Fruit Detection, Tracking and Counting

Do Kyoun Lee (u5808720), Xiaochen Liu (u6312197), Chaohuo Wu (u6516811)

Abstract - Measuring yield of crops is a crucial part of agriculture and farm management. This paper presents explores utilization of state-of-the-art detection algorithms to efficiently solve this problem. Specifically, Faster-RCNN was implemented to create a function for detection, tracking and counting 5 categories of fruits in video clips. Other than Faster RCNN, Kalman filter is used to track motion of fruits and color segmentation is applied to count fruit pixels. By using pre-trained Alexnet in Faster-RCNN, the training data required was vastly reduced. Also, design of Faster-RCNN unified region proposal and detection network has shown to reduce training and detection time. The result showed that the processing speed of this program running on Titan GPU is around 5 fps. Accuracy of detection was as high as 95% for pears, while apple was the lowest at 75%.

I. Introduction

Recognizing and counting fruits on conveyor belts is an essential task in orchards, food processing industries as well as supermarkets. Doing this job by human is ineffective and the accuracy may be low if objects move fast on the belt or fruit categories are too much. A better solution for this task is implementing computer vision method to detect, track and count fruits automatically.

In recent years deep learning has been pioneering in object classification, detection and segmentation of images. Comparing to traditional methods, it is more robust in different illumination, size and rotation. This project presents a state-of-art fruit detection, tracking and counting method. In this method deep learning is implemented to detect different fruits in the videos. Then Kalman filter is applied on detected fruit to track it. Finally when the centroid of tracked fruit cross the bottom line in the video frames, computer increments corresponding fruit count.

Section 2 presents a brief overview of the related work for this project. Section 3 introduces the methods applied to solve the tasks in this project and shows the training, detection, tracking and counting results. Moreover, section 4 demonstrates the detecting result and tracking result. The next section is to do a discussion. Final section is conclusion part.

II. Related work

The aim of this project is to detect each fruit in the videos, segment the corresponding fruit regions and track them. To solve fruit detection and segmentation task, a great amount of studies has been conducted. Zhou et al. [1] analyzed color region of fruits in RGB color space and set threshold to detect red apples in images. Wang et al. [2] thresholded hue value in HSV color space to detect red apples in vineyard. However, in our attempt to employ color threshold and morphing to classify fruits substantial limitations were found. The color region of apple and pear is extremely similar in the videos and the resolution of the video is low. Consequently, classification with fine-tuned color thresholds showed an unsuccessful performance. To find fruit color region more precisely, Yamamoto et al. [3] employed decision tree in machine learning method to segment each pixel in images in RGB, Lab, HSV and YCbCr color space. Though the result showed that the recall and precision is better than tradition color segmentation method, calculation of this approach is too huge since it segments all pixel in an image with 12 color features into 6 categories. So this approach is not suitable for fruit detection in video clips.

Recently, deep neural network has shown an impressive performance in classification and detection in images. The typical procedure of neural network classification is transforming image regions into feature spaces and using trained models to classify them as an object or background. Specifically, Region based Convolutional Neural Network (R-CNN) and Faster R-CNN have produced state-of-the-art performance. R-CNN employs Selective Search [4] to find interested pixel regions and use CNN to classify objects. Faster R-CNN on the other hand merges the computation of object classification and region proposals into a single unified network [5]. This achieved improvements in detection results as well as significant improvement in training and prediction time. Subsequent papers employing this method to detect fruits has shown performance improvements. [6] used Faster R-CNN trained using RGB color and Near-Infrared fruits training images to detect sweet pepper in images, and its F1-score rose to 0.838 comparing to 0.807 of traditional method. [7] also employed Faster R-CNN to detect mangos, almonds and apples in orchard and its F1-score was above 0.9. In this paper, we similarly explore adaptations of Faster RCNN to detection of fruits passing through a conveyor belt.

For tracking task, Camshift and Kalman filter are common methods to track each fruit in the video. Camshift stands for Continuously Adaptive Mean-Shift, is a tracking method based on color histogram [8]. It implements Mean-Shift algorithm on each frame in the video and takes the result of the last frame as the initial finding box, and iterates this procedure in the whole video. The advantage of Camshift is that when the size of tracked object is change, the tracking area can change adaptively. Besides, since it takes color as feature, this algorithm is rotation-invariant. Nevertheless, when the color of tracked object is similar to its surrounding background or other objects, the tracking performance will decrease significantly. Then we decided to use Kalman filter to track objects. Kalman filter estimates the new possible position of an object just measuring its current position and speed, thus its calculation is low and can run in real time. After that, when centroid of an object crosses the bottom line of the frame, its belonging fruit category number is incremented.

III. Approach

A. Transfer learning

In deep learning applications, transfer learning, a branch of machine learning, is commonly used. A pre-trained network which has been trained using a giant dataset can transfer its learnt features to a new task with less labels [9]. Employing this method can significantly decrease the workload and time in training neural network. In this paper, firstly a pre-trained Alexnet model that was trained on a subset of ImageNet dataset was acquired and re-trained on fruit-360 database.

