit the improvements that can be obtained by adding Branch 2. From the table, two classifiers in Branch 1, [7] in row 3 and [123] in row 4, also depend on cuboid-based feature extraction methods and they provide limited improvement. The dense trajectories [78] work poorly as Branch 1, but Branch 2 helps to improve the overall accuracy. Comparing with traditional feature extraction methods, the pre-trained C3D [90] model better captures the general video characteristics, so adding the hand-crafted Branch 2 method compensates for the lack of domain-related farming features and achieves best performance.

In general, when applying this two-branch system to practical applications, the domain-related features would generally be hand-crafted by humans based on prior knowledge. As a result, choosing general feature processing methods as the first branch can achieve better performance in the overall classification.

3.3.6 Feature efficiency experiment

This experiment explores efficiency between different features using the cuboid-based feature sampling method for farming video classification. Sampling methods such as STIP, IDT and the proposed FPT provide interest positions in the video volume, which can either be individual points or a series of points (trajectory). But when generating the feature descriptors from such positions, normally descriptors are built from cuboid regions. One interesting question is how to design the cuboid and block size, in order to get better feature efficiency with different feature sampling methods.

In a video sequence, each cuboid contributes to one feature vector and the whole video is summarized as a feature vector with shape $N \times D$. The dimension of each descriptor D is determined by the number of blocks inside the cuboid, multiplying the number of bins of the feature histogram. The number of the descriptors or cuboids per video N is determined by the sampling strategy. In dense sampling methods, N is normally the ratio between video size and cuboid size, so a smaller cuboid produces a large N. On the other hand, in STIP or trajectory-based methods, the sampling threshold decides N, and it varies based on video content and motions. Such N can be increased by dense cuboid sampling on the same volume or sampling cuboids from more than one spatial scale space. To explore this problem in our scope, we perform video classifications on the farming videos with multiple choices of N and D. Two groups of methods are tested with different videos. Dense sampling is tested for all videos, and FPT is only tested on object-motion clips.

The result is provided in Table 3.4. Notice that the results from Table 3.4 are not comparable to Table 3.2, since here we use more feature types than in the previous experiment. In the table, the upper half rows show the test for all videos and the lower four rows are for object motion videos only. We test features individually first with HOG and HOF, together with MBH and the recently proposed HMG [83]. The number after each feature in the table is the number of histogram bins of that feature. The last two columns show two different orders of feature encoding and merging. In *all merging* order, the

Table 3.4.: Feature efficiency comparison between different feature designs. Upper four rows shows the dense sampling, lower four rows shows the FPT. Note that the cuboid size is fixed for all designs.

	Block shape $m \times m \times t$	Block number	Scale	Feature per video $N \times D$	HOG (8)	HOF (9)	MBHX (8)	MBHY (8)	HMG (8)	All merged	All separate
Object motion videos	(8,8,10)	12	1	(2903.48*492)	0.711	0.858	0.599	0.488	0.77	0.819	0.677
	(8,8,6)	20	1	(2903.48*820)	0.718	0.841	0.58	0.305	0.49	0.824	0.635
	(4,4,10)	48	1	(2903.48*1968)	0.665	0.637	0.599	0.258	0.635	0.734	0.763
	(8,8,6)	20	2	(4317.44*820)	0.74	0.853	0.603	0.295	0.717	0.777	0.594
All videos	(16,16,10)	12	1	480*492	0.682	0.7	0.651	0.713	0.707	0.802	0.694
	(16,16,6)	20	1	480*820	0.768	0.698	0.643	0.677	0.485	0.782	0.645
	(8,8,10)	48	1	480*1968	0.618	0.714	0.681	0.688	0.47	0.745	0.766
	(16,16,10)	12	2	572*492	0.654	0.707	0.701	0.695	0.502	0.778	0.736

system merges all different features together as one long feature vector and then uses a Fisher vector to encode. On the other hand, *All separate* encodes different feature types individually and then concatenates all encoded vectors. All designs in the table have the same cuboid shape and each row represents one different design for the block size. *Block number* shows the number of blocks in the fixed-sized cuboid and *scale* means the number of scale spaces used to extract features. All the processing procedures are the same except the feature extraction methods. The classification accuracy is computed to be the measure of feature efficiency.

The upper half of Table 3.4 shows the results of the FPT sampling method. The HOF feature achieves the best performance among all feature types, and even outperforms the results from all features together. The lower half shows the results for the dense sampling method. The first three rows have the same number of feature vectors, but the dimension increases. The best accuracy is achieved with all five feature types with *all merged* order. AlSo this design has the minimum number of features and the smallest feature dimension.

There are some observations we can draw from the table. Firstly, neither using denser sampled cuboids nor increasing the blocks inside the cuboid to get larger feature vector improves the accuracy for either the dense sampling method or the FPT method. Also simply applying more types of features does not improve accuracy, unless a late score fusion stage is well trained to merge all decisions, like [120]. However, it is not efficient to use larger features or multiple feature types, because this adds computational burden for the encoder and classifier training process. Secondly, the order of encoding after merging five features (*all merged*) allows the GMM training stage to choose the most representative centroids in the high dimensional feature space. Therefore, it improves the encoded feature efficiency to some degree. In contrast, when merging individual encoding features together *all separate*, all feature types have the same contribution to the final feature vector. In this case, the final decision could be influenced by ineffective features. Thirdly, sampling methods such as FPT perform a spatial selection process before feature extraction, and the gradient-based features such as MBH or HMG might not provide as good performance as features that operate on the original image or optical flow, such as HOG and HOF.

3.4 Application 2: combine header control

After the videos are classified into different categories, we can develop automation applications for specific farming activities. This section introduces one video-based automation application: header-height control for a combine harvester. We select the harvesting videos which captured by dash cameras that mounted a combine harvester. The goal of this application is to automatically predict when the header needs to be lifted based on analyzing the front field. We present our prediction framework in subsections 3.4.2. Then we present the experiments on our prediction system in subsections 3.4.4, 3.4.5, and 3.4.6.

3.4.1 Background knowledge

Most farming vehicle automation applications focus on steer controls. Beside driving control, the machine operator also needs to perform the required farming work, especially for harvesting operations. For a combine harvester, the front reel (header) position should be adjusted based on crop conditions in the field. However, there are not many studies on automating the front reel header in the literature. Towards the goal of fully automated farming vehicles, this section explores the header-height prediction for a combine harvester. Note that, crop refers to different plants in the field, such as wheat, soybeans or corns.

In the combine harvester, the reel is a rotating wheel-like device that is attached to the front header to push the crop from the field. The header-height above the ground controls the position above the ground where the crops are cut off. The header should be low to the ground when there are crops to be cut, so no crops are left in the ground. But the header should be lifted when there are no crops in front to harvest, to prevent damage [108]. As a result, the number of uncut crops in front can be used to adjust the header-height. Most of the previous efforts [109, 110] for combine header are mainly passive control and their common goal is to stabilize the header-height despite vehicle vibration. However, by incorporating visual information, the header-height can be actively predicted. While continuous height adjustment is possible, here we only consider the movement from low to high. In other words, for this application, we assume the combine harvester is harvesting crops with its header low, so there are crops in the upcoming field region by default, and our goal is to predict and automatically control when the header should be lifted.

The cue when the header should be lifted is based on monitoring the upcoming field region of the combine harvester. In our settings, a dash camera is mounted on the front window by farmers. The camera not only captures the field region, but also the header and other regions. Figure 3.10 shows some sample frames that were captured from the cockpit of a combine harvester. The orange region shows the front reel (header) of the combine and the blue region is the conveyor belt that carries the cut crops into the middle. The field region, which we would like to monitor, is outlined in black. Therefore, the first task is to segment out the spatial position of the field region in the frame. The segmentation methods provided in previous sections be directly applied to select the field region.

The prediction is based on analyzing the upcoming field region, which is shown as the black box in Figure 3.10. Estimating how the crop amount changes in the field region is the key step. However, designing this automatic prediction system is not trivial. First, it needs an accurate spatial segmentation of the upcoming field region as the target. Inaccurate field regions cause missing blocks which affects crop estimation. Second, comparing the left and right columns in Figure 3.10, it is visually difficult to separate the uncut crop versus the empty field (including cut crops), or even to estimate of the amount of crops. Later in this section, two segmentation methods are evaluated for this task, and a texture-based crop-presence classifier is developed to estimate the fraction of remaining crops.

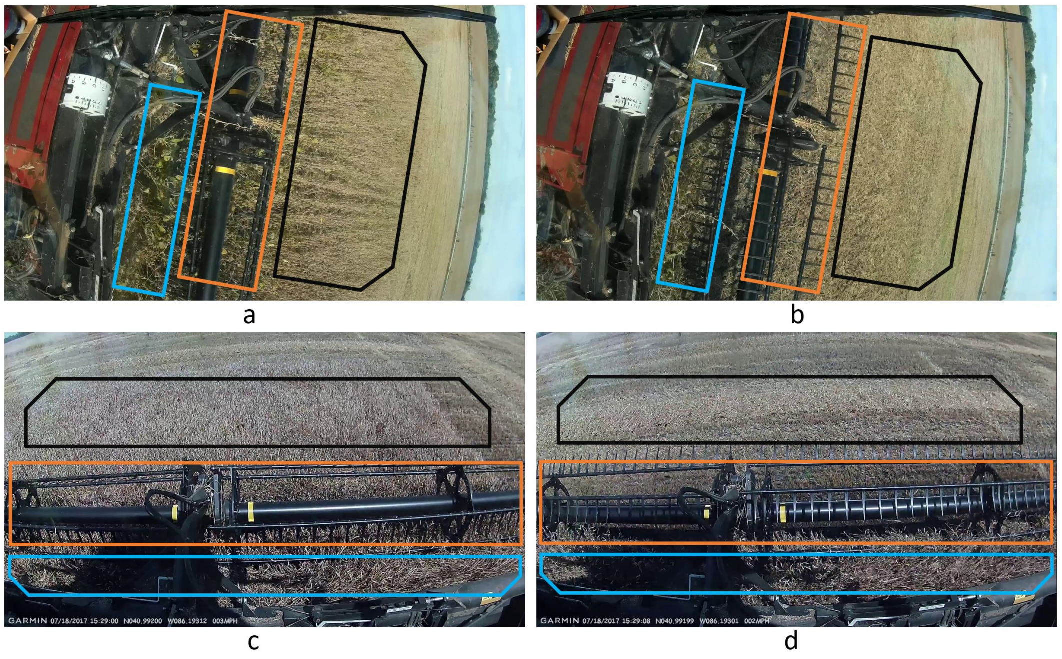


Figure 3.10.: Sample frames captured by dash camera mounted on combine harvester in bean field (top row) and wheat field (bottom row). Black box shows the front field region, red box is the front reel and green region is the conveyor belt carrying cut wheat or beans. On each row, left and right images shows the crop field and empty field in black box respectively.

For this application, the goal of monitoring the field region is to detect whether there are crops in front, and we use the fraction of the visible field that still has crops present to control the header position. Within the field region, we can classify the crops versus the empty field by training image classifiers. However, this image classification task is different than other typical classification problems. First, the camera is randomly mounted on top of the combine and far from field region. This long distance causes blurriness on the field region in the image. Secondly, from the classification point of view, the crop region and the empty field are very similar. As shown in Figure 3.10, the cut and uncut crop share a similar color for both wheat and beans. Even in the same farm field, the crops from different field regions can appear different because of the weather, lighting or the local soil condition.

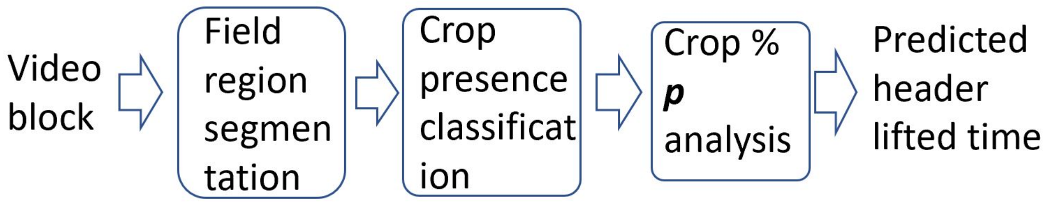

Video block
Field region segmen tation
Crop presence classificat ion
Crop % p analysis
Predicted header lifted time

Figure 3.11.: The combine harvester header prediction pipeline.

3.4.2 Proposed system

The presented header prediction system is shown in Figure 3.11. The input raw videos are processed based on a block of frames with length 30. After applying spatial segmentation, the upcoming field region is highlighted, and a crop-presence classifier is used to separate the uncut crop and empty field. Based on the classification results, each video frame produces one crop percentage value p. By analyzing a series of percentage values, we can estimate the final time when the header should be lifted.

The goal of the crop-presence classifier is to separate the crops from the empty field and estimate the crops percentages. In our design, we segment the field regions into smaller units for the classifier and then calculate the total crops percentages. As a result, the field region is divided into squares with length L based on the segmentation, and all neighboring squares overlap. We use overlapping squares because each square might not provide a complete view of the crop, and over-segmenting the field region can preserve the shape of crops. This overlap also means that each pixel is covered by multiple squares, which improves the accuracy of crop amount estimation.

To classify the squares between crop and empty field, we extract features from the divided squares. As shown in Figures 3.10 and 3.12, comparing the fields in the left and right columns, texture is more effective than color to separate the crops and the empty field. Between different texture features, we choose GLCM feature for this classification process because it better captures the directional textures in the field. As we shown in section 3.4.4, three different features including the color-based feature CBIR, the texture feature LBP and GLCM are all tested, and GLCM significantly outperforms the other two.

In the implementation of GLCM, we apply the method from [129]: four possible directions of neighbor pixel pairs are collected in each square, and the histogram contrast and homogeneity are measured as the feature vector. Each square is described by an eight-dimensional feature, and all features are trained by a decision tree classifier. Notice that all the combine harvesting videos we collected have a resolution of 1920×1080, and these higher resolution images enable the texture feature to perform well. The texture features are used to train a random forest classifier, and based on the classified squares, each pixel receives a probability representing how likely it is to be a crop instead of empty field. Then for each video frame, a final crop percentage value p is estimated using the weighted sum of all the probabilities of pixels in the upcoming field region.

After estimating all the frames in a video block, a series of crop percentage values p_t are generated. If the header of the combine harvester needs to be lifted, this percentage value should be decreasing. To capture the possible decreasing percentages, a Sigmoid function (3.5) is used to fit p_t with respect to time t:

$$\hat{p}_t = \frac{s_0}{1 + e^{-\frac{s_1}{t - s_2}}} + s_3 \tag{3.5}$$

where all the s_i are parameters, $s \in \{0, 1, 2, 3\}$. This Sigmoidal function is a natural choice given that the percentage of crop will monotonically decrease. The parameter s_2 controls the temporal position where the function decreases, s_1 indicates the dropping curvature, and s_0, s_3 normalize the range into 0 to 100. The fitted curve represents the decrease of crop percentage, which allows us to estimate the percentage value at a future time $\hat{p}_{t+\Delta T}$. As a result, the time when the header should be lifted t_{up} can be estimated by equation (3.6),

$$t_{up} = arg(\hat{p}_t \to 0) + T_{delay} \tag{3.6}$$

which predicts the time when the crop percentage decreases to 0. The T_{delay} is a constant time and it represents the delay between the time when there are no crops in the camera until the combine actually harvests the crops.

3.4.3 Experiment preparation

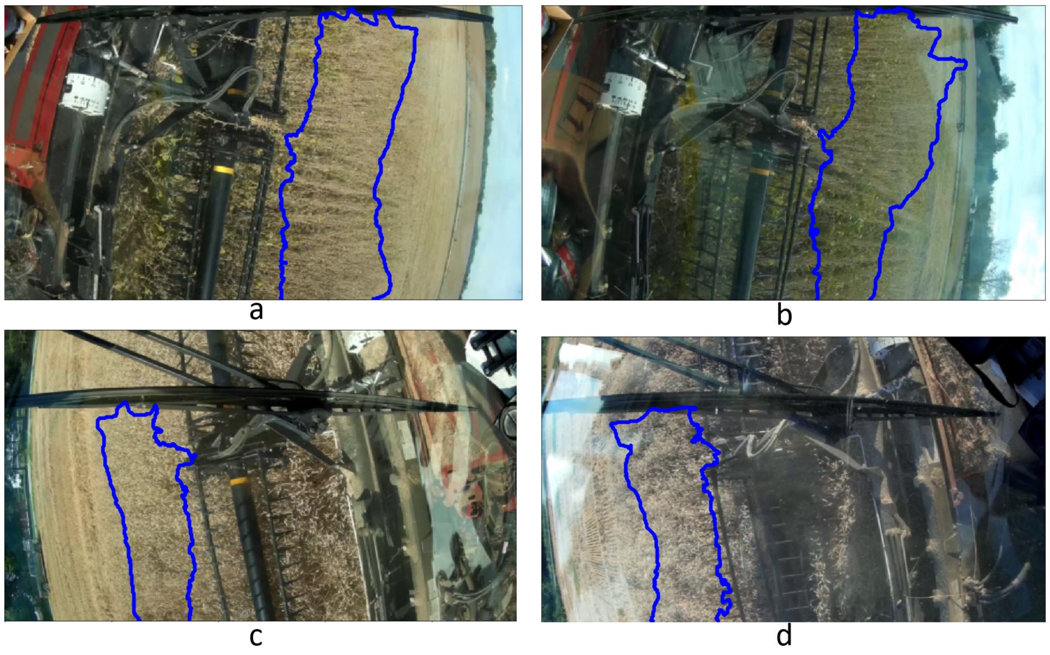


Figure 3.12.: Illumination variation of two video sets. The combine in the bean harvesting video (first row) is driving to the right and is driving to the left in the wheat harvesting videos (second row). The left column images are from the training set and right column are from the testing set. Notice the classifier is trained on squares which are selected from the coarsely-segmented field (red).

The source videos are captured by a dash camera that mounted on the front window of a combine harvester. We call a sequence of videos captured from the same day a set, and such videos normally have the same frame structure, because the camera will not move once mounted by the farmer that day. All the videos have resolution of 1920*1080 at 30 frames per second. Two sets of videos are selected in the experiment: one is on a soybean field and another is on a wheat field, as shown in Figure 3.12. For each set, we temporally segment the raw videos by hand and select those clips which contain a transition from crops to empty field. In total, we include 12 transition clips from the beans set and 20 clips from the wheat set.

Since the training and testing data are chosen from the same set of video frames, for validation we need to guarantee that the source frames of the training data and testing data are well separated in time. The different lighting conditions are considered as the separation boundary. Based on the illumination differences, we specify the first 9 clips in bean set and first 16 clips in wheat set as the source of training data, and the rest are the source for testing data. Several sample frames are shown in Figure 3.12. In the set of beans (top row), the training frame Figure 3.12a is much brighter than the testing frame Figure 3.12b, and the color of the fields are different. The lighting conditions are also different in the wheat frames set (bottom row), and the testing frame 3.12d even contains a large area of glare.

