**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**
**"Jnana Sangama",** Belagavi-590 018, Karnataka

**BANGALORE INSTITUTE OF TECHNOLOGY**
Bengaluru-560 004

**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

# CERTIFICATE

Certified that the Project work entitled "**PREDICTING YIELD OF TOMATOES USING VIDEO FEED CAPTURED BY UAV**" carried out by,

| **USN** | **NAME** |
|---|---|
| 1BI16IS002 | Abhishek Kundu |
| 1BI16IS064 | Sourav Agrawal |
| 1BI16IS075 | Devashish Bose |
| 1BI16IS079 | Shubham Dave |

the bonafide students of Bangalore Institute of Technology in partial fulfilment for the award of Bachelor of Engineering in Information Science & Engineering of the **Visvesvaraya Technological University, Belagavi** during the academic year 2019-2020. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The Project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said degree.

| | | |
|---|---|---|
| **Mrs. Prameela R**. | **Dr J. Prakash** | **Dr. M. U Aswath** |
| Assistant Professor | Professor and Head | Principal, BIT |
| Department of ISE, BIT | Department of ISE, BIT | |

# ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned our effort with success. We would like to thank all and acknowledge the help received to carry out this project.

We would like to express our sincere gratitude to **Dr. Aswath M. U.**, Principal, BIT for providing us a congenial environment and surrounding to work in.

We take this opportunity to also thank **Dr. J. PRAKASH**., Head of the Department of Information Science and Engineering, BIT for giving us the privilege of working on such an interesting topic and for the help and guidance throughout the academic semester.

We are most humbled to mention the enthusiastic influence provided by our guide **Mrs. PRAMEELA R.**, Assistant Professor for her ideas, time to time suggestions and co-operation shown during the venture and helping make our project a success.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching and non-teaching staff of the ISE department which helped us in successfully completing the project work. We would also take this opportunity to thank our friends and family for their constant support and help.

Finally, we wish to thank all those who have directly or indirectly helped us in the completion of the project.

Abhishek Kundu (1BI1IS002)
Sourav Agrawal (1BI16IS064)
Devashish Bose (1BI16IS075)
Shubham Dave (1BI16IS079)

# ABSTRACT

With the increasing global population, increasing agricultural production is the key to meet global food security goals. The imperative is to produce more food with less resources. The projected growth rate of total world consumption of agricultural products which include food, fodder and fibre is 1.1% per annum. The total world food production is proposed to increase from 2217 million tonnes in 2005-2006 to 3291 million tonnes in 2050. With land availability and water resources becoming scarcer, increases in production in such proportions will not be easy.

Estimating yield is a critical input in crop management to increase production and productivity. Our project aims to predict the yield of tomatoes in a field using video captured from an UAV. The captured video would be first preprocessed and separated into frames.Then the detection and prediction of tomatoes would be done using a Convolutional Neural Network model from the frames obtained before. The proposed system is innovative as the existing systems are limited to using images for prediction of tomatoes which would have lower efficiency and performance due to problems faced during detection of tomatoes such as occlusion of tomatoes and so on. We intend to build a system which would predict the most accurate yield of tomatoes in any type of agricultural land,big or small.

The main advantage of our application is the relative ease in predicting the yield of the tomato crop in any given area and providing a platform to integrate technology and agriculture to increase the efficiency and decrease the time significantly.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

## 1.1 General Overview

With the development of modern agriculture, intelligent agriculture has attracted more and more attention globally. Modern agriculture includes use of computer aided technologies to improve efficiency, accuracy and reliability of crop yield estimation. Tomato is the main horticulture crop of the country with an estimated production of 19396.71 metric tonnes in 2018-19 . Tomates are widely consumed in almost all parts of the world and are a large part of our day to day diet. Increase in the production of tomatoes would result in a more evenly distributed pricing of tomatoes and result in fewer spikes in prices. The growing demand of tomatoes can be met by increasing acreage of cultivation and adopting new and innovative techniques to increase the yield. Information such as health, growth rate, flowering, flower to fruit conversion rate gathered at regular intervals during the fruiting season facilitate effective decision making in tomato cultivation. Besides providing adequate irrigation and fertilization, practices like pruning promote productivity. Tomato plants send out suckers leaves that shoot out from the main stem. “Suckering” tomato plants, or removing the suckers, makes sense because it promotes air circulation, keeps down disease and focuses the plant’s energy on growing fruit. Therefore, maintaining the quality of food crops such as tomatoes is required to keep its production promising. Capturing and converting sunlight into fruit biomass is a key process in fruit production. Manual assessment of tomato plants in a vast stretch of land is painstaking and often inaccurate. Moreover, manual data collection is labor intensive, costly and time consuming. Remote sensing is the potential candidate to overcome these problems. The remotely captured images/videos processed by computer vision algorithms provide meaningful and actionable information.

UAVs short for Unmanned Aerial Vehicle is an aircraft without a human pilot on board. It can be as small as a drone or as large as a small aircraft. They are a part of an unmanned aircraft system containing UAV, a ground based controller and a communication system[. Here, the UAV used will be a drone with a camera system attached to it, a ground-based controller will be a laptop running a software to send signals to the drone. Drones usually provide a flight time of 10-15 minutes on a single charge. Here, 2-3 passes of the drone would be required which would capture videos and/or images to be processed in a later stage. Accurate early yield prediction is as relevant for farmers as it is for the entire agricultural sector. With UAVs, good yield prediction accuracies can be obtained by applying RGB-derived plant height and canopy cover or multispectral imagery. The development and steep rise of unmanned aerial vehicles (UAVs) or drones in the last decade has marked a new era in remote sensing, providing data of unprecedented spatial, spectral, and temporal resolution data[3] .

Precision agriculture can be broadly defined as an agricultural system in which the management practice is performed at the right place, with the right intensity, and at the right time. Instead of treating fields homogeneously, they are divided into management zones or blocks receiving adjusted inputs. Ideally, this should result in reduced environmental impacts and increased revenue for the farmers, because inputs are reduced while yield is maintained or increased. A prerequisite of precision agriculture is knowledge of the within-field spatial variation of edaphic factors and crop status. UAVs are unique in offering high-quality remote sensing data at the required scale and time. Accordingly, the number of studies using UAVs in precision agriculture and its applications have increased exponentially in the last 8 years. In recent times, the most favored solution is the use of RGB imagery to match human abilities to perceive fruit within a canopy on the basis of color, geometry and texture[4]. Traditional machine based vision approaches to fruit detection are based on pixel features, shape, texture features or integrated approaches. These approaches have been used to classify a variety of fruits and vegetables. Further, integrated approaches[5] that combine both pixel features and shape features provide better performance as compared to single feature based classification. Image processing approaches to fruit detection are based on color, geometry and texture features[6] . Image classification algorithms use these features to discriminate fruit from non fruit regions. In this article, videos are used as input for the detection of tomatoes because videos being a set of frames has its own vector or set of vectors which stores the information about the frame.

After the UAV has captured a video, it can be translated into a set of frames. These frames can be processed individually or cumulatively to find a point where the frame completely changes. So, by performing these steps a number of times, unique frames can be determined which in turn can be fed to the algorithm to give the estimated yield, quality and grade of tomatoes. For determining non-changing frames some suitable algorithms can be used which highlight changes frame by frame. It is a well-known fact that the production of a good quality crop not only benefits consumers but also financially benefits the farmer who works hard to produce such a good crop. In a farmer's perspective, a higher income is proportional to the quality. To predict the total yield for the season, There is a need to cluster the detected tomatoes into regions in the 2-d space defining tomatoes and non-tomatoes. This can be achieved dynamically by implementing the R-CNN(Region Convolutional Neural Network) algorithm. R-CNN is the improved version over the slow CNN(Convolutional Neural Networks) which is specially used for the purpose of multiple object detection.

RetinaNet is a single, unified network composed of a backbone network and two task-specific subnetworks. The backbone is responsible for computing a convolutional feature map over an entire input image and is an off-the-self convolutional network. The first subnet performs convo- lutional object classification on the backbone's output; the second subnet performs convolutional bounding box regression. The two subnetworks feature a simple design specifically for one-stage, dense detection,

## 1.2 Motivation

Within the framework of sustainable development modern agriculture techniques are crucial. The techniques which are being used till date are still in a naive stage. These techniques can still be improved to increase productivity and yield estimation and other factors like the grades of the tomatoes. These techniques will drastically reduce the manual efforts made to estimate the marketable yields for the current growing season. This project can also identify the quality of tomatoes during the growing system which helps the farmers identify the need for special care for the low grade tomatoes which can be a tomato of better quality with intensive care. Having accurate yield predictions of the tomato can be useful to sell tomatoes when the demand is high and tomato prices are rising, moreover it would help the farmers to estimate the financial requirements of the next growing season.

