Research Article

Detection and Localization of Ripe Tomatoes Using Machine Vision

Mahyar Gohari Moghaddam[a]

[a] Department of Mathematics and Computer Science, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran.
mahya_gm@aut.ac.ir

Article History: Received: 14 July 2020; Accepted: 2 January 2021; Published online: 5 February 2021

Abstract: Tomato is one of the most important agricultural products and the most widely used fruit in the world, whose quality for the consumer depends on its appearance. Therefore, timely delivery of tomatoes to the consumer has always been one of the major concerns of farmers. The study tried to design a model for a tomato harvester robot using machine vision to be able to detect and locate tomatoes automatically and in real-time. The algorithm implemented for this purpose is based on the You Only Looked Once Version 4 (YOLO-V4) object detection algorithm. The basic dataset used in the study was 194 images at 416×416 dimensions, increased to 1008 using data augmentation methods and other pre-processing and were used for network training and testing. The use of these methods along with the two methods of Spatial Pyramid Pooling and route aggregation network in this system, despite the small value of the initial training data, has caused the network to reach 86% accuracy according to F1-Score metric for experimental data processing.

Keywords: Machine Vision, Tomato Detection, YOLO, Deep Learning, Image Processing

1. Introduction

Tomato is known as the most widely used fruit in the world given its various uses. Hence, optimal planting, growing, and harvesting of this product has always been of great significance for researchers and farmers. Detecting and classifying ripe tomatoes in farms and greenhouses is time-consuming and costly because of the non-simultaneous ripening of this product and requires daily review. Thus, using artificial intelligence algorithms can increase the productivity of time and costs for farmers. The significance of this classification is clarified when we know that to send tomatoes to various regions, they are selected and harvested with various degrees of ripeness based on the delivery time. Thus, having a system able to automatically harvest tomatoes for a specific destination with the right degree of ripeness can be very useful.

The process of detection and localization tomatoes has been associated with some challenges.

The key challenge of deep networks is the existence or possibility of collecting a large and high-quality dataset with various types of images from various categories. Another challenge related to the nature of this problem is the various light conditions in different places of the greenhouse and the different color properties of the tomatoes relative to each other. For instance, a small portion of tomatoes may look semi-ripe for different reasons, but that tomato may be ripe. This property will make it hard to classify tomatoes.

Besides the previous two challenges, other challenges this problem faces the overlap of tomatoes with each other or with other plant components such as leaves and twigs, and the detection process should be done in a way that even when part of the tomato was visible through the camera it does not interfere with the detection, classification, and localization operations.

Another issue to consider is the significance of real-time image processing, as when a robot starts moving, as soon as a tomato with the specified conditions is in sight of the robot, it has to be detected and located to harvest the tomato. In other words, it is impossible to record images of the entire movement of the robot and then process them. This is because the slow speed of operations defined for the robot can greatly increase the harvest time so that the project execution practically loses its efficiency.

Considering the significance of tomatoes among agricultural products and the cases stated about the significance of this issue in previous sections, several studies have been conducted on this issue, which we will mention several of them in the next chapter. However, there are still no solutions to perform real-time detection operations with high accuracy in real conditions where objects overlap highly.

The model suggested to solve this problem is based on the You Only Look Once (YOLO) object detection model [1] given its high speed and accuracy. Various methods of image pre-processing and post-processing of outputs have been used to solve the problem of low quality of some image sets and overlap of image objects. This operation has enabled this deep network to reach good accuracy with only 194 images for training and testing the network.

In this model, the harvesting robot moves between the rows of tomato plants on rails at a fixed distance from the plants. The camera is placed on the robot arm. After observing the tomato, the arm will move towards the tomato until it is close enough to the tomato if the tomato belonged to the specified category for harvesting. Then the tomato harvesting operation is carried out by a robot.