Alexnet is a neural network model, which has already been trained on a subset of the ImageNet database used in ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) [10]. The Alexnet model has been trained to classify images into 1,000 object categories with more than one million images. For instance, vehicles, cakes, pencils, and numerous animals. The Architecture of Alexnet is shown in the following figure [10].

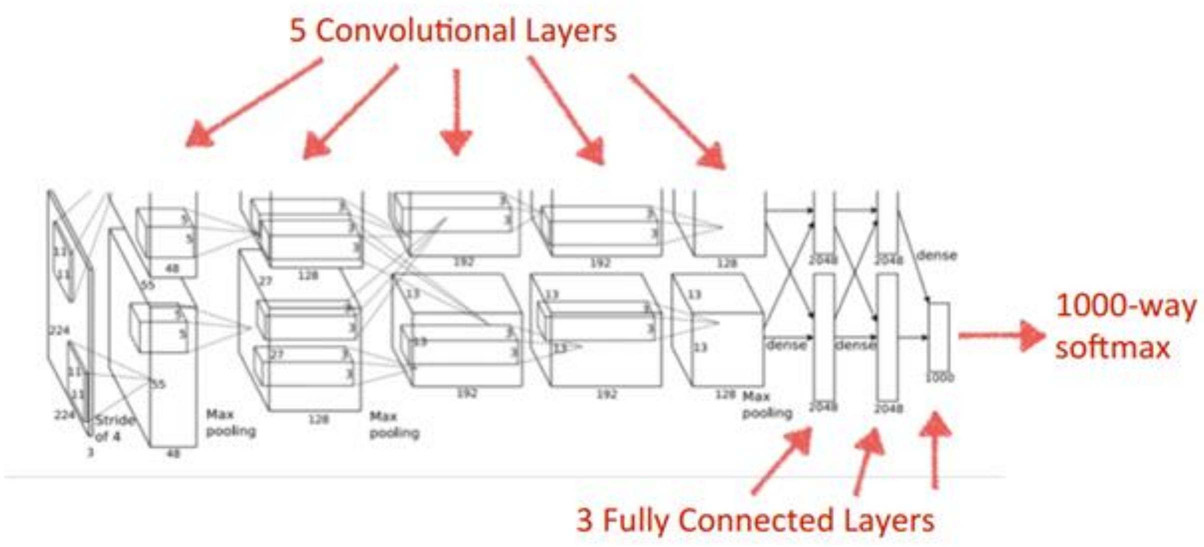


Figure 1: Architecture of Alexnet

Figure 1 shows that Alexnet has eight learned layers, which consist of five convolutional and three fully-connected. The output of the last fully-connected layer is transferred to a 1000-way softmax that can create a distribution over the 1000 class labels. The kernels of the second, fourth, and fifth convolutional layers are linked only to those kernel mapping in the previous layer. However, the kernels of the third convolutional layer are connected to all kernel maps in the second layer. The neurons in the fully-connected layers are connected to all neurons in the previous layer. Response-normalization layers follow the first and second convolutional layers. Max-pooling layers follow both response-normalization layers as well as the fifth convolutional layer. The ReLU non-linearity is applied to the output of every convolutional and fully-connected layer [10].

In order to re-train the Alexnet neural network, the training set Fruit 360 can be used. There are totally 2210 images within five labels in this training set. Firstly, the images are required to resize by 227×227×3 to meet the demand of Alexnet training format.

Figure 2 is part of the training set Fruit360.

