CUPBOARD COUNTER: VIDEO AUGMENTED OBJECT SEPARATION AND IDENTIFICATION

Zhengfu Li and Alec Li

Georgia Institute of Technology

ABSTRACT

Low-cost, easily implemented automated inventory tracking is of growing interest for industry. These systems do not require high accuracy but must be easy to integrate with the existing processes. This paper describes the implementation of one such system, relying on a mix of video frame differential comparison, watershedding-based object separation from still frames, and a neural network that identifies the pre-separated objects in the frame. The examination of each frame relative to previous frames enabled the application to work despite occlusion and enhance object isolation. The object separation protocol further augmented this performance, enabling the cleanest possible data to be passed to the neural network. Compared to the typical traditional way of features representations, deep learning tools such as deep convolutional networks reveal a better way to achieve higher accuracy and efficiency. Overall, the system shows promise for enabling this type of easily-implemented inventory tracking.

Index Terms **—** Object Separation, Video Analysis, Machine Learning, Neural Networks, Identification

1. INTRODUCTION

As consumer demands for personalization continue to grow, companies are increasingly under pressure to minimize inventories and utilize a "just-in-time" inventory system [1]. Certain key components must be carefully tracked, leading to a high level of development of inventory tracking systems. Most manufacturers' mission-critical components are currently tracked using a variety of methods ranging from manual counts to optical imaging to RFID tags. However, many other smaller components or indirect goods remain untracked. These vary widely from industry to industry, but can include nuts and bolts, screws and drill bits, paint cans, low-value components, office supplies, and food. While it would be advantageous for companies to keep track of these items, they often don't because of the cost of inventory tracking, choosing to overstock instead. While this generally works, it is inefficient and susceptible to human error, potentially resulting in production-stopping shortages. It would be advantageous to these companies to be able to track their low value items like they track their high value items, and at a reasonable cost [2].

Fortunately, for most of these items, a perfect count is not necessary. With an approximate count, some basic data about usage rate and delivery time will enable proper stocking. In applications like this, simplicity and low cost matter much more than high precision – some error in count is tolerable. Such a system could even be marketed for consumer use, helping them, for instance, keep track of the fruit in their pantry.

2. PROJECT DIRECTION

For these sorts of applications, a system was developed to analyze a mixed-inventory storage location, where objects of different types could be stored with unconstrained arrangement. The system identifies each type of object present and the number of each present. While the application is relevant for many other scenarios, the scenario that will be examined is the tracking of fruit placed in a pantry environment. This is applicable for businesses that provide food for their staff as well in a typical home. It is implemented in a manner extensible to various other scenarios and types of objects.

The field of optical object counting and tracking is by no means a new one. Various methods have been proposed to perform this task for various applications. However, occlusion is a frequent problem. Solutions based on low-level object completion using deep neural networks have been proposed [3, 4], however, these are complex and computationally expensive and not always reliable. This program is architected to mitigate this occlusion dilemma and provide the simplest input possible to the neural network recognizer.

This implementation uses a trained neural network to classify each type of object, making it extensible beyond fruits to nearly any type of inventory. As opposed to analyzing an entire frame with the neural network, this implementation separates each individual object in the scene and passes each to neural network separately, trying to keep

samples as pure as possible. To improve this separation, a knowledge of previous states will be utilized. Since the system can easily pull from a constant flow of data; i.e. a video feed, it can retain information between frames. Since objects are generally not all put into the scene at once, examining the difference between frames further aids separation of newly added objects. Additionally, this method introduces a memory between frames, which enables more data to be passed to the neural network when objects are not occluded in some frames but are occluded in others.

2. PROJECT DESCRIPTION

2.1. Video Analysis Component

The source of analyzed data will be a video stream; thus, this system accepts a video input and processes it into a tally of the inventory present. This video input need not be a high frame rate. In this implementation, a frame rate of about 4 frames per second (FPS) was used. Since this system focuses on the change in between frames, the system could remain off when, for instance, the cupboard doors are closed, and simply turn on when changes are being made to the monitoring area.