In this model, the operations in charge of the robot vision include detecting, classifying, and localizating tomatoes. The default categories are:

• Ripe tomatoes

• Semi-ripe tomatoes

• Unripe tomatoes

A laser rangefinder can be used to determine the longitudinal distance between the robot arm and the tomato, and the position of the tomato in the image obtained from the camera determines which direction the robot arm has to move (left, right, up, and down).

The proposed system

System inputs

The input of the system is as colored images with three channels: red, green, and blue. This image is at 416×416 pixels and their information can be displayed in a tensor of 3×416 × 416 dimensions. If the image dimensions vary from the above dimensions, one can convert the image dimensions to high dimensions using the resize tools.

There are different datasets for teaching object detection models, one of the most famous of which is the COCO dataset, which has attracted the attention of many scientists and engineers in the field of machine vision because of its variety and numerous updates. We used the pre-trained weights of the YOLOv4 model using this dataset.

We have used the dataset [2] to train the network. This dataset was labeled manually using LabelImg software and the tomatoes were divided into 3 categories: ripe, semi-ripe and unripe. This dataset has 194 images that have 440, 165, and 128 ripe, semi-ripe, and unripe tomatoes, respectively. In Figure 1, it is labeled so that the letters R, S, and U indicate ripe, semi-ripe, and unripe tomatoes, respectively.

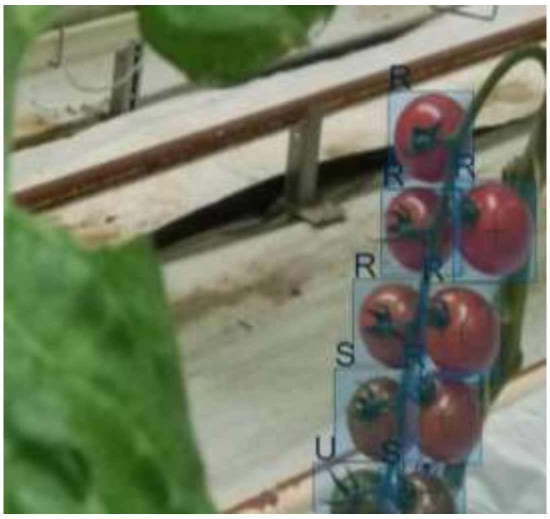


Figure 1. An example of a labeled image

Data processing

Pre-processing of the images in the machine vision field is necessary for almost any complex problem due to the noise in the images or other problems that may arise during shooting. Although today the high ability of deep networks to extract image features has led to less use of image preprocessing methods, using these methods can still help enhance the system outcomes. Thus, the images of the original dataset are randomly processed using one or more of the above methods and create new images.

2. Gaussian method

The gaussian method uses Gaussian function or normal distribution in statistics to convert each pixel of the image in image processing. A Gaussian filter approximation with a deviation of 1 and a kernel of 5×5 has been used in the study.

Cutting the images

In this method, the corners of the image are cut to prevent over-fitting. Besides creating additional artificial data, this makes the model perform well in the natural environment and in situations where part of the image is invisible. The images are cut with four various models in the study as shown in Figure 2.