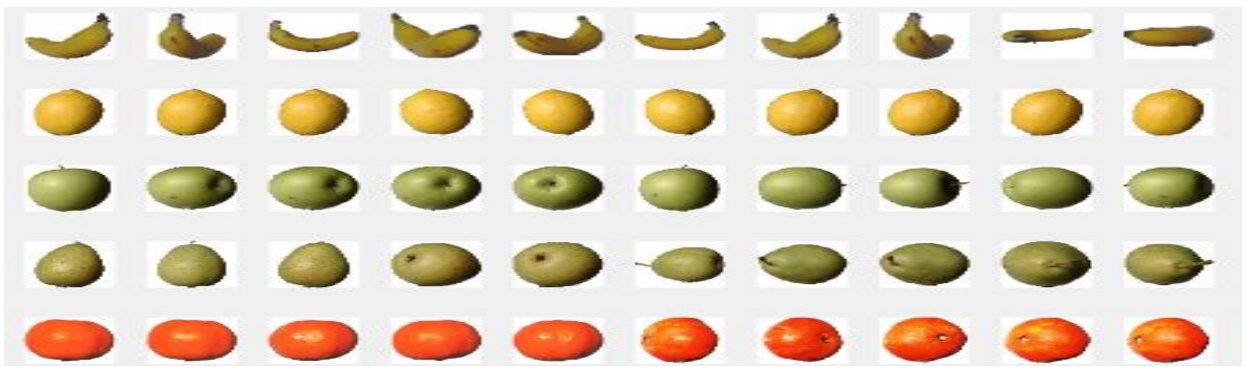
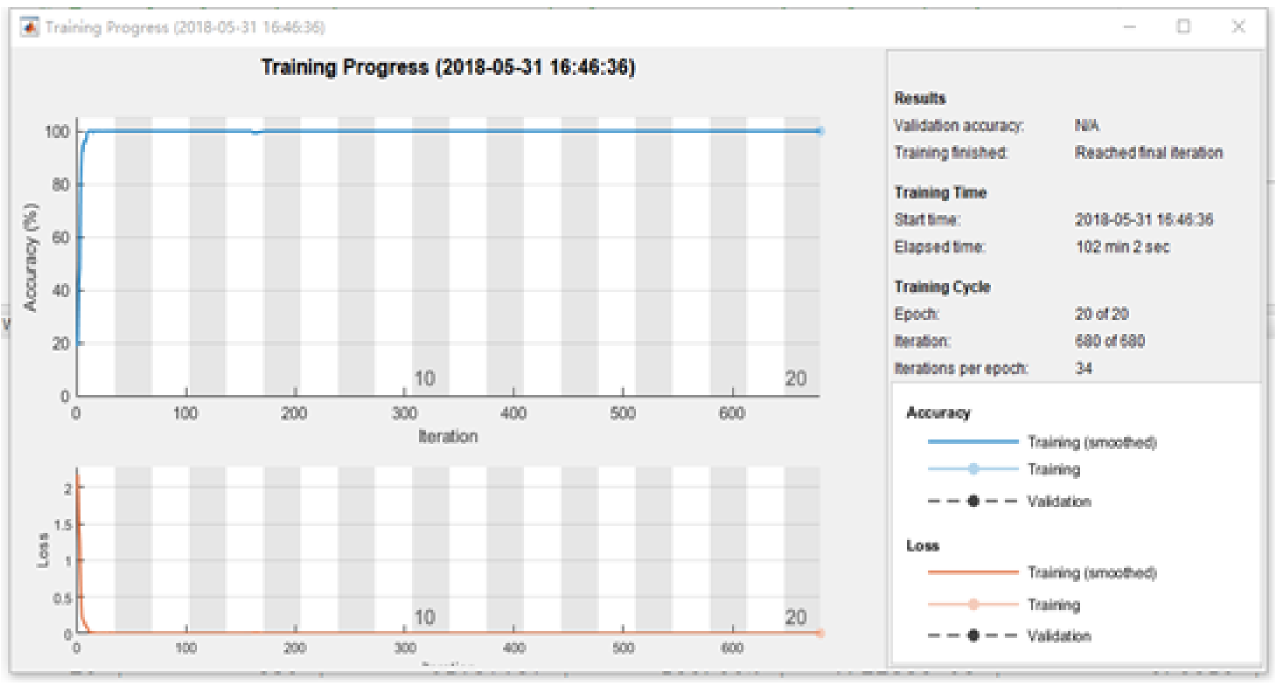


Figure 2: Part of training images from Fruit 360

The next step is to store these images into a datastore. Besides, since configured for 1000 classes, the 23rd layers and 25th layers of Alexnet network are required to be fine-tuned for the new classification task. Therefore, after extracting all layers from Alexnet, these two layers are replaced with a fully connected layer and a classification layer.



Results
Validation accuracy: N/A
Training finished: Reached final iteration

Training Time
Start time: 2018-05-31 16:46:36
Elapsed time: 102 min 2 sec

Training Cycle
Epoch: 20 of 20
Iteration: 680 of 680
Iterations per epoch: 34

Figure 3: Progress of re-training the Alexnet

Figure 3 represents that the maximum epochs is 20 and the accuracy of re-training the Alexnet can achieve to 100%. The accuracy of training initially surges by 100%, and no fluctuation before the end of training. Therefore, until this step, the neural network has already completed, which also presents that the transfer learning can save more time.

B. Faster RCNN

Faster RCNN is comprised of two modules. First is a convolutional neural network that proposes regions called Regional Proposal Network (RPN). Second is the detection network, for which a trained Alexnet, described above, was implemented [6]. The RPN modules tells the detector where to look. It is a small sub-network between the convolutional layers and the last pooling layer. The ultimate goal is to share computation between Alexnet's detection network and RPN.

The regional proposal network takes as input a n x n window of feature map produced in the last shared convolutional layer and outputs a binary classification (object or not) and also the parameters of the bounding box - x, y denoting coordinates of center, width, and height. At position of the sliding window, rectangles of *k* different scales and aspect ratios are considered, called *anchors*. By default, Faster RCNN uses k=9. This allows the RPN to be predict boxes of various sizes. During training each anchor is compared to the ground-truth bounding box and calculate the Intersection-over-Union (IoU). The IoU is calculated as the ratio of the intersecting region over the total area of bounding box and anchor [6]. If the IoU is greater than 0.7, a positive label is assigned else a negative label. In case there is no region above 0.7, the maximum region is also given a positive label. The loss function for classification is therefore calculated for each anchor as the loss over two classes (object or background). The function for the parameterized coordinates of the bounding box are also calculated. Since negative labels dominate, optimizing the loss function over all anchors creates bias. Therefore, anchors are sampled so that the ratio of positive and negative labels is 1:1. Then back-propagation and stochastic gradient descent (SGD) is used to train the network.