The video analysis component sequentially processes frames of the video, skipping frames if desired. First, each frame is analyzed for movement. A scene should only be analyzed for new objects if, for instance, a hand is in the video, or if items are rolling around after being put into the frame. Thus, if movement is detected, the frames is discarded. Movement analysis is performed using two methods. The first method is the structural similarity index (SSIM) between each frame and the preceding frame. A high SSIM, or low dissimilarity index (equal to 1 – SSIM) implies a still frame, circled in magenta in Figure 1. The second method compares each frame to the series of frames preceding it to determine if shows a local minimum in terms of active pixels in the frame. This method exploits the fact that to put an additional object into the frame, a hand or other carrying device enters the frame then exits the frame, increasing the number of active pixels when it is present. Since a local minimum is generated when the hand exits the frame, this method determines if a frame is still by whether it is a local minimum. These are circled in purple in Figure 1. The alignment of these minima is a good sign; combining these two methods results in a more robust method of determining whether a frame is still. Each minimum corresponds to a single "key frame," or still frame that indicates that an additional object has been added.

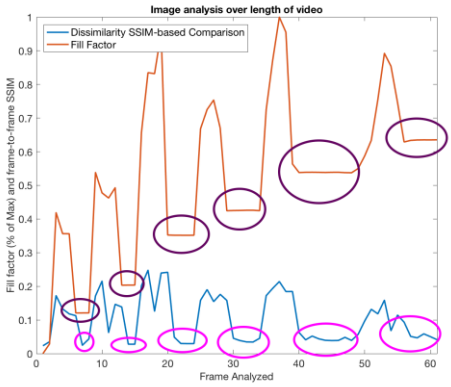


Figure 1. Fill factor, or active pixels as a percentage of the maximum number of active pictures and dissimilarity index of frame series, showing matching troughs as desired.

2.2. Object Separation Component

Each of these key frames contains the objects of the previous key frame plus the objects that have been added. The object count of the previous key frame is remembered and the difference between that frame and the next is examined. This difference, however, cannot be a simple difference due to noise and slight object movement between the frames, as shown in Figure 2. Thus, the difference must be filtered using erosion then reconstruction to eliminate undesirable, small differences, and only keep differences that are maintained over a large spatial area [5].

Figure 2. Simple difference between key frames showing effects of noise and object movement.

This difference shows the location(s) where new fruit(s) have been added. If a single object has been added, no object separation is necessary, or as in Figure 2, if two objects are added simultaneously, then the only those two objects must be separated and other potentially disruptive objects can be ignored.

However, object separation is still necessary because multiple objects can be added in the same frame. First, the program checks for non-overlapping objects by filling in a gradient magnitude. For newly added objects that overlap, a more complicated separation procedure occurs [6]. This is performed by preprocessing the image by opening and closing it with a structural element, finding local maxima, and segregating them using the gradient magnitude, and

separating them using the watershed technique, as shown in Figure 3. Each separated object is then passed on to the object identifier.

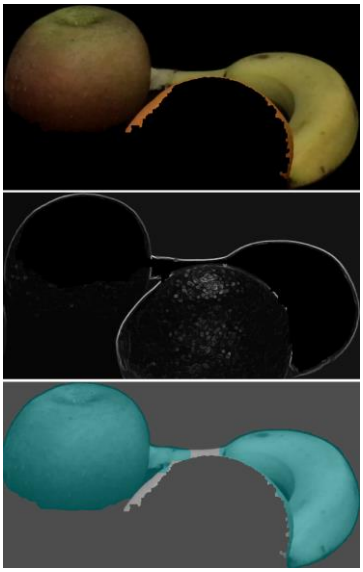


Figure 3. *First:* Difference image after erosion and reconstruction. *Second:* Gradient Magnitude. *Third:* Separation regions superimposed on image.

