Deep Learning based Crop Row Detection with Online Domain Adaptation

Rashed Doha
Department of Mechanical Engineering,
Indiana University-Purdue University Indianapolis
Indianapolis, Indiana, USA
rdoha@iu.edu

Mohammad Al Hasan
Department of Computer Science,
Indiana University-Purdue University Indianapolis
Indianapolis, IN, USA
alhasan@iupu.edu

Sohel Anwar
Department of Mechanical Engineering,
Indiana University-Purdue University Indianapolis
Indianapolis, Indiana, USA
soanwar@iupui.edu

Veera Rajendran
Equipment Technologies
Mooresville, Indiana, USA
Veera.Rajendran@etsprayers.com

ABSTRACT

Detecting crop rows from video frames in real time is a fundamental challenge in the field of precision agriculture. Deep learning based semantic segmentation method, namely U-net, although successful in many tasks related to precision agriculture, performs poorly for solving this task. The reasons include paucity of large scale labeled datasets in this domain, diversity in crops, and the diversity of appearance of the same crops at various stages of their growth. In this work, we discuss the development of a practical real-life crop row detection system in collaboration with an agricultural sprayer company. Our proposed method takes the output of semantic segmentation using U-net, and then apply a clustering based probabilistic temporal calibration which can adapt to different fields and crops without the need for retraining the network. Experimental results validate that our method can be used for both refining the results of the U-net to reduce errors and also for frame interpolation of the input video stream.

CCS CONCEPTS

• **Applied computing** → **Agriculture**.

KEYWORDS

Crop Row Detection, Semantic Segmentation

ACM Reference Format:
Rashed Doha, Mohammad Al Hasan, Sohel Anwar, and Veera Rajendran. 2021. Deep Learning based Crop Row Detection with Online Domain Adaptation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467155

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
KDD '21, August 14–18, 2021, Virtual Event, Singapore
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467155

1 INTRODUCTION

For site-specific treatments in agriculture, such as, application of herbicides, and fertilizers in the crop-fields, much of the works in the various parts of the world are still done manually via tedious and time consuming human labor. A consequence of this is greater cost, inefficiency, and the inability of on-time treatment, resulting in poor yield, and delay in the final harvest. For this reason, manual labor is increasingly being replaced by precision and autonomous agriculture technologies both in the developing and the developed countries. A vital component of precision agriculture is agricultural sprayers (also called crop sprayers), which are agricultural vehicles that maneuver in between crop rows at various stages of the crop's growth to apply water, herbicides, and fertilizers. A much needed feature in such agricultural vehicles is to detect the locations of crop rows in its view, and provide a visual projection of the crop rows on a dashboard for aiding the drivers in following a safe and optimal route along the crop fields [13]. Such detection of crop rows is also a first step for autonomous navigation of agricultural vehicles in the crop fields. To achieve this successfully, the detection and identification of crop rows from image/video data is the first hurdle—a machine learning and computer vision task, which is the focus of this paper.

Crop row identification in real-life scenario suffers from multiple challenges. For instance, outdoor agricultural environments are susceptible to a diverse range of lighting conditions which contribute to poor quality of the images taken. Furthermore, if the camera is on-board a moving vehicle, the turbulence caused by the motion of the vehicle causes perturbations in the perspective of the camera—another contributing factor to poor image quality. Besides this, high weed density with spectral signature similar to the crop rows can confuse the vision algorithms in their row detection task. Different crops have different row widths; in addition, the growth stage of the crops in the rows adds another degree of freedom to the shape and size of the crop rows; this causes crop rows to have a variable distance between them resulting in inconsistent densities of crop row pixels over different crop row images. Finally, imperfections and occlusions are very common in real world scenarios, as such, ground patches in between crop rows, artifacts in the form of objects that occlude portions of the crop rows and, in general, a diverse range

of pixel densities that are a result of different soil conditions—all make accurate crop row identification a very difficult task.

In existing works, different computer vision based approaches have been adopted as possible solutions in dealing with diverse range of crop row images. The solutions were designed to cope with a specific set of challenges that we discussed in previous paragraph. For example, the Hough Transform[29] is a feature extraction algorithm that maps the image from the pixel space to the parameter space in order to determine geometric shape, such as, lines. Fitting straight lines through crop rows could prove to be fruitful for cases where the rows are perfectly straight and parallel but the solution fails in the case of curved crop rows. Pixel accumulator based algorithms [10] help in determining centroids of regions of a certain color thereby being robust to different curvatures a crop row may exhibit. But since the color of rows varies and their spectral signatures can often be similar to those of inter-crop row weed clusters, building a model robust and consistent enough for practical use becomes a difficult task.

Some of the existing methods have demonstrated impressive results on smaller datasets consisting of only one type of crop and with little to low variation [5, 23, 25]; however, they lack the ability to adapt to new crop fields that differ significantly from the training set images. Existing methods also work on image data only, ignoring temporal sequence of frames coming from the video feed, which could provide additional information to aid in improving continuous detection of rows. These are the key challenges for bringing crop row detection based driver assistance tool on-board with the agricultural vehicles.

In this work, we present our ongoing effort to build a practical crop row detection system to be deployed in agricultural sprayers. We discuss the challenges that we have faced while using off-the-shelf methodologies for solving this task and also discuss our solutions to overcoming these challenges. Our overall solution is a novel crop row detection method that augments the predictions from a U-net based convolutional neural network model trained on a handful of crop row images. The critical component of our system is a clustering-based probabilistic temporal calibration, which can adapt to different fields without the need for retraining the network. This need is vital for industrial deployment of such an equipment in order to reduce dependence on data collection throughout the year. In terms of methodologies, our contribution is to augment a supervised machine learning based method with clustering based online pattern detection mechanism, so that the overall system can be used for adaptation of a detection method in an unsupervised manner.