In order to combine RPN with the pretrained Alexnet, a 4-step alternating training method, proposed by the authors of Faster RCNN [6], is used. Fruit images extracted from 4 of the given videos and were used as a data set. Using Alexnet's convolutional neural network as initialization, RPN is trained. RPN takes as input the feature map produced by the convolutional layers below. Fast-RCNN detector is trained separately also using Alexnet as initialization. Then, in the third step, Fast-RCNN detector is used as initialization for further fine-tuning the layers of RPN, keeping the shared convolutional network fixed. Finally, we also fine-tune the layers unique to Fast-RCNN while the shared layers are kept fixed. The result is one unified neural network.

C. Segmentation

Another task of this project was the segment bananas and oranges and to count the number pixels. To segment the pixels of bananas and oranges from the background, pixels were filtered through RGB values. The colour threshold was manually fine-tuned to R > 150 and G > 150. This effectively filtered the pixels from the black conveyor. While some parts of the conveyor were still recognized, this problem was overcome by implementing the counting of pixels within the bounding box of each fruit.

D. Tracking

After detection in each frame, Kalman filter is implemented in this project to track and estimate the motion of each detected fruit. The tracking procedure contains following steps: firstly apply Kalman filter to predict the possible positions of each detection result in the frame, then calculate the Euclidian distance between each predicted centroid of the tracks and each centroid of the detection. After that, matlab built-in *assignDetectionsToTracks* function is applied, which takes the distances mentioned above as input and implements Hungarian algorithm to minimize the distances, and this function can return the assignment between tracks and detections. Given that in some frames fruits are wrongly detected, the tracks are regard reliable only when the track exist few frames (in this case the minimal number of frames is set to 8).

Each track is an object that contains attributes including id (the number of track), boundary box, label (fruit label in the track), exist age, total visible count and consecutive invisible count. Total visible count enables the track be visible until object is tracked in several frames, which helps to denoise tracking result. The track will be deleted after several frames when tracked object is not detected anymore, and the number of these frames is maintained by key 'consecutive invisible count'.

E. Counting

This section is to introduce the counting method. In this project, counting should be a final step, which cover all 5 types of fruits. Utilizing the known track and setting a cutting line, the number of fruit can be accumulated. Counting the number of fruit in the videos is based on the tracking result instead of detection result, thus a fruit will be counted only once. The counting method is illustrated below: firstly extract reliable tracks whose existing age is longer than 8 frames from the whole tracks set.

Next check if the attribute 'counted' of each reliable track is 0, if yes, then check if its top boundary reaches the line y=100, and if the result is true then read the label of the track and increment is corresponding fruit count and its 'counted' attribute will be set 1. After this flag labeled 1, the program will skip this counting part so that a single track can be only counted once.

IV. Experiment Results

A. Detection Result

The detection and classification are required in this project. Based on the above method description, Faster-RCNN can effectively perform detection and classification for the interesting object in provided videos, whatever for blob analysis or segmentation analysis. Faster-RCNN works frame by frame. The appropriate performance is detector should detect each type of fruits correctly and label them with correspondent classification. Moreover, bounding boxes are demanded to circle the interesting object, rather than drifting to wrong area. Missing the object is also serious problem for this design. We consider the result is true if the performance can meet the standard we mention above, instead, it is wrong including false positive, true negative and bounding box shifting.