After the training and testing sources are prepared, we label the ground truth of crops (positive) and empty field (negative) for all source data. Every training or testing transition clip has around 300 frames and they all have the same decreasing trend of crops. Therefore, each transition clip produces one positive and one negative video block. For each clip, we label the first 30 frames as a positive video block and last 30 frames as a negative video block.

We prepare one pair of training data and testing data on the beans set, and another pair of training and testing data on the wheat set. There are 18 training video blocks and 6 testing blocks for the bean set, and 32 training blocks and 8 testing blocks in the wheat set. Each video block has 30 frames.

3.4.4 Crop classification experiment

The basic units of the classifier are the squares divided from the field region in the image frames. To generate the squares from the labelled video blocks, first, the field region is located in each frame by performing the spatial segmentation method described above. Notice that to achieve real-time processing speed, during segmentation, the frames are downsampled for optical flow analysis. Then the segmented field mask is up sampled back to the original size, in order to extract texture features in high-resolution images. Here we only use the motion-based coarse segmentation method to locate the closer field region. There are two reasons for this. Firstly, the closer field region provides more robust texture features than the blurry faraway field. Secondly, the goal of this detection is to predict the crop percentage for the near future, so the distant field is less critical. The red boxes in Figure 3.12 show the coarsely-segmented results of each frame.

Next the field region of each frame is divided into squares of 100 by 100 pixel with overlap of 50 pixel. Notice the pre-labelled positive video blocks only produce positive squares and all negative video squares are from negative video blocks. In total we prepared more than 60,000 squares for each training set. But for each video set, we randomly select 5000 positive squares and 5000 negative squares for training because the squares overlap.

To compare different features, for each square we extract three different types of features: CBIR, LBP and GLCM. CBIR computes the histogram in HSV color space with a fixed-bin range. The LBP feature is formed by computing the histogram of normalized LBP at every position in the square. GLCM is computed with the method in section 3.4.2. For each type of feature, we train three different classifiers. One general classifier is trained on the combined bean and wheat sets and other two classifiers are individual trained on video sets: one on the bean set and other on the wheat set. We generate two groups of testing data in the same way (one from each set). Finally, all the classifiers are tested with both two test groups.

Table 3.5.: Crop presence classification results. Notice the number below features is the feature dimension.

Feature	Trained dataset	Test on beans set	Test on wheat set
CBIR [70] (14D)	Both	0.498	0.641
	Beans	0.601	0.486
	Wheat	0.651	0.671
LBP (8D)	Both	0.601	0.783
	Beans	0.633	0.506
	Wheat	0.540	0.856
GLCM (8D)	Both	0.809	0.902
	Beans	0.961	0.712
	Wheat	0.655	0.935

The classification results are shown in Table 3.5. This is a two-class classification, so the base line accuracy is 0.5. Comparing three different features, the color feature CBIR does not work. LBP has a better performance on wheat than beans, because the wheat field texture is less directional, as shown in Figure 3.12. GLCM achieves the best performance in both sets of videos.

Comparing the three classifiers on one feature, we can observe that the general classifier that was trained on both video set, has a worse performance than the specifically-trained classifiers. In addition, the classifier trained on beans does not work as well on wheat and vice versa. The reason could be the feature difference between the two types of fields. The specific conditions from different fields make it difficult to generate one single pre-trained classifier that solves the classification problem for all fields. As a result, we need to develop a dynamic classifier that can adapt to the conditions in the specific fields. In future system design, the general trained classifier (as shown in the table) can be applied as an initialization step. Then based on the specific conditions of the field, the classifier can be improved and fine-tuned to achieve better accuracy.

3.4.5 Field analysis experiment

This section shows how we apply the pre-trained classifier to analyze crop percentage and make predictions. This simulates a real-time field analysis of header-height prediction. For preparation, first, we use the testing transition clips in the bean set to represent the real-time captured videos. All these videos begin with driving towards the end of the field and end with the reel being lifted. We choose the pre-trained GLCM classifier (eighth row in Table 3.5) for this prediction experiment. Notice the performance of the classifier in this section is not comparable with results in Table 3.5, because in this experiment, the classifier is used to analyze all the squares from every frame in the test video sequence, and there is no direct ground truth for the classification results. The only ground truth we know is that the trend of crop percentage should decrease in every testing video sequence.

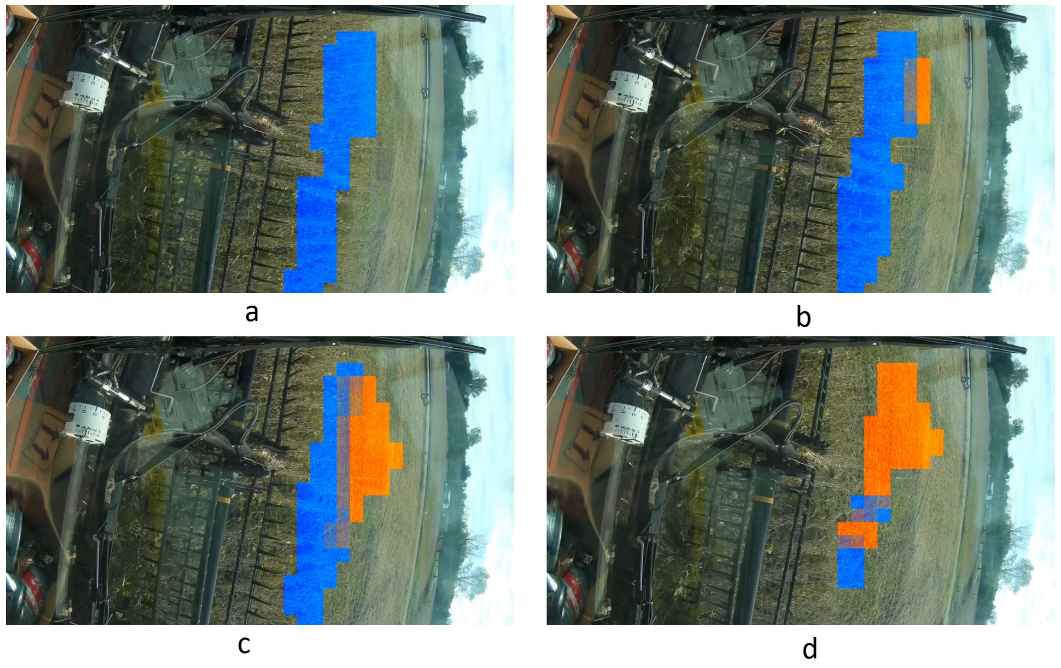


Figure 3.13.: Probability maps of beans in the field. Color represent the probability of beans: blue indicates beans and orange indicates empty field. (a) frame 90, (b) frame 120, (c) frame 150, (d) frame 180.

The experiment steps are explained as follows. We first temporally divide the input testing clips into video blocks and perform spatial segmentation to find the field region. Then for every frame in the block, the classifier is used to separate all the squares that can be extracted from the field region. Next, we apply the voting method explained above and generate a probability map. This process is repeated for every frame in the whole video sequence and finally we generate a series of probability maps.

Figure 3.14 shows some probability maps at frame number 90, 120, 150 and 180 in one testing clip. The colored regions (both orange and blue) represent the coarsely-segmented field region, and the color shows the probability: blue indicates more likely to be crop area; orange indicates no crop. Notice the combine harvester is driving to the right, and the crop region (blue) is gradually shrinking from right to left over time. In Figure 3.13b and 3.13c, we can clearly observe the border line between the crops and the empty field. But in Figure 3.13d, there are some uncertain regions in the lower half

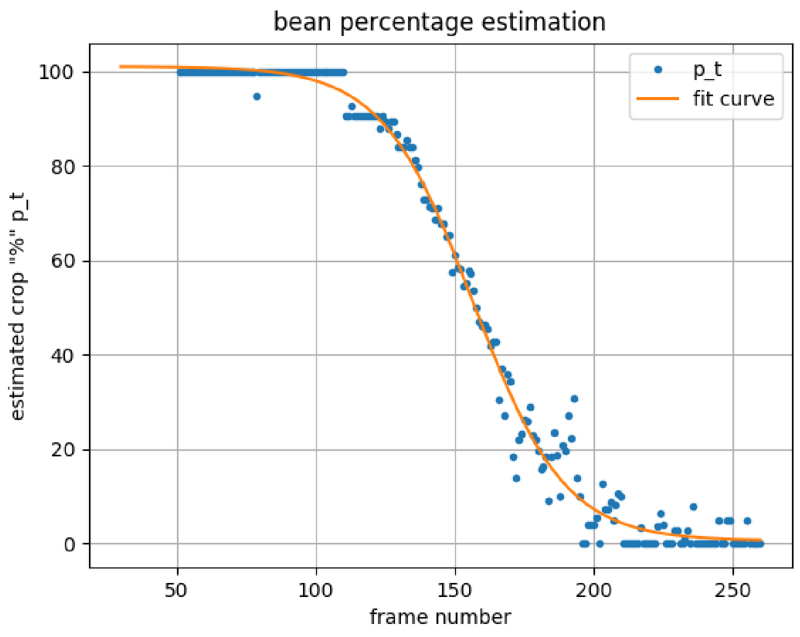


Figure 3.14.: The fitted curve of crop percentage p_t

of the field. One possible reason could be the pre-trained classifier has a bias that makes it more effective at classifying crops than classifying empty field.

The crop percentage p_t can be estimated based on the probability maps and we can fit the percentage using equation (3.5). The percentage plot of one testing clip is shown in Figure 3.14, where the x axis represents the frame number and the y axis shows the percentage p_t. We can observe that before frame 110, the crop percentage is around 100% and after the decrease between frame 110 to around 220, the percentage reaches 0 after frame 220. The fitted sigmoid function of $\hat{p}_t$ is shown in orange, which correctly captures the decreasing trend of the bean percentage in the field.

Considering the prediction of time to lift the header t_{up}, according to equation (3.6), we need to find the time when $\hat{p}_t$ reaches to 0. By setting a low cut-off percentage, we estimate that this time is at frame 220. Adding the constant time offset $T_{dealy} = 2$ seconds, the predicted lift-up time is around 280, which agrees with the actual time when the header is lifted.

The performance of our overall system depends on both the classifier and the segmentation process. Errors from the classifier lead to inaccurate probability estimation. The spatial segmentation result is also critical, and the inconsistency of segmentation affects the later steps. The colored (blue and orange) regions from Figure 3.13 show the coarse segmentation results in consecutive frames. The field region varies slightly between the four different frames, which introduces errors when counting the crop percentage. One reason is that the global motion magnitude at different times is changing, but the threshold value θ in equation (3.2) is fixed. To minimize the error in future work, designing a dynamic threshold value that is adaptive to the global motion magnitude can smooth the segmentation difference and further improve the overall system robustness.

3.4.6 Segmentation method comparison

This experiment compares the influence of different segmentation methods to the header control system. We use both wheat harvesting and soybean harvesting videos in this comparison experiment. The crop-presence classifier is trained from the manually labelled masks used in the segmentation experiment in Section 3.4.2. These ground truth masks are further labelled into crops and empty fields to train the crop-presence classifier. In the implementation, the square length L is 100 pixels, the overlap between squares is $L/2$, and the crop-presence classifier is trained from multiple harvesting clips.

When preparing the testing videos for header prediction, the raw harvesting clips are manually segmented into transition clips. All these clips contain the moment when the farmer lifts the header. Each transition clip is 10 seconds (300 frames) long with the transition happening around frame 250 to 280. We also select some normal harvesting videos (without header lifted) as negative clips. In total, we prepare 37 positive transition clips and 38 negative clips.

Three segmentation methods are compared below: the fine-tuned Deeplab [50]*, the method from 3.1.3*, and the proposed random forest method from 3.1.4. To measure

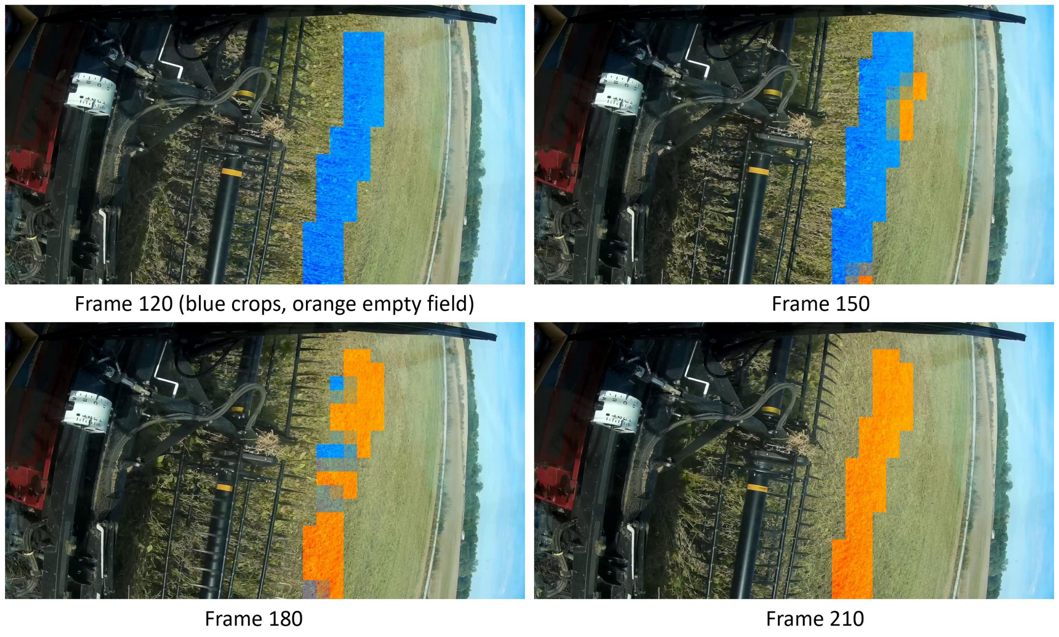


Figure 3.15.: Example output of the crop-presence classifier. The color represents the probability of crops: blue indicates crops, and orange indicates empty field.

the performance of this system, we first evaluate whether the system can make a correct decision when to lift the header. This measure is based on the difference between the average value of $\hat{p}_t$ of the first N frames and the average value of the last N frames. Here N is the number of frames that the header is stabilized to either high or low position, which is 30 frames (1 second) in the experiment. The accuracy is computed by the ratio of correctly predicted clips. If the header needs to be lifted, the system also predicts the time when the header needs to be lifted. As a result, another measure is the averaged error time between the actual rising time and the predicted T_{up}.

Figure 3.15 shows another result of the crop-presence classifier using the segmentation in 3.1.3* on a soybean-harvesting video. Similar to Figure 3.13, the blue color indicates regions that are more likely to have crop and the orange regions represents the empty field. But this example shows a smaller field region which also shows the decreasing trend as time increases. Note that this machine is driving toward to the right side, which is the end of a row in the field. As the frame number increases, the area of crop (blue) is shrinking to the left (frame 150 and 180 in Figure 3.15), and disappears in

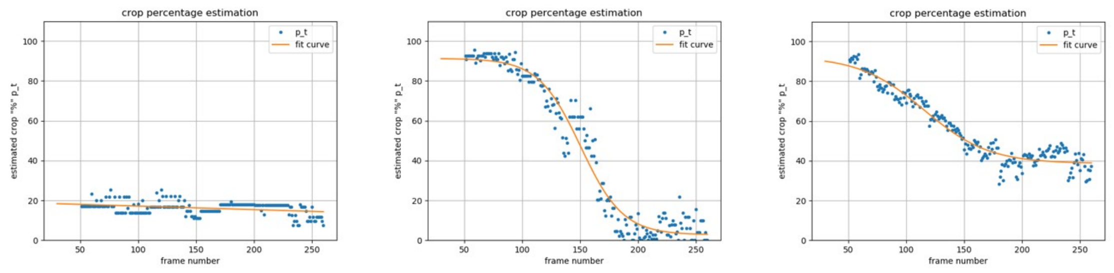


Figure 3.16.: The crop amount prediction comparison on an example video clip. The X axis shows the frame number, and Y axis is the estimated crop percentage.

the end. It can be observed that the classifier is applied on squares, and the light blue squares in frame 180 shows the uncertainty of the classifier. Each frame produces one percentage value p, and after fitting these values with time, the future crop percentage can be estimated.

On the same soybean harvesting clip, the crop percentage curves estimated by three segmentation methods are shown in Figure 3.16. All three curves present the decreasing crop percentages, but the decreasing moment of the Deeplab* [50] curve is difficult to detect. One possible reason could be the segmentation method miss-classifies other blurry regions such as the sky or some far-away blurry fields. The texture features in those regions confuse the crop percentage estimation which leads to the flat curve. The other two methods provide clear decreasing curves which can be easily interpreted to create a header control signal.

Table 3.6.: The comparison of three segmentation methods in combine header prediction. Notice all videos are 30 frame per second.

Method	Prediction accuracy	Averaged error frames
Deeplab [50]*	0.778	121.04
Liu et al. [5]*	0.984	40.67
RF_motion	0.953	66.72

The quantitative comparison between the segmentation methods is shown in Table 3.6. It can be observed that apart from the Deeplab [50]*, the other two methods have almost the same performance on both decision accuracy and averaged error time. But for real applications, no incorrect prediction is allowed. Therefore, for further improvement, this system needs to have more accurate spatial segmentation results for the field region. The crop-presence classifier needs to gain robustness by training data from various farms.

An extra experiment is performed to evaluate the sensitivity of the parameters of the fitting function in Eq. (3.5). All four s_i values are adjusted to $s_i \pm 5\%$ and we perform the same evaluation as in Table 3.6. The results show that only the variation of s_2 cause 1% variation compared to the original results in Table 3.6, and all other parameter variations result in no changes. This indicates that the method is robust to small perturbations generated by the parameters in the fitting model.

3.4.7 Limitations and potential improvements

There are some limitations of this current prediction framework. The camera-based system is only able to predict the time when the reel should be lifted, but the time when reel should be lowered is not considered. Also, we only consider the header-height to be in either high or low position, but for an accurate control system, predicting precise height positions needs further exploration. Instead of detecting the crops in front of the vehicle, there are other cues that can be applied to predict the header position, for example the load on the conveyor belt. Our presented system could be improved by considering more cues for better prediction.

This current framework can be improved to be a fully automated and adaptive prediction system. In the current framework, during the pre-processing stage, the raw harvesting video data need to be manually labelled. Then we need to temporally segment the transition clips by hand. For future work, we can further automate the hand labelling work of the temporal segmentation, which enable the system to achieve better performance on to different fields.

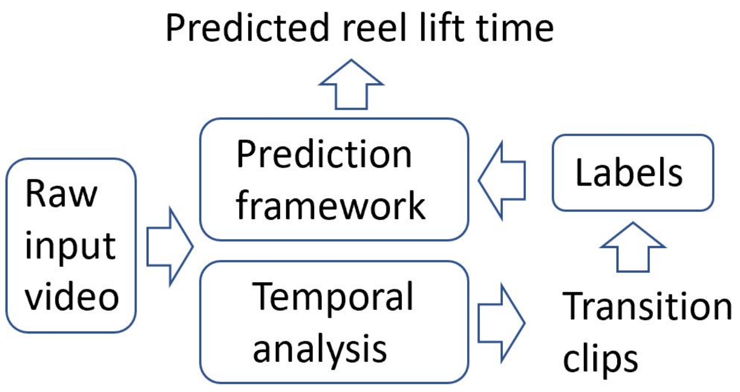


Figure 3.17.: The fully autonomous header-height prediction framework.