The farmers can easily make appropriate deals if they know the yield of each grade and size of tomato. Currently, images require multiple passes of the field and images need to be captured at a particular elevation and angle. Video would greatly improve this and reduce the time taken. Also it will not require much intervention as it is with images. Video can capture parts that may have been missed in images so a more accurate estimation can be made by the use of videos instead of images as the same tomato is captured in more than one frame and through different angles. Also the state of farmers is very poor in our country as the government fails to provide proper facilities to aid the farmer. This system will allow the farmer to remotely monitor the crops and help them prosper. Farmers can check everything and give special care or attention to special areas of the field which would boost the yield and bring them out of their current state. This is our largest motivation to bring about a change in society through our project**.**

## 1.3 Proposed Solution

Capturing video feed over the field with a UAV such that the alignment of the field of view of the camera mounted on the UAV is aligned with the boundaries of the tomato plantation so that it captures video efficiently and no frames are processed multiple times. This video feed can be preprocessed to remove redundant pixels in the frames which are used for the background extracting just the tomato feed from the video feed.

This preprocessed video will further be fed into the system which will use clustering and classification algorithms to process the yield and quality of the tomatoes which will further be explained where the system is trained to predict the yield and quality of the yield using different machine learning and deep learning technologies.

CHAPTER 2

# LITERATURE SURVEY

[1]Senthilnath, J, Dokania, Akanksha, Kandukuri, Manasa, Ramesh, KN, Anand, Gautham, Omkar, SN, 2016. Detection of tomatoes using spectral-spatial methods in remotely sensed RGB images captured by UAV. Biosyst. Eng. 146, 16–32.

## Methodology

Spectral-spatial classification of tomatoes and non-tomatoes to UAV images was applied. To achieve better classification of spectral-spatial methods majority voting on the pixel wise classification is performed using an adaptive neighbourhood.

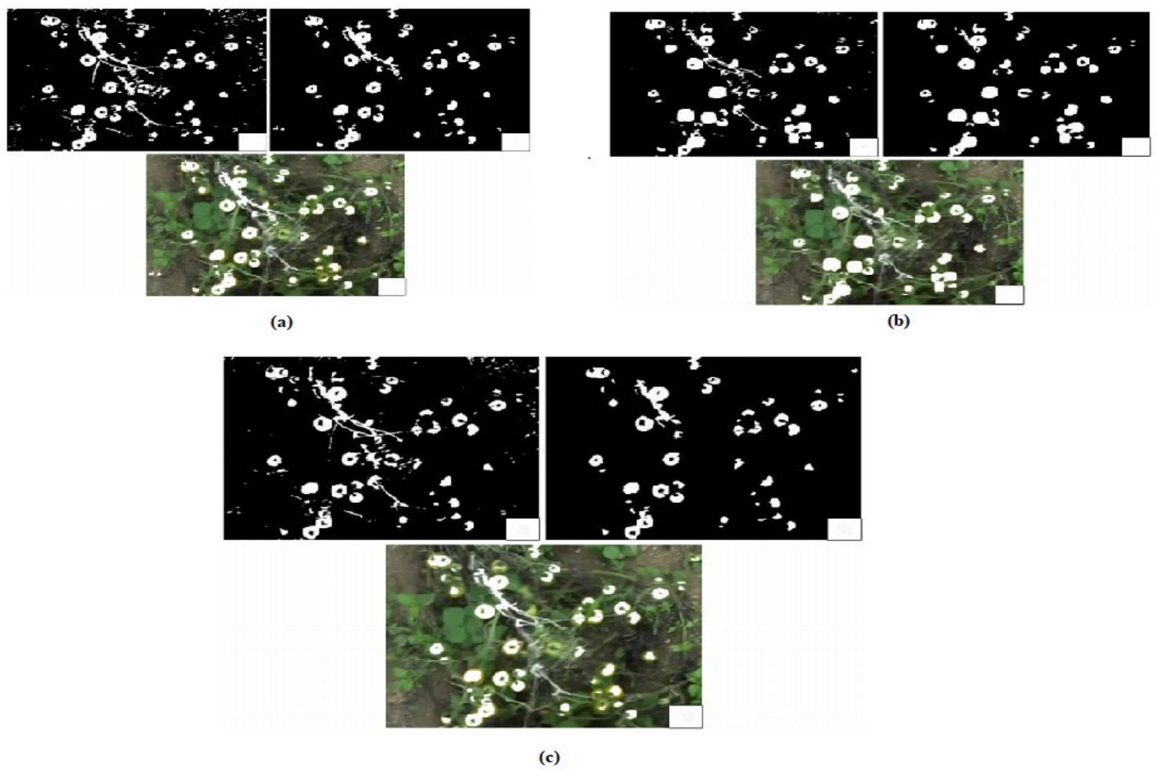


Figure 2.1: (a) Spectral Clustering Using K-Means (b) Spectral Clustering Using EM (c) Spectral Clustering Using SOM

Bayesian information criterion (BIC) was used to determine the optimal number of clusters for the image. Spectral clustering was carried out using K-means, expectation maximisation (EM) and self-organising map (SOM) algorithms to categorise the pixels into two groups i.e. tomatoes and non-tomatoes. The RGB value of any pixel in an image depends on the reflectance properties of the object it represents. Due to resemblance in spectral intensities, some of the non-tomato pixels were grouped into the tomato group and in order to remove them, spatial segmentation was performed on the image. Spatial segmentation was carried out using morphological operations and by setting thresholds for geometrical properties. Since threshold values chosen for carrying out spatial segmentation are shape and size dependent, different threshold values are applied to different methods of clustering. A synthetic image of 12 x 12 pixels with different labels is created to illustrate the effect of each method used for spatial segmentation on the clustered image. This is shown in Figure 2.1. Two representative UAV images captured at different heights from the ground were used to demonstrate the performance of the proposed methodology.

### Limitations

The performance of this system is weak when the tomatoes are overlapped or when the lighting conditions are bad.

[2]Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition.*

## Architecture

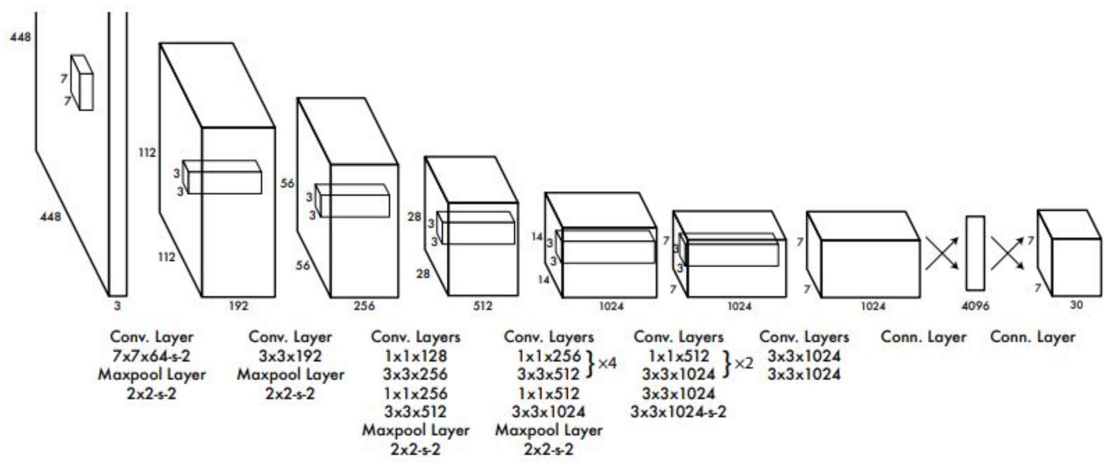

448
7
7
448
3
112
3
3
112
192
56
3
3
56
256
28
3
3
28
512
14
3
3
14
1024
7
3
3
7
1024
7
7
1024
4096
7
7
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 }×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 }×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 2.2: The Architecture of Unified, Real-Time Object Detection

The detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. It pretrains the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then doubles the resolution for detection.

## Methodology

Our network uses features from the entire image to predict each bounding box. Our system divides the input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. Our network has 24 convolutional layers followed by 2 fully connected layers. Simply, 1 × 1 reduction layers followed by 3 × 3 convolutional layers have been used. The final output of our network is the 7 × 7 × 30 tensor of predictions.

## Limitations

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. The model struggles with small objects that appear in groups, such as flocks of birds.

Since the model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. The model also uses relatively coarse features for predicting bounding boxes since architecture has multiple downsampling layers from the input image.

Finally, while training on a loss function that approximates detection performance, the loss function treats errors the same in small bounding boxes versus large bounding boxes.

[3]Haotian Zhang, Gaoang Wang, Zhichao Lei, Jenq-Neng Hwan, 2019. Eye in the Sky: Drone-Based Object Tracking and 3D Localization. In Proceedings of the 27th ACM International Conference on Multimedia, 899-907.

