Tracking the movement of plants

Introduction

Aditya Adya

New York University

Abstract

In this report, we review a new offline approach to track the visual plant tracking by using the General Object Tracking Using the Regression Network. The *circumnutating* motion of the plants are located using the recurrent convolution network and distinctively visual features are classified by the neural networks. The network models a generic relationship between the motion of the object and the visual features and it can track any novelties excluded from the training set.

Keywords: Deep learning, Tracking objects, Plant tracking, GOTURN

Tracking the movement of plants

Introduction

Once we have the object to be tracked in one frame of the video the main purpose of the object tracking is to detect the object in the next frame of the video despite the change in location, angle of the video or any variations in the lighting of the video. We primarily focus on the object detection as part of the motion tracking model. The model should be robust enough to detect the movement for an object through the environment. The model should be able to accurately predict the motion of the object in the next frames especially for the applications like autonomous driving.

Now if we compare the object tracking models that are generalized and are trained online to its entirety from the start. They have lower performance and less accurate results because they are not focused on a special class of object, and they lack the advantage of having a large dataset of the video that has changes in location, angle of the video or any variations in the lighting of the video for the object frames.

In this report, we are going to track the object offline with the help of offline videos (dataset) to track the in motion real-time for the real-world entity. We are going to use the General Object Tracking Using Regression Networks (GOTURN) architecture for this purpose. In this report, the neural networks are trained completely offline. We are going to use frozen weights to detect any new objects in the frame without any online supervision or training of the network.

The offline unsupervised model is more accurate, robust, and quicker in the detection that the online model. Our model can track objects as fast as up to 100 FPS making it a state-of-the-art object tracking architecture.

Most of the “fastest” Neural networks are trained online and use a classification-based approach to classify the image and the required objects. However, on the contrary, we use the offline training for the tracker to model a generic relationship for the object and its motion making it completely offline. Instead of the classification-based approach, we are going to use the regression-based approach, using the single feed it forward pass to regress the apex of the leaf directly and locate it.

The robustness and speed of the model in detection are significantly better than the previous models. We can track up to 100 FPS using the GOTURN object neural network tracking algorithm which is quite state-of-the-art.

Datasets

This dataset contains 12 short time-lapse videos of the circumnutating movement of a plant kept in a pot, with a scale as a reference in a dark solid background. The link for the dataset can be found along with Code under the project GitHub link for **Plant tracker**

The videos of the dataset are divided into:

(cir01 – cir08) 8 videos: Training

(cir09, cir10) 2 videos: Validation

(circ11, cir12) 2 videos: Testing

The circumnutating movement of the plant is recorded as a time-lapse MP4 video at 30 FPS, the videos were further converted into a zip file containing each frame that was

recorded for the sake of input to the model. The videos were also annotated into an XML file, the annotation file contains each frame in the video. The annotations and the zip file were both used for the sake of input to the model.

The annotation and zip file can both be obtained for each video from an open-source website ‘vatic.JS’ by providing the MP4 video of the plant from the dataset.

The annotation makes it easier to track the object from the previous frame that can be used as an input to the model.

The video frame contains a single plant with a single apex of the plant. The annotation takes care of tracking the apex. The XML file will contain each frame with the bounding box which indicates the apex of the plant. The bounding box will have four coordinates, x1, y1, x2, and y2 representing top-left and bottom-right respectively. Each frame will be annotated with a title and bounding box as an input to train the model. The deep learning model then helps in predicting the bounding box for the next frame.

	A	B	C	D	E	F	G	H	I
1	**Index**	**Original File**	**Length(s)**	**Frame number**	**Frame rate(f/s)**	**read in mm**	**clicked pixel**	**Pixel Scale (mm/pixel)**	**Comment**
2	cir1		47.6	1428	30			#DIV/0!	Training data
3	cir2		59.2666667	1778	30			#DIV/0!	Training data
4	cir3		30.7	921	30			#DIV/0!	Training data
5	cir4		35.5666667	1067	30			#DIV/0!	Training data
6	cir5		35.6	1068	30			#DIV/0!	Training data
7	cir6		35.5666667	1067	30			#DIV/0!	Training data
8	cir7		16.5	495	30			#DIV/0!	Training data
9	cir8		37.6333333	1129	30			#DIV/0!	Training data
10	cir9		37.6333333	1129	30			#DIV/0!	Validation
11	cir10		35.7333333	1072	30			#DIV/0!	Validation
12	cir11		35.5666667	1067	30			#DIV/0!	TESTING
13	cir12		35.5666667	1067	30			#DIV/0!	TESTING
14	**Total:**		**442.933333**	**13288**					

Figure 1: Dataset Info excel sheet

As seen in the table above each video has a frame rate of 30 FPS. The training validation and testing division can be seen in the above chart. The videos contain several frames from 495 up to 1428.