Figure 3.17 shows the fully autonomous header prediction system. The incoming video is processed by two branches. The proposed header-height prediction framework is applied to estimate the time when the reel should be lifted. At the same time, a temporal analysis module detects the changes of header position. The temporal analysis will automatically crop the transition clips, and further process them into labels of crops and empty fields. The new data and labels provide a feedback to update the crop presence classifier in the prediction framework. With the automatic temporal segmentation system, the whole framework can produce accurate classification result and make better predictions.

There are some potential issues need to be considered for this fully autonomous system. Firstly, in this dynamic system, the prediction framework is updating but temporal analysis part should be fixed. And the temporal analysis should be robust enough because it is used to replace human labelling process. Secondly, we need to assume that all header lifting movements are caused by the fact that there are no crops in front. To overcome this limitation, we need to apply different rules for predicting when the header is lifted, such as the crop load on the conveyor belt. Thirdly, this system is designed to learn from the operator. So, we need to assume that the operator is performing correct operations, otherwise the reliability of video data is not guaranteed.

4. DAIRY COW ANALYSIS

This chapter introduces video processing techniques related to animal health analysis. We mainly focus on dairy cows in the project. In Section 4.1, we present our proposed cow structural model, followed by the cow model detection system in Section 4.1. The proposed evaluation metrics are described in Section 4.3. Then we present three different experiments based on the detection system in Section 4.4. Based on this proposed method, we also introduce an cow-weight estimation application in Section 4.5.

4.1 Cow structural model

This section describes our proposed side-view cow structural model. We first introduce the motivation of this model in subsection 4.1.1. Then we present the keypoints in our cow structural model in detail in subsection 4.1.2, and then describe the spatial constraints between the keypoints in subsection 4.1.3.

4.1.1 Motivation and Target

As Section 1.4 mentions, video analytic techniques could be applied to monitor and detect cow health conditions. However, the first step to analyze cows using visual data is to detect and segment the cows within the video sequences. This is a straightforward task when each cow walks individually on a well-lit pathway with a very clear background and no obstructions. However, if a practical camera system is installed on a commercial farm, with the goal of not interrupting daily operations, unrelated objects such as humans and fences are often captured. As a result, identifying the spatial locations of cows is a fundamental first step for further analysis. However, location is not enough. For the purpose of assessing an aspect of the animal (i.e. body size or gait), simply having a binary mask that labels the cow's location is inadequate. Further information of the cow's structure is required, such as the locations of all body parts or joints [1]. This information can then be converted to human interpretable knowledge, or further processed by autonomous health monitoring systems. In summary, we need a video system that not only isolates the cow's spatial location, but also detects its body structure and tracks its movements.

Designing a video processing system that satisfies these two requirements is not trivial. Many previous segmentation methods focus on object detection, which generates a binary mask of the objects and their corresponding labels. But this is not enough for further cow health analysis. Recently, additional methods have been proposed to detect keypoint-based object structures, like human skeleton models. These models are formed with a series of keypoints or joints connected in a particular order. But these methods are designed with the knowledge of human structure, which is difficult to adapt to other animals like cows. While new methods such as DeepLabCut [65] focus on animal-related keypoint detection methods, this method requires clear video sequences with a single object and a clear background. It cannot be directly applied for practical cow applications on the farm. Finally, there are some visual applications [23, 26, 130] for cows in the literature, but they are designed for videos captured in a specifically-designed environment, which requires extra efforts and costs for the farm to collect video data.

Processing cow videos collected from a practical farm also poses specific challenges. First, the environment in which video is captured environment cannot be fully controlled without interrupting the daily operation of the farm. The cameras need to be installed with specific positions and viewing angles, so that the cows can be clearly observed with few obstructions. Issues such as poor illumination [26] and heavy obstacles [27] largely influence the performance of existing detection algorithms. In addition, the environment also limits the choice of capturing devices. Surveillance cameras are the most suitable devices to install and deploy on typical farms, but the quality of the

[1]In this paper, we use the term body parts or keypoints to refer to the points that represent joints or specific regions on the cow.

videos is limited. Distortions such as low color saturation, low frame rate, and heavy compression deteriorate the performance of detection algorithms.

In later sections, we introduce our proposed cow structural model and a detection system. we combine deep learning with domain knowledge about cows to develop a cow structure detection system that operates on videos captured from a practical dairy farm. Our system estimates the number of cow objects in a frame and detects the body parts of every individual cow. For each cow object, the detected body parts compose a side-view cow structural model as shown in Figure 4.1. This model describes both the spatial location of the cow and additional structural information such as the body contour, the positions of major joints, and the trajectories of their movement. This detailed information provides interpretable knowledge for further health analysis.

This method also overcomes some practical issues. First, all the videos are captured without interfering with the daily work on the dairy farm, requiring only surveillance cameras and no specialized hardware. Second, by incorporating domain knowledge about cows, the video processing algorithm overcomes practical video challenges, such as poor video quality, bad lighting conditions, and heavy occlusion from fences. Later experiments show that our method provides robust results under these conditions.

4.1.2 The body structural model

This proposed structural model is designed to represent a detected cow object in the frame more effectively than using a binary cow mask. It is designed to provide both the spatial location and cow structural details, such as the body shape and positions of the body parts. For consecutive video frames, this model should also provide information so that we can track the movement or motions of these body parts. Inspired by recent approaches to model the human skeleton [59], we combine some anatomical cow joints with other spatial keypoints to represent the cow pose, and the cow structural model is built by connecting the keypoints.

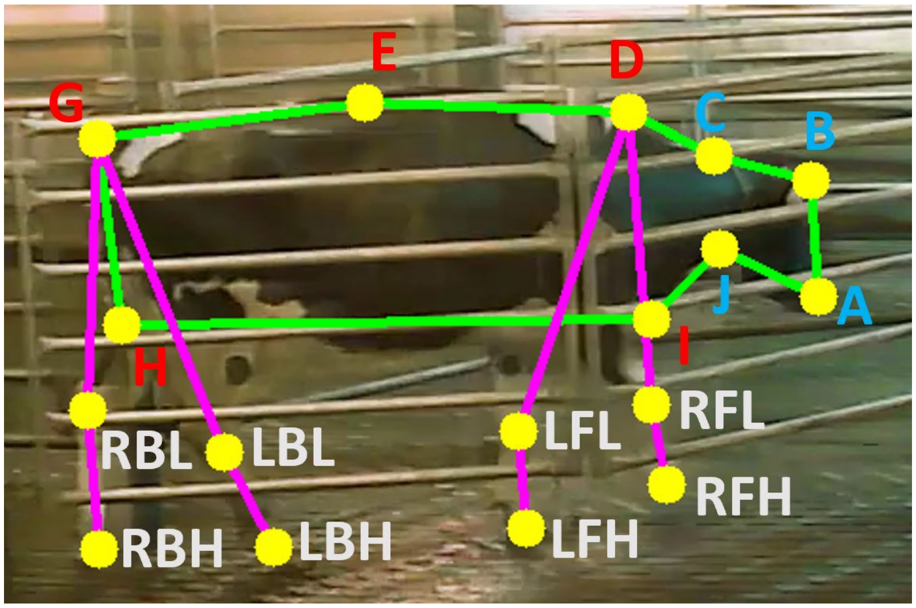


Figure 4.1.: The proposed cow structural model. 4 blue head region points: A:nose, B:head, C:top of neck, J:bottom of neck. 5 red body region points, D:shoulder, E:spine, G:tailhead, H:mid-thigh, I:bottom of shoulder. 8 white leg and hoof points, with name format: Right/Left + Front/Back + Leg/Hoof.

Figure 4.1 shows our proposed side-view cow structural model. There are 17 points in total to describe the important locations of a cow object from this angle. The upper body region has 9 points, including the head region (blue) and the main body region (red). Connecting these points forms the contour of the upper body region (green lines). Another 8 points are in leg-hoof regions which represent the four limbs, and each limb has a pair of leg and hoof joints. Comparing to the anatomical 51-point cow skeleton model [131], we only select visually-observable joints. Some joints such as the elbow and stifle joints are neglected because their positions are hidden in the body and difficult to estimate visually. In addition, we also add some points such as the two bottom corners H and I show in Figure 4.1. Even though they are not physical joints, connecting them with other joints forms a closed contour which spatially locates the body region. The point E on the spine is also an added point, because connecting three spine points provides information about the back curvature which is useful for lameness detection.

There are two general observations about the keypoints in this cow structural model. First, the points in the main body region (red in Figure 4.1) are always visible from the side-view, and their relative spatial locations do not change dramatically when the cows are walking. Second, the leg and hoof points are more difficult to detect compared to the upper body region points because of the practical issues such as bad illumination, shadows, and fast leg movement. Distinguishing between the points from the left or the right leg is also difficult when there are obstacles in front, for example the horizontal fences shown in Figure 4.1.

4.1.3 keypoints constrains

Practical constraints limit the potential relationships among the keypoints in both space and time. When the cows are walking between the fences, the cameras located at a fixed position on the side wall always capture the side-view of the cow. In this case, all the cows shown in the video have the similar pose, and the keypoints of their upper body region are always located at relatively fixed spatial positions. For example, the cow's head always appears on the right side of the body, and the body does not change size. As a result, we can compute general relationships that constrain the keypoints in the cow structural model.

To model the constraints, we first define the center of the cow's body. This center point is computed as the spatial center of all the keypoints from the cow's upper body region. Note that the points in the leg-hoof region are not used to compute the center point because their positions are not relatively fixed when the cow moves. Then we can estimate the relative spatial relationship between the center and all the keypoints.

Figure 4.2 visualizes the keypoint constraints. The middle X shows the cow center c, and the relative spatial locations of the upper body parts appear surrounding the center. Notice each body part mapping function F_j is a 2D Gaussian probability distribution, which is shown as the ellipse in the figure.

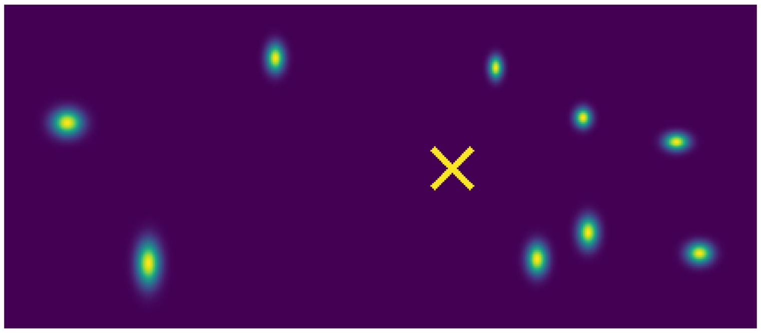


Figure 4.2.: The constraints between the upper body keypoints of the cow structural model. The yellow X is the cow center, and the surrounding points shows the relative positions of the keypoints in the upper body region.

Formally, for a fixed cow center point $c^* = (x, y)$, we define a set of mapping functions $F_j(\cdot)$ that describe the relative spatial locations of every upper body-part point p_j^* to the center,

$$p_j^* = F_j(c^*) \tag{4.1}$$

where j is the index of the body part. Each mapping function F_j is characterized by a 2D Gaussian model, and the parameters are trained using all ground-truth labels. During the training process, the approximate cow center c is computed first by averaging all labelled body parts, and the parameters in each F_j are estimated individually based on their relative spatial locations to c.

In the next section, we show how these constraints can be used to separate cows which are spatially close together in the frame. They also provide a reference when assigning body-part candidates to each individual cow object in the post-processing module.

4.2 Cow structure model detection system

This section introduces our proposed system to detect the structure of cows. We first review one popular work for keypoint extraction and then describe the components of our proposed system. Then we explicitly introduce two main processing components: the body part extraction module and the post-processing module.

4.2.1 The DeepLabCut tool box

The DeepLabCut toolbox [65] is a recent popular method to extract keypoints from video sequences. Its inputs are color images from videos, and it applies a CNN to generate confidence maps that represent the potential keypoint locations. One advantage of the DeepLabCut toolbox is that it provides simple access for users to manually define the output body parts, and the toolbox automatically alters the last layer of the CNN based on the number of body parts. For example, there are 17 confidence maps generated in our case because we have 17 keypoints in our cow structural model. In our system, we apply the network created by the toolbox to extract the keypoints of our cow structural model.

However, other modules from the toolbox are less suitable for our application because of two major limitations. First, this platform is designed and evaluated with videos captured from a laboratory environment with clear objects and background. But our cow videos, generated from a commercial farm, have low video quality and the view of the cows are often blocked by obstructions. Later experiments show that the original DeepLabCut does not provide robust detection results on our videos. Second, this method assumes there is only one object in a frame, so it only chooses one body part from each confidence map. If there are multiple body-part candidates detected, only the position with the highest confidence score will be selected. But in practical cow videos, there could be multiple cows and obstructions like fences easily cause false detection. We address these two limitations and build a general keypoints detection system which extracts robust keypoints on our cow videos.

4.2.2 Proposed system

This detection system is targeted to extract the structural model for every cow object from video sequences. Figure 4.3 presents the overall system; its primary components are two CNNs for the extraction and a post-processing module. The body part extraction module uses trained networks to convert each single image into a group of confidence maps. Each map shows the potential locations of a particular body part, and the values of the map represent their detection confidence. The post-processing module generates the final structural model based on two groups of confidence maps and the trained keypoint constraints. Both modules are discussed in detail in the next two sections.

In this figure, both the training process and the testing process are labelled using colored arrows. During the training process (indicated by the green arrow in Figure 4.3), the ground-truth labels are used to fine-tune both CNNs and the keypoint constraints. During operation (indicated by the yellow arrow), the system takes the input of both the color image and the frame difference image on the left and generates the cow structural model for a single image. After all the frames from a video sequence are processed, the post-processing module refines all the detected cow structures based on temporal information.

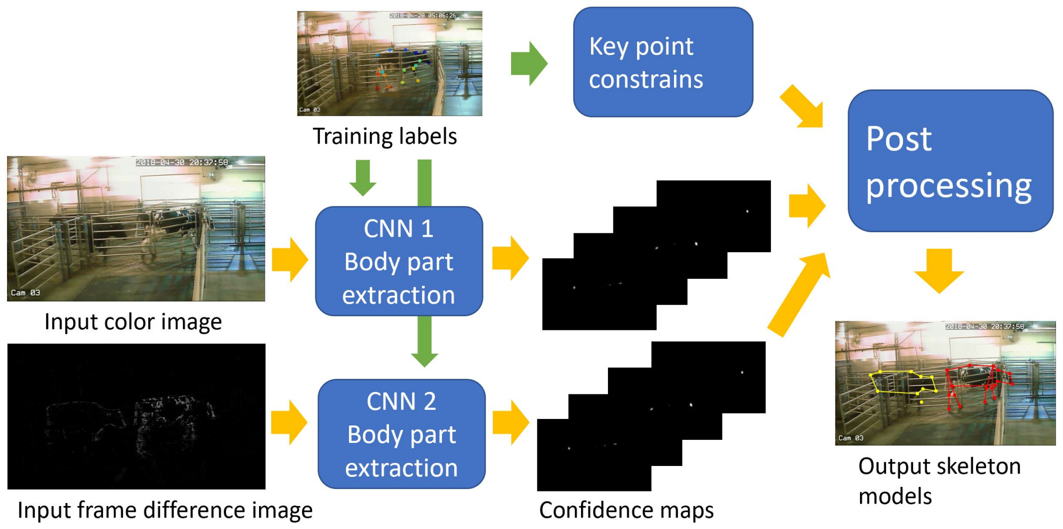


Figure 4.3.: A diagram of the proposed system. The green arrows show the training process and yellow arrows present the process during operation.

4.2.3 Body part detection module

The goal of this module is to find the spatial locations of all potential keypoints from raw images. In our system, we apply the original DeepLabCut network [65], labeled CNN1, to extract keypoints from color images. This network structure follows DeeperCut [132] and is implemented using ResNet [127] for the convolution stages, followed by one de-convolution layer before the output layer to recover the target spatial locations of the keypoints. The last two convolution layers apply atrous convolution, which increases effective fields-of-view of the applied convolution and preserves spatial resolution [50]. By default, the DeepLabCut network is pre-trained on ImageNet [133] for image classification tasks, and we use our own cow labels to fine-tune the last deconvolution layer for keypoint detection.

However, as mentioned above, low video quality and heavy obstacles influence the performance. To overcome this issue, we add an extra network, CNN2, into the system. The architecture of this network is same as the first, but it processes frame difference images. There are three major advantages of using frame difference images for our cow videos. First, because we have fixed cameras, the frame difference image better captures the moving objects and eliminates the stationary obstacles such as fences. Second, many of our target keypoints are on the contour of the cow body, and the frame difference highlights these edges of a walking cow.

Third, frame difference also reduces the influence of color variation. This is useful, because the color responses of different cameras are different especially under poor illumination. In addition, the majority of the cows have color variations introduced by the patterns on the cows, but there are some cows have single coloring, such as pure while or pure brown. These patterns are unlikely to be included in the training frames, so the color-based CNN methods would likely fail to detect cows with unknown colors. As a result, using frame differences provides robustness to these factors.

However, using the frame difference images alone is not enough because they eliminate too much spatial information, especially for legs and hooves. This is because most of the legs are stationary even when the cows are moving. As a result, our system merges both networks together to improve the body part detection accuracy.

4.2.4 Post processing module

The post-processing module collects and merges the confidence maps from the two CNNs and assigns the cow body-part candidates to each cow object instead of just to one cow per frame. This step enables the system to detect multiple cows together and track their temporal movements. There are three major steps in this post-processing module: body part extraction, spatial clustering, and temporal filtering.

Body part extraction

This step extracts the spatial locations of all body-part candidates from the confidence maps generated by the CNNs. Notice that at this stage, the number of cow objects in the image is unknown and we want to extract all possible candidates. So, for each body part, the confidence map from the two networks are merged together, and we use non-maximum suppression to select all the points whose confidence scores are higher than their neighbors.

The output of this step are lists of body-part candidates. Formally, for a given frame at time t, all these body-part candidates can be represented as $p_j^{i,t} = (x, y)$, where j is the index of that body part, and $i \in \{1, 2...\}$ indicates the count of all possible keypoints extracted for this body part. The total number of i is not determined because there could be multiple objects in the frame, or some candidates could be incorrectly detected. All these candidates are further selected and clustered in the next step.

The confidence maps from the two networks are treated differently during this process. For keypoints from the upper body region, the two confidence maps are directly merged to find body parts. But since color information is more useful to detect the legs and hooves, only the confidence map from the color image network is used unless this map contains no candidates.