## Architecture

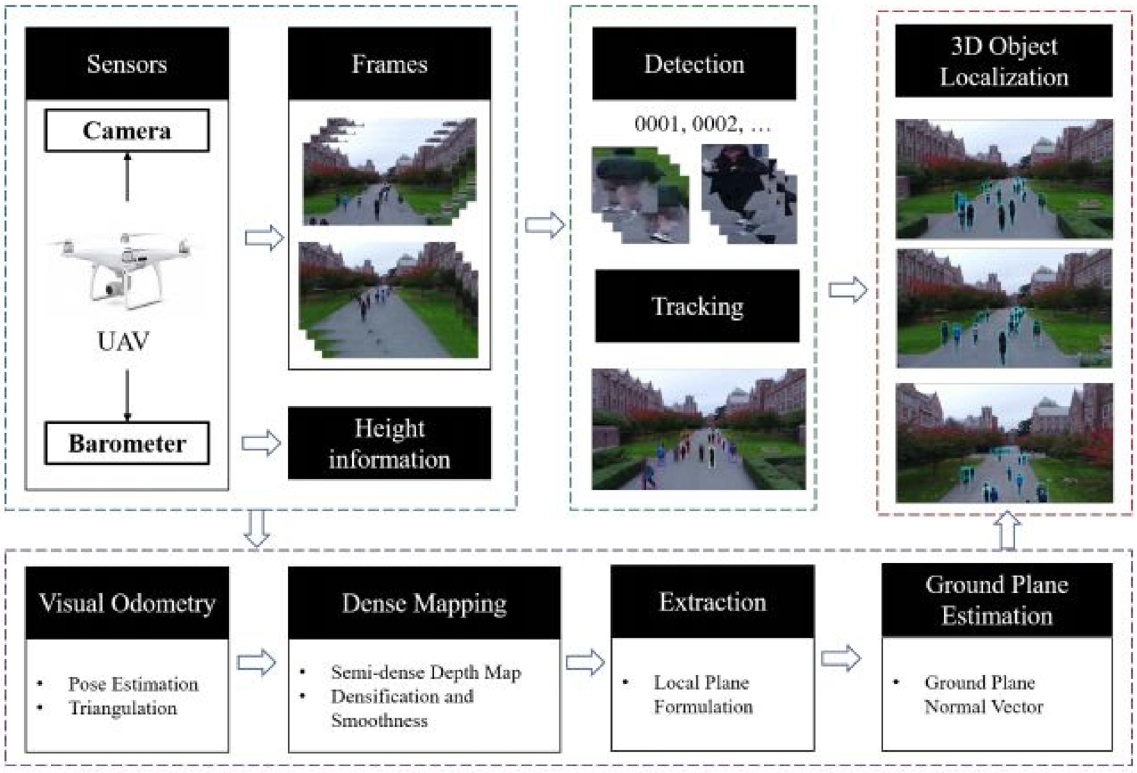


Figure 2.3 The Flow Chart of Proposed System, Which Integrates Object Detection, Multi-Object Tracking and 3D Localization.

## Methodology

The system implementation is divided into four stages:

1. Accurate object detection: The system detects objects of interest based on the modified RetinaNet, which provides a better prior for tracking by detection.
2. Multi-object tracking: A robust TrackletNet Tracker (TNT) for multiple object tracking (MOT), which takes into account both discriminative CNN appearance features and rich temporal information, is incorporated to reduce the impact from unreliable or missing detections and generate smooth and accurate trajectories of moving objects
3. Visual odometry and ground plane estimation: The effective semi-direct visual

odometry (SVO) is used to get the camera pose between views. The ground plane is then estimated from dense mapping based on the multi-view stereo (MVS) method. It minimizes photometric errors across frames and uses a regularization term to smoothen depth map in low-textured regions.

4. 3D object localization: Based on the self-calibrated drone camera parameters, available camera height and estimated ground plane, the detected and tracked objects can be backprojected to 3D world coordinates from 2D image plane. The distance between objects and drones can thus be obtained.

## Limitations

The limitation of this method is that it gives incorrect ground plane estimation quite often. The abrupt camera motion with blurring can cause failure and incorrect detection.

[4]Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar,´ Facebook AI Research (FAIR), 2017. Focal Loss for Dense Object Detection. The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988.

## Architecture

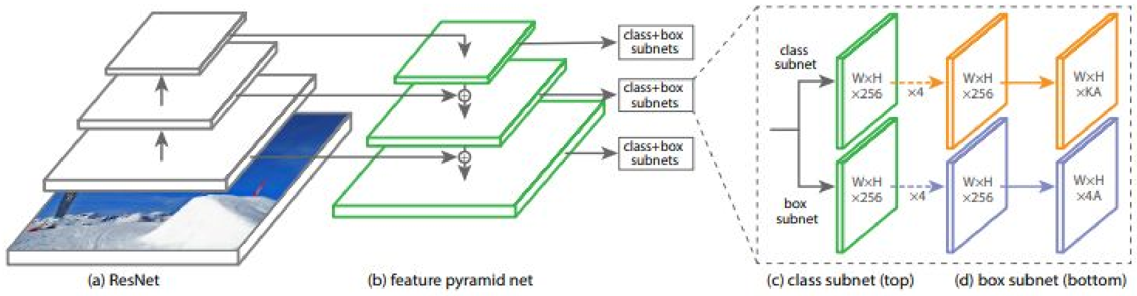


Figure 2.4: The One Stage RetinaNet Architecture

It uses a Feature Pyramid Network (FPN) backbone on top of a feedforward ResNet architecture Figure 2.4 (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN while running at faster speeds.

## Methodology

The first subnet performs convolutional object classification on the backbone's output; the second subnet performs convolutional bounding box regression. The two subnetworks feature a simple design that proposes specifically for one-stage, dense detection. FPN augments a standard convolutional network with a top-down pathway and lateral connections so the network efficiently constructs a rich, multi-scale feature pyramid from a single resolution input image. Each level of the pyramid can be used for detecting objects at a different scale. It constructs a pyramid with levels P3 through P7, where l indicates pyramid level (Pl has resolution 2 l lower than the input). It uses translation-invariant anchor boxes. Box regression targets are computed as the offset between each anchor and its assigned object box, or omitted if there is no assignment.

## Limitations

Although this method seems perfect for extracting objects from blur images, the ratio between positives and negatives is still 3:1 i.e, 75%.

[5]Qianhui Luo, Huifang Ma, Yue Wang, Li Tang, Rong Xiong 3D-SSD: Learning Hierarchical Features from RGB-D Images for Amodal 3D Object Detection.

## Architecture

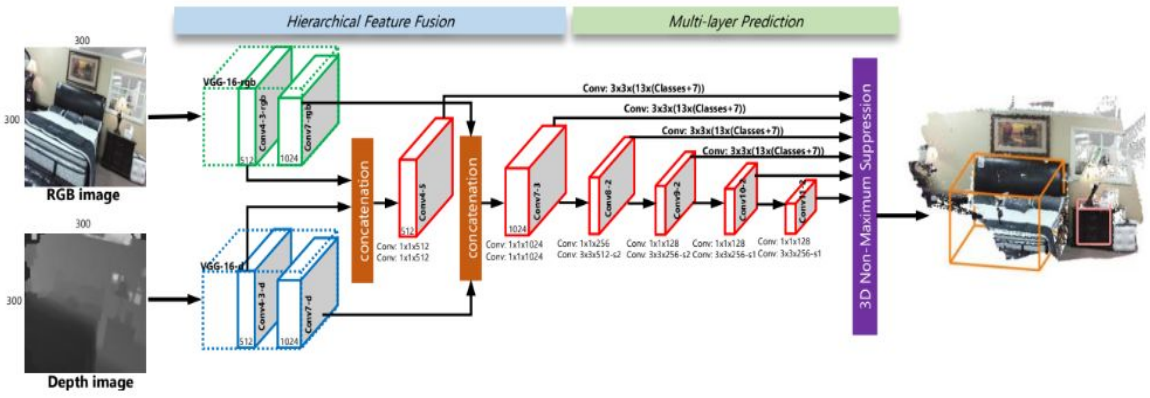


Figure 2.5 - 3D-SSD Architecture

The network takes a pair of RGB-D images as the input and hierarchically fuses appearance and geometric features from the RGB and depth images. Then, a set of 3D anchor boxes with manually-set sizes are attached to every location of the prediction layers, whose 3D positions are estimated from the depth image. Without using 2D bounding box proposals, these multi-scale prediction layers are used to estimate category scores for each anchor box and regress its position, size and orientation offsets.

## Methodology

The 3D-SSD network takes a pair of RGB-D images as input and hierarchically fuses appearance and geometric features from the RGB and depth images. The input RGB and depth images (all resized to 300 × 300) separately go through two VGG-16 networks to learn appearance and geometric features, as shown in Figure 2.5. Then, the selected two pairs of layers with identical receptive fields in the network to fuse their features. Specifically, in the hierarchical feature fusion part, firstly concatenate the convolution layers of conv4-3-rgb and conv4- 3-d, which are later followed by two 1 × 1 convolution layers to shuffle and choose appearance and geometric features that come from the same region on the original input RGB and depth images. Then is the concatenation of convolution layers conv7-rgb and conv7-d, also followed by two 1 × 1 convolution layers. The fused feature maps on these two stages are parts of the multiple feature layers used for prediction in the multi-layer prediction part. The 2D features from images cannot be directly used for 3D object detection without other calibration information due to the scaling problem. Without using 2D bounding box proposals, apply a small(3×3) convolutional filter(ConvFilter) on every location of the prediction layers to detect objects. When the small ConvFilter acts on the lower level feature maps, its receptive field is limited and can only capture local features of some of the big objects. As is shown in Figure 2.5, six convolution layers conv4-5, conv7-3, conv8-2, conv9- 2, conv10-2 and conv11-2 in the multi-layer prediction part are used to produce a number of 3D bounding boxes, which are later fed to a 3D non-maximum suppression to estimate the final object detection results.

### Limitations

The system takes a significant amount of time when working with big datasets.

CHAPTER 3

# PROBLEM STATEMENT

An application to extract frames at predefined rate using and then creating an algorithm to process these frames using the object detection with RatinaNet to detect tomatoes and predict the total yield.