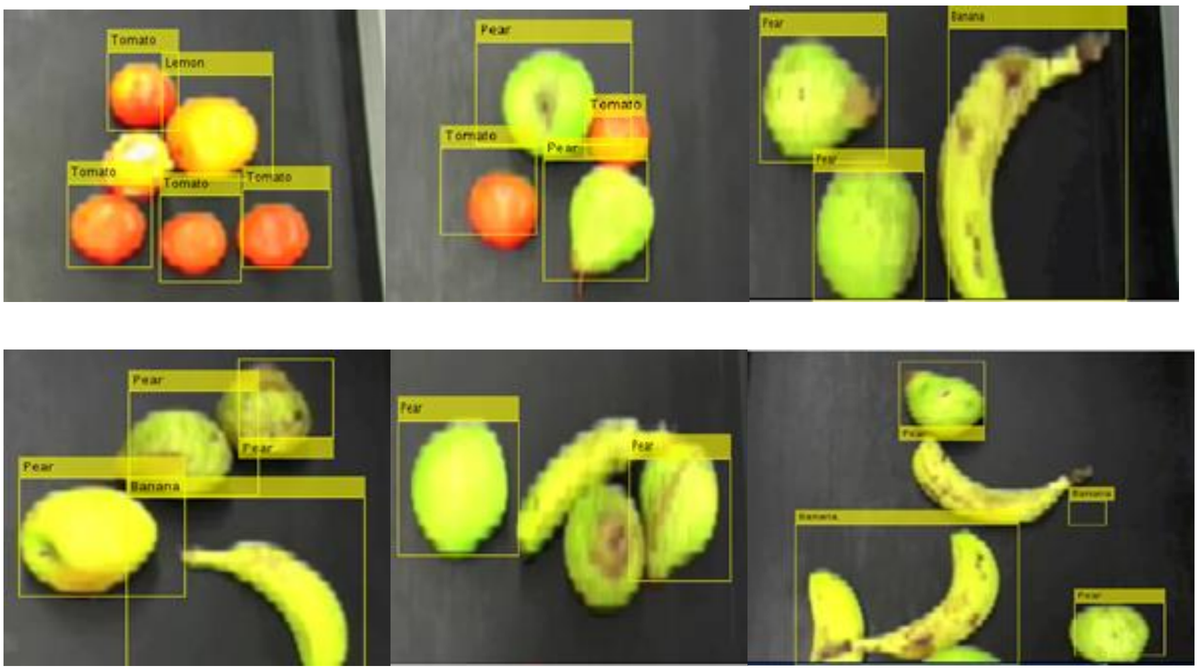


Figure 4: Detection performance for bananas, apples, pears, tomatoes, and lemons. The first row shows the correct detection result. The second row shows the incorrect detection performance, including false positive and true negative detection as well as bounding box shifting.

Figure 4 shows that most objects can be detected and classified, but there are still a few wrong detection and classification. Calculating the accuracy of this design are needed. Therefore, type and number of fruits provided in videos are calculated manually. Then by dividing the total time of error for each type of fruits by the total number of each kind of fruit, the error rate and accuracy rate can be obtained. The errors include detected errors, the classified error, bounding boxes drifting, and missing fruit.

Fruit	Number	Accuracy	Error
Pear	86	95.38%	4.41%
Banana	67	84.44%	15.56%
Lemon	23	86.96%	13.04%
Apple	26	75%	25%
Tomato	64	88.34%	11.66%

Table 1: Calculating the number of each type of fruit in all provided videos, and also calculating the accuracy and error rate with specific formula.

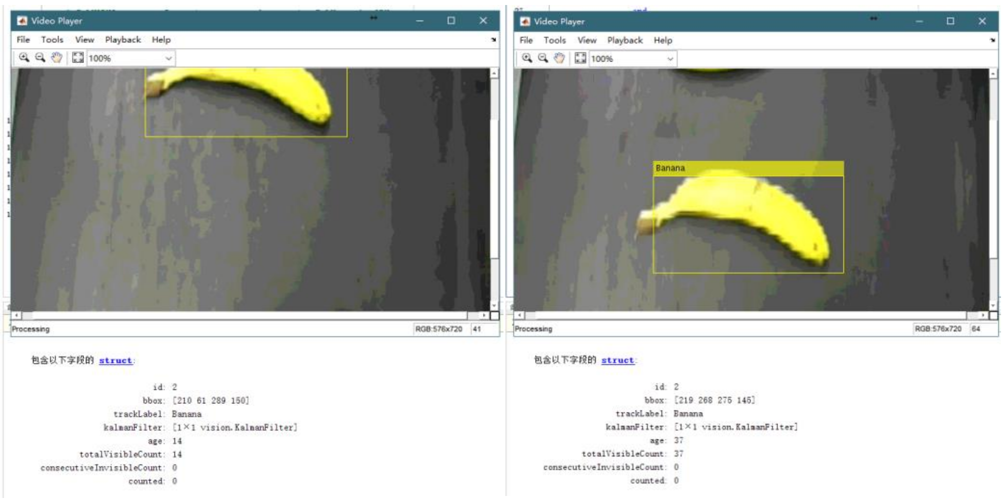
According to following formula, we can compute the accuracy and error rate.

$$\frac{Num_{error}}{Num_{TOTAL}} = Error \qquad \frac{Num_{true}}{Num_{TOTAL}} = Accuracy$$

Error rate is equal to the total number of wrong performance divided by the total number of each type of fruits. Similarly, accuracy rate is equal to the total number of correct performance divided by the total number of each type of fruits. Meanwhile, as table 1 shown, the most accurate result appears in the pear detection, followed by the tomatoes detection, which can up to 95.38% and 88.34% respectively. For banana detection and lemon detection, this Faster-RCNN we trained performs unsatisfactorily with 13.56% and 13.04% error rate separately. But most surprisingly, new trained Faster-RCNN only has 75% accuracy rate in apple detection, which means up to one quarter error rate for this type of fruits in this design. A deep analysis of this performance in the following discussion part.