Spatial clustering

The second step in the post-processing module is spatial clustering. This step selects the correct body parts and clusters them into different cow objects. The first task before clustering is to determine the number of cows in the frame by counting cow centers. Given a set of extracted keypoint candidates $p_j^{i,t}$ from the upper body parts, the corresponding cow center positions can be estimated based on the constraints of the keypoints, shown in Equation 4.2.

$$c_j^{i,t} = F_j^{-1}(p_j^{i,t}) \tag{4.2}$$

Then a mean-shift clustering method is applied to the 2D spatial positions of all the cow centers $c_j^{i,t}$. Based on the clustering results of the center points, the corresponding body parts are labelled into separate cow objects. We ignore a cow object if the system cannot detect more than half of its keypoints.

The cow centers are also used to predict the location of missing body parts that the network fails to detect. After all the keypoints are clustered into distinct cow objects, then for each cow object, we compute the averaged cow centers based on the detected points, and the miss-detected keypoints can be estimated using the keypoint constraints F_j. The predicted body parts based on the constraints may not be always accurate, but they provide a rough estimate of the cow's spatial location, which is useful for searching for keypoints in leg-hoof regions.

The final process in this step is to match the leg-hoof points. Similar to [26], we indicate the region of all possible leg-hoof points using a rectangle that is somewhat wider than the rectangle of the upper body. Candidates outside this region will not be considered. The search process relies on the structural model. We follow the order of *shoulder/tail base, leg, hoof* along each limb, and search the joints from among the candidates that lie in the search range. We also reject inappropriate points by applying the rule that each limb should have a certain rotation range; the angle between *shoulder* to *leg* and *leg* to *hoof* must be greater than 90 degrees for valid keypoints. Finally, all the selected

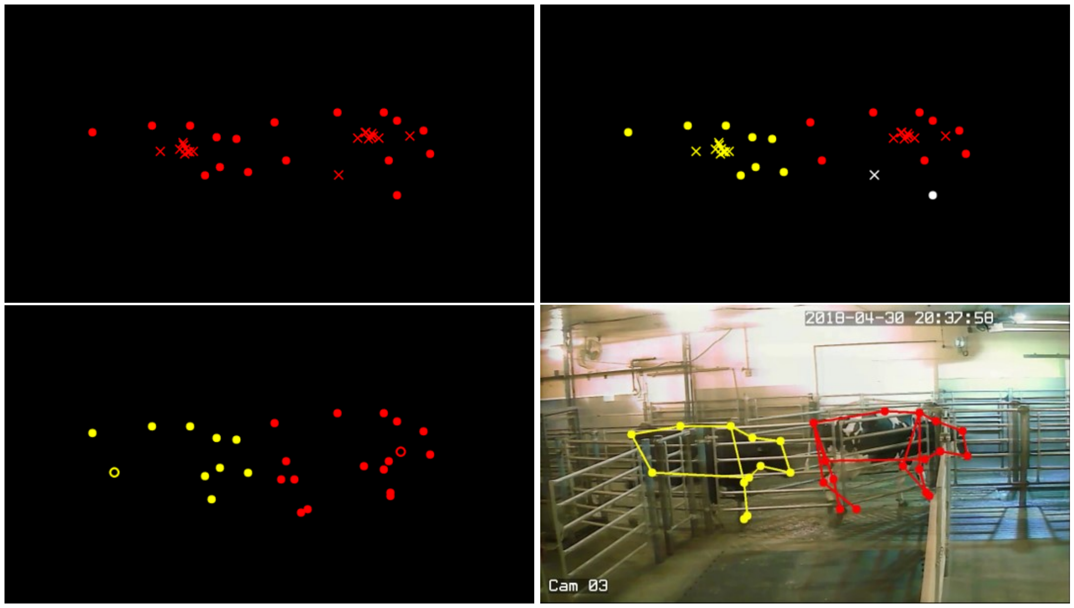


Figure 4.4.: The procedure of spatial clustering during post-processing. Top left: the keypoints and centers, Top right: Center points clustering. Bottom left: Adding leg region points, Bottom right: The output structural model. Circles represent the body parts p and crosses are the estimated cow centers c. Empty circles are the predicted body parts. Each color indicates a different cow object.

leg-hoof joints are connected to the body contour to complete the final cow structural model.

Figure 4.4 illustrates the procedures of the spatial clustering step. The top left image shows the original extracted body parts from the previous step. The red circles are the extracted candidates, and each is converted to a corresponding cow center, shown as crosses. Then in the top right image, all center crosses are clustered using mean shift to produce three clusters shown in distinct colors. Here the incorrect cluster (white) is eliminated because there are not enough candidates. Next in the bottom left image, points in the leg and hoof region are assigned to each cow object. Notice the empty circles are predicted points; the yellow one is blocked by the fences. Finally, by connecting all keypoints together, we form two cow structural models as shown in the bottom right image.

Temporal filtering

The final step in post-processing module refines the detection results using temporal information and matches cow objects across different frames. The two previous steps each operate on a single image, but the relationship between neighboring video frames is helpful to refine keypoint positions. It is reasonable to assume that the cows walk on an identical path between the fences and that they move steadily and slowly. This means that for a specific keypoint in the upper body, its trajectory over time should be smooth and any points far from the trajectory line can be considered outliers.

Based on this idea, we refine the positions of every upper body-part point across time to improve the temporal smoothness of the output. Before this step starts, all the frames in a video have been processed, so we know the number of cow objects in each frame. Then for every body-part in the upper body region, we temporally filter each trajectory to remove and correct the outliers. In our experiment, we use a median filter, which is simple and provides robust prediction. Other filters such as the Kalman filter do not work well especially when there are too many missing points from the previous steps. Notice that the leg-hoof region points are not involved in this process, since their trajectories are much more complicated.

Based on the trajectories of each cow object, the cow objects can be matched between neighboring frames. After this process, the system detects the total number of cows shown in a complete video sequence, and parameters about how every cow move can be inferred, including the speed and rhythm [134].

4.3 Evaluation metrics

This section introduces our evaluation metrics. Although our method uses few ground-truth labels for training, ground truth is typically also required for performance evaluation. Therefore, in this paper we propose to use both supervised measures, which compare the detected results with ground-truth labels, and unsupervised measures, which directly evaluate the results without labels. Adding unsupervised measures to the evaluation process improves its thoroughness in the presence of insufficient labels. We first discuss the supervised measures for the cow structural model, and then introduce two unsupervised metrics.

4.3.1 Supervised metrics

Quantifying the performance of the cow structural model requires more than the typical measures used to quantify object detection. As mentioned in Section 1, the cow structural model is designed to provide two types of information: the spatial location of the body region, and the detailed positions of body parts. Both information is represented in terms of the keypoints of the cow body parts, and our ground-truth labels are also in terms of keypoints. As a result, we separately evaluate the area of the cow body region and the points in the leg-hoof region. Two metrics are developed and described below in detail: the *Body F1 score* and the *Leg-hoof F1 score*. In each case, the F1 score is harmonic mean of precision and recall when comparing the detection results to the ground truth. Notice that both metrics compare accuracy at the keypoint level. In a later experiment in Section 4.4, we also propose a method to convert the cow structural model to a binary mask with both body region and extended limbs, for the sole purpose of comparing our detected keypoint model with other mask-based segmentation methods.

Body F1

This metric measures the spatial area formed by the body region points. We connect the keypoints in the upper body region and generate one polygon mask for both the detected structural models and the ground-truth keypoints. Then we compare the two masks using the typical Intersection Over Union (IOU) metric and report the F1 score.

Leg-hoof F1

For the legs and hooves, a single pixel position represents each keypoint. However, physical joints typically extend for a larger spatial region. Therefore, the evaluation metric must accommodate this discrepancy, which may introduce systematic errors to both the labelling and detection process. For this reason, when measuring the distance between ground truth and the detected leg and hoof keypoints, we set a threshold distance of 30 pixels. If the distance between the points is less than this threshold, we consider the joint to be detected, and points further away are considered to be miss-detected. After thresholding, we determine how many leg-hoof points are successfully detected and summarize this using the F1 score computed from the precision and recall. Since we do not create ground-truth labels for keypoints that are completed blocked by obstacles, these blocked joints do not affect the evaluation result.

4.3.2 Non-supervised metrics

Unsupervised measures allow performance evaluation without ground-truth labels. This is particularly critical for video, where exhaustive application-specific labeling becomes even more onerous. Without labels, previously proposed metrics such as mean of region similarity, contour accuracy [135], and temporal stability metric [42] cannot be computed. Here, we apply prior knowledge to evaluate the performance when the ground-truth labels are not provided.

We consider two rules for the cow structural model. First, the spatial locations of the keypoints in a model should always form a cow-shaped object. Second, the shape of the cow body should be stable during the walk and the keypoints should have similar smooth trajectories. Based on these two constraints, we introduce two unsupervised metrics: the valid cow percentage and temporal consistency.

The Valid Cow Percentage (VCP)

This metric counts the fraction of detected cow models that are valid. Here valid means that the positions of the keypoints in the structural model can form a cow-shaped object. Like the supervised measure, we validate the upper body region and leg-hoof region separately.

For the upper body region, we use the trained keypoint constraints (Figure 4.2) as a reference, and compute the similarity between the detected contour and the reference using the Fréchet distance [136]. We choose this distance because it better captures the similarity between two curves, which are the body contour in our case. The computed distance is thresholded to form a binary decision whether the upper body region is valid or not. For points in the leg-hoof region, we define two interpretable rules to validate their spatial positions: all leg-hoof points should be lower than the body region points, and all hoof points should be lower than their corresponding leg points. If all leg-hoof points satisfy these two rules and the upper body region contour is also validated, the cow structure is considered valid.

This validation scheme is applied to all the detected cow objects in a video sequence, and the Valid Cow Percentage (VCP) is computed as the number of valid cow objects divided by the number of detected cows. The absolute VCP score is directly related to the actual number of cows in the testing video sequence, so the score is only meaningful when compared with other methods on the same testing dataset.

Temporal Consistency (TC)

The second unsupervised metric evaluates the Temporal Consistency (TC), which reflects the smoothness of the motion of moving objects in a video sequence. It is reasonable to assume that at a certain camera angle, the points from the body region always share the same translational motion because the shape of the cow body is stable. So ideally, the motion vector between every keypoint generated from one frame to the next frame should be the same. The Temporal Consistency (TC) metric evaluates this co-movement and computes the difference between the motion vectors generated by the body parts.

Formally, for each body part p_j^t in a cow object, we compute its motion vector from time t to $t+1$ and summarize the variations d between all the motion vectors as

$$d^t = std(p_j^{t+1} - p_j^t), \forall j \in \{j_1, j_2 ...\} \tag{4.3}$$

where std is the standard deviation, and j_i represents the index of the body part from the upper body region. Then the temporal consistency is computed as the average motion vector differences for all the frames in a video sequence.

$$TC = mean(d^t), \forall t \tag{4.4}$$

Notice this measure is applied to every individual cow object in a video sequence, and smaller TC values imply smoother object movements.

4.4 Structural model experiments

This section presents the validation experiments. We first give a high-level summary of how we collect and prepare the video data from a practical farm. Then we present three different experiments. The system output experiment compares the results of every stage in the proposed method, to demonstrate the importance of each component. Next, the dataset robustness experiment is performed on three different sets of videos, to demonstrate the robustness of the method. Finally, we compare our method with other popular object segmentation methods, to demonstrate the advantages of our proposed method for cow detection.

4.4.1 Data collection

All cow videos in these experiments are collected from the Purdue Animal Sciences Research and Education Center located in West Lafayette, IN, USA from 2018 to 2019. All procedures were approved by the Institutional Animal Care and Use Committee (PACUC #1803001704). The cameras are mounted at fixed positions include a side-view of the path where cows walk every day. This path has fences on both sides and only allows one cow to walk through at a time. This limits the amount of cow-overlap; however the dense fences partly block the view of the cows, and some body parts are not visible behind the fences, as shown in Figure 4.1. This walking path is a typical component of many dairy farms.

During data collection, we used three different capture devices: a commercial surveillance camera with Digital Video Recorder (DVR), a GoPro camera, and a high-quality IP camera. Table 4.1 shows the detailed information of the three video sets captured from the three cameras. The DVR videos have the worst quality with low frame rate and low resolution. The GoPro videos provide higher frame rate, but they are spatially cropped with less spatial details. The IP camera captures high quality videos with both high frame rate and rich spatial information.

Table 4.1 compares several factors among the cameras that will influence detection performance. As noted, the video resolution and frame rate are different between the three sets and Set 3 has the best quality. The number of pixels per cow refers to the average number of pixels that each cow occupies in a image, which is an indication of the spatial detail in each set. Notice that Set 2 only has 0.29 million pixels per cow, which is less than a third of the other two sets. The field-of-view each camera are also different. Set 1 videos only capture the center of the walking path where there are fewer fences, while the other two sets capture a wider view which includes two sides that have denser fences. In addition, the typical number of cows in one video are different across the sets. Narrow field-of-view videos normally captures a single cow in the frame, but the wider-angle videos could contain multiple cows, which challenges the detection method. In general, Set 3 has more video clips than the others with the greatest variety, so we will further divide this set into subsets in a later experiment described in Section 4.4.

To prepare the videos, we temporally segment the hours-long sequences into 10-second clips, on average, where all cows walk from left to right. In each set, we separate the clips into training and testing groups, where the number of training clips per set are shown in parentheses in Table 4.1 after the number of video clips. All multiple-cow clips are testing clips, so the training clips all contain only a single cow object. Non-consecutive frames are chosen randomly for labeling from both training and testing clips.

Table 4.1.: Summary of three sets of video data used in the experiments. The # Pixel per cow is in units of millions.

	Set 1	Set 2	Set 3
Capture Device	DVR	GoPro	IP camera
Video info	1280*720 @12fps	1232*384 @30fps	1920*1088 @30fps
# Pixels per cow	0.88m	0.29m	1.35m
Image Quality	low	low	high
Field of view	narrow	wide	wide
# cow per clips	single	multiple	multiple
# video clips (# for training)	87 (5)	18 (2)	114 (5)
# training frames	100	40	100
# testing frames	585	59	611

4.4.2 System validation experiment

This experiment compares all the internal outputs from our proposed system shown in Figure 4.3, to demonstrate the importance of each individual module. We choose the output of CNN1 as the baseline method, which is the original method in the DeepLab-Cut (DLC) toolbox [65]. However, this method can only detect one object per frame, so for a fair comparison, we only use the videos in Set 1 since these only contain one cow object. We compare the baseline method with four other internal outputs from the system: the CNN2 output from the difference videos, the CNN1 output plus the Post-Processing (PP) stages, the CNN2 output plus the PP stages, and the final merged result.

The implementation details are explained below. The frame difference images are generated by the sum of differences between the current frame and both the previous and next frame. The training labels from Set 1 are used to fine-tune both CNNs in the system. Recall that CNN1 processes the color images and CNN2 processes the frame-difference images. Both networks are pre-trained on ImageNet [133] and their final up-sampling layers are fine-tuned with our cow images. For the two CNN methods without PP stage, we follow the extraction method from the DeepLabCut toolbox by setting a hard threshold and finding the location in the confidence maps with the maximum probability.

Both supervised and unsupervised evaluation metrics are used, but their testing data are different. For unsupervised measures, we compare the Valid Cow Percentage (VCP) and Temporal Consistency (TC) for all the frames in the testing videos because no labels are required. But for supervised measures, only the 585 labelled testing frames are used for evaluation. Among these labelled images, we report the body F1 score and leg-hoof F1 score, and the VCP score is also computed to compare the cow detection capability of each module in the system. Both qualitative and quantitative results are presented below.

Figure 4.5 shows an example of all five outputs of one testing image in Set 1. The direct outputs from the two CNNs without post-processing (top middle and bottom left) miss-detect some body parts, because they apply the strategy from the original DLC method that only selects one maximum point. Our proposed post-processing module uses non-maximum suppression to select all local maximum values from the confidence map, and all body-part candidates are detected (see bottom right of Figure 4.5). Considering the leg-hoof points, some joints of the swing leg are missed by CNN1 based on color image, because of motion blurriness and heavy compression. But these points are

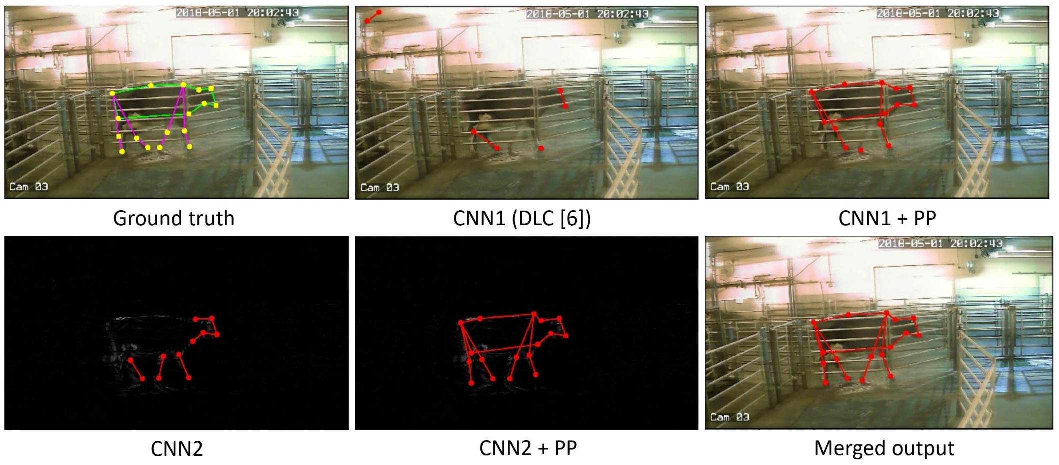


Figure 4.5.: The outputs of different stages in the proposed system.

detected by CNN2 using the frame difference image, and the merged result generates a complete cow structural model.

Table 4.2.: Comparison of the outputs of the system components on Set 1 videos (single-cow). Notice smaller TC value means smoother object movement in the video. Bold numbers show the best performance method in each column.

	Unsupervised		Supervised		
	VCP	TC	VCP	Body F1	Leg-hoof F1
CNN1 (DLC [65])	0.447	102.8	0.714	0.260	0.391
CNN2	0.408	155.0	0.673	0.366	0.252
CNN1+PP	0.632	**8.92**	0.846	0.772	0.373
CNN2+PP	0.667	10.19	0.929	0.841	0.333
Merged output	**0.705**	9.0	**0.960**	**0.879**	**0.434**

The numerical comparison results are presented in Table 4.2. In general, our complete system (last row) improves the performance compared to the method in the DLC toolbox (first row). It can be observed that adding a Post-Processing (PP) module largely improves the system performance. The temporal and spatial prediction in the PP module improves the cow-detection ability demonstrated by the increasing VCP scores. Notice the two VCP scores from supervised measure and unsupervised measures are not comparable because their test sets are different. In addition, the temporal filtering process in the PP module largely improves the Temporal Consistency (TC), because the original CNN method purely operates on an image without considering temporal information. Comparing two F1 scores in the supervised measures, the PP step improves the detection accuracy for the cow structural model because more body-part candidates are selected from the intermediate CNN output.