2.3. Object Identification Component

The Object identification is based on a deep convolutional neural network, which is designed from scratch and trained to recognize 7 kinds of fruits. The network is implemented by MatConvNet and adopts SimpleNN, a simple but efficient wrapper suitable for networks consisting of linear chains of computational blocks. Plenty of experiments are conducted at initial period to decide which method to use, as fine-tuning a pre-trained deep network, namely VGG-F [7], remains a powerful approach. It turned out that adapting the pre-trained network to our small target dataset was rather difficult and a merely 74.4% of accuracy was obtained. Three SimpleNN models are designed corresponding to three different sizes of input images and comparison between them are displayed in the next section. Figure 4 shows their network structures.

layer	0	1	2	3	4	5	6	7	8	9	10	11
type	input	conv	bnorm	mpool	conv	bnorm	mpool	conv	bnorm	relu	conv	softmxl
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	layer9	layer10	layer11
support	n/a	5	1	2	10	1	2	10	1	1	1	1
filt dim	n/a	3	n/a	n/a	30	n/a	n/a	70	n/a	n/a	700	n/a
num filts	n/a	30	n/a	n/a	70	n/a	n/a	700	n/a	n/a	7	n/a
stride	n/a	1	1	2	1	1	2	1	1	1	1	1
pad	n/a	0	0	0	0	0	0	0	0	0	0	0
rf size	n/a	5	5	6	24	24	26	62	62	62	62	62
rf offset	n/a	3	3	3.5	12.5	12.5	13.5	31.5	31.5	31.5	31.5	31.5
rf stride	n/a	1	1	2	2	2	4	4	4	4	4	4
data size	64	60	60	30	21	21	10	1	1	1	1	1
data depth	1	30	30	30	70	70	70	700	700	700	7	1
data num	43	43	43	43	43	43	43	43	43	43	43	1
data mem	688KB	18MB	18MB	4MB	5MB	5MB	1MB	118KB	118KB	118KB	1KB	4B
param mem	n/a	9KB	480B	0B	820KB	1KB	0B	19MB	11KB	0B	19KB	0B

parameter memory|20MB (5.1e+06 parameters)|
data memory| 52MB (for batch size 43)|

(64 × 64 pixels)

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
type	input	conv	bnorm	mpool	conv	bnorm	mpool	conv	bnorm	mpool	conv	bnorm	relu	conv	softmxl
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	layer9	layer10	layer11	layer12	layer13	layer14
support	n/a	9	1	2	7	1	2	7	1	2	3	1	1	1	1
filt dim	n/a	3	n/a	n/a	30	n/a	n/a	70	n/a	n/a	280	n/a	n/a	2800	n/a
num filts	n/a	30	n/a	n/a	70	n/a	n/a	280	n/a	n/a	2800	n/a	n/a	7	n/a
stride	n/a	1	1	2	1	1	2	1	1	2	1	1	1	1	1
pad	n/a	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rf size	n/a	9	9	10	22	22	24	48	48	52	68	68	68	68	68
rf offset	n/a	5	5	5.5	11.5	11.5	12.5	24.5	24.5	26.5	34.5	34.5	34.5	34.5	34.5
rf stride	n/a	1	1	2	2	2	4	4	4	8	8	8	8	8	8
data size	72	64	64	32	26	26	13	7	7	3	1	1	1	1	1
data depth	1	30	30	30	70	70	70	280	280	280	2800	2800	2800	7	1
data num	43	43	43	43	43	43	43	43	43	43	43	43	43	43	1
data mem	871KB	20MB	20MB	5MB	8MB	8MB	2MB	2MB	2MB	423KB	470KB	470KB	470KB	1KB	4B
param mem	n/a	28KB	480B	0B	402KB	1KB	0B	4MB	4KB	0B	27MB	44KB	0B	77KB	0B

parameter memory|31MB (8.2e+06 parameters)|
data memory| 70MB (for batch size 43)|

(72 × 72 pixels)