Architecture

Traditionally for object tracking the models are trained online and they start from the 1st frame of the video, and later the tracker will sample the target object in batches known as foreground. Then these patches will be used to train the classifier, but the drawback of this method is that since it is trained online the model cannot use a large data set that is available. One of the other methods to do this is to use neural networks for object tracking but the neural networks are trained slower as compared to our model and hence they can track the objects up to only 15 FPS but in our case, we can track objects up to 100 FPS.

In our architecture the model will be given input in the form of two randomly chosen consecutive annotated frames, these frames will have the bounding box which indicates the apex of the plant. The model will then crop the image of the apex and perform data augmentation on the cropped image.

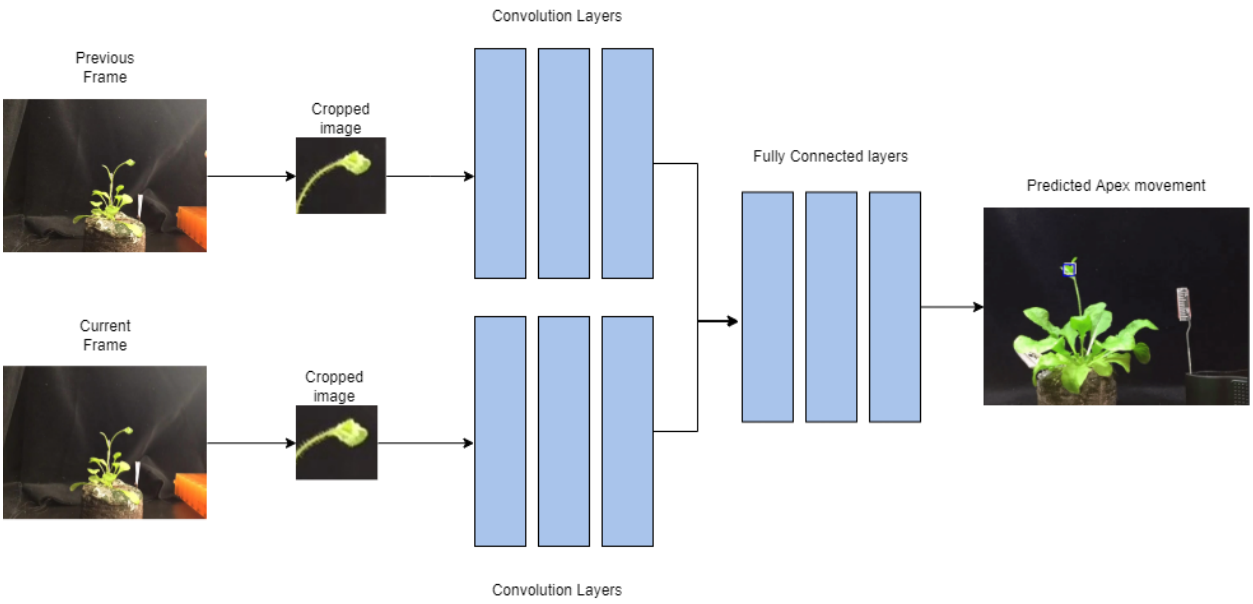


Figure 2: GOTURN Architecture for plant tracking

As shown in the image both the cropped images both will enter convolution layers separately to train the model. The two convolution layers are then fully connected to make a prediction of the movement of the apex of the plant. This is repeated for all the annotated frames in the dataset.

Data Augmentation:

While training the model, both the cropped image goes through data augmentation techniques to increase the robustness of the architecture. We used the PyTorch library to implement data augmentation techniques. We performed image flip, transforms, scaling, rotations, filtering, and cropping to get a more robust model. The data augmentation techniques improved the accuracy by just over 1.3% on our model.

Training and Loss Function:

To the train the model we perform the following steps:

1. Select a random frame f_t and its previous frame f_{t-1} and obtain the ground truth for f_{t-1}
2. Crop f_t and f_{t-1} into the window of size 2 * (k * max (height and width)). Here we observe that the optimal value for the integer k for the model is k=7.
3. The ground truth and window are compared to reduce the loss function to train the model.
4. The model gets trained for the f_t and its previous frame f_{t-1}.
5. The process is repeated for every available frame (N)

The model uses stochastic gradient descent to optimize the objective function iteratively with appropriate smoothness.

The Loss equation L1 can be represented as:

$$L1 = \frac{|x1 - \hat{x}1| + |y1 - \hat{y}1| + |x2 - \hat{x}2| + |y2 - \hat{y}2|}{4}$$

(x1, y1) are the coordinates for the top left corner of the ground truth box

(x2, y2) are the coordinates for the bottom right corner of the ground truth box

$(\hat{x}1, \hat{y}1)$ are the coordinates for the top left corner of the prediction box

$(\hat{x}2, \hat{y}2)$ are the coordinates for the bottom corner of the prediction box

The figure below shows the mean square error of the training loss over 50 epochs.