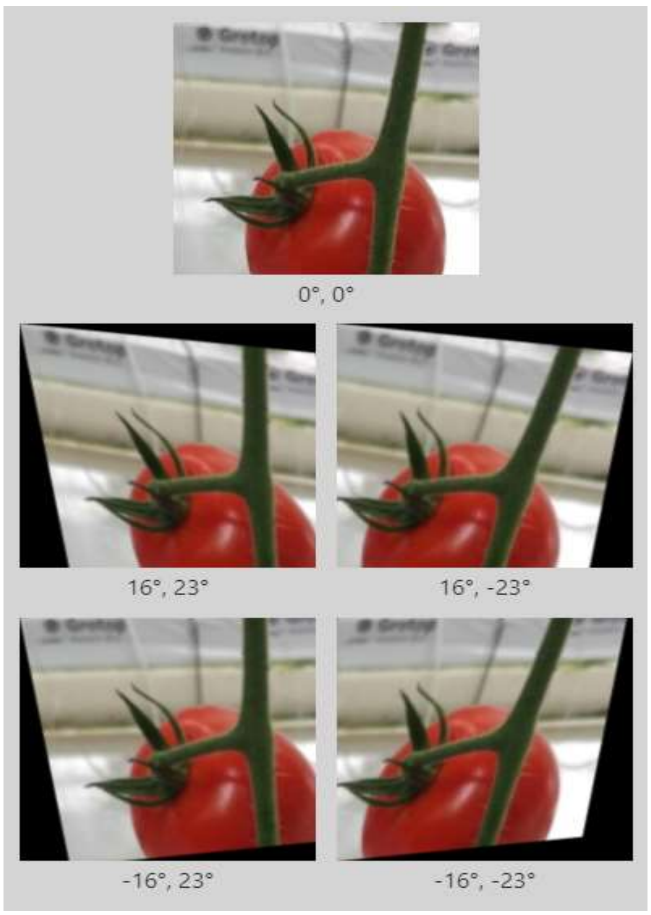


Figure 2: Image cut

Changes in luminosity

This preprocessing method can be very useful in cases where various lighting conditions can affect the results. In this study, the initial images remain randomly or unchanged, or their brightness is increased by 10%, or their brightness is reduced by 10%.

The structure of the proposed model

Two important factors have been considered to select the appropriate algorithm to solve the problem. The first factor is the accuracy of the model and then the speed of its execution. The YOLO series of algorithms [4, 8, 3], introduced and developed by Redmond et al., became one of the most popular object detection models because of their two features simultaneously. The fourth version of this algorithm, proposed by Bochkovskiy et al. [5], further enhanced the results of previous models. We will discuss how to implement and use this model to solve the mentioned problem. This method, and almost all deep networks that detect objects, are composed of three parts: the backbone, the neck, and the head, which will be examined below.

Backbone

The backbone or elementary part of the model is responsible for feature extraction. Models like VGG [6], ResNet [7], and DarkNet [8] are usually used for this purpose, each with its advantages and disadvantages. For instance, the ResNet network, although smaller in size than other networks, has a high computational cost because of its recursive nature, which can reduce processing speed. This study uses the DarkNet network, which has a much lower computational cost with an accuracy almost equal to the ResNet network. The weights of this network are pre-trained on the ImageNet dataset [9]. The architecture of this network is shown in Table 1.

Table 1. Dark DarkNet53 architecture

Frequency	**Filter size**	**Number of filters**	**Layer type**
1×	3 × 3 2/3 × 3	32 64	Convolutional Convolutional
1×	1 × 1 3 × 3 -	32 64 -	Convolutional Convolutional Residual
1×	2/3 × 3	128	Convolutional
2×	1 × 1 3 × 3 -	64 128 -	Convolutional Convolutional Residual
1×	2/3 × 3	256	Convolutional
8×	1 × 1 3 × 3 -	128 256 -	Convolutional Convolutional Residual
1×	2/3 × 3	512	Convolutional
8×	1 × 1 3 × 3 -	256 512 -	Convolutional Convolutional Residual
1×	2/3 × 3	1024	Convolutional
4×	1 × 1	512	Convolutional

	3 × 3	1024	Convolutional
	-	-	Residual

Neck

Neck layers of the network are the extra layers that lie between the backbone and the head. These layers are responsible for extracting various mapping features of different stages of the backbone. YOLOv4 uses Spatial Pyramid Pooling (SPP) [10] and the Path Aggregation Network (PAN) [11], but the PAN used in this method slightly differs from the standard PAN - internal multiplication is used instead of addition. Using these two methods simultaneously can significantly enhance the accuracy of the model and the formation of the feature map with a small increase in runtime.