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
type	input	conv	bnorm	mpool	conv	bnorm	mpool	conv	bnorm	mpool	conv	bnorm	relu	conv	softmxl
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	layer9	layer10	layer11	layer12	layer13	layer14
support	n/a	7	1	2	11	1	2	12	1	2	3	1	1	1	1
filt dim	n/a	3	n/a	n/a	30	n/a	n/a	70	n/a	n/a	280	n/a	n/a	2800	n/a
num filts	n/a	30	n/a	n/a	70	n/a	n/a	280	n/a	n/a	2800	n/a	n/a	7	n/a
stride	n/a	1	1	2	1	1	2	1	1	2	1	1	1	1	1
pad	n/a	0	0	0	0	0	0	0	0	0	0	0	0	0	0
rf size	n/a	7	7	8	28	28	30	74	74	78	94	94	94	94	94
rf offset	n/a	4	4	4.5	14.5	14.5	15.5	37.5	37.5	39.5	47.5	47.5	47.5	47.5	47.5
rf stride	n/a	1	1	2	2	2	4	4	4	8	8	8	8	8	8
data size	96	90	90	45	35	35	17	6	6	3	1	1	1	1	1
data depth	1	30	30	30	70	70	70	280	280	280	2800	2800	2800	7	1
data num	43	43	43	43	43	43	43	43	43	43	43	43	43	43	1
data mem	2MB	40MB	40MB	10MB	14MB	14MB	3MB	2MB	2MB	423KB	470KB	470KB	470KB	1KB	4B
param mem	n/a	17KB	480B	0B	993KB	1KB	0B	11MB	4KB	0B	27MB	44KB	0B	77KB	0B

parameter memory| 39MB (1e+07 parameters)|
data memory|128MB (for batch size 43)|

(96 × 96 pixels)

Figure 4. Structures of SimpleNN network models for different image sizes.

A few tricks are tried and employed in the design process: (1) A tiny part of images are mirrored and rotated to manually augment the training set. (2) Subtract the mean from every image to make images zero-centered. (3) Regularize the network by adding dropout layer (not adopted in the structure shown above) (4) Add more hidden layers. (5) Use Batch Normalization and increase learning rate. One major problem, which is also a frequent problem in most neural networks, is overfitting, shown in Figure 5. There are a few solutions to that (such as the dropout layer) and a reasonable distribution of train and validation/test set is crucial.

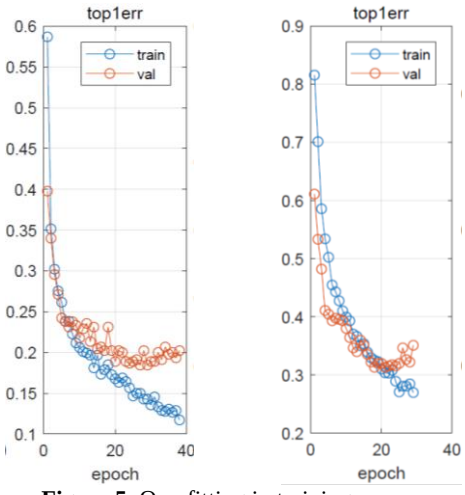


Figure 5. Overfitting in training process.

3. RESULTS

3.1. Video Analysis Results

The video analysis described performed well in determining which frames to take. Its goal is to designate a freeze frame if and only if there is no movement in the scene. As missed freeze frames only result in heavier reliance on Component 2, it is preferable to miss some freeze frames as opposed to accepting some frames with motion. In a sample of videos containing 108 motionless periods, the program correctly identified 82% of them with a 3% rate of false positives, as shown in Table 1.

TABLE 1. Video Analysis Performance.

Total Motionless Frames	Correctly identified	Missed Frames	False Positives
108	89 or 82%	19 or 18%	3 or 3%

Missed frames typically occurred when one fruit was rapidly placed in after another fruit. False positives typically occurred when the hand placing the fruit lingered, motionless.

Figure 6 illustrates an example of a missed frame. At around the 5th frame analyzed, the image was still between fruit insertions. This is visible in the drop seen in both SSIM and fill factor (circled in purple); however, SSIM does not drop below the required threshold to determine a freeze frame. While these thresholds could be raised, that would increase the chance of false positives, and thus they were set as is.