## 3.1 Existing System

- In the current existing system spectral-spatial classification of tomatoes and non-tomatoes to UAV images is applied. To achieve better classification of spectral-spatial methods majority voting on the pixel wise classification is performed using an adaptive neighbourhood.
- Bayesian information criterion (BIC) is used to determine the optimal number of clusters for the image.
- Spectral clustering is carried out using K-means, expectation maximisation (EM) and self-organising map (SOM) algorithms to categorise the pixels into two groups i.e. tomatoes and non-tomatoes.
- The RGB value of any pixel in an image depends on the reflectance properties of the object it represents.
- Due to resemblance in spectral intensities, some of the non-tomato pixels are grouped into the tomato group and in order to remove them, spatial segmentation was performed on the image.
- Spatial segmentation is carried out using morphological operations and by setting thresholds for geometrical properties. Since threshold values chosen for carrying out spatial segmentation are shape and size dependent, different threshold values are applied to different methods of clustering.

## 3.2 Problem Description

Although there exist systems to produce good yield results with images, there can be anomalies and inaccurate predictions since only images are considered to predict the yield. On the other hand if a video is used, the area is available in two or more frames and at different angles. In image areas can be left out due to improper boundaries and can lead to inaccurate predictions about the yield. Producing quality tomatoes is also a very important aspect of making the tomatoes marketable and to do this, the system should be able to grade tomatoes based on their optical quality aspects and which can later be used to predict the estimated earning for the farmer.

## 3.3 Objectives

The system can be used to predict the yield of the tomato crop using an unmanned aerial vehicle. The system can do interpolation of the video captured by the UAV using context aware synthesis. The system can identify the different stages of the tomato crop growing as well identify the defective ones.

The system can suggest the farmer to make changes based on the color of the tomatoes in different stages of the tomato growing, so that the farmer can give special attention to specific parts of the field. The system can help the farmer detect suckering in tomato plants without actually visiting each and every part of the field. Removal of suckering would help tomatoes grow fuller and receive more sunshine. The system can provide specific yields for specific grades of tomatoes which would help the farmer make an estimate of the earnings made from the growing season.

# CHAPTER 4

# SYSTEM REQUIREMENTS

A software requirements specification (SRS) is a detailed description of a software system to be developed with its functional and non-functional requirements. The SRS is developed based on the agreement between customer and contractors. It may include the use cases of how the user is going to interact with the software system. The software requirement specification document is consistent with all necessary requirements required for project development. To develop the software system one should have a clear understanding of Software system. To achieve this need continuous communication with customers to gather all requirements is needed.

A good SRS defines how the Software System will interact with all internal modules, hardware, communication with other programs and human user interactions with a wide range of real life scenarios. Using the Software requirements specification (SRS) document on QA lead, managers create a test plan. It is very important that testers must be cleared with every detail specified in this document in order to avoid faults in test cases and its expected results.

It is highly recommended to review or test SRS documents before starting writing test cases and making any plan for testing. Let’s see how to test SRS and the important point to keep in mind while testing it.

1. **Correctness of SRS should be checked**. Since the whole testing phase is dependent on SRS, it is very important to check its correctness. There are some standards with which can be used to compare and verify.
2. **Ambiguity should be avoided**. Sometimes in SRS, some words have more than one meaning and this might confuse tester’s making it difficult to get the exact reference. It is advisable to check for such ambiguous words and make the meaning clear for better understanding.
3. **Requirements should be complete**. When a tester writes test cases, what exactly is required from the application, is the first thing which needs to be clear. For e.g.

if an application needs to send the specific data of some specific size then it should be clearly mentioned in SRS how much data and what is the size limit to send.

4. **Consistent requirements.** The SRS should be consistent within itself and consistent to its reference documents. If you call an input “Start and Stop” in one place, don’t call it “Start/Stop” in another. This sets the standard and should be followed throughout the testing phase.
5. **Verification of expected result**: SRS should not have statements like “Work as expected”, it should be clearly stated that what is expected since different testers would have different thinking aspects and may draw different results from this statement.
6. **Testing environment**: some applications need specific conditions to test and also a particular environment for accurate results. SRS should have clear documentation on what type of environment is needed to set up.
7. **Pre-conditions defined clearly**: one of the most important parts of test cases is pre-conditions. If they are not met properly then the actual result will always be different than the expected result. Verify that in SRS, all the pre-conditions are mentioned clearly.
8. **Requirements ID**: these are the base of the test case template. Based on requirement Ids, test case ids are written. Also, requirements make it easy to categorize modules so just by looking at them, the tester will know which module to refer to. SRS must have them such as an id that defines a particular module.
9. **Security and Performance criteria**: security is priority when a software is tested especially when it is built in such a way that it contains some crucial information when leaked can cause harm to business. Tester should check that all the security related requirements are properly defined and are clear to him. Also, regarding performance of a software, it plays a very important role in business so all the requirements related to performance must be clear to the tester and he must also know when and how much stress or load testing should be done to test the performance.
10. **Assumption should be avoided**: sometimes when requirement is not cleared to tester, he tends to make some assumptions related to it, which is not a right way to do testing as assumptions could go wrong and hence, test results may vary. It is

better to avoid assumptions and ask clients about all the “missing requirements” to have a better understanding of expected results.

11. **Deletion of irrelevant requirements**: there are more than one team who work on SRS so it might be possible that some irrelevant requirements are included in SRS. Based on the understanding of the software, testers can find out which are these requirements and remove them to avoid confusions and reduce workload.
12. **Freeze requirements**: when an ambiguous or incomplete requirement is sent to client to analyse and tester gets a reply, that requirement result will be updated in the next SRS version and client will freeze that requirement. Freezing here means that the result will not change again until and unless some major addition or modification is introduced in the software.

Most of the defects which are found during testing are because of either incomplete requirements or ambiguity in SRS. To avoid such defects it is very important to test software requirements specification before writing the test cases. Keep the latest version of SRS with you for reference and keep yourself updated with the latest change made to the SRS. Best practice is to go through the document very carefully and note down all the confusions, assumptions and incomplete requirements and then have a meeting with the client to get them clear before the development phase starts as it becomes costly to fix the bugs after the software is developed. After all the requirements are cleared to a tester, it becomes easy for him to write effective test cases and accurate expected results.

## 4.1 Functional Requirements

### 4.1.1 Unmanned Aerial Vehicle

An unmanned aerial vehicle (UAV) or uncrewed aerial vehicle, commonly known as a drone, is an aircraft without a human pilot on board and a type of unmanned vehicle. UAVs are a component of an unmanned aircraft system (UAS); which include a UAV,a ground-based controller, and a system of communications between the two. The flight of UAVs may operate with various degrees of autonomy: either under remote control by a human operator, autonomously by onboard computers  or piloted by an autonomous robot.

### 4.1.2 OpenCV Library

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source BSD license. Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel. It is an advanced vision-based commercial application by making portable, performance-optimized code available for free – with a license that did not require code to be open or free itself.

OpenCV's application areas include:

- 2D and 3D feature toolkits
- Egomotion estimation
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding
- Object identification
- Segmentation and recognition

### 4.1.3 LabelImg

LabelImg is a free, open source tool for graphically labeling images. It’s written in Python and uses QT for its graphical interface. It’s an easy, free way to label a few hundred images to try out. LabelImg supports labelling in VOC XML or YOLO text file format. At Roboflow, it is strongly recommended that you use the default VOC XML format for creating labels. Thanks to ImageNet, VOC XML is a more universal standard as it relates to object detection whereas various YOLO implementations have slightly different text file formats. Moreover, you can always easily convert from VOC XML to any other format using Roboflow, like VOC XML to COCO JSON.

### 4.1.4 RetinaNet

RetinaNet is one of the best one-stage object detection models that has proven to work well with dense and small scale objects. For this reason, it has become a popular object detection model to be used with aerial and satellite imagery. It has been formed by making two improvements over existing single stage object detection models - Feature Pyramid Networks (FPN) and Focal Loss.

Traditionally, in computer vision, featured image pyramids have been used to detect objects with varying scales in an image. Featured image pyramids are feature pyramids built upon image pyramids. Focal Loss (FL) is an enhancement over Cross-Entropy Loss (CE) and is introduced to handle the class imbalance problem with single-stage object detection models.

## 4.2 Non-Functional Requirements

### 4.2.1 Data Integrity

Given the importance of the contents in the documents, it is our foremost priority to maintain the integrity of the data. If the integrity of the data of these documents are compromised, then the value provided by the application will not be significant irrespective of all the features being provided.

### 4.2.2 Usability

Since this is an application, the main metric of success to be considered is its usability. To achieve this aspect, looking beyond having an intuitive interface is suggested. The interface will be intuitive when the user is able to understand the flow of the control through the application when they are using. This intuitive flow forms a crucial role to design an intuitive interface. Hence making the application highly user-friendly and can be used without having a steep learning curve.

### 4.2.3 Performance

Performance forms another important feature for an application. Once a video is recorded and uploaded, a lot of processing needs to happen such as frame extraction, creation of bounding boxes, detection of tomatoes and prediction. These are computationally expensive tasks. Instead of performing all these operations on the device, which further affects the battery performance. These processes can be performed in the cloud infrastructure which is used to host the database. This will reduce the load on the local device and by making use of the high-power computation available as part of the infrastructure helps in faster processing. This results in a very high performance and thus meeting an important feature of the application.