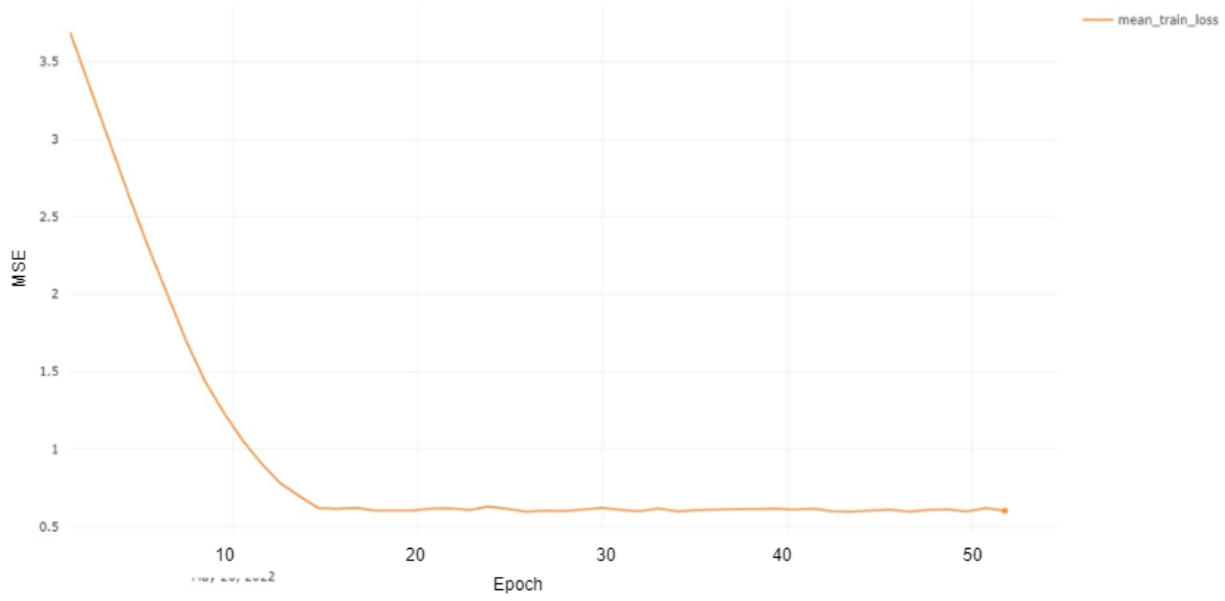


Figure 3: MSE vs Epochs for training

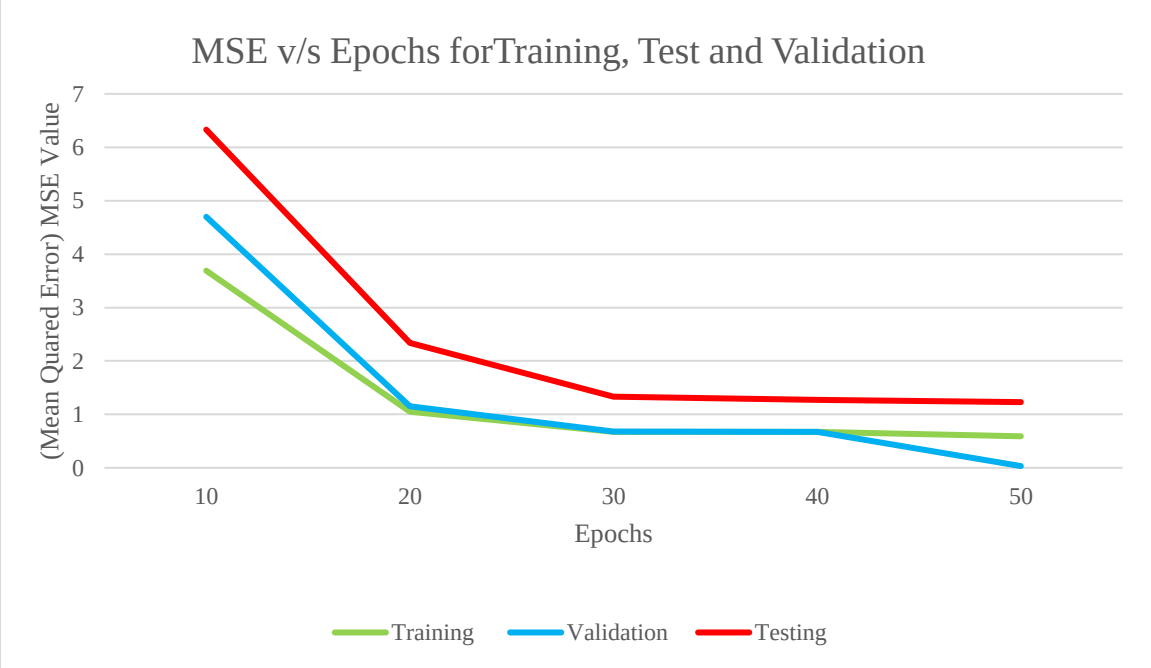
The MSE decreases significantly after 15 epochs. There is no major change in the MSE after 15 epochs and training the dataset for more than 50 epochs results in over fitting of model.