We summarize our main contributions as follows:

(1) We design a robust system for fast detection of crop rows from video frames.
(2) We design a mechanism for adapting the model to unseen crop fields through a brief calibration phase.
(3) We introduce a method to extrapolate the model's predictions by taking into account past predictions thereby taking advantage of the temporal aspect of the data.

The overall system diagram is illustrated in Figure 1. As can be seen, continuous video data frame from an on-board camera is streamed into a prediction system, which is calibrated initially for a given crop field. The detection system detects slope, intercept, and confidence value of central four crop rows in the angle of view of the camera. The confidence value for each row represents how confident the system is in its detection of that specific row. Using these data, these rows are displayed on a dashboard screen for assisting the driver.

2 RELATED WORK

For dealing with variations in weed pressure, Zhang et al.[34] designed a crop row detection algorithm to fit straight lines through the rows. For fitting straight lines, several Hough Transform (HT) based solutions were also proposed in [2, 4, 5, 19, 27]. A variation of HT in the form of gradient-based Random Hough Transform (RHT) was proposed by Basso et al.[6] as well as Ji and Qi[14]. Gee et al.[11] proposed a double HT to improve on the Hough Transforms performance in determining the right lines to fit. Inspired by these ideas, Olsen[22] and Romeo[25] developed algorithms that considered accumulating green pixels in the image under the assumption that these are the locations most likely to contain crop rows. Authors of [32] have shown that a green pixel accumulation based method works well at solving the curved crop row detection problem. Sogaard and Olsen[31], Fontaine and Crow [9], Sainz-Costa et al.[28], Burgos-Artizzu et al[7] and Jiang et al.[15] all worked with determining gravity centers of horizontal strips to fit lines using either least squares or HT.

In terms of working with pattern recognition methods, Vidovic et al. [32] used template matching with optimization to solve the curved crop row problem. Machine learning solutions to tackle the detection problem was investigated in [24] where they compared the performances of various supervised, unsupervised and semi-supervised machine learning algorithms for solving crop row detection task.

Hough Transform based methods only consider edge information when detecting lines. In order to compensate for this limitation, Jiang et al. [16] used a sequence of preprocessing stages to extract features that attempt to capture the global geometric pattern of the crop rows. Zhao et al. [35] used a Convolutional Neural Network (CNN) [17] based feature extractor along with a differential hough transform module to translate high level CNN features from the image space to the parameter space. A similar idea was proposed by Bah et al. [3] where two neural networks were employed to solve the subtasks of segmentation and line detection. Lin et al. [18] considered the processes of detecting crop rows as a special case of object detection using a Faster R-CNN architecture where the bounding boxes can take on the form of 2D lines. All the existing crop row detection works used homogeneous images for training their model, so, more often, their performance suffers when the test images are from different crops, or from the same crops but at a substantially different growth stage. On the other hand, our work, although uses U-net—a traditional method for semantic segmentation, can adapt for varying crops and varying growth stages through calibration, without requiring to retrain the model.

Several works have been proposed which consider other precision agriculture tasks, besides crop row detection. For example, a real time semantic segmentation method using the features extracted from a CNN is used for discriminating between crop and

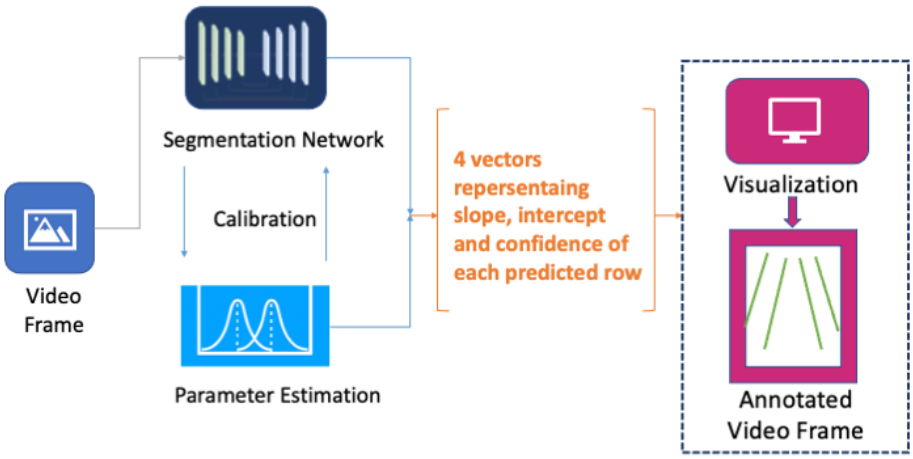


Figure 1: Illustration of proposed method.

weed [21]. Zhao et al. [36] compares the performance of U-net [26] against R-CNN [12] for the task of segmenting pomegranate tree canopy.

3 METHODS

Our method for crop row detection uses a supervised learning method, which given an image classifies a region of image pixels into either of two classes, crop-row or background; such a task is commonly known as semantic segmentation. We use a U-net based neural network architecture for solving this task. After semantic segmentation of the images, we use density based clustering of crop-row pixels to detect clusters resembling crop rows. Then for each such clusters, we fit a straight line to obtain slopes and intercept values. Using the slope values, central four crop rows are identified and returned. By using the distribution of slopes and intercept values over a sequence of frames coming from video frame, domain adaptation is performed.

3.1 Crop Row Benchmark Dataset

We use the open source Crop Row Benchmark Dataset[1], hereby referred as CRBD to train our model. The dataset consists of 281 crop row images of maize, celery, potato, onion, sunflower and soya bean crops taken with a Panasonic LUMIX DMC-F2 digital camera in JPEG format during Spring 2014 in Slavonia, Croatia. The images are also diverse in terms of crop row thickness, inter crop row distances, curvature of the rows and the crop growth stage.

Besides raw images, the CRBD dataset also provides ground truth segmentation masks in the form of data files that provide coordinates of pixels that correspond to crop rows in the original image. We can use the coordinates to generate binary segmentation maps to annotate the location of crop row pixels. A sample from the dataset alongside its ground truth segmentation map is shown in Figure 2.