### 4.2.4 Availability

Another significant criterion to be considered for an application is the availability. Both data and service should be available to the user at all times. If the user is unable to access the documents that have been uploaded or unable to access the documents when the search has been made then the impact of the application is affected. At the same time, when the user provides an input to make use of the service being provided, it is to be made sure that the service is readily available at all times. Users must be able to make use of the service on demand, which in turn affects the likeability of the system.

## 4.3 Hardware Requirements

Table 4.1: Hardware Requirements

| Name of the Component | Specification |
|---|---|
| Processor | Intel Core 2 Duo or above / AMD 6 or above |
| RAM | 2 GB or above |
| Hard Disk | 100 GB or above |

| | |
|---|---|
| Graphics | Hardware Accelerated Graphics card supporting OpenGL 3.3 with 1GB GPU |
| Unmanned Aerial Vehicle | Handmade with Camera, Wireless Control,Flight Stabilizer, Altimeter, Gimbal Stabilizer, Raspberry Pi |

## 4.4 Software Requirements

Table 4.2: Software Requirements

| **Name of the Component** | **Specification** |
|---|---|
| Operating System | Windows 7+ |
| Scripting Language | Python |
| LabelImg | Annotation for Data Set |
| Browser | Google Chrome/Mozilla etc. |

**CHAPTER 5**

# DESIGN

Software design is the process of defining software methods, functions, objects, and the overall structure and interaction of your code so that the resulting functionality will satisfy your user's requirements. It is the method of creating a representation of the complete software components and behavior before implementing the actual software. You need to follow design principles, which will lead you to develop more robust, maintainable & flexible software.

Software design is basically a mechanism of preparing a plan, a layout for structuring the code of your software application. It is a multi-stage concept. The software itself is a multi-layer, multi-dimensional spectrum and its design has multiple intermediate steps therefore; different types of software level design.

- **Architectural Design:** Architectural design is the first step in software level design. This involves the construction of a software application structure in order to have the correct interconnection of each element of code according to the required function.
- **High-level Design:** This is the second step in software level design. In this step, the designer splits the theoretical concepts into multiple entities and ensures their functions are interrelated to get an optimum result. This module identifies a modular structure of different entities and creates a design to inter-connect all modules for a specific output.
- **Detailed Design:** This is the third and the last step of software level design. It involves the compilation of all modular outputs and arrangement or rearrangement of functional modules for final outcome. This level of software design determines more of a logical structure out of constructed modules. The software outcome accomplishment part is the motive of the detailed design level of software design.

## 5.1 Architecture

A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.

A system architecture can consist of system components and the sub-systems developed, that will work together to implement the overall system. There have been efforts to formalize languages to describe system architecture, collectively these are called architecture description languages (ADLs). Various organizations can define systems architecture in different ways, including:

The fundamental organization of a system, embodied in its components, their relationships to each other and to the environment, and the principles governing its design and evolution.

A representation of a system, including a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.

An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.

An architecture consists of the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behavior.

A description of the design and contents of a computer system. If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software.

A formal description of a system, or a detailed plan of the system at component level to guide its implementation. The composite of the design architectures for products and their life-cycle processes.

The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time.

One can think of system architecture as a set of representations of an existing (or future) system. These representations initially describe a general, high-level functional organization, and are progressively refined to more detailed and concrete descriptions

### 5.1.1 Proposed System Architecture

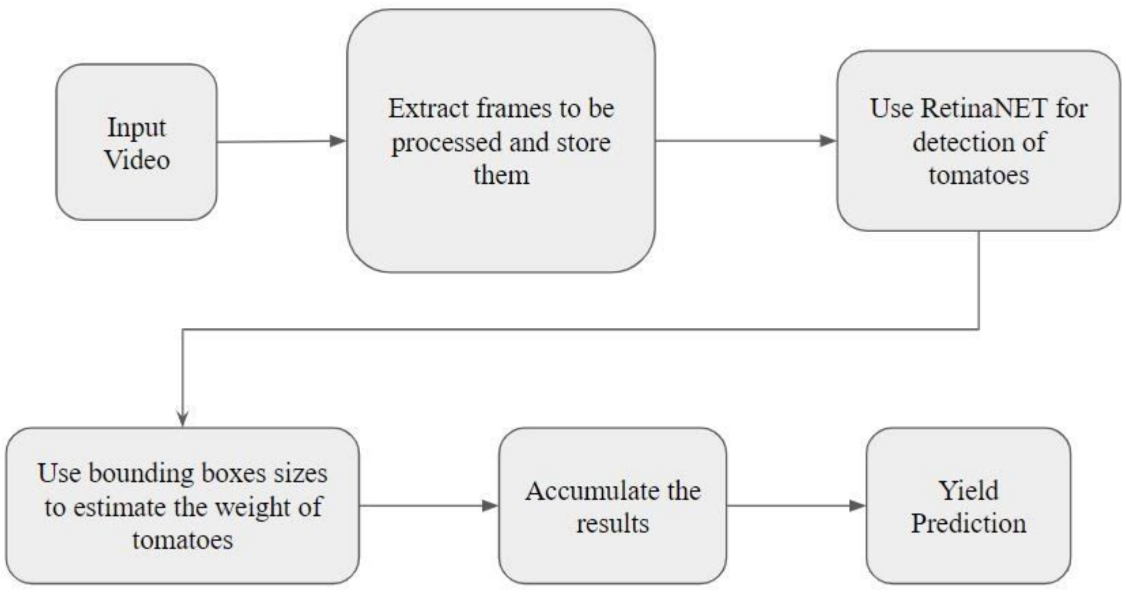

Input Video
Extract frames to be processed and store them
Use RetinaNET for detection of tomatoes
Use bounding boxes sizes to estimate the weight of tomatoes
Accumulate the results
Yield Prediction

Figure 5.1: System Architecture

To come up with a solution for the problems faced in the previous systems,an advanced way to solve this problem was by capturing video feed over the field with a UAV such that the alignment of the field of view of the camera mounted on the UAV is aligned with the boundaries of the tomato plantation so that it captures video efficiently and no frames are processed multiple times. The video feed can be preprocessed to remove redundant pixels in the frames which are used for the background extracting just the tomato feed from the video feed. Preprocessed video will further be fed into the system which will use clustering and classification algorithms to process the yield and quality of the tomatoes, where the system is trained to predict the yield and quality of the yield using different machine learning and deep learning techniques.

Firstly, the field of the farmer is analyzed to get the exact dimensions of the field. Exact dimensions of the field are required for the operation of the UAV to capture the video feed. The UAV is flown once over the field and the video feed of the field is captured in the RGB color space using a multispectral camera. After the video has been uploaded to the system,the video is pre-processed to a suitable format like mp4. Also the video can be cropped or length can be reduced to remove unnecessary frames from the video.The processed video frame is now ready.

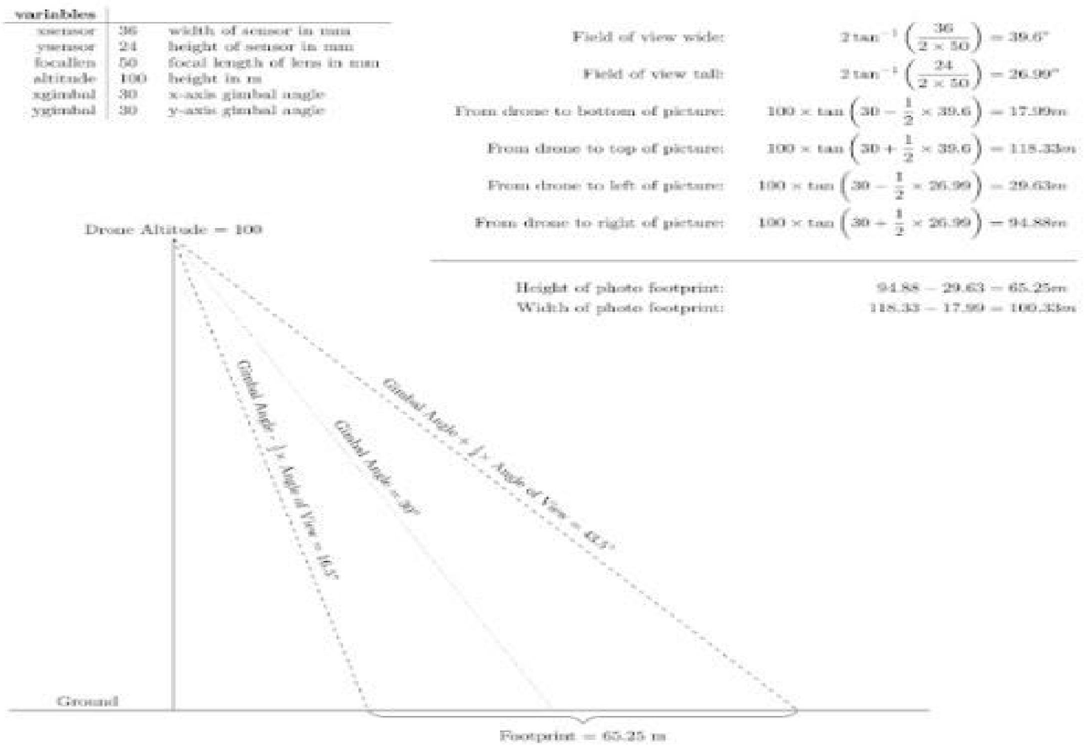
Then, a RetinaNET system is proposed for multi-stage object detection.. It consists of FNP and The first subnet performs convolutional object classification on the backbone's output; the second subnet performs convolutional bounding box regression. The two subnetworks feature a simple design that is proposed specifically for one-stage, dense detection. FPN augments a standard convolutional network with a top-down pathway and lateral connections so the network efficiently constructs a rich, multi-scale feature pyramid from a single resolution input image. Each level of the pyramid can be used for detecting objects at a different scale. It constructs a pyramid with levels P3 through P7, where l indicates pyramid level (Pl has resolution 2 l lower than the input). It uses translation-invariant anchor boxes. Box regression targets are computed as the offset between each anchor and its assigned object box, or omitted if there is no assignment.. At inference, the quality of the hypotheses is sequentially improved, by applications of the same cascade procedure, and higher quality detectors are only required to operate on higher quality hypotheses. The results are analyzed and stored for further use to improve the accuracy of our model. The analysis is then conveyed to the farmer and suggestions are made so that he can take necessary actions.