B. *Tracking results*

Tracking result is demonstrated in figure 5. From that figure it can be seen that a single track is shared by different frames for a single object since the track id is same. When a fruit is detected in the first time in the video, a new track will be assigned to that object. Then the track uses Kalman filter to estimate its predicted location in the next frame and draws a boundary box for that. When detected fruit has disappear from the video, its track will be labeled 'predicted' and after several frames (in this project we set this threshold 8 frames) the track for this fruit will be deleted.



```
包含以下字段的 struct:

                        id: 2
                      bbox: [210 61 289 150]
                trackLabel: Banana
              kalmanFilter: [1×1 vision.KalmanFilter]
                       age: 14
         totalVisibleCount: 14
consecutiveInvisibleCount: 0
                   counted: 0
```

```
包含以下字段的 struct:

                        id: 2
                      bbox: [219 268 275 145]
                trackLabel: Banana
              kalmanFilter: [1×1 vision.KalmanFilter]
                       age: 37
         totalVisibleCount: 37
consecutiveInvisibleCount: 0
                   counted: 0
```

Figure 5: Tracking result.

Figure 6 shows the wrong tracking results, which is due to wrong detection as well as wrong estimation by Kalman filter. However, these wrong tracking result shows a small impact on counting result. This is because these wrong tracks will be recognized as unassigned tracks and will be deleted quickly. Besides, they tend to move to the top and sides edges instead of the bottom edge.

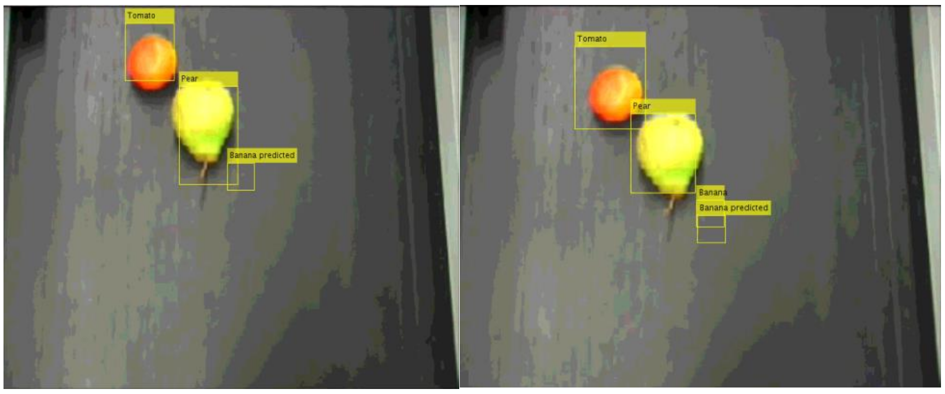


Figure 6: Wrong tracking result.

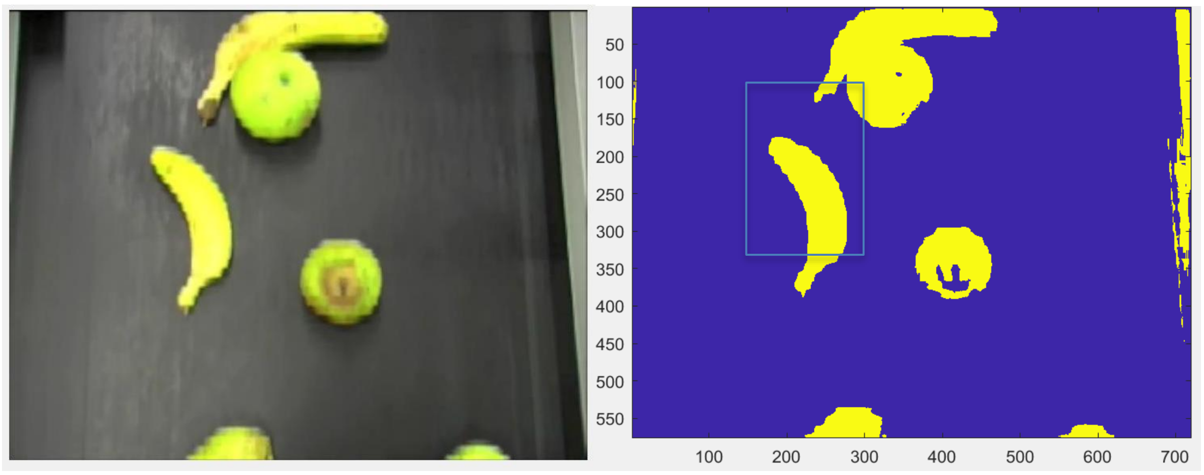


Figure 7: Original frame on the left, segmentation from background on the right. Some parts of conveyor on the right were falsely detected. However, this problem was resolved by only counting the pixels of fruits in each of their bounding boxes.

C. Segmentation and Pixel Counting Result

The result of segmentation can be seen in Figure 7. The fruits are clearly segmented from the background. The number of pixels within the bounding box of the banana in this case was 6421. Although it is difficult to confirm the exact number of pixels, it is possible to infer from the grid that this value is correct.