3.2 Crop Row Modeling and Assumptions

Based on the samples in the CRBD dataset and because of the fixed position of the camera mounted on the vehicle, we assume that the

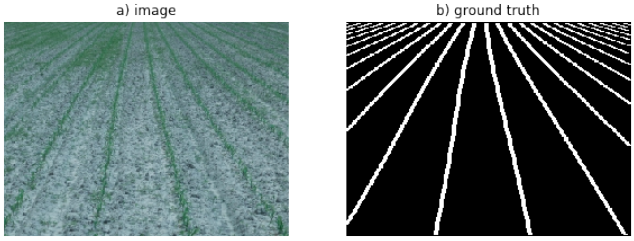


Figure 2: Sample crop row image with label

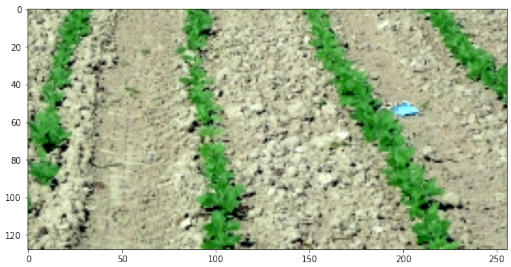


Figure 3: ROI for Crop row image

perspective of the crop row images to be fixed. Since the underlying application is used for navigating, similar to [15, 23], we adapted an ROI to reduce both the complexity of the algorithm and the variability in the images. The ROI selection process assumes that only the crop rows facing the vehicle on it's path are relevant for determining the right path to follow.

A central cropped window of size 128x256 pixels were chosen as the region of interest or ROI for the crop fields as shown in Figure 3. Moreover, since inter crop row distances vary so does the number of rows within the ROI. The rows of interest were chosen to be the central 4 rows that pass through the ROI.

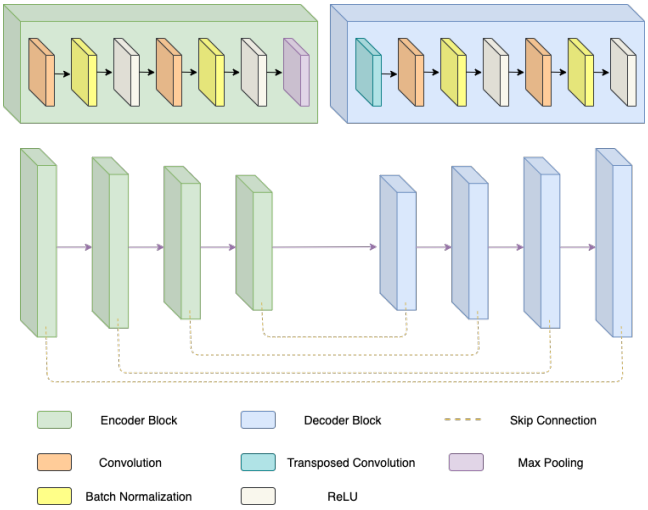


Figure 4: U-net Architecture for segmenting crop rows. The decoder consists of a downsampling path made up of four blocks (green) and the encoder consists of an upsampling path made up of four blocks (blue). Skip connections are added between corresponding encoder-decoder blocks to pass on features learned by the encoder to the decoder.

In order to approximate the change in directions of the rows, instead of directly detecting the exact pixel positions of the rows, we model our task to detect the first derivative of the curvatures that run along the shape of each of the rows. This simplifies the curved shape of the rows into a piece-wise linear form. This process abstracts away the low level semantic segmentation process into a model for detecting straight lines denoting the change of direction of the crop rows.

3.3 Backbone: U-Net Architecture

Initially we train a deep neural network that follows a U-Net Architecture [26]. The U-net architecture differs from a simple encoder-decoder system by its introduction of skip connections between the encoder layers and corresponding decoder layers. If there are a total of n layers in the network then the U-net adds a connection between the ith and $(n-i)^{th}$ layer. These connections provide information of corresponding stages of encodings to the decoding blocks thus helping to improve the decoder's ability to output a final segmentation map with finer features.

Since the size of our labeled dataset is extremely small, we took caution with the network's complexity. To reduce the effect of overfitting as much as possible, we restricted the size of the network by using a total of 4 encoder blocks and 4 decoder blocks. The network architecture along with the composition of layers within the encoder and decoder blocks is shown in Figure 4. The generalization ability of the model is primarily attributed to the sparsity of parameters followed by the downstream adaptation module.

Let the ith encoder block of the U-net takes as input a tensor with dimensions $f \times h \times w$, where f denotes the the number of feature maps in the input, and h and w are the dimensions of the image data. The encoder extracts features from this input through convolution operations and successively, batch normalisation to reduce covariate shift. They are followed by a ReLU layer for incorporating non-linearity. The input-output relation of an encoder block can be shown by the following equation.

$$output_i = ReLU[BatchNorm2d\{Conv2d(input_i)\}] \tag{1}$$

The resultant output expands the number of feature maps but retains the spatial dimensions h and w. The max pooling operation is used to increase the receptive field of a Convolutional Neural Network and we pass the output of the ReLU activation through a pooling layer which halves the spatial dimensions to $h/2$ and $w/2$ respectively.

$$encoding_i = MaxPool(output_i) \tag{2}$$

The jth decoding blocks perform a similar operation as shown in (1) but is preceded by the transposed convolution. This reverts the spatial dimensions of the input to its original size thus completing one stage of the decoding operation.

$$decoding_j = TransposedConv(input_j) \tag{3}$$

$$output_j = ReLU[BatchNorm2d\{Conv2d(decoding_i)\}] \tag{4}$$

For passing information about learned features between encoder and decoder blocks, skip connections are used as can be seen in Figure 4. The skip connections denote a concatenation operation where the output of encoder block and the input to the decoder block are concatenated along the first axis before being passed through the decoder block.