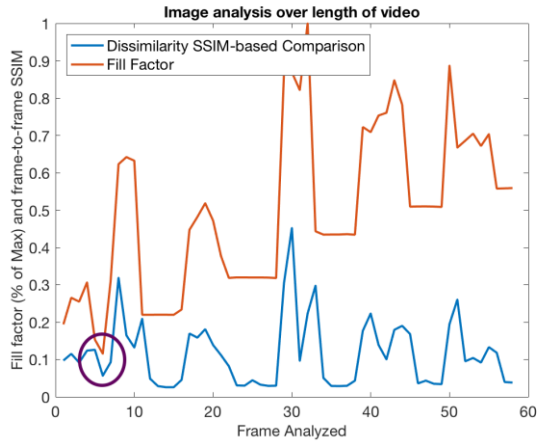


Figure 6. Graph of fill factor and dissimilarity showing a missed frame (circled in purple).

This level of performance was deemed acceptable. Higher performance could be attained at the tradeoff of processing time by increasing the number of frames analyzed per second of video, tightening the tolerances, and comparing each new frame with multiple of the previous frames.

3.2. Object Separation Results

The object separator serves to separate the fruits if either multiple fruits are added at the same time, or if single fruits are added sequentially but the video analysis component does not separate the additions, making the object data as if both were added simultaneously. Thus, the overall accuracy increases when both the object separator and the video analyzer are considered, reaching 87% correctly separated and 3% falsely identified fruits, as shown in Table 2.

TABLE 2. Combined Video Analysis and Object Separation Performance.

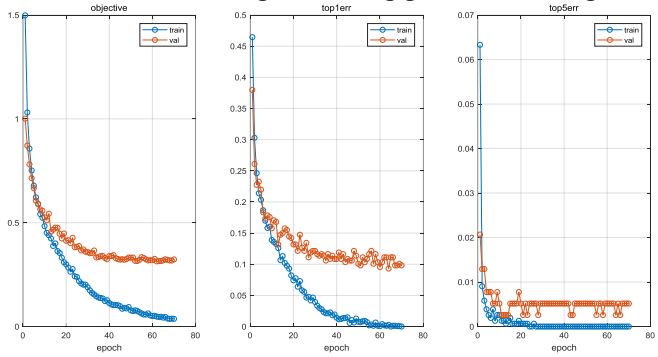
Total Fruits Added	Correctly Separated	Missed Fruits	False Positives
111	97 or 87%	14 or 13%	3 or 3%

Most missed fruit errors arose from failure to separate two adjacent, overlapping fruits because of lack of a hard edge in between them. False positives are either propagated through from things like hands passed through the video analysis or when a single fruit is separated into two fruits.

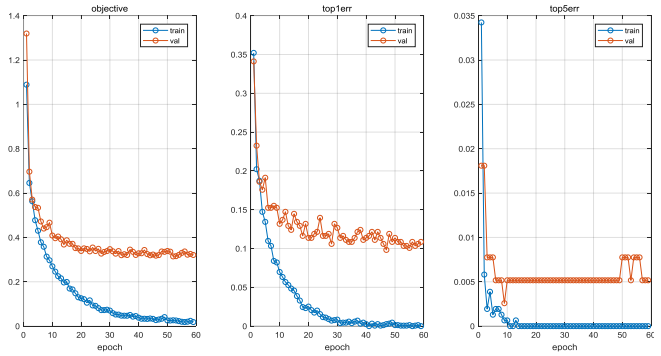
3.2. Object Identification Results

A small amount of 1935 images of 7 kinds of fruits (apple, banana, dragon fruit, orange, peach, pear, persimmon) are fed to the network and 1/5 of them are used as validation/test set, with no overlap on the train set. For all three models, batch size is set to 43 so that the train set can divided into 36 batches. The learning rate is 0.001 for 64×64 and 72×72 models and 0.0001 for 96×96 model. Training results are displayed in Figure 7.

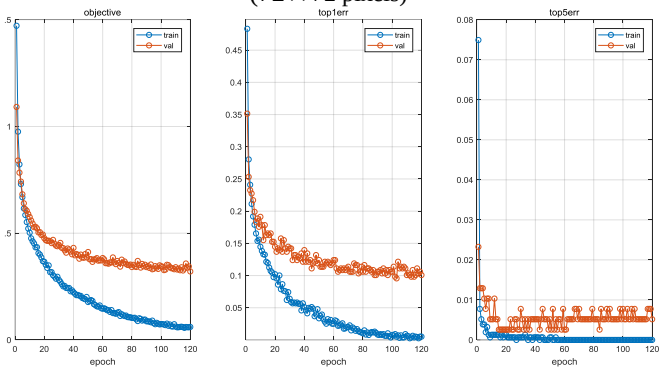
The left plot shows the training and validation loss across training epochs. Each training epoch is a pass over the entire train set broken up into batches. When the network makes mistakes, it incurs a loss and back propagation updates the weights of the network in a direction that should decrease the loss. The middle plot shows top 1 error, how often the highest scoring guess is wrong, thus standing for training and testing accuracy. The last plot shows top 5 error, how often all of the 5 highest scoring guesses are wrong.