## 5.2 Module Description

A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program. Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application. Modules are typically incorporated into the program (software) through interfaces. The various modules of our system are as follows:

**Frame Extraction**

The input video is captured using an Unmanned Aerial Vehicle(UAV). Which is flown over a tomato field at a nearly constant velocity and the altitude in patches such that the whole field is covered and velocity is noted.



| variables | | |
|---|---|---|
| xsensor | 36 | width of sensor in mm |
| ysensor | 24 | height of sensor in mm |
| focallen | 50 | focal length of lens in mm |
| altitude | 100 | height in m |
| xgimbal | 30 | x-axis gimbal angle |
| ygimbal | 30 | y-axis gimbal angle |

Field of view wide: $2\tan^{-1}\left(\frac{36}{2\times 50}\right) = 39.6°$

Field of view tall: $2\tan^{-1}\left(\frac{24}{2\times 50}\right) = 26.99°$

From drone to bottom of picture: $100 \times \tan\left(30 - \frac{1}{2} \times 39.6\right) = 17.99m$

From drone to top of picture: $100 \times \tan\left(30 + \frac{1}{2} \times 39.6\right) = 118.33m$

From drone to left of picture: $100 \times \tan\left(30 - \frac{1}{2} \times 26.99\right) = 29.63m$

From drone to right of picture: $100 \times \tan\left(30 + \frac{1}{2} \times 26.99\right) = 94.88m$

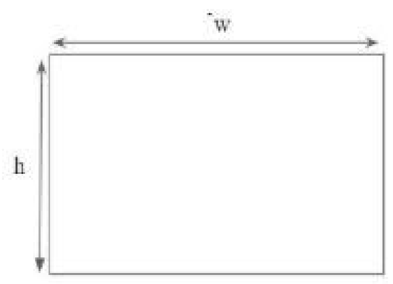
Height of photo footprint: $94.88 - 29.63 = 65.25m$

Width of photo footprint: $118.33 - 17.99 = 100.33m$

Drone Altitude = 100
Gimbal Angle - ½ × Angle of View = 16.5°
Gimbal Angle = 30°
Gimbal Angle + ½ × Angle of View = 43.5°
Ground
Footprint = 65.25 m

Figure 5.2: Photographic Footprint of Camera on a Drone

The input is captured using a spectral camera and then stored for processing. The frame extraction is crucial because it extracts frames that would be processed by the system. The frames are extracted such that when the frames are stitched together such that it can nearly regenerate the whole tomato field.



Total no of frames = video length (s) x fps (frames/sec)

Let the fps be x, so the granularity = 1/x sec/frame

Let the frequency of velocity capture be $f_v$.

Distance covered = velocity x ($1/f_v$).

So the number of frames in that distance = (1/x) x ($1/f_v$).

Figure 5.3: Frame Extraction Procedure

**Tomato Detection**

RetinaNet has been used, a single stage detector for detection of tomatoes. RetinaNET is trained on the videos captured from the tomato field. The frames extracted from the previous step are used and the tomatoes are detected with their bounding boxes and probabilities. It uses translation-invariant anchor boxes. Box regression targets are computed as the offset between each anchor and its assigned object box, or omitted if there is no assignment.

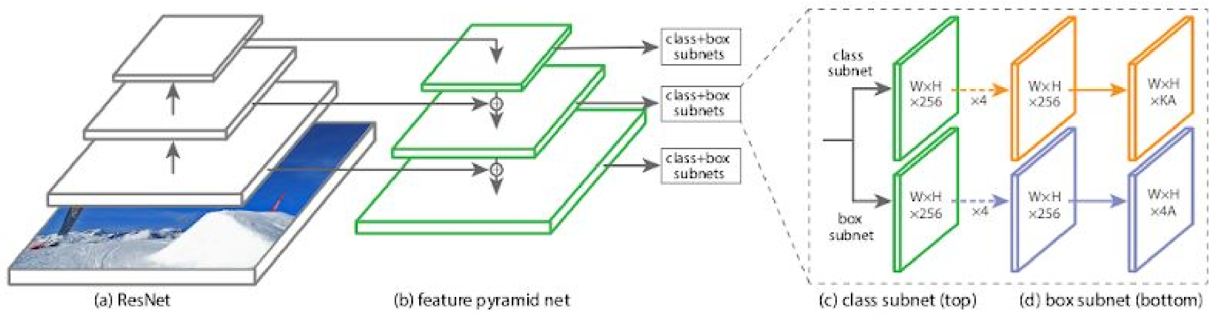


Figure 5.3: RatinaNET Architecture

## 5.3 Algorithm Design

An algorithm is a series of instructions, often referred to as a “process,” which is to be followed when solving a particular problem. While technically not restricted by definition, the word is almost invariably associated with computers, since computer-processed algorithms can tackle much larger problems than a human, much more quickly.

An algorithm is the best way to represent the solution of a particular problem in a very simple and efficient way. If given a particular algorithm for a specific problem, then it can be implemented in any programming language, meaning that the algorithm is independent from any programming languages.

### 5.3.1 Algorithm : Frame Separation:

1. Load the video using cv2.VideoCapture
2. Calculate the frames using vidcap.get(cv2.CAP_PROP_FRAME_COUNT)
3. determine snap frequency as :
   $f_s = w_d / v_d$
4. Calculate the frame(s) to be taken as :
   $FPS * f_s$

### 5.3.2 Image and Multiple Bounding Boxes Augmentation for Deep Learning

1. Convert all XML files into one CSV file
2. Resize all images together with the corresponding object bounding boxes
3. Augment images to upsample our dataset. Corresponding object bounding boxes should be augmented accordingly
4. Document augmented images' new sizes and bounding boxes' coordinates pascal voc format.