If x_i is the output of the ith encoder block, y_j is the input to the jth decoder block and $j = N - i$ where N is the total number of decoder blocks, then a skip connection between these two blocks exist to perform the following operation-

$$output = Concat(x_i, y_j) \tag{5}$$

3.4 Data Pipeline

In this section we discuss the data pipeline of our system. Since the dataset is small, in order for our network to learn the underlying general geometric pattern of the rows as well as possible, we chose a split prioritizing a larger portion for training. Moreover, to reduce overfitting to the data, we used random augmentations to increase both the size and variability of the dataset. The following are some of the key steps used in the pipeline before training the network:

(1) Low level processing and ROI extraction
(2) Training and test set split
(3) Training set augmentation

3.4.1 Low Level processing. For low level processing, we normalize the raw images to have pixel intensities with a fixed mean and standard deviation per color channel. This helps stabilize the training of the network and convergence of the model. A Gaussian blur with kernel size of 3 and $\sigma = 1$ is also used to reduce noise in the image.

The images from CRBD are each of size 320×240 pixels. As shown in Figure 2, there are more than 10 crop rows in this image, the majority of them are densely situated towards the edge and horizon of the canvas, due to the vanishing point effect. Detection of these rows are not important in real-life when using such a

system in the agricultural vehicle navigation system. So, we extract an ROI (region of interest) from each image of size 128×256 pixels covering the central rows. Doing so helps discard irrelevant crop rows near the edges. Also, after extracting the ROI we obtained images with fairly equalized inter crop row distances as shown in Figure 3.

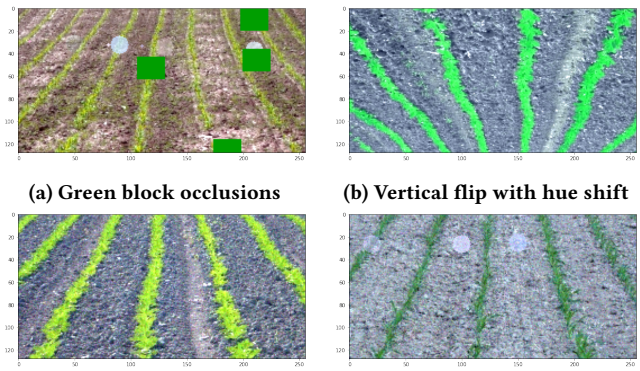
3.4.2 Train and test set split. In order to test the performance of the algorithm on unseen data, we randomly shuffle the dataset before splitting it where 80% of the dataset is used to train the model and the remaining 20% is held out for testing different algorithms for quantitative comparison. During training, we use 20% of training data instance as validation set for parameter tuning.

3.4.3 Training set augmentation. Overfitting [33] to training data is one of the central challenges in deep learning. Especially for small datasets such as CRBD, it is difficult for the network to generalize to unseen data and much easier to memorize the entire dataset instead. To mitigate this, we employed data augmentation methods that result in a significant increase to the variability of data seen by our model during training.

The augmentation methods were chosen so they would emulate real world conditions of crop fields as closely as possible. Some examples of these include solar flares and occlusions due to weed or other objects. This helped the network improve its capability to learn the continuity of each of the rows. We also made random lighting changes to the overall image to emulate the various lighting conditions our system could be exposed to during test time.

Finally to improve spatial variance of the underlying dataset, images were randomly flipped both horizontally and vertically.

All of the augmentations were performed with the help of the albumentations library [8] in Python. The augmented datasets produced samples similar to ones shown in Figure 5.



(a) Green block occlusions **(b) Vertical flip with hue shift**

(c) Saturation and brightness shift **(d) Solar Flares**

Figure 5: Some samples from the augmented dataset

3.5 Tuning Network Predictions

3.5.1 Refining the output. Although the network performed well on the test set for CRBD, it performed poorly in the case of real world video frames. The network output segmentation map included significant noise in its predictions that consist of both false positives within the inter crop row spaces as well as false negatives within the crop rows themselves. A sample prediction of the network on a video frame is shown in Figure 6.

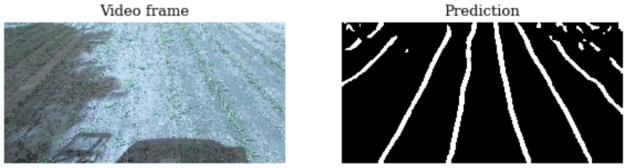


Figure 6: Sample input/output of the U-Net

We mitigate this issue to some extent by clustering the pixel predictions for crop rows in the segmentation map using a Hierarchical Density Based clustering routine. We set a threshold of 50 pixels to filter out the clusters of false positives in the binary image leaving us with clusters that are large enough to qualify as crop rows. This process also has the added benefit of removing the rows at the edges of the ROI which are smaller in size due to the view perspective.

3.5.2 Choosing central rows. We choose the central four rows to be relevant to our operation the most as they help to keep the vehicle centered while also reducing the effect of camera's viewpoint center shifting within a reasonably small range. To find this, we initially fit first order polynomials or straight lines through the clusters in the segmentation map that are obtained after the refining stage.

Because of the consistency in length and slope of the rows from the left to the right of the ROI, we assume that the central four rows should have two properties that distinguish them from the others:

- The four longest along the vertical axis
- The four smallest slopes along the vertical axis

As such we find the central four rows by first sorting the lines based on the length of the line segments and picking the largest four. Finally, in order to know line corresponds to what position of the four rows (left, left-center, right-center, right), we sort them by their slopes along the vertical axis. This results in the leftmost row having the smallest slop with a gradual increase up to the rightmost row. Algorithm 1 demonstrates this process.

This post processing routine converts our output from a binary segmentation map to four pairs of m_i, c_i where m_i is the slope of the ith row from the left and c_i is the intercept.