Comparing the first two rows from the table, we can see CNN2 has better performance than CNN1 for the cow body region but works poorly on the leg and hoof regions. As explained in Section 4.2.2, CNN2 operates on gray-scale edges generated by the frame difference and better captures smoothly moving objects like the body region. But it cannot work in isolation because it eliminates too much information contained in the original images, such as the stationary legs. As a result, merging the two networks together obtains better detection for the leg-hoof region points.

4.4.3 Robustness evaluation experiment

This experiment evaluates the system robustness with different datasets. Training-based detection methods normally perform worse when they are applied to testing data that is substantially different from the training set. In this experiment, we evaluate the performance of our system when testing on frames collected from the three different cameras, that capture the same region of the farm but with different capture angles. This experiment also explores the influence of image quality on our system, since the video qualities from the 3 sets are also different.

For the training images in each video set, we fine-tune three detection systems, $S1$, $S2$, and $S3$, based on each individual corresponding dataset, respectively. An extra system S_all is trained on all the training frames together. In the testing phase, each trained system is applied to the images from the three sets separately. We also test each system on all testing images together for an overall comparison. All training and testing data are separated regardless of their dataset, and no images used for both training and testing. In total, there are 4 trained models testing on 4 groups of test sets, which forms 16 training/testing pairs. For each pair, we measure the final system output using supervised metrics: body F1 score and leg-hoof F1 score. Table 4.3 shows the comparison results.

Table 4.3.: System performance comparison on different video sets. The bold numbers show the best performance of each column.

Trained system	Body F1 score on				Leg-hoof F1 score on			
	Set1	Set2	Set3	All	Set1	Set2	Set3	All
$S1$	0.80	0.42	0.51	0.64	0.61	0.18	0.35	0.46
$S2$	0.72	**0.65**	0.58	0.65	0.16	0.59	0.33	0.26
$S3$	**0.82**	0.56	0.59	0.69	0.61	0.52	**0.56**	0.58
S_all	**0.82**	0.64	**0.61**	**0.71**	**0.62**	**0.65**	**0.56**	**0.59**

In Table 4.3, each row represents a system trained from one dataset, and each column shows the system performance on one corresponding test set. Comparing the four systems, it can be observed that S_all achieves similar and slightly better performance than the others, and this merged system even works better than when each individual system is both trained and tested on its own videos (diagonal values). This demonstrates that adding training data from other similar video sets helps to improve the detection performance.

The results in Table 4.3 also allow us to examine the performance of the method when the input videos have different qualities. While both Set 1 and Set 2 have low quality, the images in Set 2 (see Figure 4.6) have a small spatial resolution while the images in Set 1 (see Figure 4.5) are blurry with poor illumination. Therefore, the results of system $S1$ on Set 2 and of system $S2$ on Set 1 images are poor, especially for the leg and hoof regions. However, system $S3$, which is trained on high quality images, provides better results on both these two datasets. This demonstrates that using higher quality images or increasing the variation of training data can improve system performance.

A final observation from the table is that the body region F1 scores are more stable across different systems than the leg-hoof F1 scores, due to the post-processing module that only operates on the body region. The spatial and temporal prediction in the post-processing model improves the estimates of missing and incorrectly detected points,

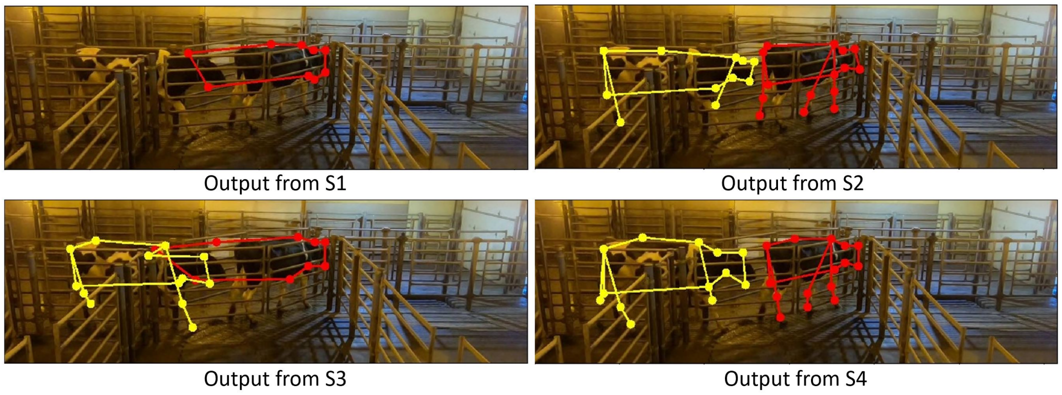


Figure 4.6.: The detection comparison between systems trained on different video sets. This example image is from Set 2.

which compensates for poor CNN performance. Since the legs and hooves are estimated directly from the CNN outputs, the performance variation is primarily due to the variation of training data.

In addition to numerical comparison, in Figure 4.6 we also present some visual results from all 4 trained systems applied to a test image from Set 2 that contains two cows. Comparing the outputs, system $S1$ fails to detect two cow objects and $S3$ is confused with some body parts between the two cow objects. However, system $S2$ and S_all both detect two cow objects and present an accurate cow shape, because these two systems are both trained with data from Set 2. But the merged result from S_all is more accurate on some body parts, for example the points on each cow's back, because of the additional training data involved. However, for the leg and hoof region, none of the systems detect all the points, due to the difficulty of observing them and the lack of post-processing process.

4.4.4 Detection performance evaluation

This experiment compares the detection performance between our system and other popular object detection methods. Recall that the motivation for our system is not only to segment the spatial location of the cow, but also to detect critical keypoints about its body parts. Therefore, ideally comparison methods should also target these two goals. However, as mentioned in Section 1.6, most previous keypoint detection methods focus on human objects and incorporate knowledge about human body parts, and it is difficult to adapt them to cow bodies for a comparison. On the other hand, there are many popular object detection methods which can be fine-tuned to segment cows, and these make for an effective comparison. In this experiment, we compare the cow object detection performance between our system and other three popular pixel-wise object detection methods: One Shot Video Object Segmentation (OSVOS) [41], DeepLab [50], and Mask R-CNN [49].

Structural model to mask

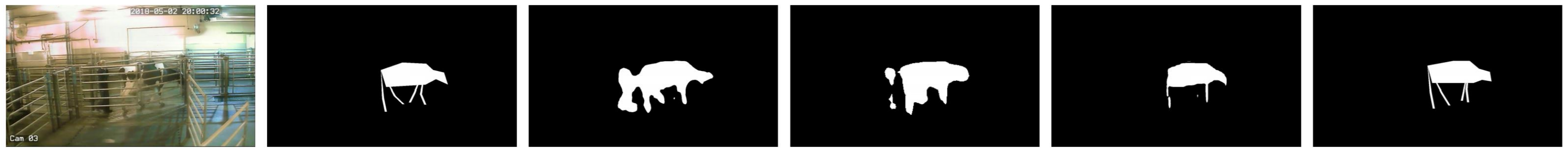
To create a performance comparison that does not disadvantage the object detection methods, we convert the output of our structural model into a binary cow mask, with two steps. First, all keypoints from the upper body region are connected to form a closed area representing the cow body. Second, every leg-hoof limb is expanded from a line into a polygon with a horizontal width of 10 pixels, as shown in the second column of Figure 4.7. This expansion process is applied to both the ground-truth labels and the detection results. The newly expanded ground-truth masks are then used to fine-tune the object detection methods, as well as to compute performance metrics. Still the point-to-mask conversion is not perfect. Notice the approximated masks cannot exactly cover the cow object from the original image; see for example the inaccurate edges of the cow body and the straight legs.

Performance comparison

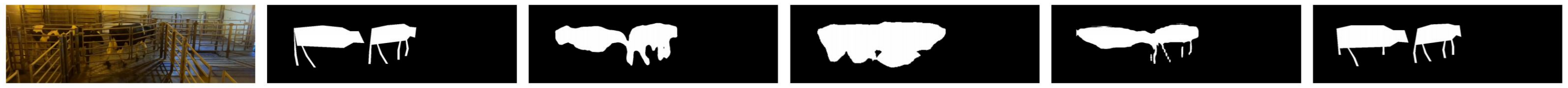
We use all the training and testing data from the three video sets in this experiment. In total, there are 240 single-cow frames for training and 1255 images for testing. Each of the three comparison methods are fine-tuned with the approximate cow masks, with different implementation details. For OSVOS [41], we use the parent network pre-trained on the DAVIS 2016 [42] dataset and fine-tune it with our data. The output results are binarized using Otsu [137] threshold. For DeepLab [50], we use the pre-trained network from the COCO dataset [58], and we modify the last layer to produce two classes: cows and background. The fine-tuning process is applied only on the last atrous spatial pyramid pooling layers with binary entropy loss. For Mask R-CNN [49], we use the network pre-trained on the COCO dataset and fine-tune its region proposal network and feature pyramid network. The classifier outputs are also adjusted to the two classes of cows and background.

Figure 4.7 shows some visual examples of the detection results. From left to right are the original image, ground truth, and the results from OSVOS, DeepLab, Mask R-CNN, and our system, respectively. Each row shows an example which is selected from a different test set. Example (a) includes a human wearing black clothes who is walking right behind the cow. This confuses OSVOS which considers it to be part of the moving foreground object. Example (c) shows a special case which contains a pure white cow, and this color is not present in the training data. The DeepLab method completely misses the cow, because it directly extracts information from the color image and this rare color has not been seen before. The OSVOS method detects part of this cow using motion information, but Mask R-CNN works well because its region proposal network determines there is an object candidate and segments the cow object correctly.

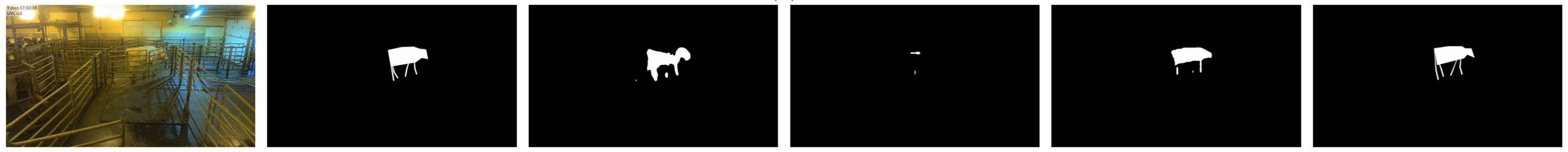
Examples (b) and (d) contain multiple cows, and each method does detect multiple cow objects. However, the three masked-based methods merge all detected cow objects together because the objects are close to each other, and we need further effort to count the number of cows or to extract other detailed information. But our result provides a clear delineation between the cow objects, due to the use of the structural model. Another observation about these two examples is that the cow positions in these two images are different. Some cows are in the middle with fewer fences and others are on either the left or right side with denser fences blocking the view. Every method can detect the middle cow, but the cows on the sides are more difficult due to the obstacles. We further analyze the influence of fences in later paragraphs.



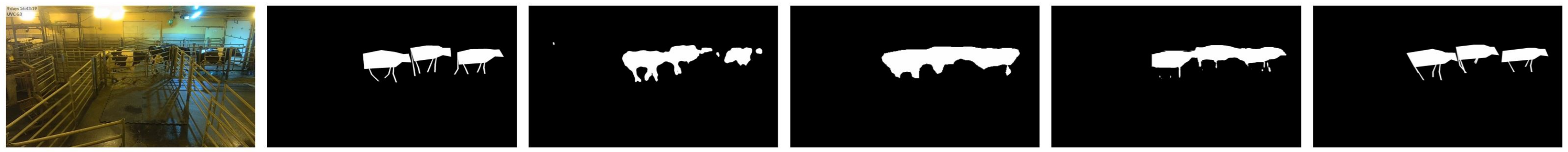
(a)



(b)



(c)



(d)

Figure 4.7.: Results using different detection methods. From left to right: original image, ground truth, OSVOS [41], DeepLab [50], Mask R-CNN [49], and ours. Example (a) and (b) are from Set 1 and Set 2, respectively; example (c) and (d) are both from Set 3.

Table 4.4.: Comparison of methods on different test sets.

	Set 1	Set 2	Set 3	All
OSVOS [42]	0.571	0.580	0.570	0.571
DeepLab [50]	0.655	0.513	0.577	0.610
Mask R-CNN [49]	0.735	**0.692**	0.630	0.682
ours	**0.750**	0.668	**0.662**	**0.703**

Numerical comparison results among the methods are also reported using the F1 scores of the IOU between the detection results and the ground-truth masks. The measures are reported based on every test set separately in Table 4.4, and on distinct subsets of Set 3 in Table 4.5.

From Table 4.4, it can be observed that our method achieves the highest accuracy for most sets, although its performance relative to the fine-tuned Mask R-CNN is similar. There are three factors which may influence these scores. First, when comparing the masks using IOU, we use a merged mask containing both the cow body and leg regions. Since the body region occupies a larger area of the ground-truth mask, the IOU score can still be high even if the legs are miss-detected. Second, because the masks for our method and the ground truth are both converted from keypoints, it is highly sensitive to the positions of the keypoints, especially for the narrow leg regions. Small position shifts can lead to a large change to the converted mask, which will influence the IOU score. Third, when our system does not detect a leg or hoof point, the mask will be empty in this region. This will also decrease the IOU of our system. Nonetheless, our system performs well in comparison.

Detection comparison under fences

As mentioned above, a main consideration of our system is to obtain acceptable performance even when there are multiple cows, and when there are obstacles like fences. We use Set 3 videos to further explore the influence of these issues, to eliminate any performance variations due to video quality. As Figure 4.7 shows, Set 3 images have a wider

Table 4.5.: Comparison of methods on subsets of Set 3. *Middle* means the cow is in the image center which has fewer obstacles, while *Side* means the cows are on the two sides with denser fences.

	Middle	Side	Single-cow	Multiple-cows
OSVOS [42]	0.672	0.589	0.650	0.547
DeepLab [50]	0.644	0.537	0.616	0.518
Mask R-CNN [49]	**0.749**	0.574	0.703	0.520
ours	0.734	**0.645**	**0.711**	**0.587**

view of the walking path, and cows in the center have fewer fences while cows on the left or right sides are blocked with denser fences. So we separate the testing frames from Set 3 into four subsets: cows in the middle, cows on either side, single-cow frames, and multiple-cow frames. Among the four subsets, images with cows in the middle and with a single cow set will be easier than images from the other two subsets. The qualitative comparison F1 scores of these subsets are shown in Table 4.5. From the table, Mask R-CNN has better performance on the easier test case when the cows are blocked by fewer fences. But for difficult test sets like denser obstacles, our proposed system works better. The OSVOS method also performs well when there are more obstacles because this method only considers the foreground and background, which allows it to separate the stationary fences from the moving cows.

Experiment discussion

In general, compared to the other three mask-based object detection methods, our proposed system has three advantages. First, based on the keypoints detection, our method can correctly detect the cow structure even when the cows are behind the fences or there are humans nearby. Second, when there are multiple cow objects, this system can explicitly isolate each cow even when they are close to each other. Third, it can detect cows with color patterns that do not exist in the training data through the use of frame difference images. However, our system also has two limitations. First, the cow structural model completely depends on the accuracy of the body parts, and one inaccurate detection can cause large errors for the body contour and influence the overall spatial location. Second, the prediction system in our method is based on the keypoint constraints from the cow structural model, which is fixed after the training process. So if there are not enough cow body parts detected, the prediction system still forces the results to conform to a particular shape, which could cause incorrect results.

4.5 Application 1: cow weight estimation

This section introduces a video analytic application for cow weight estimation, which applies the side view cow structural model proposed in previous sections. We first introduce the target and motivation of this application in subsection 4.5.1. Then we explain our developed weight estimation system in subsection 4.5.2. Next, we present three experiments in the following subsections.

4.5.1 Motivation

As explained in previous sections, dairy cows are one of the production animals which is economically important. One of the major purposes of monitoring cows is weight estimation [138]. However, accurately measuring the cow's weights using animal weighing devices requires large amount of human efforts for large dairy farms. For years, people are discovering easy-to-measure body features that can predict the cow's weight. The heart girth [139–141], withers height [142,143], and body length [142] are the most widely used features, which are considered closely correlated to the cow's weight. Figure 4.8 shows the examples of some these features. Inspired by this idea, it is possible to measure these visual features based on our proposed cow structural model from Section 4.1. Based on these features, we can predict the cow's weights automatically without extra human labor.

In the literature, many systems are proposed to automatically measure the cow weights using visual information. To achieve accurate measurements, many methods include the specifically-designed cow capturing environment. For example, [145] designs a cow-

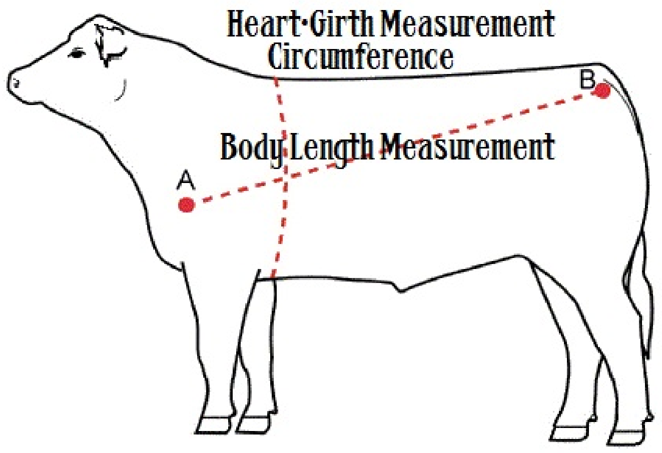


Figure 4.8.: Typical weight estimation features from [144].

image taking unit with a multi-camera system covered with good lighting conditions. To capture more robust cow visual features, [146] designed a 3D box over the weighing scale, so that the captured videos and images have less noise. Researchers from [138] also install depth cameras on top side to capture the height of the joints on the cow's back, which could be further applied for body condition scores estimation [147]. However, all these methods require either building a special hardware system or install expensive depth cameras. Extra efforts are required for the farmers to control the cows to walk through the weighing unit.

However, in this section, our proposed weighing estimation method is purely automated without any human involvement. The videos are captured under the normal daily work of a typical dairy farm using commercial surveillance cameras. As Chapter 2 explains, there are four cameras installed in the dairy farm at different positions. This method uses two viewing angles from the system: the front view which captures the face and back of the cow, and the side view which we can extract the cow structural model proposed from Section 4.1. We extract six different human-interpretable cow body features and fit them into a linear model to estimate the body weights.

4.5.2 Weight estimation system

This section introduces the general flowchart of our proposed weight estimation system. Figure 4.9 shows the system pipeline. It takes input of two sets of videos captured from different viewing angles, front-view and side-view, which are the top and bottom branch in the figure. For both branch from left to right, first we need to locate the cow object in the video sequence including the useful frames and the cow's spatial position. Next, based on the viewing angle and their spatial positions, several visual features are defined and extracted. Finally, all features from both views are merged together to fit a weight estimation model, which predicts the cow weight.