# CHAPTER 6

# IMPLEMENTATION

## 6.1 Image Augmentation

```
# Installing ImgAug

!pip install imgaug

# Load the required Libs

# imgaug library for augmentation

import imgaug as ia

ia.seed(1)

%matplotlib inline

from imgaug.augmentables.bbs import BoundingBox, BoundingBoxesOnImage

from imgaug import augmenters as iaa

# Using imageio for image input/output

import imageio

import pandas as pd

import numpy as np

import re

import os

import glob
import math

#Library for reading xml files and converting it into CSV

import xml.etree.ElementTree as ET

import shutil # Used for copying

# Add to input_images dir

%%time

!mkdir input_images

for index, file in enumerate(glob.glob('/content/drive/My Drive/combined/*')):

  shutil.copy(file,'/content/input_images/'+os.path.basename(file))

images = []

for index, file in enumerate(glob.glob('input_images/*.jpg')):

  images.append(imageio.imread(file))

  # Print image sizes

  print('Image {} has size of {}'.format(file[:], images[index].shape))

print('We have {} images'.format(len(images)))

# Take a look at smaller images

ia.imshow(images[0])

demo_file = ''

for index, file in enumerate(glob.glob('input_images/*.xml')):

  print(file)

  #Check if size is zero
  tree = ET.parse(file)

  root = tree.getroot()

  changed = False

  if int(root.find('size')[0].text) == 0:

    root.find('size')[0].text = str(imageio.imread(file[:-3]+'jpg').shape[0])

    changed = True

  if int(root.find('size')[1].text) == 0:

    root.find('size')[1].text = str(imageio.imread(file[:-3]+'jpg').shape[1])

    changed = True

  if changed == True :

    tree.write(file)

  demo_file = file

# have a look how one of the XML annotation file looks like

shutil.copy(demo_file,demo_file[:-3]+'txt')

annotation_text = open(demo_file[:-3]+'txt','r')

print(annotation_text.read())

annotation_text.close()

def xml_to_csv(path):

  xml_list = []

  print('path',path + '/*.xml')

  for idx, xml_file in enumerate(glob.glob(path + '/*.xml')):
   tree = ET.parse(xml_file)

   root = tree.getroot()

   for member in root.findall('object'):

     value = (

             root.find('filename').text,

             int(root.find('size')[0].text),

             int(root.find('size')[1].text),

             member[0].text.upper(),

             member[1].text,

             member[2].text,

             member[3].text,

             int(member[4][0].text),

             int(member[4][1].text),

             int(member[4][2].text),

             int(member[4][3].text)

             )

     xml_list.append(value)

                                                        column_name                          =
['filename','width','height','class','pose','truncated','difficult','xmin','ymin','xmax','ymax']

  xml_df = pd.DataFrame(xml_list, columns=column_name)

  return xml_df
```

```
# apply xml_to_csv() function to convert all XML files in images/folder into labels.csv

labels_df = xml_to_csv('input_images')

labels_df.to_csv(('img_labels.csv'), index=None)

print('Successfully converted xml to csv')

labels_df

# Group each image bbs by filename

grouped = labels_df.groupby('filename')

print(grouped.groups)

# Testing a group

group_df = grouped.get_group('frame296.jpg')

group_df = group_df.reset_index()

group_df = group_df.drop(['index'], axis=1)

print(group_df.head())

# Visualization of bounding boxes

bb_array = group_df.drop(['filename','width','height','class','pose','truncated','difficult'],
axis=1).values

print(bb_array)

# pass the array of bounding boxes coordinates to the imgaug library

image = imageio.imread('input_images/frame296.jpg')

bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)

#display the image and draw bounding boxes
```

```
ia.imshow(bbs.draw_on_image(image, size=2))

# To resize the images two augmenters shall be created


# for height

height_resize = iaa.Sequential([

                                iaa.Resize({"height": 600, "width": 'keep-aspect-ratio'})

])

#for width

width_resize = iaa.Sequential([

                               iaa.Resize({"height": 'keep-aspect-ratio',"width": 600})

])

# Convert BoundingBoxesOnImage object into DataFrame

def bbs_obj_to_df(bbs_object):

  # Convert BoundingBoxesOnImage object into array

  bbs_array = bbs_object.to_xyxy_array()

  # Convert array into a DF ['xmin', 'ymin', 'xmax', 'ymax'] columns

  df_bbs = pd.DataFrame(bbs_array, columns=['xmin', 'ymin', 'xmax','ymax'])

  return df_bbs

def resize_imgaug(df, images_path, aug_images_path, image_prefix):

  # create df which er're going to populate with augmented_image info

  aug_bbs_xy = pd.DataFrame(columns=
                    ['filename','width','height','class','pose','truncated','difficult','xmin','ymin',
'xmax','ymax'])
  grouped = df.groupby('filename')


  for filename in df['filename'].unique():
    #Get separate data frame grouped y file name
    group_df = grouped.get_group(filename)
    group_df = group_df.reset_index()
    group_df = group_df.drop(['index'], axis=1)


    if(filename == 'frame100.jpg'):
      print('frame100 received',group_df['height'].unique()[0],group_df['width'].unique()[0])


    # Use of height_resize and width_resize makes the difference
              if group_df['height'].unique()[0] >= group_df['width'].unique()[0] and
group_df['height'].unique()[0] > 600:
      #image read
      image = imageio.imread(images_path+filename)
      #get bb coordinates and write it into array
                                                                  bb_array =
group_df.drop(['filename','width','height','class','pose','truncated','difficult'],axis=1).values
      #pass to imgaug
      bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)
      #Apply augmentation on image and on the bounding boxes
      image_aug, bbs_aug = height_resize(image=image, bounding_boxes=bbs)
      #Wrie augmented image to a file
      imageio.imwrite(aug_images_path+image_prefix+filename, image_aug)
      #create a data frame with augmented values of image width and height
      info_df = group_df.drop(['xmin', 'ymin','xmax','ymax'], axis=1)
      for index,_ in info_df.iterrows():
        info_df.at[index, 'width'] = image_aug.shape[1]
        info_df.at[index, 'height'] = image_aug.shape[0]
      # rename filenames by adding the predefined prefix
      info_df['filename'] = info_df['filename'].apply(lambda x: image_prefix+x)
      # create a df with augmented values of image width and height
      bbs_df = bbs_obj_to_df(bbs_aug)
      #concat all new augmented info into thr new data frame
      aug_df = pd.concat([info_df, bbs_df],axis=1)
      #append rows to aug_bbs_xy df
      aug_bbs_xy = pd.concat([aug_bbs_xy, aug_df])


          elif group_df['width'].unique()[0] >= group_df['height'].unique()[0] and group_df['width'].unique()[0] > 600:
```

```
#image read
image = imageio.imread(images_path+filename)
#get bb coordinates and write it into array
bb_array = group_df.drop(['filename','width','height','class','pose','truncated','difficult'],axis=1).values
#pass to imgaug
bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)
#Apply augmentation on image and on the bounding boxes
image_aug, bbs_aug = width_resize(image=image, bounding_boxes=bbs)
#Wrie augmented image to a file
imageio.imwrite(aug_images_path+image_prefix+filename, image_aug)
#create a data frame with augmented values of image width and height
info_df = group_df.drop(['xmin', 'ymin','xmax','ymax'], axis=1)
for index,_ in info_df.iterrows():
  info_df.at[index, 'width'] = image_aug.shape[1]
  info_df.at[index, 'height'] = image_aug.shape[0]
# rename filenames by adding the predefined prefix
info_df['filename'] = info_df['filename'].apply(lambda x: image_prefix+x)
# create a df with augmented values of image width and height
bbs_df = bbs_obj_to_df(bbs_aug)
#concat all new augmented info into thr new data frame
```

```
      aug_df = pd.concat([info_df, bbs_df],axis=1)
      #append rows to aug_bbs_xy df
      aug_bbs_xy = pd.concat([aug_bbs_xy, aug_df])
    #append image info without any changes
    else:
      aug_bbs_xy = pd.concat([aug_bbs_xy, group_df])
  #return df with updated images and bounding boxes annotations
  aug_bbs_xy = aug_bbs_xy.reset_index()
  aug_bbs_xy = aug_bbs_xy.drop(['index'], axis=1)
  return aug_bbs_xy
#make aug_images dir
!mkdir resized_images
# apply resizing augmentation to our images and write the updated mages and bounding boxes annotations to the DF
resized_images_df = resize_imgaug(labels_df, 'input_images/', 'resized_images/', '')
# Create csv
resized_images_df.to_csv('resized_labels.csv', index=False)
# visualize the resized image with bb to make sure resizing worked
grouped = resized_images_df.groupby('filename')
group_df = grouped.get_group('frame296.jpg')
group_df = group_df.reset_index()
group_df = group_df.drop(['index'], axis=1)

bb_array = group_df.drop(['filename', 'width', 'height', 'class', 'pose', 'truncated', 'difficult'],
axis=1).values

image = imageio.imread('resized_images/frame296.jpg')

bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)

ia.imshow(bbs.draw_on_image(image, size=2))

# This setup of augmentation parameters will use four augmenters and apply them

aug = [

    iaa.Affine(rotate=(-25,25)),

    iaa.Fliplr(1),

    iaa.Flipud(1),

    iaa.AdditiveGaussianNoise(scale=(0.03*255, 0.05*255))

]

def image_aug(df, images_path, aug_images_path, image_prefix, augmentor):

  #Create df which is going to be populated with augmented image info

  aug_bbs_xy = pd.DataFrame(columns=

                              ['filename','width','height','class','pose','truncated','difficult','xmin','ymin',
'xmax','ymax']

                             )

  grouped = df.groupby('filename')



  for filename in df['filename'].unique():
    # get separate df grouped by file name
    group_df = grouped.get_group(filename)
    group_df = group_df.reset_index()
    group_df = group_df.drop(['index'], axis=1)
    #image read
    image = imageio.imread(images_path+filename)
    #get bb coordinates and write it into array
                                                          bb_array                    =
group_df.drop(['filename','width','height','class','pose','truncated','difficult'],axis=1).values
    #pass to imgaug
    bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)
    #Apply augmentor
    image_aug, bbs_aug = augmentor(image=image, bounding_boxes=bbs)
    # if filename == 'frame10.jpg':
    #   print('pre clipping\n',bbs_aug)
    #Remove bbs which have fallen out of image pane
    bbs_aug = bbs_aug.remove_out_of_image()
    #Clip bbs partially outside of image pane
    bbs_aug = bbs_aug.clip_out_of_image()
    # if filename == 'frame10.jpg':
    #   print('post clipping\n',bbs_aug)
```

```
#Skip actions on image if there are no bounding boxes left
if re.findall('Image...',str(bbs_aug)) == ['image([]']:
  pass
else:
  # Write augmented image to a file
  imageio.imwrite(aug_images_path+image_prefix+filename, image_aug)
  # create df with augmented values of image widht and height
  info_df = group_df.drop(['xmin','ymin','xmax','ymax'], axis=1)
  for index, _ in info_df.iterrows():
    info_df.at[index, 'width'] = image_aug.shape[1]
    info_df.at[index, 'height'] = image_aug.shape[0]
  #Rename filenames by adding predefined prefix
  info_df['filename'] = info_df['filename'].apply(lambda x: image_prefix+x)
  #create df with augmented bbs coordinates using the function created earlier
  bbs_df = bbs_obj_to_df(bbs_aug)
  index_diff = np.setdiff1d(info_df.index,bbs_df.index)
  # if filename == 'frame84.jpg':
  #   print('post df conversion\n',bbs_df)
  #   print('info df len',info_df.shape,'bbs df len',bbs_df.shape)
  #   print('index diff',np.setdiff1d(info_df.index,bbs_df.index))
  #Perform reshaping
```

```
    info_df = info_df.drop(index_diff)

    #concat all new augmented info into new df

    aug_df = pd.concat([info_df, bbs_df],axis=1)

    # if filename == 'frame10.jpg':

    #   print('post df concat\n',aug_df)

    #append rows to aug_bbs_xy df

    aug_bbs_xy = pd.concat([aug_bbs_xy, aug_df])


  #return df with updated images and bbs

  aug_bbs_xy = aug_bbs_xy.reset_index()

  aug_bbs_xy = aug_bbs_xy.drop(['index'], axis=1)

  return aug_bbs_xy

#print(augmented_images_list)

augmented_images_df = []

for idx, augmented_images in enumerate(augmented_images_list):

  augmented_images_df.append(pd.DataFrame(augmented_images))

print(augmented_images_df[0].drop(['width','class','pose','truncated','difficult'],axis=1))

# visualize the augmentation image with bb to make sure augmentation worked

def visualize_augmentation(filename,index):

  grouped_resized = resized_images_df.groupby('filename')

  group_df_resized = grouped_resized.get_group(filename)
  group_df_resized = group_df_resized.reset_index()

  group_df_resized = group_df_resized.drop(['index'], axis=1)

    bb_array_r = group_df_resized.drop(['filename', 'width', 'height', 'class', 'pose', 'truncated', 'difficult'], axis=1).values

  resized_image = imageio.imread('resized_images/'+filename)

  print(resized_image.shape,bb_array_r.shape)

        bbs_r = BoundingBoxesOnImage.from_xyxy_array(bb_array_r, shape=resized_image.shape)


  grouped_aug = augmented_images_df[index].groupby('filename')

  group_df_aug = grouped_aug.get_group('aug'+str(index)+'_'+filename)

  group_df_aug = group_df_aug.reset_index()

  group_df_aug = group_df_aug.drop(['index'], axis=1)

   bb_array_a = group_df_aug.drop(['filename', 'width', 'height', 'class', 'pose', 'truncated', 'difficult'], axis=1).values

  aug_image = imageio.imread('new_aug_images/aug'+str(index)+'_'+filename)

  print(aug_image.shape,bb_array_a.shape)

        bbs_a = BoundingBoxesOnImage.from_xyxy_array(bb_array_a, shape=aug_image.shape)


  ia.imshow(np.hstack([

          bbs_r.draw_on_image(resized_image, size=2),
          bbs_a.draw_on_image(aug_image, size=2)
          ]))
#Copy all the images to output_images folder
!mkdir output_images
for file in os.listdir('resized_images'):
  shutil.copy('resized_images/'+file,'output_images/'+file)
for file in os.listdir('new_aug_images'):
  shutil.copy('new_aug_images/'+file,'output_images/'+file)
# Combine all the images df (augmented and resized)
all_images_df =  resized_images_df.copy()
print(all_images_df.shape,all_images_df.columns)
for df in augmented_images_df:
  print('Appending df',df.columns)
  all_images_df = pd.concat([all_images_df,df],ignore_index=True)
  #all_images_df.append(df,ignore_index=True)
print(all_images_df.shape)
#Converting the CSV files to Pascal VOC XML format

grouped_images = all_images_df.groupby('filename')
for i,df in enumerate(grouped_images):
  augmented_df = df[1]
```

```
 #print(augmented_df)
 root = ET.Element('annotation')
 ET.SubElement(root,"folder").text = os.path.basename('/content/output_images')
 ET.SubElement(root,"filename").text = augmented_df['filename'].iloc[0]
 ET.SubElement(root,"path").text = '/content/output_images/'
+augmented_df['filename'].iloc[0]
 source = ET.SubElement(root,"source")
 ET.SubElement(source,"database").text = "unknown"
 size = ET.SubElement(root,"size")
 ET.SubElement(size,"width").text = str(augmented_df['width'].iloc[0])
 ET.SubElement(size,"height").text = str(augmented_df['height'].iloc[0])
 ET.SubElement(size,"depth").text = str(3)
 ET.SubElement(root,"segmented").text = str(0)
 for i,z in enumerate(augmented_df['width']):
   object_elem = ET.SubElement(root,"object")
   ET.SubElement(object_elem,"name").text = augmented_df["class"].iloc[i]
   ET.SubElement(object_elem,"pose").text = augmented_df["pose"].iloc[i]
   ET.SubElement(object_elem,"truncated").text = augmented_df["truncated"].iloc[i]
   ET.SubElement(object_elem,"difficult").text = augmented_df["difficult"].iloc[i]
   bnd_box = ET.SubElement(object_elem,"bnd_box")
   ET.SubElement(bnd_box,"xmin").text = str(augmented_df["xmin"].iloc[i])
```

```
    ET.SubElement(bnd_box,"ymin").text = str(augmented_df["ymin"].iloc[i])

    ET.SubElement(bnd_box,"xmax").text = str(augmented_df["xmax"].iloc[i])

    ET.SubElement(bnd_box,"ymax").text = str(augmented_df["ymax"].iloc[i])

  tree = ET.ElementTree(root)

  tree.write('/content/output_images/'+augmented_df['filename'].iloc[0][:-4]+'.xml')

  print('/content/output_images/'+augmented_df['filename'].iloc[0][:-4]+'.xml','written'.
```