3.6 Pre-Calibration and density estimation

The pre-calibration stage is what helps the model to adapt to an unseen crop field. In this phase, we use a clustering based algorithm to fit a distribution over each of the four predicted rows. This process is only employed once for every crop field to adapt the model to a new domain without the need for supervised training.

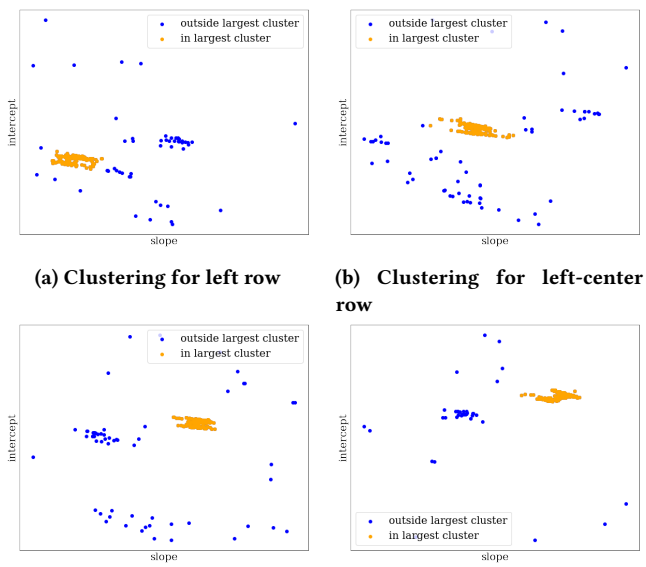
3.6.1 Rows of interest and largest clusters. We first run the model on the first 250 frames of the video feed. The number 250 is a tuneable hyperparameter and increasing or decreasing it would result in higher or lower quality of adaptation respectively. After

Algorithm 1: ROWS-OF-INTEREST RETRIEVAL

Input: Binary segmentation mask y_{pred} predicted by the backbone U-Net architecture where pixel positions corresponding to crop rows have a value of 1 and 0 elsewhere. s_m, the minimum size for valid row clusters

Use Hdbscan[20] to identify N_m clusters of positive values in y_{pred} with a minimum cluster size s_m
for $i \leftarrow 1$ ***to*** N_m **do**
  $m, c \leftarrow$ get slope-intercept of best fit line through cluster C_i
  $N_{rows} \leftarrow min(N_m, 4)$
  $center\_rows \leftarrow$ longest N_{rows} best fit lines through clusters C_i s.t $i \in \{1, ..., N_{rows}\}$
  $sorted\_rows \leftarrow$ sort $center\_rows$ by the slopes m_i
end
return $sorted\_rows$

running for the first 250 frames, we record the model's output as 1000 (m_i, c_i) pairs where $i \in \{1, 2, 3, 4\}$. These were then feature scaled to eliminate the effect of magnitudes.



(a) Clustering for left row **(b) Clustering for left-center row**

(c) Clustering for right-center row **(d) Clustering for right row**

Figure 7: Density based clustering of predicted row parameters (feature scaled)

We make the assumption that among the 250 frames, for each of the four rows at least over half of the predictions would be fairly accurate. This does not imply that half of the frames would be good predictions, but that each of the rows would have over half of their predictions to be accurate representations of their position in the frame.

Based on this, for each of the four rows the corresponding (m, c) values were then clustered using density based clustering. This gives us four different scatter plots each consisting of 250 points in 2D space corresponding to the 250 calibration frames. The plots are shown in Figure 7. After clustering the points, we pick the largest cluster as the one most likely to correspond to the correct predictions for that specific row. We also chose density based clustering here because for correct predictions, the variance within the (m, c) pairs would be small compared to the incorrect predictions and density based clustering captures that property. The parameters were chosen to be $\epsilon = 0.35$ and $minsamples = 5$.

Algorithm 2: LARGEST CLUSTER SELECTION

Input: X, list of 2D points; ϵ, maximum distance to nearest neighbor for density based clustering; $min\_samples$, minimum number of nearest neighbors for a core point

$N_x \leftarrow$ population of list X
$C \leftarrow$ density based clustering of points $x_i; \forall x_i \in X$, parameterized by ϵ and $min\_samples$
$N_c \leftarrow$ number of clusters obtained
$largest\_cluster \leftarrow \emptyset$
for $c_i \in C$ **do**
  if $Population(c_i) > Population(largest\_cluster)$ **then**
    $largest\_cluster \leftarrow c_i$
  end
end
return $largest\_cluster$

3.6.2 Validating correct row predictions. We extracted the points from the largest clusters for each row and then used K-means to plot them in the same 2D space. The scatter plot is shown in Figure 8.

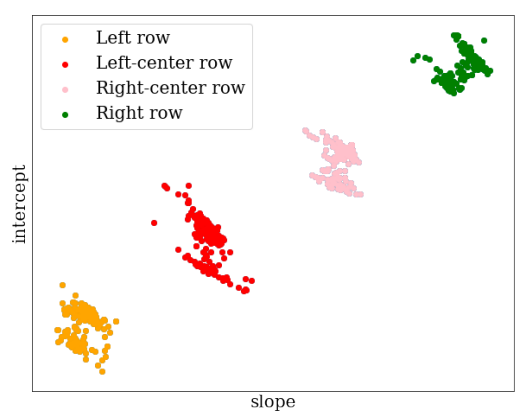


Figure 8: Kmeans clustering of row representations

As shown in the plot, the four clusters are well separated and are very concentrated along each of the four means. This helps validate our assumption that the largest clusters are good representations for the correct row predictions.

3.6.3 Density Estimation of row clusters. To find the closeness of a new row prediction to a good prediction, we need to know its likelihood of belonging to the corresponding cluster. We do this by fitting an EM clustering over the parameter space of the four clusters. This results in a continuous density function that gives a high confidence if the prediction for a row is likely to have come from the same distribution as the correct rows obtained during the calibration stage. The entire calibration phase is shown in Algorithm 3.