Two videos provide different views of the cow and their target body features also varies, as a result, we use two separate branches to process them. For the front-view, the cow body width is the main feature to extract from this angle, and we design a spatial-temporal segmentation process to only select the frames where the cow's spatial location are within a particular region. But for the side-view videos, more detailed information can be extracted, such as the body length and height. This detailed information requires the location of some specific joints from the cow's body, which requires a side-view cow structure based on keypoints. In this case, our previously-proposed keypoint-based detection system [148] is applied to extract the joint locations. The detailed feature extraction process is explained in the next section.

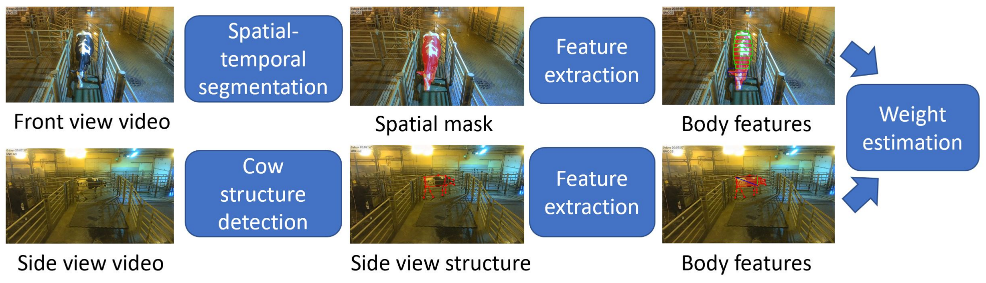


Figure 4.9.: The general flowchart of the weight estimation system.

Front-view video feature

This section introduces the procedure to process the front-view videos. This viewing angle captures the cow walking from a front top side, which clearly sees a complete cow including both the head region and the back of body region. Weight-related features such as the cow's back width and size can be measured in the unit of image pixels.

The fundamental step for cow feature extraction is to locate the cow objects from the video sequences. Because of the fixed camera position, a fixed Region Of Interest (ROI) is defined to locate the walking path (black box in Figure 4.10), which is further divided into 4 vertical boxes (black dash lines) from top to bottom. Within this ROI, we apply Gaussian Mixture Model to detect background changes within every vertical box. We assume a cow is included in this area if all four black boxes have the changing background, and these frames are selected for further analysis. After the temporal segmentation, all the selected frames for spatial segmentation. Since our target feature is mainly the body width, a pixel-level cow mask is needed. The Mask-RCNN [49] is applied to detect the spatial position, which is shown in red shaded area in Figure 4.10. Notice that the Mask-RCNN is pre-trained on COCO [58] which includes a cow category.

Based on the detected cow mask, we can extract the body width $\boldsymbol{w}$ directly. We vertically divided the cow's mask into n rectangulars, and each rectangular has the width of the maximum length of the cow's mask at that vertical position. As shown in Figure 4.10, the red cow mask is equally cut into small rectangulars. However, only the width of the body region is needed, and we select top m green boxes and record the median of their length of pixels as feature body width $\boldsymbol{w}$. In later experiments, we choose n as 14 and m as 9 in the system. Apart from the body length, we also record the area of the mask as another feature represented by $\boldsymbol{a\_f}$, which is the area of the red mask in Figure 4.10. After all the selected frames from the video are processed, we average both feature values from all the frames as the final features to represent this cow in the video.

Note that the angle of this surveillance camera is not straight down, and the width on the image are not consistent due their distance to the camera. As a result, an extra

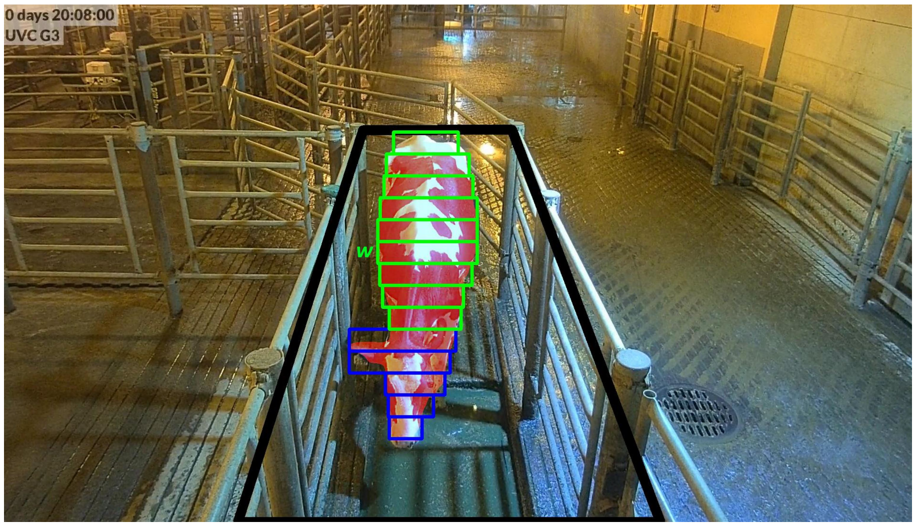


Figure 4.10.: The feature analysis for the front-view video. Notice the body width ***w*** is the median length of the green rectangulars, and another feature is the front area ***a_f*** shown in red.

process is added to normalize the body width features. We use the shape of the ROI box as reference because its left and right edges, or the fences, are parallel and the length between edges are the same. Based on this fact, different scale factors are computed based on the pixel vertical position in the frame, and then multiplied to the extracted body width ***w*** features to unify their represented physical distance.

Side-view video analysis

This section introduces the side-view video analysis. Figure 4.11 shows an example frame from the side-view videos, which shows the cows to walk from the left to right side through the path bounded by fences. Unlike the front-view videos, this view provides more body size information related weight estimation such as the like body length and height. In addition, this view does not cause geometric length-measuring problems, because the camera is far from the cows, and it is reasonable to assume the length on the frame is independent of the cow's spatial location. However, detecting cows in this view needs to handle the fences which always partly-block the cow. There are also unrelated background noise that challenges the cow detection, such as the crowned cows in behind shown in Figure 4.11.

As mentioned above, the length measurement on this view requires the location of joints from the cow. Here we apply our previously-proposed cow structure detection system [148]. This method is based on CNNs, which takes the input of a video sequence and extracts the joint keypoints of all possible frames to generate a structural model. Shown in Figure 4.11, the yellow dots around the cow body edges are the detected structural model, which consists of the upper body boundary and the position of all 4 leg limbs. In this model, we mainly focus on the three joints labelled in the figure: the shoulder, tail base, and the bottom front. We define these lines connected by the joints with the following names: the black line is the body length ***l***, the green line is the body height ***h***, and the blue line is called the body diagonal length ***d***. The area of the body region mask shown in red in Figure 4.11 is also used for weight estimation, which is denoted by ***a_s***.

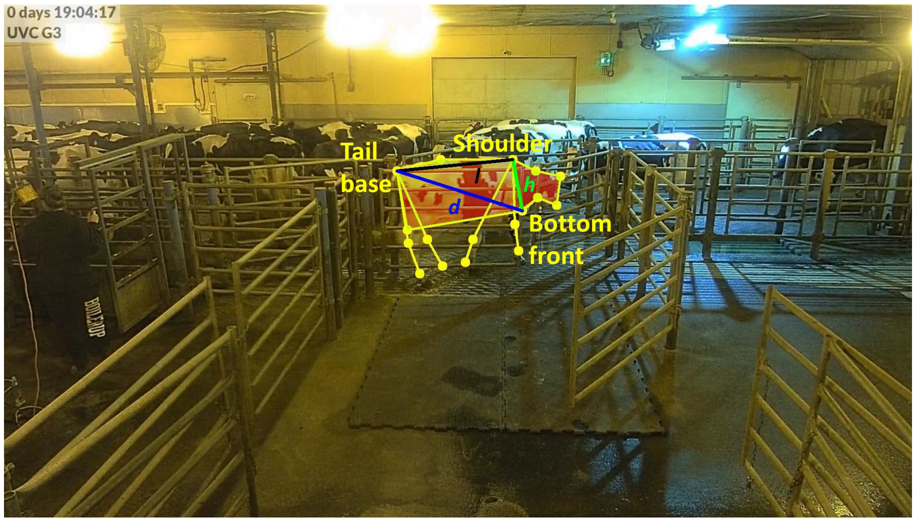


Figure 4.11.: The feature analysis for the side-view video. The yellow color shows the output of cow structural detection system, and the black, green, and blue lines represents the body length ***l***, body height ***h***, and body diagonal length ***d***. The feature side area ***a_s*** is the shown in red.

When applying the structural model detection system, we detect the target body lengths from all possible frames and each frame produces a vector of visual features. To overcome the inconsistency of the detection process on different frames, we computed the median value of every length feature collected from a video sequence and result in a final set of features to represent the cow.

Detection model and fitting

This section introduces the weight estimation model based on visual features. Based on analyzing videos from both viewing angles, six features are extracted. Four of these features represent the physical lengths but in unit of pixels, and another two are the areas of the cow body. All these features have physical meanings and they are generally all proportional to the cow's weight. Here we use a simple linear model to fit the features and predict the Body Weight (BW) shown in Equation 4.5:

$$BW = \sum_i \alpha_i f_i + \beta \tag{4.5}$$

where the f_i is the visual feature, and α_i and β are the parameters. The parameters are estimated based on the training weights using linear regression.

The weighing model could variate by having different choice of visual features, and it is possible to use one single feature only to predict the cow's weight. In later sections, apart from single feature fitting, we mainly consider four different combinations: the front-view features only, the side-view features only, four length features, and all features together. All these combinations are trained and evaluated in later experiment section.

4.5.3 Experiment preparation

All cow videos used in this paper were collected from a local U.S. dairy farm during 2017 to 2019. We install high-quality IP cameras on different positions in the farm, so that we can collect both the front-view and side-view of the walking path. The cameras provide videos with the resolution of 1920*1080 at 30 frames per second. The raw captured videos from both views are temporal segmented into single cow walking clips manually, and they are synchronized with time. As a result, every cow identity is captured by two clips, where the front-view one (Figure 4.10) captures the cow move from far to near, and the side-view one (Figure 4.11) shows the cow move from left to right.

Table 4.6.: Summary of video clips used in the experiments.

Day	# of cows	Min weight lbs.	Mean weight lbs.	Max weight lbs.	weight SD lbs.
1	125	1010	1449.2	1851	167.6
2	105	1132	1460.1	1841	170.1
3	137	1114	1447.9	1857	181.2
4	135	1021	1447.6	1931	183.3
All	502	1010	1447.9	1931	176.6

Most of these single cow walking clips are captured during the monthly weighting process, and the weight of these cows are recorded as ground truth labels in this experiment. As Table 4.6 shows, we separate these clips by their captured days, and the ground truth weight statistics are also presented. Notice each cow identity has two video clips, one from side-view and one from the front-view. In addition, the unit of weights in this paper is pound (lbs.), so all the weight estimation models used later try to fit the visual features into the scale of pound.

Apart from the cow weighting clips, we also extract 100 frames from the side-view clips and label the keypoints of their joints. These labels are used to fine-tune the cow structural model detection system for the side-view video analysis.

4.5.4 Visual features comparison

This experiment evaluates every extracted visual feature individually. From two viewing angles, we select six visual features to estimate the body weight: the median body width w, front-view body area a_f, body diagonal d, body length l, body height h, and side-view body area a_s. All the features are shown in Figure 4.10 and 4.11.

For each feature, two major aspects are evaluated: the statistics of the feature and its correlation with body weight. First, we show the basic value ranges of each feature and their Standard Deviation (SD) to provide a sense of the range of the feature values. Second, the connection between the feature and the body weight is measured using the Pearson Correlation Coefficient (PCC). In addition, we assume every feature has a linear correlation with the body weight, which fits the feature values and weights into a linear function. Then the Root Mean Square Error (RMSE), Mean Absolute Percentage Error (MAPE), and the R square error (R2) are reported to show the fitting errors. The RMSE and MAPE are defined below,

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{4.6}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \tag{4.7}$$

$$r_2 = \frac{\sum_{i=1}^{N}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{4.8}$$

where N is the number of samples, y_i is the ground truth weight, $\hat{y}_i$ is the predicted weight, and $\bar{y}$ is the mean value. Notice, these errors are generated in the processing of linear parameter fitting, and we do not separate the data into training and testing sets in this experiment.

Table 4.7.: Statistics of every single feature and their correlation versus body weights. Notice the unit of the RMSE is lbs..

name	abv	min	mean	max	sd	pcc	RMSE	MAPE	r2
width	w	0.337	0.446	0.522	0.027	0.721	120.1	12.14	0.52
area_f	a_f	8.942	24.24	32.52	2.136	0.504	149.7	11.05	0.254
diagonal	d	232.9	261.8	308.8	10.95	0.55	144.8	11.3	0.302
length	l	192	235.6	276	11.99	0.583	140.8	11.45	0.34
height	h	84.25	98.55	125.5	4.905	0.426	156.8	10.79	0.182
area_s	a_s	23952	29803	35760	2153	0.615	136.6	11.6	0.379

Table 4.7 shows the statistics and correlation of every feature with the body weights. The first two features width and area_f are extracted from the front video videos. Both features are normalized because of the geometric of the viewing angle, so their values are generally smaller and do not have units. Other four features from the side-view videos are all in unit of pixel, which are directly measured from the video frame. Comparing all the features based on the last four columns, it can be observed that the front view width w has the highest correlation with the body weight. It also achieves the minimum RMSE and highest r2. This is mainly because this width is computed based on the median of 9 horizontal bins from the cow's back, which is more robust to errors. In the side-view features, the area of the cow a_s shows the best correlation with the body weight. Among other three length-based features, the side-view body length has less error.

The scattered plots between the feature and weights are shown in Figure 4.12. Among all six features, it can be observed that the front-view width w has the strongest linear correlation with the weight. Two area-based features a_f and a_s have more non-linear correlations. The weight prediction models can be further explored with more complicated non-linear fitting functions, but this is beyond the scope of this thesis. In the next section, we select some combinations based on these features and computer their performance on weight estimation.

4.5.5 Weight estimation experiment

In this experiment, we separate all the weighting data into training and testing sets and evaluate the weighting prediction errors which are tested on unknown dataset. We first compare the feature and the models based on the feature combinations. Then we split the data by different days and simulate a practical situation that we have our fitted model based on current data, and try to estimate cow weights for new collected videos.

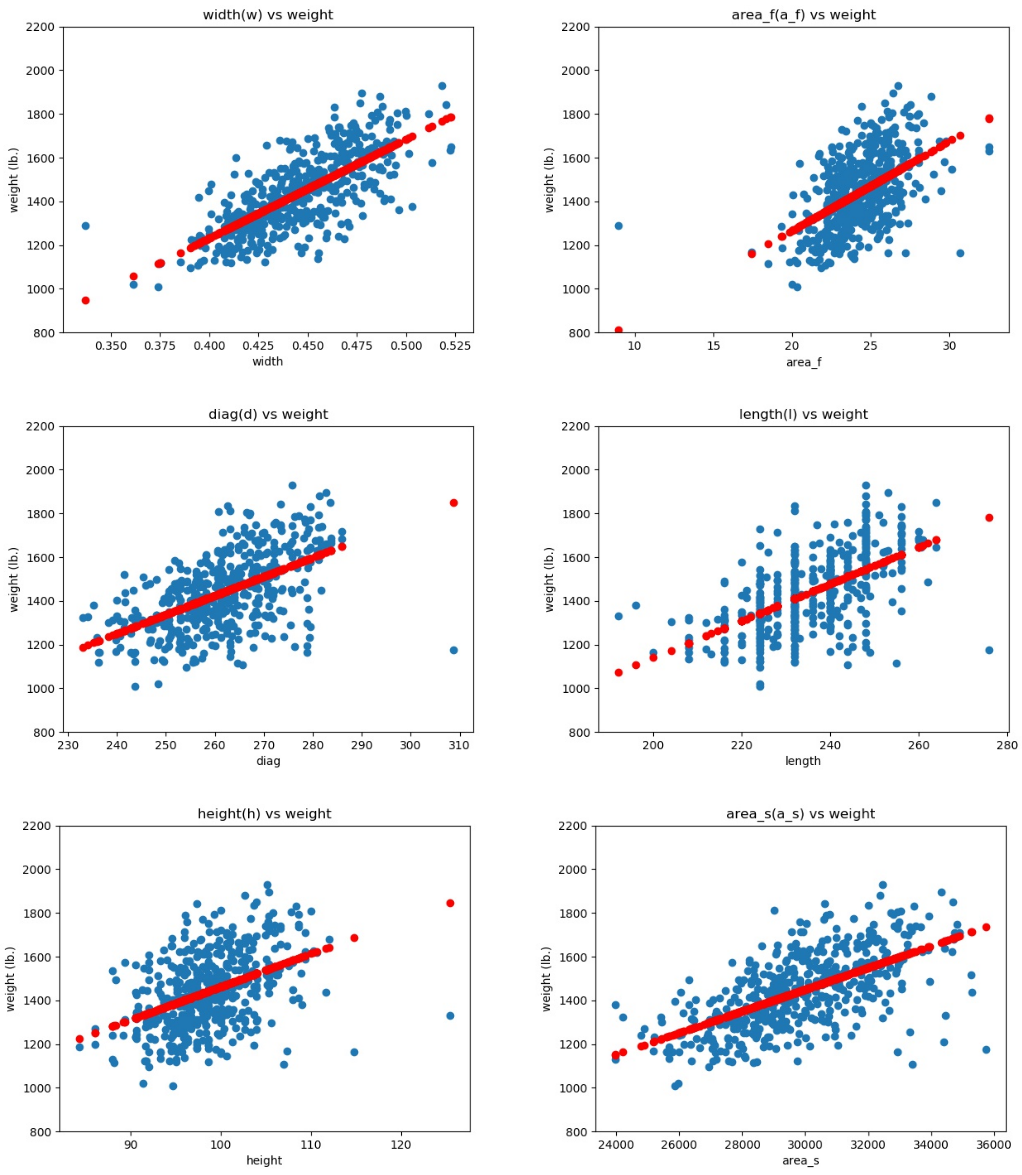


Figure 4.12.: Visual features versus the cow body weight. The red dots show the predicted weight using linear model.

Test with different feature combinations

In this subsection, we randomly separate features to evaluate the performance of different linear models using multiple features. There are 502 samples collected from all four different days, and we randomly split these samples at the percentage of 80%

and 20% at each test. In other words, 400 samples are used to fit the linear data model proposed in Section 4.5.2, and 102 testing samples are used to compute the prediction errors. In the testing process, we report the RMSE and MAPE score to evaluate the prediction errors. In addition, the prediction Error Percentile (EP) are also reported. The EP is computed by sorting the prediction errors and select the values at corresponding percentile in an increasing order. In the experiment, we report EP at 50% and at 95%. The test is repeated for 100 times, and every value reported are averaged over all the tests.

Different feature models are considered in this experiment. Apart from the single features, we also select their combinations, such as the two features from the front-view video, four features from the side-view video, four line-based features, and all features together. All combination of features are merged using the weighted linear sum shown in Equation 4.5. The prediction results are shown in Table 4.8.