(64×64 pixels)



(72 × 72 pixels)



(96 × 96 pixels)

Figure 7. Training results: total loss, top 1 error and top 5 error.

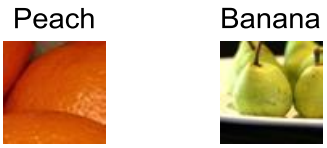
It is observed that with the same batch size, higher learning rate does help networks achieve the highest accuracy faster, as the first two networks achieve the accuracy of 90% at about 45 epochs while it takes the last one another 45 epochs. However, either batch size or learning rate improves accuracy. Specific data are shown as follows in Table 3.

TABLE 3. Highest accuracy achieved by each model of CNN and total time consumed.

Accuracy/Efficiency	64 × 64	72 × 72	96 × 96
Accuracy(Top 1 Error)	91.6%	90.2%	90.8%
Efficiency(Time Cost)	0h28m35s	0h45m08s	4h13m19s

Tested performances of the networks conform to the results shown in above graphs. Take the 64 × 64 pixels model for example, it is capable of recognizing single fruit items accurately even if there is overlap or cut-off. But too much zooming in or other noises still prevent correct predictions. Figure 8 shows successful and failed examples of using trained network to classify single items.

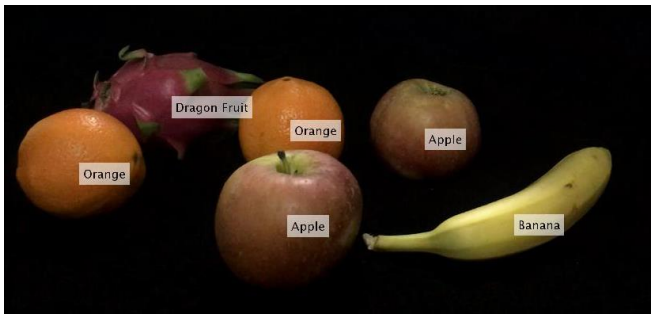
Successful examples



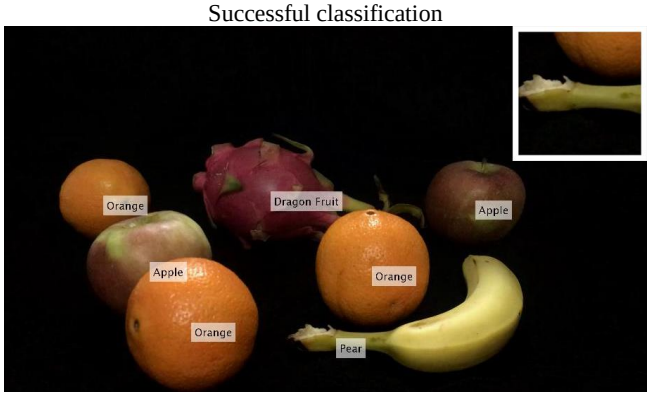
Failed examples

Figure 8. Successful and failed examples of single fruit classification using trained network.