By putting a threshold on the confidence value, we can control the tradeoff between temporal smoothness of a crop row's parameter values (slope and intercept) and the network's prediction accuracy for that row in the current frame. Poor confidence value acts as a trigger that causes the U-Net to disengage from the prediction system and in turn predict the row of interest not as a pixel level classification but as an extrapolation of previous high confidence predictions of the parameter values of the same row. In a sense the calibration stage can be defined as a type of self-supervision that provides our network with an estimate of its performance for the current frame.

Algorithm 3: Calibration: Estimation of valid row distribution

Input: V_s, video stream; N_{frames}, Number of frames to use for calibration;

Samples ← dictionary mapping row indices to 2-*D* arrays of shape $(N_{frames} \times 2)$
for $i \leftarrow 1$ *to* N_{frames} **do**
  $f_i \leftarrow$ retrieve current frame from V_s
  $pred_i \leftarrow$ get predicted segmentation map from U-Net
  sorted_rows ← retrieve 4 central rows from $pred_i$
  for $j \leftarrow 1$ *to* 4 **do**
    append parameters (m, c) of jth row to $Samples_j$
  end
end
largest_clusters ← empty list
for $i \leftarrow 1$ *to* 4 **do**
  $\mathbf{C}_i \leftarrow$ get_largest_cluster($Samples_i$)
  append all elements (m_i, c_i) in $\mathbf{C}_i$ to *largest_clusters*
end
$\mu_{em}, \Sigma_{em} \leftarrow$ parameters from EM clustering of *largest_clusters*
return μ_{em}, Σ_{em}

3.6.4 Frame extrapolation and temporal corrections. The model was tested using a camera that recorded videos on-board a moving vehicle at 25fps. Since the slopes and intercepts of each of rows change uniformly along subsequent frames, the derivative of these can be regarded as constant. This facilitates the ability to extrapolate a prediction based on previous predictions to a reasonable degree of accuracy.

Since the change in slopes and intercepts of each row varies consistently, the derivative can be approximated as constant through a small number of subsequent frames. Therefore the change in parameters can be assumed to be linear. As such, given frames f_{i-1} and frame f_i with parameters (m_{i-1}, c_{i-1}) and (m_i, c_i) respectively, the parameters for the next frame f_{i+1} can be approximated as shown in equation 6.

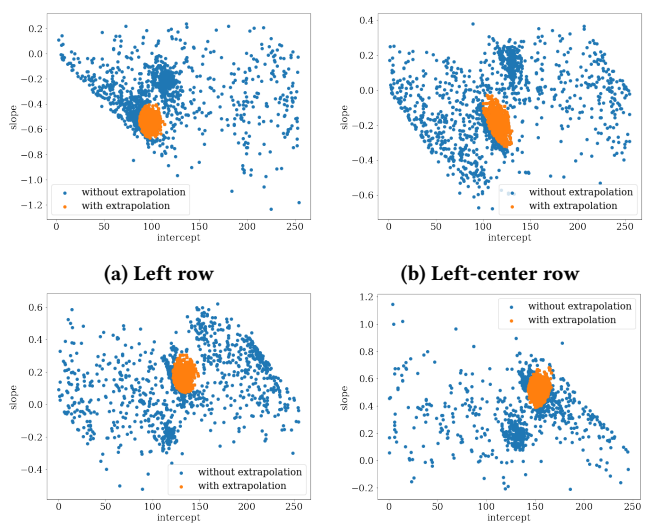
$$x_{i+1} = 2x_i - x_{i-1}; x_j \in \{\mathbf{m}_j, \mathbf{c}_j\}, \forall j \in \{i-1, ...i+1\} \quad (6)$$

We can control the temporal smoothness of our model's predictions at runtime by choosing when to extrapolate a prediction and when to rely on the network's current prediction for a row. This choice is made by putting a threshold on the confidence value of a prediction that can be obtained from previously estimated probability densities for each of the rows. For our tests, we found that by rejecting predictions that produce confidence scores of under 30% and replacing them with a rolling average of the previous 3 rows, the model can be made robust to difficult frames that has shadows, occlusions or other obstructions.

4 EXPERIMENTS AND RESULTS

The U-Net backbone was trained for 100 iterations in a traditional semantic segmentation setting to minimize the cross-entropy loss between the output and the ground truth segmentation map.

Experiments were performed on the Nvidia Jetson TX2. In Figure 9 we can see the differences in the distribution of the best fit line parameters for each row upon the introduction of frame extrapolation based on parameters obtained from the calibration phase. As shown, the calibration phase pushes the distribution of the points towards a concentrated region (annotated in orange)



(a) Left row (b) Left-center row (c) Right-center row (d) Right row

Figure 9: Distribution of row predictions in parameter space

On the other hand, without going through the calibration phase and solely relying on the raw predictions from the trained network, we can see a large variance in the predictions for each of the rows. By controlling the threshold on the learned confidence metric, the weight on frame extrapolation can be controlled to tune

this difference in variance. This in turn controls the trade-off between temporal smoothness and the U-Net's current prediction accuracy. A higher threshold on confidence would concentrate the predictions towards the cluster means thereby enforcing a smoother inter-frame transition of the predictions. A lower threshold would reduce the effects of the previous frames and lean more towards the network's predictions for the current frame only.

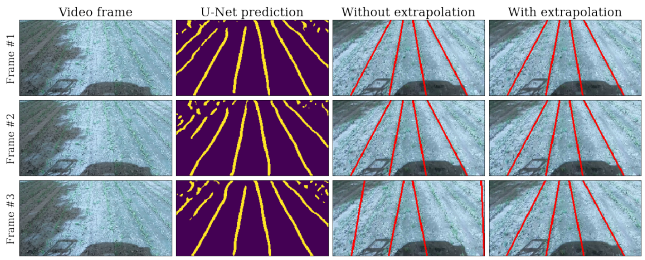


Figure 10: Example of frame extrapolation to fix errors in predicted rows. Here frames 1 and 2 are used to extrapolate the third frame since confidence of prediction for the left and right row did not cross the threshold in the case of the original prediction.