Table 4.8.: The weight prediction test on different feature models. Notice the units are all in lbs. Notice that the training sample number and testing sample number in this table are 400 and 102.

model	RMSE	MAPE	EP50	EP95
w	118.7	6.557	79.43	234.3
a_f	146.9	8.449	108.8	268.4
d	141	7.893	93.42	269.3
l	135.5	7.441	84.98	271.8
h	151.3	8.551	103	276.8
a_s	133.4	7.31	87.4	254.2
w,l	104.6	5.679	65.04	211.5
front only (w,a_f)	118.3	6.568	82.09	231.9
side only (d,l,h,a_s)	126.7	6.93	81.57	246.3
four lines (w,d,l,h)	100.5	5.451	61.96	195
all	**99.39**	**5.406**	**61.72**	**192.7**

From the table, it can be observed that merging features together generally improves the prediction errors. Specifically, the merged model with all six features together achieves the best prediction. There are three observations from this table. First, the front-view features have better predictions than the side-view features, and one reason could the viewing distance. The cows appear far away from the camera, which makes one pixel equal to larger physical distance. So minor pixel error could be amplified which further increases detection noise.

Second, the model based on front-view body width w and side-view body length l is an approximation of the popular physical measuring method using heart girth and body length from [144]. This model predicts a reasonable result with two lines since it is easier to measure these two values physically. However, based on our visual analysis, adding feature dimension improves the prediction results. This shows the potential that for practical weight estimation, measuring more physical quantities of the cow also improves the accuracy and robustness.

Third, the Error Percentile (EP) shows range of the absolute prediction errors. For example, in the last row, this value means half of the prediction errors are less than 61 lbs., which is useful and meaningful in real application. But the 95% prediction error is less than 200 lbs., which is too large to deploying in practices according to the weighting data from Table 4.6. The robustness of the prediction system needs to be improved in the future.

Multi-day prediction experiment

This experiment separates the training and testing data by days to simulate a practical usage. In total we collect data from four different days, and we trained four different linear models based on the combination of all six visual features together (last row from Table 4.8). Each model is trained on the video data from three different days and test on the videos from the other day. We report the RMSE and the mean difference of the errors. This mean difference error is used to check if there are any bias on different days. The estimation results are shown in Table 4.9.

It can be seen from the table that RMSE scores variate between four different models with the range around 20 lbs.. The best results are achieved on testing day 2 videos, which has the largest training samples. This implies that training with more data has

Table 4.9.: The weight estimation experiment based on different days. The unit of the weights are lbs.

Train days	Train N	Test date	Test N	RMSE	mean dif
2,3,4	377	1	125	96.60226	34.69979
1,3,4	397	2	105	90.74556	21.96487
1,2,4	365	3	137	112.6376	-28.1958
1,2,3	367	4	135	106.6216	-22.3129

the potential to improve the prediction results. Another observation is that the mean difference value or bias error is not close to zero. This shows the fact that there could be capturing issues on specific days which influence the feature extraction, such as the lighting condition or the position of the obstacle fences in front.

4.5.6 Discussion and potential improvements

This preliminary weight prediction method has two major advantages. First, the whole measuring system is fully automatic, which means no extra human efforts are required. The videos are captured during the daily milking section, and the system is running in the background and predict the weights automatically. Second, this system uses normal surveillance cameras for video capturing, which is economically efficient. Using pure video data, our best prediction result achieves the RMSE error of 90 to 100 lbs.. This error level also achieves the similar performance comparing to a recent previous work such as [146], which applies complicated hardware system.

However, there are many limitations to this weight prediction system. The most important one is the large prediction error, while the ideal error level suggested by farmers is 30 lbs.. Another limitation is the sensitivity of the system. A good weight prediction system can compare the weights between two cows, but our system has a high comparison error. This issue is more common to observe when comparing two cows with weight difference less than 100 lbs.. In addition, the extracted visual features largely depend on the pose of the walking cows. If special poses happen, for example the cow is turning their head or trying to rotate in the walking path, the detection system will fail and cannot provide the correct visual features.

There are many potential procedures to improve the prediction errors. Based on our current video capturing system, there are four viewing angles available which provides four different views of the cow, but our current method only applies two of them. For example, the top view which captures vertically above the cow could provide more direct and robust body width information. Second, the segmentation methods could be improved to extract visual features more accurately. Some rare situations, such as the cows are blocking each other or an uncommon cow body pose, could be solved with better spatial localization methods. Third, better models could be applied to estimate the cow body weights. For example based on the fitting plots shown in Figure 4.12, some features such as the areas could be fitted with second order functions. In addition, more complicated regression model could be applied, such as neural network.

From a practical point of view, estimating the cow body weights of every walking through cow is not enough. This work could be extended into a fully automated cow monitoring system. One further potential application is cow re-identification (Re-ID). With the help of Re-ID, the estimated weights could be automatically saved into the farm recording system, because we know which cow is walking through and its weight. This can further improve efficiency of a dairy cow farm using cameras and video analytics.

5. CONCLUSION

This chapter summarizes the projects in the thesis and provides some general discussion of video analytics in practical applications. In Section 5.1, we first highlight the key features from the two video processing projects in agricultural industry: farming video processing and animal video analysis. Next we discuss more about the general challenges in practical applications and how we overcome them in Section 5.2.

5.1 Project summary

5.1.1 Farming video project summary

In farming video processing project, we first introduce the multi-sensor data logging system ISOBlueHD in Section 2.1. Next the spatial segmentation problem is defined for farming machine videos. Unlike typical video object segmentation, the goal of segmenting farming videos is to partition the frame into different regions. The practical farming environment also requires the system to be both computationally efficient and easily adaptable to different farming applications. Based on these conditions, three spatial segmentation methods are presented. The first two methods in Section 3.1.2 and 3.1.3 based on thresholding motion features select the specific regions from the frame. The third method from Section 3.1.4 uses a trained random forest classifier. It extracts basic color and motion features, and the system can be quickly trained on machines with limited computational power.

In addition, two video-based farming applications are presented, each focuses on a different region of the image. First, we develop a generalized two-branch pipeline for farming video classification in Section 3.3. The system uses a general video classifier as Branch 1 and merges a specifically-designed classifier in Branch 2 based on domain knowledge. When applying this system to classify farming activity videos, the second branch selects features from the attachment region and improves the overall performance. Apart from farming, this two-branch pipeline can be further applied to video classification problems for other areas with domain knowledge involved.

Another application is to predict and control the header-height of a combine harvester in Section 3.4. This system uses the segmentation results and focuses on analyzing the upcoming field region. Based on crop presence classification, the system estimates the crop amount in the field, which can be further parsed to indicate and adjust the combine header-height. The results show that the crop percentages in the field regions can be successfully detected, and the sensitivity test shows the system can identify the decreasing percentage and it is robust to minor crop detection errors.

5.1.2 Cow health analysis

In cow health analysis project, we design a side-view cow structural model based on the joints or keypoints in Section 4.1. This model shows three levels of information: the spatial location of every cow object in the frame, the mask of the cow's upper body region, and the keypoint positions of some specific joints. A cow structural model is also proposed to automatically detect cow keypoints from the video sequences. The detection system is based on deep learning techniques, and it first extracts all possible keypoints from the frame and then assign them to every cow object. The miss-detected keypoints could be predicted based on spatial and temporal prediction using the keypoints constrains and temporal motion consistency.

We also show some preliminary results of an application related to cow weight estimation. Videos from two different views are collected for cow weight estimation: the front view and the side view. Related cow body features are extracted from two different views, such as the cow body width and cow length. Notice the features from the side-view videos are extracted from the proposed cow structural model directly. We use a linear function to fit the features to cow weights, and the predict weight error range is around 100 Lbs.. Further improvement process is required to decrease the prediction error rate.

5.2 General discussion of practical applications

There are some common challenges for practical video applications. First, the video capturing process is not easy. Depending on the application, there are many limits on the camera selection and placement. One typical challenge for practical application is the limited video data. Second, the collected raw videos require large amount of effort for pre-processing. Temporal segmentation and spatial segmentation are crucial to locate where the target objects or regions are in the long video sequences. The performance and accuracy of the pre-processing steps directly influence the further steps. Third, the ground truth labels are very expensive. For industries with specific domain knowledge, only trained personals or special operations can provide the labels. For example, the weight of cows from the dairy farm is collected based on weighing process which involves large amount of human efforts.

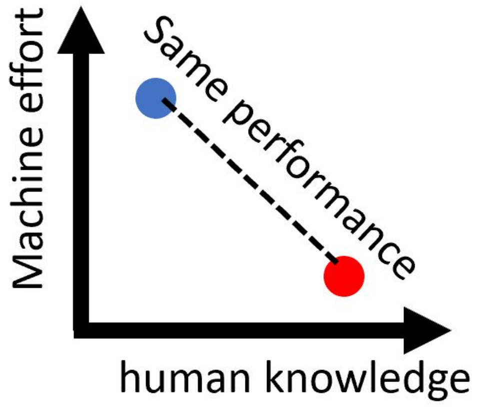


Figure 5.1.: General relation between human knowledge and machine effort.

A typical video analytic system is often designed to read input data and provides the required targets, like event detection or object classification. Nowadays, it is possible to build self-learning systems using large amount of data and labels using a complicated model such as a CNN. In this case, only the data and labels are required to solve the problem, and little human knowledge is needed since it is converted into the labels. As long as the number of data and labels is enough, normal video applications such as classification or event detection can be solved with a reasonable performance.

However, for practical applications which have limited video data and labels, the data-driven models are not suitable. In this case, we can apply the prior human knowledge to reduce the complexity of models and huge requirement of data. In the projects we presented in this thesis, different levels of human knowledge are applied: defining the machinery farming videos into three different regions, hand-crafting motion features for segmentation purposes, and applying the cow body features for weight estimation. All the human knowledge is implemented in the analytic systems, which is more general and robust to data-driven methods, especially when the videos or labels are limited.

Figure 5.1 shows these two type of systems. The machine effort on the Y axis represents the number of data and labels, plus the computation power and model complexity. The human knowledge on the X axis shows how much knowledge used when developing the system. The blue point in Figure 5.1 represents the data-driven systems. It uses less human knowledge but needs large amount of data and computational power. On the other hand, the applications shown in the thesis are represented as the red point. We use domain knowledge to develop the applications, which requires generally less data and labels for industries such as agriculture. Our experimental results from previous chapters show that our systems achieve similar or better performance comparing to data-driven methods.

REFERENCES

REFERENCES

[1] A. King *et al.*, “The future of agriculture,” *Nature*, vol. 544, no. 7651, pp. S21–S23, 2017.

[2] E. Benson, J. Reid, and Q. Zhang, “Machine vision-based guidance system for agricultural grain harvesters using cut-edge detection,” *Biosystems Engineering*, vol. 86, no. 4, pp. 389–398, 2003.

[3] M. O’Connor *et al.*, “Automatic steering of farm vehicles using GPS,” *Precision Agriculture*, pp. 767–777, 1996.

[4] J. Chen, Y. Wang, X. Wang, Y. Wang, and R. Hu, “Development and application of remote video monitoring system for combine harvester based on embedded linux,” in *Seventh International Conference on Electronics and Information Engineering*, vol. 10322. International Society for Optics and Photonics, 2017, p. 1032223.

[5] H. Liu, A. R. Reibman, A. C. Ault, and J. V. Krogmeier, “Video-based prediction for header-height control of a combine harvester,” in *IEEE Conference on Multimedia Information Processing and Retrieval*, 2019, pp. 310–315.

[6] H. Kurita, M. Iida, M. Suguri, and R. Masuda, “Application of image processing technology for unloading automation of robotic head-feeding combine harvester,” *Engineering in Agriculture, Environment and Food*, vol. 5, no. 4, pp. 146–151, 2012.

[7] H. Liu, A. R. Reibman, A. C. Ault, and J. V. Krogmeier, “Video classification of farming activities with motion-adaptive feature sampling,” in *IEEE International Workshop on Multimedia Signal Processing*, 2018, pp. 1–6.

[8] H. Cho *et al.*, “A multi-sensor fusion system for moving object detection and tracking in urban driving environments,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1836–1843.

[9] J. Levinson *et al.*, “Towards fully autonomous driving: Systems and algorithms,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, pp. 163–168.

[10] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, “Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds,” in *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, pp. 241–246.

[11] R. Lenain *et al.*, “High accuracy path tracking for vehicles in presence of sliding: Application to farm vehicle automatic guidance for agricultural tasks,” *Autonomous Robots*, vol. 21, no. 1, pp. 79–97, 2006.

[12] T. Litman, *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

[13] J. Wei, J. M. Snider, J. Kim, J. M. Dolan, R. Rajkumar, and B. Litkouhi, "Towards a viable autonomous driving research platform," in *Intelligent Vehicles Symposium (IV), 2013 IEEE*. IEEE, pp. 763–770.

[14] M. Bojarski *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[15] H. Xu *et al.*, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2174–2182.

[16] X. Song, G. Zhang, F. Liu, D. Li, Y. Zhao, and J. Yang, "Modeling spatio-temporal distribution of soil moisture by deep learning-based cellular automata model," *Journal of Arid Land*, vol. 8, no. 5, pp. 734–748, 2016.

[17] M. F. Kragh, P. Christiansen, M. S. Laursen, M. Larsen, K. A. Steen, O. Green, H. Karstoft, and R. N. Jørgensen, "Fieldsafe: dataset for obstacle detection in agriculture," *Sensors*, vol. 17, no. 11, p. 2579, 2017.

[18] H. Yalcin, "Plant phenology recognition using deep learning: Deep-pheno," in *2017 6th International Conference on Agro-Geoinformatics*. IEEE, 2017, pp. 1–5.

[19] M.-T. Yang and J.-Y. Zheng, "On-road collision warning based on multiple FOE segmentation using a dashboard camera," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 11, pp. 4974–4984, 2015.

[20] L. Fleishman and J. Endler, "Some comments on visual perception and the use of video playback in animal behavior studies," *Acta ethologica*, vol. 3, no. 1, pp. 15–27, 2000.

[21] N. B. Cook and K. V. Nordlund, "The influence of the environment on dairy cow behavior, claw health and herd lameness dynamics," *The Veterinary Journal*, vol. 179, no. 3, pp. 360–369, 2009.

[22] S. Kumar and S. K. Singh, "Visual animal biometrics: survey," *IET Biometrics*, vol. 6, no. 3, pp. 139–156, 2016.

[23] A. Poursaberi, C. Bahr, A. Pluk, A. Van Nuffel, and D. Berckmans, "Real-time automatic lameness detection based on back posture extraction in dairy cattle: Shape analysis of cow with image processing techniques," *Computers and Electronics in Agriculture*, vol. 74, no. 1, pp. 110–119, 2010.

[24] S. Viazzi, C. Bahr, A. Schlageter-Tello, T. Van Hertem, C. Romanini, A. Pluk, I. Halachmi, C. Lokhorst, and D. Berckmans, "Analysis of individual classification of lameness using automatic measurement of back posture in dairy cattle," *Journal of Dairy Science*, vol. 96, no. 1, pp. 257–266, 2013.

[25] X. Song, T. Leroy, E. Vranken, W. Maertens, B. Sonck, and D. Berckmans, "Automatic detection of lameness in dairy cattle-vision-based trackway analysis in cow's locomotion," *Computers and Electronics in Agriculture*, vol. 64, no. 1, pp. 39–44, 2008.

[26] K. Zhao, J. Bewley, D. He, and X. Jin, "Automatic lameness detection in dairy cattle based on leg swing analysis with an image processing technique," *Computers and Electronics in Agriculture*, vol. 148, pp. 226–236, 2018.

[27] A. Ter-Sarkisov, R. Ross, and J. Kelleher, "Bootstrapping labelled dataset construction for cow tracking and behavior analysis," in *2017 14th Conference on Computer and Robot Vision (CRV)*. IEEE, 2017, pp. 277–284.

[28] W. Andrew, C. Greatwood, and T. Burghardt, "Visual localisation and individual identification of Holstein Friesian cattle via deep learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2850–2859.

[29] K. Zhao, X. Jin, J. Ji, J. Wang, H. Ma, and X. Zhu, "Individual identification of Holstein dairy cows based on detecting and matching feature points in body images," *Biosystems Engineering*, vol. 181, pp. 128–139, 2019.

[30] W. Shao, R. Kawakami, R. Yoshihashi, S. You, H. Kawase, and T. Naemura, "Cattle detection and counting in UAV images based on convolutional neural networks," *International Journal of Remote Sensing*, pp. 1–22, 2019.

[31] B. L. Price *et al.*, "Geodesic graph cut for interactive image segmentation," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 3161–3168.

[32] P. Arbelaez *et al.*, "Contour detection and hierarchical image segmentation," *IEEE transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[33] P. F. Felzenszwalb *et al.*, "Efficient graph-based image segmentation," *International Journal of Computer Vision*, vol. 59, no. 2, pp. 167–181, 2004.

[34] M. Grundmann *et al.*, "Efficient hierarchical graph-based video segmentation," in *2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 2141–2148.

[35] W. Wang, J. Shen, R. Yang, and F. Porikli, "Saliency-aware video object segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 1, pp. 20–33, 2017.

[36] W. Wang *et al.*, "Saliency-aware geodesic video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3395–3402.

[37] A. Faktor and M. Irani, "Video segmentation by non-local consensus voting." in *BMVC*, vol. 2, no. 7, 2014, p. 8.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 580–587.

[39] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440.

[40] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan, "Learning features by watching objects move," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6024–6033.

[41] S. Caelles, K.-K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "One-shot video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 221–230.

[42] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 724–732.

[43] F. Galasso, N. Shankar Nagaraja, T. Jimenez Cardenas, T. Brox, and B. Schiele, "A unified video segmentation benchmark: Annotation, metrics and analysis," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3527–3534.

[44] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1742–1750.

[45] Y. Wei, X. Liang, Y. Chen, X. Shen, M.-M. Cheng, J. Feng, Y. Zhao, and S. Yan, "STC: A simple to complex framework for weakly-supervised semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 11, pp. 2314–2320, 2016.

[46] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1796–1804.

[47] Y.-H. Tsai, M.-H. Yang, and M. J. Black, "Video segmentation via object flow," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3899–3908.

[48] Y. J. Lee, J. Kim, and K. Grauman, "Key-segments for video object segmentation," in *IEEE International Conference on Computer Vision*, 2011, pp. 1995–2002.

[49] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2961–2969.

[50] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[51] J. Redmon and A. Farhadi, "YOLO v3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[52] K.-K. Maninis, S. Caelles, Y. Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool, "Video object segmentation without temporal information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 6, pp. 1515–1530, 2018.

[53] P. Tokmakov, K. Alahari, and C. Schmid, "Learning video object segmentation with visual memory," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 4481–4490.

[54] J. Cheng, Y.-H. Tsai, S. Wang, and M.-H. Yang, “SegFlow: Joint learning for video object segmentation and optical flow,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 686–695.