Head

This is a network that performs the operation of detection the localization of objects within the image or Bounding Box and performs the classification. Methods such as YOLO, based on an anchor box, produce one output for each defined auxiliary framework. In this study, 9 anchor boxes have been considered and the size of these auxiliary frameworks has been obtained using the k-means clustering method on all COCO dataset boxes. The size of these frames is shown in Table 2. And since the network output of the 5 parameters of the coordinates of the upper right corner of the object (x, y), the length of the object, the width of the object and the probability of belonging to a category and also the classes with one-hot coding need 3 parameters, the network output It will have 72 channels. This part of the network remains the same as YOLOv3 and has not changed.

Table 2. Anchor boxes

Width box	**Length box**	**Row**
13	10	1
30	16	2
23	33	3
61	30	4
45	62	5
119	59	6
90	116	7
198	156	8
326	373	9

3. Results

Software and libraries

OpenCV libraries [12] have been used to pre-process the initial images and prepare the dataset. Furthermore, Tensorflow [13] and Keras [14] libraries have been used to implement the model and train the network. LabelImg software [15] was used to label the dataset. All network training steps are done on the Google Colab system.

Network training

This section examines the training and the results of deep neural network training for the problem.

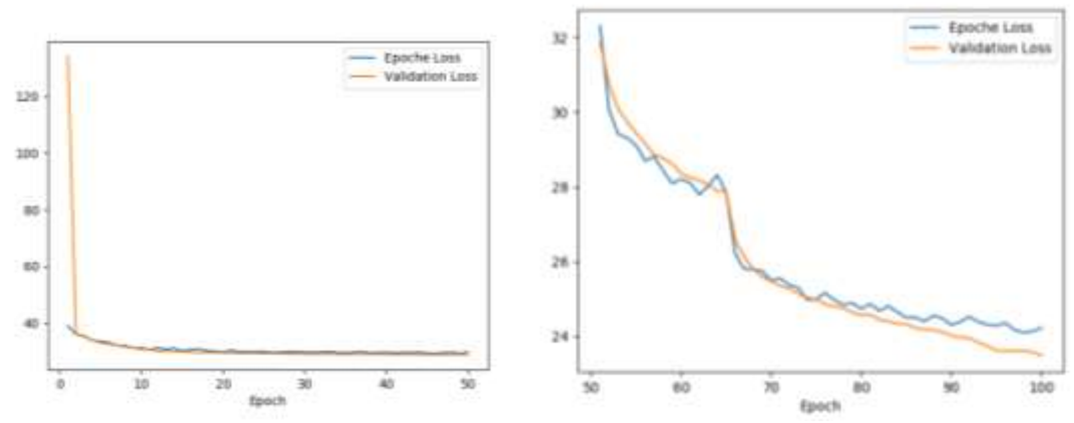
We will detect and classify tomatoes.

Firstly, we converted all the images of the initial dataset to the size of the network input - 416 × 416 - and then performed the labeling operation on the whole dataset. After performing all the pre-processing steps randomly on the initial dataset, the number of images in the dataset increased to 1008. We divided the generated data into three categories: training, experimental, and validation with 70, 15, and 15%, ratios respectively.

After several unsuccessful experiments, we used freezing the initial layers of the network to train the network and the lack of output convergence, which resulted in better network convergence. In this method, network training is done in two stages. In the first stage, the weights of the used Darknet53 network become untrained and the network training is done only on the head and neck of the network. After the network error is reduced to a certain value, all network weights can be trained and training is carried out on all layers. This will lead to better training of the head and neck of the network.

Network batch training has been used because of memory limitations. Thus, only a part of the training data is loaded in memory in each stage of training. For the first stage of training, the size of category 16 was considered and the size of category 8 was considered for the second stage of training.

Adam optimizer [16] has been used to optimize this network and the network training was done in the first stage with a learning rate of 0.001 and 50 times and in the second stage with a learning rate of 0.0001 and 50 times. In both stages of training, a method called automatic reduction of learning rate is used. In this method, if the error rate is not reduced in two consecutive stages, the learning rate is multiplied by 0.1 so that the network can better reduce the error rate. Figure 3 shows the graph of error rate reduction during training in the first and second stages, respectively.