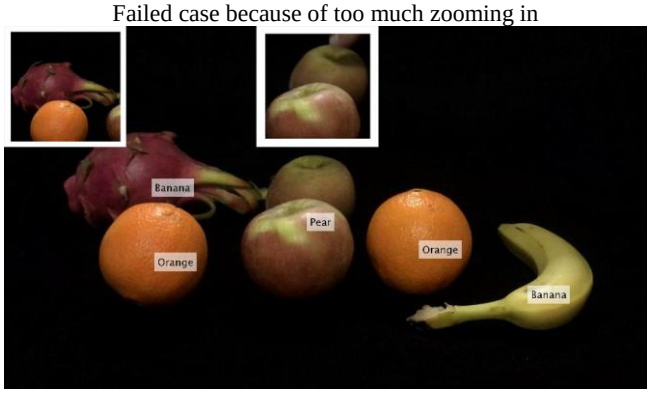
Overall performance of the system is worse due to network limitations stated above, separation error and noises in the filming process. Lack of light in experiment videos is also a factor for error increase, but in principle, darkness problems should be overcome in the training process, which illustrates lack of volume and diversity of training samples. Figure 9 shows results of overall system tests on video input.



Successful classification



Failed case because of too much zooming in



Failed cases because of multiple items in one isolated image

Figure 9. Overall system test with video input.

4. DISCUSSION AND CONCLUSION

Though automated inventory tracking has grown substantially for key process components, there remain many less critical inventories that are not tracked though doing so would be economical. The option of a simple, low-cost tracking system with minimal set up is what is necessary to bring automated inventory tracking to these untapped applications. The focus for these is simplicity and ease of use as opposed to accuracy. Thus, a best-effort optical tracking system that does not require any training or adjusting of user behavior is proposed to fit the market need.

With this directive in mind, the inclusion of frame differential analysis proved to be an effective way of getting reasonably accurate results from a simple camera setup. Even without the use of multi-frame comparison and at a low frame rate of about 4 frames per second, the algorithm to detect new object additions showed 82% accuracy. Since the missed additions only result in more work for the object separator, including it increased accuracy to 87% and left a total of 3% of false objects. This enabled simplified objects to then be passed on to the neural network.

Three models of convolutional neural networks designed for different sizes of input images have very similar highest accuracy of around 90%, while higher resolution should provide more details like the fruit surface textiles and help improve it. With better design on filter sizes and hidden layers, there is potential for superior performance. Besides, actual tests conducted show that trained models are vulnerable to certain kinds of image distortions and noises. As stated above, increase the volume and diversity of training samples is a good solution and can also help prevent overfitting in the training process.

In future development, different algorithms could be tested to detect still frames. A multi-frame comparison or a higher frame rate could be used, in addition to de-noising to improve performance. For object separation, additional tuning or the combination of multiple methods could be implemented to attempt to better separate objects. SimpleNN wrapper is used as the approach to construct the networks in this paper though, retraining the pre-trained deep network model like VGG-F on target classes still remains a promising way to achieve higher accuracy and efficiency. Modifying the last few layers of VGG-F to adapt it to our own dataset has been tried unsuccessfully but future work may reveal better results. While the overall accuracy rate of 91.6% is enough to offer some help in tracking, further increasing accuracy with these improvements would even further elevate its effectiveness.

5. REFERENCES

[1] H. Basodan, "Just In Time Inventory In Operations Management,", *International Journal of Scientific & Engineering Research*, pp. 327-329, April 2016.

[2] Ross D.F, *Distribution Planning and Control*, Springer, Boston, MA, 2015.

[3] B. Chandler and E. Mingolla, "Mitigation of Effects of Occlusion on Object Recognition with Deep Neural Networks through Low-Level Image Completion", *Computational Intelligence and Neuroscience*, pp. 1-15, 2016.

[4] B. Pepik, M. Stark, P. Gehler, B. Schiele, "Occlusion Patterns for Object Class Detection," Computer Vision Foundation, 2013.

[5] Z. Ye, H. Mohamadian, Y. Ye, "Gray level image processing using contrast enhancement and watershed segmentation with quantitative evaluation," *International Workshop on Content-Based Multimedia Indexing*, IEEE, June 2008.

[6] "Marker-Controlled Watershed Segmentation - MATLAB & Simulink Example", Mathworks.com, 2017. [Online]. Available: https://www.mathworks.com/help/images/examples/marker-controlled-watershed-segmentation.html?prodcode=IP&language=en. [Accessed: 27- Nov- 2017].

[7] "Return of the Devil in the Details: Delving Deep into Convolutional Networks", Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman, BMVC 2014.
Infor etc.