[55] P. Voigtlaender and B. Leibe, “Online adaptation of convolutional neural networks for video object segmentation,” *arXiv preprint arXiv:1706.09364*, 2017.

[56] N. Xu, L. Yang, Y. Fan, J. Yang, D. Yue, Y. Liang, B. Price, S. Cohen, and T. Huang, “Youtube-VOS: Sequence-to-sequence video object segmentation,” in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 585–601.

[57] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, “2D human pose estimation: New benchmark and state of the art analysis,” in *IEEE Conference on Computer Vision and Pattern Recognition*, June 2014.

[58] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[59] A. Toshev and C. Szegedy, “Deeppose: Human pose estimation via deep neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1653–1660.

[60] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation,” in *European Conference on Computer Vision*. Springer, 2016, pp. 483–499.

[61] E. Insafutdinov, M. Andriluka, L. Pishchulin, S. Tang, E. Levinkov, B. Andres, and B. Schiele, “ArtTrack: Articulated multi-person tracking in the wild,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6457–6465.

[62] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, “Realtime multi-person 2D pose estimation using part affinity fields,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 7291–7299.

[63] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, “OpenPose: realtime multi-person 2D pose estimation using part affinity fields,” *arXiv preprint arXiv:1812.08008*, 2018.

[64] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. V. Gehler, and B. Schiele, “DeepCut: Joint subset partition and labeling for multi person pose estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4929–4937.

[65] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, M. W. Mathis, and M. Bethge, “Deeplabcut: markerless pose estimation of user-defined body parts with deep learning,” Nature Publishing Group, Tech. Rep., 2018.

[66] T. D. Pereira, D. E. Aldarondo, L. Willmore, M. Kislin, S. S.-H. Wang, M. Murthy, and J. W. Shaevitz, “Fast animal pose estimation using deep neural networks,” *Nature Methods*, vol. 16, no. 1, p. 117, 2019.

[67] S. Günel, H. Rhodin, D. Morales, J. Campagnolo, P. Ramdya, and P. Fua, “Deepfly3D: A deep learning-based approach for 3D limb and appendage tracking in tethered, adult drosophila,” *bioRxiv*, p. 640375, 2019.

[68] H. Liu *et al.*, “Video classification of farming activities with motion-adaptive feature sampling,” in *IEEE 20th International Workshop on Multimedia Signal Processing (MMSP)*, 2018.

[69] C.-H. Lin *et al.*, “A smart content-based image retrieval system based on color and texture feature,” *Image and Vision Computing*, vol. 27, no. 6, pp. 658–665, 2009.

[70] J. Yue *et al.*, “Content-based image retrieval using color and texture fused features,” *Mathematical and Computer Modelling*, vol. 54, no. 3-4, pp. 1121–1127, 2011.

[71] T. Ahonen *et al.*, “Face description with local binary patterns: Application to face recognition,” *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 12, pp. 2037–2041, 2006.

[72] A. Baraldi and F. Parmiggiani, “An investigation of the textural characteristics associated with gray level cooccurrence matrix statistical parameters,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 33, no. 2, pp. 293–304, 1995.

[73] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[74] P. Dollár *et al.*, “Behavior recognition via sparse spatio-temporal features,” in *2005-2nd Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005, pp. 65–72.

[75] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, “Action recognition by dense trajectories,” in *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3169–3176.

[76] J. Liu *et al.*, “Recognizing realistic actions from videos “in the wild”,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1996–2003.

[77] V. Kantorov and I. Laptev, “Efficient feature extraction, encoding and classification for action recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 2593–2600.

[78] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *2013 IEEE International Conference on Computer Vision (ICCV)*, pp. 3551–3558.

[79] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 886–893.

[80] I. Laptev *et al.*, “Learning realistic human actions from movies,” in *2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1–8.

[81] N. Dalal, B. Triggs, and C. Schmid, “Human detection using oriented histograms of flow and appearance,” in *European Conference on Computer Vision (ECCV)*, 2006, pp. 428–441.

[82] A. Klaser, M. Marszałek, and C. Schmid, “A spatio-temporal descriptor based on 3d-gradients,” in *2008-19th British Machine Vision Conference (BMVC)*. British Machine Vision Association, pp. 275–1.

[83] I. C. Duta *et al.*, “Histograms of motion gradients for real-time video classification,” in *14th International Workshop on Content-Based Multimedia Indexing (CBMI)*, pp. 1–6.

[84] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[85] H. Bay, T. Tuytelaars, and L. Van Gool, “Surf: Speeded up robust features,” in *2006 European Conference on Computer Vision (ECCV)*. Springer, pp. 404–417.

[86] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *IEEE International Conference on Computer Vision*, 2011, pp. 2564–2571.

[87] I. Laptev, “On space-time interest points,” *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[88] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, “Places: A 10 million image database for scene recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1452–1464, 2018.

[89] A. Karpathy *et al.*, “Large-scale video classification with convolutional neural networks,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1725–1732.

[90] K. Hara, H. Kataoka, and Y. Satoh, “Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet?” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6546–6555.

[91] J. Sivic and A. Zisserman, “Video google: A text retrieval approach to object matching in videos.” IEEE, 2003, p. 1470.

[92] H. Wang *et al.*, “Evaluation of local spatio-temporal features for action recognition,” in *2009-British Machine Vision Conference (BMVC)*. BMVA Press, 2009, pp. 124–1.

[93] L. Shao *et al.*, “Spatio-temporal Laplacian pyramid coding for action recognition,” *IEEE Transactions on Cybernetics*, vol. 44, no. 6, pp. 817–827, 2014.

[94] J. C. Niebles *et al.*, “Modeling temporal structure of decomposable motion segments for activity classification,” in *2010 European Conference on Computer Vision (ECCV)*, pp. 392–405.

[95] F. Perronnin, J. Sánchez, and T. Mensink, “Improving the Fisher kernel for large-scale image classification,” in *2010 European Conference on Computer Vision (ECCV)*, pp. 143–156.

[96] A. Vedaldi and B. Fulkerson, “VLFeat: An open and portable library of computer vision algorithms,” http://www.vlfeat.org/, 2008.

[97] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 568–576.

[98] K. Soomro *et al.*, “UCF101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.

[99] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[100] T. Kounalakis, G. A. Triantafyllidis, and L. Nalpantidis, "Deep learning-based visual recognition of rumex for robotic precision farming," *Computers and Electronics in Agriculture*, vol. 165, p. 104973, 2019.

[101] H. Kuehne *et al.*, "HMDB: a large video database for human motion recognition," in *2011 IEEE International Conference on Computer Vision (ICCV)*, pp. 2556–2563.

[102] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre.

[103] M. Marszalek, I. Laptev, and C. Schmid, "Actions in context," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 2929–2936.

[104] J. Xiao *et al.*, "SUN database: Large-scale scene recognition from abbey to zoo," in *2010 IEEE conference on Computer vision and pattern recognition (CVPR)*, pp. 3485–3492.

[105] N. Sainz-Costa *et al.*, "Mapping wide row crops with video sequences acquired from a tractor moving at treatment speed," *Sensors*, vol. 11, no. 7, pp. 7095–7109, 2011.

[106] F. Rovira-Más *et al.*, "Hough-transform-based vision algorithm for crop row detection of an automated agricultural vehicle," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 219, no. 8, pp. 999–1010, 2005.

[107] M. Li *et al.*, "Review of research on agricultural vehicle autonomous guidance," *International Journal of Agricultural and Biological Engineering*, vol. 2, no. 3, pp. 1–16, 2009.

[108] Y. Xie *et al.*, "Fundamental limits in combine harvester header height control," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 3, p. 034503, 2013.

[109] Y. Xie and A. Alleyne, "Two degrees of freedom control for combine harvester header height control," in *ASME 2012 5th Annual Dynamic Systems and Control Conference joint with the JSME 2012 11th Motion and Vibration Conference*. American Society of Mechanical Engineers, 2012, pp. 539–547.

[110] G. Lopes *et al.*, "Ae–automation and engineering technologies: Optimal header height control system for combine harvesters," *Biosystems Engineering*, vol. 81, no. 3, pp. 261–272, 2002.

[111] A. W. Layton, A. D. Balmos, S. Sabpisal, A. Ault, J. V. Krogmeier, and D. Buckmaster, "ISOBlue: An open source project to bring agricultural machinery data into the cloud," in *2014 Montreal, Quebec Canada July 13–July 16, 2014*. American Society of Agricultural and Biological Engineers, 2014, p. 1.

[112] H. Jiang, G. Zhang, H. Wang, and H. Bao, "Spatio-temporal video segmentation of static scenes and its applications," *IEEE Transactions on Multimedia*, vol. 17, no. 1, pp. 3–15, 2015.

[113] Y. Poleg, C. Arora, and S. Peleg, "Temporal segmentation of egocentric videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2537–2544.

[114] M. Pal, "Random forest classifier for remote sensing classification," *International Journal of Remote Sensing*, vol. 26, no. 1, pp. 217–222, 2005.

[115] P. L. Correia and F. Pereira, "Classification of video segmentation application scenarios," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 5, pp. 735–741, 2004.

[116] L. Onofri, P. Soda, M. Pechenizkiy, and G. Iannello, "A survey on using domain and contextual knowledge for human activity recognition in video streams," *Expert Systems with Applications*, vol. 63, pp. 97–111, 2016.

[117] M. H. Kolekar, "Bayesian belief network based broadcast sports video indexing," *Multimedia Tools and Applications*, vol. 54, no. 1, pp. 27–54, 2011.

[118] M. Kafai and B. Bhanu, "Dynamic bayesian networks for vehicle classification in video," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 100–109, 2012.

[119] S. Petscharnig and K. Schöffmann, "Learning laparoscopic video shot classification for gynecological surgery," *Multimedia Tools and Applications*, vol. 77, no. 7, pp. 8061–8079, 2018.

[120] X. Peng *et al.*, "Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice," *Computer Vision and Image Understanding*, vol. 150, pp. 109–125, 2016.

[121] J. M. Chaquet, E. J. Carmona, and A. Fernández-Caballero, "A survey of video datasets for human action and activity recognition," *Computer Vision and Image Understanding*, vol. 117, no. 6, pp. 633–659, 2013.

[122] G. Hee Lee, F. Faundorfer, and M. Pollefeys, "Motion estimation for self-driving cars with a generalized camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2746–2753.

[123] J. Uijlings *et al.*, "Video classification with densely extracted HOG/HOF/MBH features: an evaluation of the accuracy/computational efficiency trade-off," *International Journal of Multimedia Information Retrieval*, vol. 4, no. 1, pp. 33–44, 2015.

[124] C. Liu, "Beyond pixels: exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[125] J. Krapac, J. Verbeek, and F. Jurie, "Modeling spatial layout with Fisher vectors for image categorization," in *2011 IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 1487–1494.

[126] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[127] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[128] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.

[129] M. Hall-Beyer, "GLCM texture: A tutorial v. 3.0 March 2017." https://prism.ucalgary.ca/handle/1880/51900.

[130] A. Pluk, C. Bahr, T. Leroy, A. Poursaberi, X. Song, E. Vranken, W. Maertens, A. Van Nuffel, and D. Berckmans, "Evaluation of step overlap as an automatic measure in dairy cow locomotion," *Transactions of the ASABE*, vol. 53, no. 4, pp. 1305–1312, 2010.

[131] G. Aujay, F. Hétroy, F. Lazarus, and C. Depraz, "Harmonic skeleton for realistic character animation," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation.* Eurographics Association, 2007, pp. 151–160.

[132] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schiele, "DeeperCut: A deeper, stronger, and faster multi-person pose estimation model," in *European Conference on Computer Vision.* Springer, 2016, pp. 34–50.

[133] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[134] H. Whay, "Locomotion scoring and lameness detection in dairy cattle," *In Practice*, vol. 24, no. 8, p. 444, 2002.

[135] X. Li, Y. Qi, Z. Wang, K. Chen, Z. Liu, J. Shi, P. Luo, X. Tang, and C. C. Loy, "Video object segmentation with re-identification," *arXiv preprint arXiv:1708.00197*, 2017.

[136] H. Alt and M. Godau, "Computing the Fréchet distance between two polygonal curves," *International Journal of Computational Geometry & Applications*, vol. 5, no. 01n02, pp. 75–91, 1995.

[137] P.-S. Liao, T.-S. Chen, P.-C. Chung *et al.*, "A fast algorithm for multilevel thresholding," *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING*, vol. 17, no. 5, pp. 713–727, 2001.

[138] M. F. Hansen, M. L. Smith, L. N. Smith, K. A. Jabbar, and D. Forbes, "Automated monitoring of dairy cow body condition, mobility and weight using a single 3d video capture device," *Computers in Industry*, vol. 98, pp. 14–22, 2018.

[139] M. Goe, J. Alldredge, and D. Light, "Use of heart girth to predict body weight of working oxen in the ethiopian highlands," *Livestock Production Science*, vol. 69, no. 2, pp. 187–195, 2001.

[140] A. Nesamvuni, J. Mulaudzi, N. Ramanyimi, and G. Taylor, "Estimation of body weight in nguni-type cattle under communal management conditions," *South African Journal of Animal Science*, vol. 30, no. 4, pp. 97–98, 2000.

[141] M. Lesosky, S. Dumas, I. Conradie, I. G. Handel, A. Jennings, S. Thumbi, P. Toye, and B. M. de Clare Bronsvoort, "A live weight–heart girth relationship for accurate dosing of east african shorthorn zebu cattle," *Tropical Animal Health and Production*, vol. 45, no. 1, pp. 311–316, 2012.

[142] A. J. Heinrichs, G. Rogers, and J. Cooper, “Predicting body weight and wither height in holstein heifers using body measurements,” *Journal of Dairy Science*, vol. 75, no. 12, pp. 3576–3581, 1992.

[143] S. Sawanon, P. Boonsaen, P. Innuruk *et al.*, “Body measurements of male kamphaengsaen beef cattle as parameters for estimation of live weight,” *Kasetsart J.(Nat. Sci.)*, vol. 45, pp. 428–434, 2011.

[144] “How to measure cattle estimating weight of beef cattle with measurements.” [Online]. Available: http://miniature-cattle.com/wt.htm

[145] S. Tasdemir, A. Urkmez, and S. Inal, “Determination of body measurements on the holstein cows using digital image analysis and estimation of live weight with regression analysis,” *Computers and Electronics in Agriculture*, vol. 76, no. 2, pp. 189–197, 2011.

[146] X. Song, E. Bokkers, P. van der Tol, P. G. Koerkamp, and S. van Mourik, “Automated body weight prediction of dairy cows using 3-dimensional vision,” *Journal of Dairy Science*, vol. 101, no. 5, pp. 4448–4459, 2018.

[147] J. Bewley, A. Peacock, O. Lewis, R. Boyce, D. Roberts, M. Coffey, S. Kenyon, and M. M. Schutz, “Potential for estimation of body condition scores in dairy cattle from digital images,” *Journal of Dairy Science*, vol. 91, no. 9, pp. 3439–3453, 2008.

[148] H. Liu, A. R. Reibman, and J. P. Boerman, “A cow structural model for video analytics of cow health,” 2020.

APPENDIX

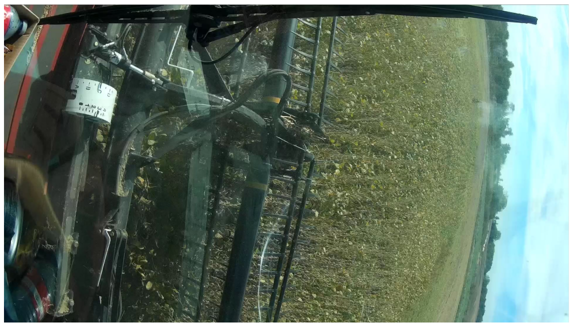
A. APPENDIX FIGURES

(a) Harvest corn(snaplage)

(b) Harvest corn(snaplage)

(c) Harvest corn(chopper)

(d) Harvest corn(chopper)


(e) Combine beans

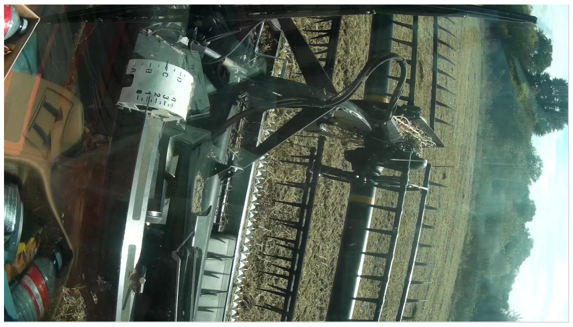

(f) Combine beans

Figure A.1.: The example images captured from framing vehicles (1)

(a) Tillage

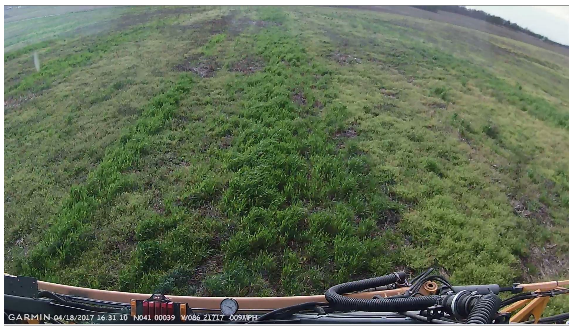
(b) Tillage

(c) Planting

(d) Planting

(e) Spraying (1)

(f) Spraying (1)


(g) Spraying (2)

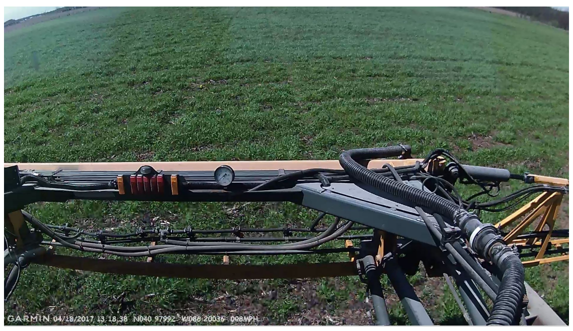
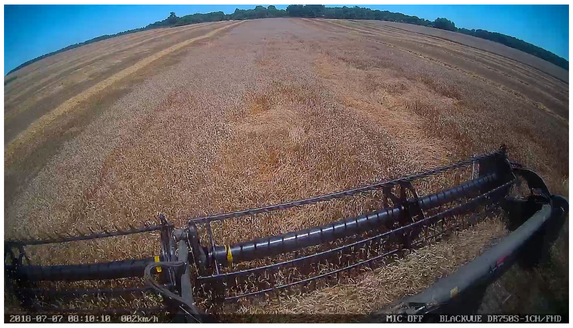

(h) Spraying (2)

Figure A.2.: The example images captured from framing vehicles (2)



(a) Wheat harvesting (1)

(b) Wheat harvesting (2)

(c) Wheat harvesting (1)

(d) Wheat harvesting (2)

(e) Hauling Bales (1)

(f) Hauling Bales (2)

(g) Planting corn (1)

(h) Planting corn (2)

Figure A.3.: The example images captured from framing vehicles (3)