V. Discussion

A. Performance

This section analyzes the detection performance, including incorrect detection and overlapping detection. There are three reasons behind incorrect detection of fruits. Firstly, in order to training the neural network Alexnet and detector Faster-RCNN, a large number of training set are needed. Nevertheless, the Fruit 360 training set used to train the neural network

Alexnet in this design only has 2210 images. Each fruit has few hundred training images, which is insufficient for re-training the neural network. According to Bargoti and Underwood, as the number of training images increasing, detection performance will be better, even though it will be close to convergence finally [11]. Therefore, inputting more training data into neural networks and detector can improve the precision of the detection and classification.

In addition to the number of training images, the quality of provided videos is also significant for detection and classification. Size of frames of provided videos is 576 x 720, which means that resolution of these videos is low. This can lead to lower precision in detection. Therefore, providing higher resolution videos can improve the performance of detection, by allowing the detector to recognize detailed features.

Finally, Apple detection has the lowest accuracy shown in the above result figure. The apple and pear images are quite similar in training set, and also the pears and apples have similar color depth and features in videos, which might be the reasons why the detector performs imperfectly.

As can be seen in figure 8, the Faster RCNN detector in this project can detect overlapping objects robustly, while in tradition methods such as color segmentation and SIFT, this task is extremely hard since features from different objects are difficult to be separated.

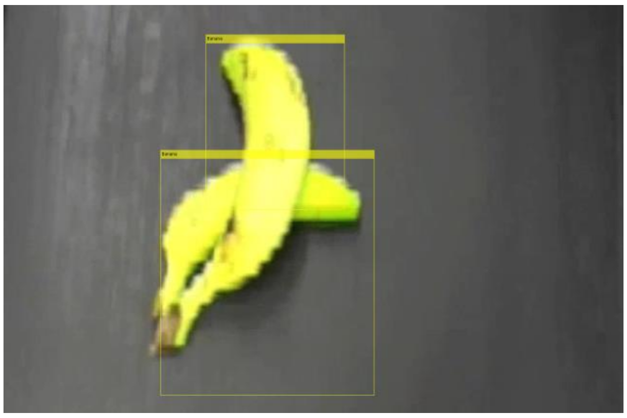
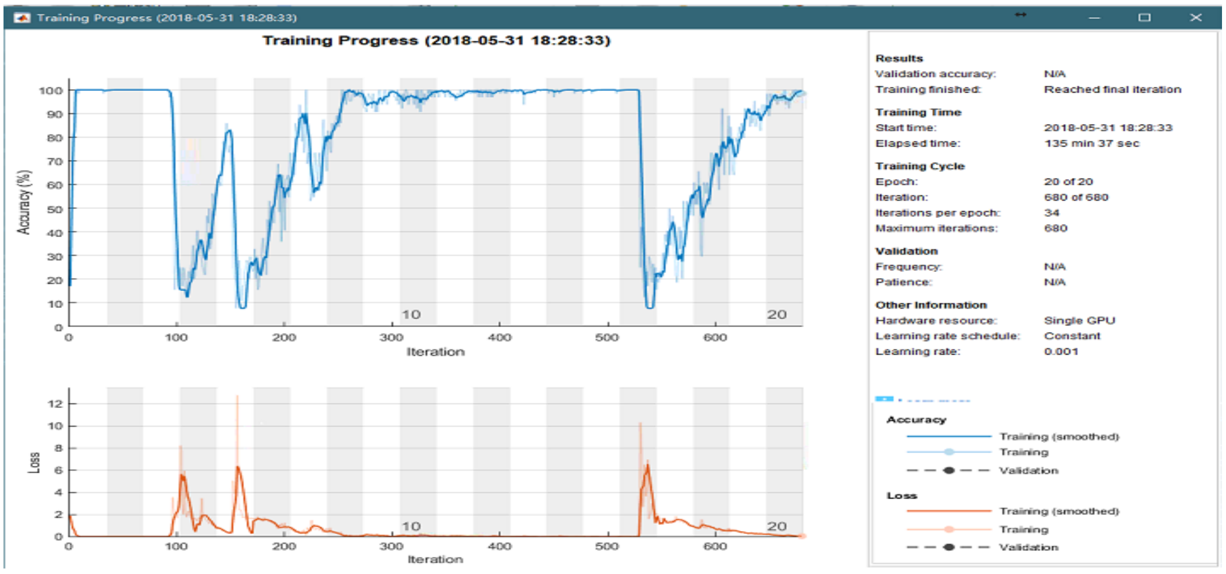


Figure 8: Overlapping detection.

B. Comparison between Alexnet and VGG19

This section compares the Alexnet and VGG19 neural network. In the above introduction of neural network, after training, the Alexnet is the better option compared to the VGG19. Since not only the accuracy is stable, but also there is no fluctuation during the training progress for Alexnet.