Name	Last			Min			Max		
	Value	Step	Time	Value	Step	Time	Value	Step	Time
epoch	49		5/20/22 08:07 PM	0		5/20/22 03:02 PM	49		5/20/22 03:02 PM
loss	0.464		5/20/22 08:10 PM	0.215		5/20/22 03:02 PM	3.883		5/20/22 03:02 PM
mean_train_loss	0.602		5/20/22 08:10 PM	0.595		5/20/22 03:06 PM	3.691		5/20/22 03:06 PM
val_loss	0.846		5/20/22 08:13 PM	0.033		5/20/22 03:06 PM	4.699		5/20/22 03:06 PM

Figure 4: Loss chart for training and validation.

As shown in the figure a minimum of 0.59 MSE was observed with a minimum loss of 0.215 for training and 0.033 loss on validation.

The model was then tested on two videos with over 2100 frames and the following graph indicates the overall performance of the model over the training, validation, and test dataset.



Here we can observe that both training and validation loss are less than the training loss, also the validation loss is better than the actual training loss which means our model is a good fit.

Visual analysis:

Now from the screenshots, we can observe that:

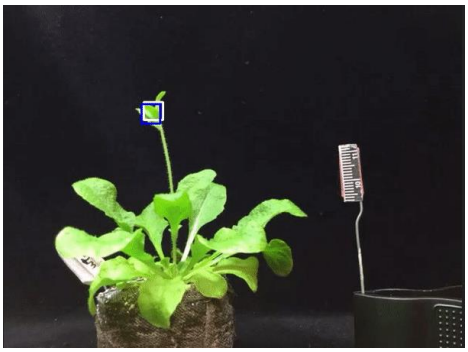


Figure 5: Frame 100

For frame 100, there is not much movement in the apex in terms of the previous frame and hence the model was able to predict quite accurately and boxes almost overlap.

Figure 6: Frame 200

In the frame 200, there was small amount of movement for the apex of the plant and the model was able to predict the position of the apex with minor inaccuracy.

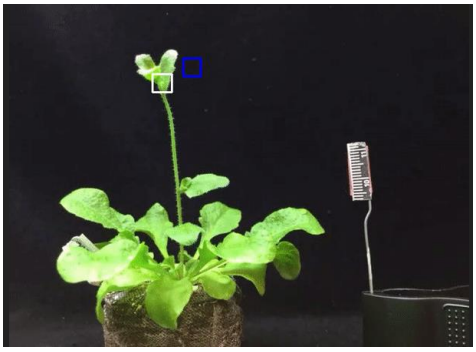


Figure 7: Frame 300

While, in frame 300, the apex of the plant moved significantly more from the previous frames and completely changed its position from far left to center, thus the error is more in terms of the prediction box.

Figure 6: Frame 400

In the frame 400 there was some movement in the apex on the plant, but the model was able to predict the position of the apex with minor inaccuracy and boxed still overlap.

Conclusion

We were able to train the model to classify and track the circumnutating movement of the apex of the plant with the help of a deep neural network by using the GOTURN architecture. The algorithm was able to correctly crop the apex in most of the frames. The size of the crop window was selected optimally such that it was able to detect the apex accurately but not include other features in the frame.

References

[1]David Held, Sebastian Thrun, Silvio Savarese (2016). Learning to Track at 100 FPS with Deep Regression Networks (GOTURN)

[2] G. Ning et al., "Spatially supervised recurrent convolutional neural networks for visual object tracking," 2017 IEEE International Symposium on Circuits and Systems (ISCAS), 2017, pp. 1-4, doi: 10.1109/ISCAS.2017.8050867.

[3] Xingyi Zhou1, Vladlen Koltun2, and Philipp Kr¨ahenb¨uhl1, 1UT Austin, 2Intel Labs, " Tracking objects as Points", 2020, doi: arXiv:2004.01177v2

[4] Joseph Redmon, Santosh Divvala†, Ross Girshick¶, Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", 2018 IEEE http://pjreddie.com/yolo/

[5] Yi Wu, Jongwoo Lim, Ming-Hsuan Yang, "Online Object Tracking: A Benchmark", 2013

[6] Tanzeel U. Rahman[a] , Libo Zhang[a], Liangju Wang[a], Dondong Ma[a] , Hideki Maki[a], "Automated leaf movement tracking in time-lapse imaging for plant phenotyping"(2020)