Table 1: Changes in variance of parameters in subsequent frames, averaged across central four crop rows

	σ		err
	Without calibration σ_{nc}	With calibration σ_c	Abs. difference $\lvert\sigma_{nc} - \sigma_c\rvert$
m	0.011	0.0026	0.0083
c	148.02	13.82	134.2

From Table 1 it is evident that our calibration phase had little effect on reducing the variance in slopes of the rows. However, the reduction in the case of the intercept is significant. This is caused by the fact that the due to highly discontinuous segments predicted by the U-Net and the selection process for the rows of interest being dependent on the size of the clusters in the predicted segmentation map, prediction for certain rows shifts to an adjacent row.

This also explains the small difference in the variance in slopes as adjacent crop rows have slopes that are nearly identical. In order to ensure that the algorithm can be employed for real-time applications, we tested its performance on an Nvidia Jetson TX2. We ran the inference on each frame of a stream fed by a camera mounted on a vehicle. The neural network's inference was performed on the on-board GPU and the frame extrapolation based on calibrated parameters were processed on the CPU. Table 2 shows the average processing time per frame and the average FPS achieved for both the calibration and inference phases.

5 DISCUSSION

Our proposed method provides a way to influence a neural network's predictions based on properties of its history of correct predictions. This simultaneously preserves the affine properties of the crop rows that are the same for all crop fields (multiple straight/curved lines converging towards the horizon) and also has the added flexibility of adapting to a new domain where the underlying scene has a shift in distribution from the training set.

Table 2: Processing time of algorithm on Nvidia Jetson TX2

	Calibration 250 frames	Inference 10000 frames
Avg time per frame	32ms	54ms
Avg FPS	31	17

To evaluate the potential of the algorithm to be deployed on edge devices for real time detection, the inference time was tested on an Nvidia Jetson TX2.

6 FUTURE WORK

Future work in this area could use a more sophisticated method for refining the CNN's output. This can be done by conditioning on a geometric property of the clusters that take into account the sizes along different axes instead of just considering the size as the population of a cluster. Moreover, discontinuities in the segmentation map and as a result large number of clusters in the output could be reduced by enforcing a topological constraint on the CNN's output as proposed by [30]. Finally, In the case of high frame rates, the derivative of the parameters of each row can be assumed to be constant. However, this assumption is not robust as with decreasing frame rates, there are more dramatic shifts in the row's positions on in the scene. A sophisticated sequence model could be used to learn the temporal nature of the crop rows for that particular frame rate during the calibration phase to tackle this challenge.

7 CONCLUSION

In this work, we introduced a method to augment a neural network's inference capabilities in the case of scarcity of labeled training data. Our study has shown that this method can serve as a mechanism of self-supervision for the segmentation engine by applying confidence scores to it's own predictions based on its consistency with the underlying distribution of valid predictions. Besides quantitative and qualitative evidence on the accuracy of the result, we investigated the processing speed for video frames on a low power edge device.

8 ACKNOWLEDGEMENT

This research was performed at Indiana University-Purdue University Indianapolis (IUPUI). Funding for this research came from a research grant by Equipment Technologies, Moorseville, Indiana. Dr. Hasan's research is also funded by National Science Foundation through the grant IIS-1909916.

REFERENCES

[1] [n.d.]. *Crop Row Benchmark Dataset.* http://www.etfos.unios.hr/r3dvgroup/index.php?id=crd_dataset
[2] Björn Åstrand and Albert-Jan Baerveldt. 2005. A vision based row-following system for agricultural field machinery. *Mechatronics* 15, 2 (2005), 251–269.

[3] Mamadou Dian Bah, Adel Hafiane, and Raphael Canals. 2019. CRowNet: Deep network for crop row detection in UAV images. *IEEE Access* 8 (2019), 5189–5200.

[4] Tijmen Bakker, Kees van Asselt, Jan Bontsema, Joachim Müller, and Gerrit van Straten. 2011. Autonomous navigation using a robot platform in a sugar beet field. *Biosystems Engineering* 109, 4 (2011), 357–368.

[5] Tijmen Bakker, Hendrik Wouters, Kees Van Asselt, Jan Bontsema, Lie Tang, Joachim Müller, and Gerrit van Straten. 2008. A vision based row detection system for sugar beet. *Computers and electronics in agriculture* 60, 1 (2008), 87–95.

[6] Maik Basso and Edison Pignaton de Freitas. 2019. A UAV Guidance System Using Crop Row Detection and Line Follower Algorithms. *Journal of Intelligent & Robotic Systems* (03 2019). https://doi.org/10.1007/s10846-019-01006-0

[7] Xavier P Burgos-Artizzu, Angela Ribeiro, Maria Guijarro, and Gonzalo Pajares. 2011. Real-time image processing for crop/weed discrimination in maize fields. *Computers and Electronics in Agriculture* 75, 2 (2011), 337–346.

[8] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. 2020. Albumentations: Fast and Flexible Image Augmentations. *Information* 11, 2 (2020). https://doi.org/10.3390/info11020125

[9] V Fontaine and TG Crowe. 2006. Development of line-detection algorithms for local positioning in densely seeded crops. *Canadian biosystems engineering* 48 (2006), 7.