Training Progress (2018-05-31 18:28:33)

```
Results
Validation accuracy:        N/A
Training finished:          Reached final iteration

Training Time
Start time:                 2018-05-31 18:28:33
Elapsed time:               135 min 37 sec

Training Cycle
Epoch:                      20 of 20
Iteration:                  680 of 680
Iterations per epoch:       34
Maximum iterations:         680

Validation
Frequency:                  N/A
Patience:                   N/A

Other Information
Hardware resource:          Single GPU
Learning rate schedule:     Constant
Learning rate:              0.001
```

Figure 9: Progress of retraining the VGG19 neural network

Figure 9 shows the training process of VGG19. The parameters of training options are identical with Alexnet, for example, learning rate, maximum epoch. As the image shown, although in the first three epochs can be 100%, accuracy is erratic until the 11st epoch that again achieve to 100% and maintain it. However, in the last four training epochs, accuracy increase steadily from around 10% to 99%.Generally, VGG19 is better than Alexnet, embodied in deeper DCNN, larger number of channel and lower error rate. Arising the above result might be blamed to the learning rate or batches arguments. Different neural network should have different training options so that using same arguments for training options of different network might result in the unsatisfied outcome. Meanwhile, since training the VGG19 neural network requires more time and higher computing equipment. For example, Alexnet can be re-trained via single CPU in around one hours while retraining VGG19 takes over 2 hours via single GPU. Therefore, based on above analysis, Alexnet is an optimal option for this project.

VI. Conclusion

This paper briefly presented an implementation of fruit detection, tracking, and counting. The Alexnet neural network, Faster-RCNN were used to do this. Mainly, we introduced the transfer leaning, Faster-RCNN, tracking and counting method. By using the pre-trained Alexnet, the number of required training data and time was vastly reduced. Based on Alexnet, Faster-RCNN is used to detect the interest object in provided videos. By implementing Alexnet into Faster-RCNN and training it further, detection performance was greatly improved as well as the detection speed. It is likely that the low resolution of the videos hindered the detection performance as detailed features of the performance cannot be extracted by the model, resulting in false detection and bounding box shifting. Using Kalman filters can predict the next position of interest object consequently this design can implement the tracking. In addition, counting can be done by tracking result, setting the cutting line and checking whether the top boundary of fruits reach the line. Utilizing this method, the number of each kind of fruits can be accumulated and outputted in real time.

This design can stimulate the improvement in many related fields, including the agriculture industries, sorting factory, even though it is required to improve the accuracy and speed. Additionally, through learning transfer learning, Faster-RCNN and other methods used in this design, other applications, for example, car detecting, can also implemented.

Reference

[1] R. Zhou *et al*, "Using colour features of cv. 'Gala' apple fruits in an orchard in image processing to predict yield," *Precision Agriculture,* vol. 13, *(5),* pp. 568-580, 2012.

[2] Wang *et al*, "Design of Crop Yield Estimation System for Apple Orchards Using Computer Vision. Presented at 2012 ASABE Annual International Meeting; Dallas, TX, USA, 29 July–1 August 2012.

[3] K. Yamamoto *et al*, "On plant detection of intact tomato fruits using image analysis and machine learning methods," *Sensors (Basel, Switzerland),* vol. 14, *(7),* pp. 12191-12206, 2014.

[4] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," International journal of computer vision, vol. 104, no. 2, pp. 154–171, 2013.

[6] S. Bargoti and J. P. Underwood, "Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards," *Journal of Field Robotics,* vol. 34, *(6),* pp. 1039-1060, 2017.

[7] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster RCNN: Towards Real-Time Object Detection with Region Proposal Networks," CoRR, vol. abs/1506.0, 2015, Available: http://arxiv.org/abs/1506.01497

[8] Sa *et al*, "DeepFruits: A Fruit Detection System Using Deep Neural Networks," *Sensors (Basel, Switzerland),* vol. 16, *(8),* pp. 1222, 2016.

[9] Mureşan H, Oltean M, "Fruit recognition from images using deep learning", arXiv preprint arXiv:1712.00580, 2017.

[10] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, (6), pp. 84-90, 2017. . DOI: 10.1145/3065386.

[11] S. Bargoti and J. Underwood, "Deep fruit detection in orchards," in 2017, DOI: 10.1109/ICRA.2017.7989417.

Peer Review

In this project, our team worked effectively. Do Kyoun Lee （u5808720） mainly took responsible for segmentation and pixels counting. Coding for segmentation and pixels counting are done by him. And also, Corresponding contents in report are also completed by him. Xiaochen Liu (u6312197) worked for detecting and tracking. Part of detection task are implemented by him. Also he did the tracking for the project. As for report, he also did the corresponding part. Finally, I (u6516811) did the training neural network and did part of detection task. Also I wrote the related part in the report and also wrote the readme file. Each member also helps other members in need. I think every member has the same contribution in this project. Finally, thanks to team members help me in this project, I can learn a lot from this project.