A B

Figure 3. a) Error reduction chart in the first stage of training, b) Error reduction chart in the second stage of training

Calculating the accuracy of the outputs

In this section, we calculate the accuracy of the outputs using various validation criteria such as detection precision and recall accuracy. An artificial dataset containing 875 images and 5643 tomatoes has been prepared to better evaluate the network for the category of ripe tomatoes. Table 3 is the network accuracy separately for each category and the artificial data generated.

Table 3. Network accuracy separately for each category

Categories	Precision	Recall accuracy	F1-SCORE criteria
Ripe tomatoes	88%	90%	89%
Semi-ripe tomatoes	90%	77%	83%
Unripe tomatoes	86%	79%	82%
Total	88%	83%	86%
Artificial datasets	87%	86%	86%

Recall accuracy

The accuracy of the recall is important as it indicates how well the objects in the image are detected. This means the ratio of the number of detected objects that were correctly classified to the ratio of all the objects in the image. The system presented in the study could reach 83% accuracy for educational data with this criterion.

Accuracy of detection

Precision is important as it shows how valid the system diagnoses are. This means that a system with a lot of wrong diagnoses will not be very efficient despite its high recall accuracy because its diagnostics cannot be trusted. Precision is calculated by the ratio of the number of correctly detected objects to the total number of system diagnoses. The system presented in the study has a precision of 88% for experimental data.

F1-SCORE criteria

One of the most valid criteria used in most machine learning problems, especially machine vision problems, is the F1-SCORE criterion, as although the two accuracies of recalling and precision of each can provide a great deal of information about the performance of a network, neither can be cited alone. Thus, the F1-SCORE criterion, obtained from the harmonic mean of the accuracy of calling and diagnosis, has been used to evaluate this system, and the results obtained for this criterion and all categories are equal to 86% that proves its efficiency even for commercialization in this regard.

Comparison of the proposed method with methods based on YOLOv3 and Faster R-CNN

As already stated, YOLOv3 and Faster R-CNN are the two most widely used networks in various object detection problems. In this study, we have compared the proposed method with the results obtained using these networks for experimental data. All steps performed for data preprocessing and network training have been performed like the provided system, except for the network structure that differs from the system provided.

The results of this experiment indicate a significantly better performance of the proposed network compared to other popular networks. Table 4 compares the results obtained for these 3 methods.

Table 4ç: Comparison of the proposed method with other ones

F1-SCORE criteria	Recall accuracy	Precision	Method
86%	83%	88%	The proposed method
76%	72%	81%	YOLOv3
69%	66%	73%	Faster R-CNN

4. Conclusion

The purpose of the study was to design a model accordıng to machine vision that can detect tomatoes on plants from the images received from the harvesting robot, determine their degree of ripeness and report their location to the robot. Three classes were considered, including fully ripe, semi-ripe, and unripe to classify the ripening, and all the tomatoes that could be harvested for the robot in one image were labeled.

The tomatoes belongıng to each category can be discerned by their color. For ınstance, unripe tomatoes are almost green and semi-ripe tomatoes are pale yellow or red. In the study of this type of classification, a proposed model has been presented which, by spending a high processing cost on a single image, could extract all the tomatoes in the image and specıfy its processing rate based on the mentioned categories. The purpose of extracting tomatoes is to provide frameworks that cover all of them.