[10] Iván García-Santillán, José Guerrero, Martín Montalvo Martínez, and Gonzalo Pajares. 2017. Curved and straight crop row detection by accumulation of green pixels from images in maize fields. *Precision Agriculture* 19 (01 2017). https://doi.org/10.1007/s11119-016-9494-1

[11] Ch Gée, Jérémie Bossu, Gawain Jones, and Frederic Truchetet. 2008. Crop/weed discrimination in perspective agronomic images. *Computers and Electronics in Agriculture* 60, 1 (2008), 49–59.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2013. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (11 2013). https://doi.org/10.1109/CVPR.2014.81

[13] José Guerrero, José Ruz, and Gonzalo Pajares. 2017. Crop rows and weeds detection in maize fields applying a computer vision system based on geometry. *Computers and Electronics in Agriculture* 142 (11 2017), 461 – 472. https://doi.org/10.1016/j.compag.2017.09.028

[14] Ronghua Ji and Lijun Qi. 2011. Crop-row detection algorithm based on Random Hough Transformation. *Mathematical and Computer Modelling* 54, 3-4 (2011), 1016–1020.

[15] Guoquan Jiang, Zhiheng Wang, and Hongmin Liu. 2015. Automatic detection of crop rows based on multi-ROIs. *Expert systems with applications* 42, 5 (2015), 2429–2441.

[16] Guo-Quan Jiang, Cui-Jun Zhao, and Yong-Sheng Si. 2010. A machine vision based crop rows detection for agricultural robots. In *2010 International Conference on Wavelet Analysis and Pattern Recognition*. IEEE, 114–118.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems* 25 (01 2012). https://doi.org/10.1145/3065386

[18] Shaomin Lin, Yu Jiang, Xueshen Chen, Asim Biswas, Shuai Li, Zihao Yuan, Hailin Wang, and Long Qi. 2020. Automatic Detection of Plant Rows for a Transplanter in Paddy Field Using Faster R-CNN. *IEEE Access* 8 (2020), 147231–147240.

[19] JA Marchant. 1996. Tracking of row structure in three crops using image analysis. *Computers and electronics in agriculture* 15, 2 (1996), 161–179.

[20] Leland McInnes, John Healy, and Steve Astels. 2017. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software* 2, 11 (2017), 205.

[21] A. Milioto, P. Lottes, and C. Stachniss. 2018. Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2229–2235.

[22] HJ Olsen. 1995. Determination of row position in small-grain crops by analysis of video images. *Computers and Electronics in Agriculture* 12, 2 (1995), 147–162.

[23] Vignesh Raja Ponnambalam, Marianne Bakken, Richard JD Moore, Jon Glenn Omholt Gjevestad, and Pål Johan From. 2020. Autonomous Crop Row Guidance Using Adaptive Multi-ROI in Strawberry Fields. *Sensors* 20, 18 (2020), 5249.

[24] María Pérez-Ortiz, José M Peña-Barragán, Pedro Antonio Gutiérrez, Jorge Torres-Sánchez, Cesar Martínez, and Francisca López-Granados. 2015. A semi-supervised system for weed mapping in sunflower crops using unmanned aerial vehicles and a crop row detection method. *Applied Soft Computing* 37 (12 2015), 533–544. https://doi.org/10.1016/j.asoc.2015.08.027

[25] J Romeo, G Pajares, M Montalvo, JM Guerrero, M Guijarro, and A Ribeiro. 2012. Crop row detection in maize fields inspired on the human visual perception. *The Scientific World Journal* 2012 (2012).

[26] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[27] F Rovira-Más, Q Zhang, JF Reid, and JD Will. 2005. 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* 219, 8 (2005), 999–1010.

[28] Nadir Sainz-Costa, Angela Ribeiro, Xavier P Burgos-Artizzu, María Guijarro, and Gonzalo Pajares. 2011. Mapping wide row crops with video sequences acquired from a tractor moving at treatment speed. *Sensors* 11, 7 (2011), 7095–7109.

[29] Allam Shehata, Sherien Mohammad, Mohamed Abdallah, and Mohammad Ragab. 2015. A Survey on Hough Transform, Theory, Techniques and Applications. (02 2015).

[30] Suprosanna Shit, Johannes C Paetzold, Anjany Sekuboyina, Andrey Zhylka, Ivan Ezhov, Alexander Unger, Josien PW Pluim, Giles Tetteh, and Bjoern H Menze. 2020. clDice–a Topology-Preserving Loss Function for Tubular Structure Segmentation. *arXiv preprint arXiv:2003.07311* (2020).

[31] Henning Tangen Søgaard and Hans Jørgen Olsen. 2003. Determination of crop rows by image analysis without segmentation. *Computers and electronics in agriculture* 38, 2 (2003), 141–158.

[32] Ivan Vidović, Robert Cupec, and Zeljko Hocenski. 2016. Crop Row Detection by Global Energy Minimization. *Pattern Recognition* 55 (01 2016). https://doi.org/10.1016/j.patcog.2016.01.013

[33] Wikipedia contributors. 2020. Overfitting — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Overfitting&oldid=953810115 [Online; accessed 1-May-2020].

[34] Xiya Zhang, Xiaona Li, Baohua Zhang, Jun Zhou, Guangzhao Tian, Yingjun Xiong, and Baoxing Gu. 2018. Automated robust crop-row detection in maize fields based on position clustering algorithm and shortest path method. *Computers and Electronics in Agriculture* (09 2018). https://doi.org/10.1016/j.compag.2018.09.014

[35] Kai Zhao, Qi Han, Chang-Bin Zhang, Jun Xu, and Ming-Ming Cheng. 2021. Deep hough transform for semantic line detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021).

[36] Tiebiao Zhao, Yonghuan Yang, Haoyu Niu, Dong Wang, and YangQuan Chen. 2018. Comparing U-Net convolutional network with mask R-CNN in the performances of pomegranate tree canopy segmentation. In *Multispectral, Hyperspectral, and Ultraspectral Remote Sensing Technology, Techniques and Applications VII*, Allen M. Larar, Makoto Suzuki, and Jianyu Wang (Eds.), Vol. 10780. International Society for Optics and Photonics, SPIE, 210 – 218. https://doi.org/10.1117/12.2325570