The model proposed has an accuracy of more than 86% accordıgn to the F1-SCORE criterion. As this neural network-based model costs less than 9 million decimal operations, it could be executed immediately. Given the high accuracy and the possibility of immediate execution is required for the operation of this system and any other system that acts at the moment, the proposed model can be used in a commercial harvesting robot. The high accuracy of this model and its fast performance is because of using useful preprocessing done on the basic training data and has made the model with low-quality images and low volume to have a very good performance. Using these methods as well as the highly efficient model presented has made the model have a good speed despite the high accuracy with a limited number of educational data that is unique in similar examples. The advantages of the proposed system compared to the previous studies are as follows:

1. In many past studies, the proposed systems, particularly those that only use image processing, just detect tomatoes and do not classify them, both of which perform both operations simultaneously.

2. The system needs less training data compared to similar systems using deep networks because of its lower parameters as well as the basic processing it performs on images.

3. Besides the above, given the low parameters of the system, the problem that existed in most similar systems and to increase accuracy, the network had to be made heavier and thus reduction in the speed that is solved.

4. Because of the variety of optical properties as well as the application of efficient filters, the model has a much lower sensitivity to the type and time of shooting than similar models.

Ultimately, one has to state that the proposed system has some limitations. By removing these limitations, this proposed model can be used for commercialization too. Here are some limitations along with the suggestions to solve them, by using which the limitations in future studies will be solved.

1. Using two cameras simultaneously will add the image depth feature to the images, and by using it and a shallow neural network, besides the features stated, the feature of estimating the longitudinal distance to the tomatoes could be added to this system. This feature can facilitate designing the tomato harvesting robot if implemented properly.

2. Another limitation of this system is its inability to use it at night. Despite the low sensitivity to light conditions, the system is effective only during the day and when the imaging quality is acceptable. One can use other visual sensors that provide a very clear image even in low light to use this system at night and in poor lighting conditions.

3. It is necessary to implement it on light operating systems that can be installed on robots to commercialize this system. The final suggestion is to implement this model on these operating systems to build a commercial tomato harvester robot.

4. The dataset used in the study was not designed to classify tomatoes and thus the categories were not balanced in terms of number. For future studies, it is suggested to provide a wider and higher quality collection for this purpose.

The suggestions set forth to overcome the existing limitations of the system suggested are probably effective. Different tasks can be carried out to enhance the operating and computational costs as well as the accuracy of the system. The neural networks presented in the study can be improved and fine-tuned. The development and commercialization of the system rely on these improvements.

References

1. Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. 2016.

2. Makeml.app. 2021. "Tomato Dataset | MakeML - Create Neural Network with ease." [online] Available at: <https://makeml.app/datasets/tomato> [Accessed 13 March 2021].
3. Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788. 2016
4. Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.
5. Bochkovskiy, Alexey, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection." *arXiv preprint arXiv:2004.10934,* 2020.
6. Simonyan, K. and Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556.* 2014.
7.] He, K., Zhang, X., Ren, S. and Sun, J."Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition,* pp. 770-778, 2016.
8. J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database.", *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248-255, 2009.
9. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767,* 2018.
10. He, K., Zhang, X., Ren, S. and Sun, J., "Spatial pyramid pooling in deep convolutional networks for visual recognition." In *IEEE transactions on pattern analysis and machine intelligence,* pp.1904-1916, 2015.
11. Liu, Shu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. "Path aggregation network for instance segmentation." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759-8768. 2018.
12. Bradski, Gary and Kaehler, Adrian. *Learning OpenCV: Computer vision with the OpenCV library.* " O'Reilly Media, Inc.", 2008.
13. Abadi, Martín, Barham, Paul, Chen, Jianmin, Chen, Zhifeng, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, Ghemawat, Sanjay, Irving, Geoffrey, Isard, Michael, et al. "Tensorflow: A system for large-scale machine learning." In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16),* pages 265–283, 2016.
14. GitHub. 2021. keras-team/keras. [online] Available at: <https://github.com/fchollet/keras> [Accessed 13 March 2021].
15. GitHub. 2021. tzutalin/labelImg. [online] Available at: <https://github.com/tzutalin/labelImg> [Accessed 13 March 2021].
16. Kingma, Diederik P and Ba, Jimmy. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*, 2014.