## 6.2 Training on RetinaNet Dataset

```
import detectron2

from detectron2.utils.logger import setup_logger

setup_logger()

# import some common libraries

import numpy as np

import cv2

import random

from google.colab.patches import cv2_imshow

# import some common detectron2 utilities

from detectron2 import model_zoo

from detectron2.engine import DefaultPredictor

from detectron2.config import get_cfg

from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog

from detectron2.data.catalog import DatasetCatalog

!gdown --id 1johg1Plrb4q0difgxiiduq2pgcbts34i

!unzip -q Tomato.coco.zip

#As our dataset is in COCO format we use the following lines

from detectron2.data.datasets import register_coco_instances

register_coco_instances("tomato_train", {},
'/content/train/_annotations.coco.jsonmodified.json', "/content/train/")

register_coco_instances("tomato_val", {},
"/content/valid/_annotations.coco.jsonmodified.json", "/content/valid/")

register_coco_instances("tomato_test",{},"/content/test/_annotations.coco.jsonmodified.js
on","/content/test")

#visualize training data

my_dataset_train_metadata = MetadataCatalog.get("tomato_train")

dataset_dicts = DatasetCatalog.get("tomato_train")

import random

from detectron2.utils.visualizer import Visualizer

for d in random.sample(dataset_dicts, 3):

   img = cv2.imread(d["file_name"])

   visualizer = Visualizer(img[:, :, ::-1], metadata=my_dataset_train_metadata, scale=0.5)

   vis = visualizer.draw_dataset_dict(d)

   cv2_imshow(vis.get_image()[:, :, ::-1])
```

```
anchor_sizes = eval("[[x, x * 2**(1.0/3), x * 2**(2.0/3) ] for x in [32, 64, 128, 256, 512
]]")

print(anchor_sizes)

from detectron2.engine import DefaultTrainer

from detectron2.config import get_cfg

import os

cfg = get_cfg()

#Using RetinaNet

cfg.MODEL.META_ARCHITECTURE = "RetinaNet"

cfg.MODEL.BACKBONE.NAME = "build_retinanet_resnet_fpn_backbone"

cfg.MODEL.RESNETS.OUT_FEATURES = ["res3", "res4", "res5"]

#cfg.MODEL.ANCHOR_GENERATOR.SIZES = anchor_sizes

cfg.MODEL.FPN.IN_FEATURES = ["res3", "res4", "res5"]

cfg.MODEL.RETINANET.IOU_THRESHOLDS = [0.4,0.5]

cfg.MODEL.RETINANET.IOU_LABELS = [0, -1, 1]

cfg.MODEL.RETINANET.NUM_CLASSES = 1

cfg.DATASETS.TRAIN = ("tomato_train",)

cfg.DATASETS.TEST = ("tomato_val",)

cfg.SOLVER.IMS_PER_BATCH = 2

cfg.SOLVER.BASE_LR = 0.01

#cfg.SOLVER.STEPS = (3000,4000)
```

```
#cfg.OUTPUT_DIR = "retinanet_train"

cfg.SOLVER.MAX_ITER = 2000

cfg.SOLVER.NESTEROV = True

cfg.INPUT.MIN_SIZE_TRAIN = (640, 672, 704, 736, 768, 800)

cfg.VERSION = 2

cfg.MODEL.WEIGHTS                                                                    =
"https://dl.fbaipublicfiles.com/detectron2/COCO-Detection/retinanet_R_50_FPN_3x/1903
97829/model_final_5bd44e.pkl"

cfg.MODEL.RESNETS.DEPTH = 50

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 128

cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1

cfg.DATALOADER.NUM_WORKERS = 2

cfg.SOLVER.CHECKPOINT_PERIOD = 500

from detectron2.engine import DefaultTrainer

from detectron2.evaluation import COCOEvaluator

class CocoTrainer(DefaultTrainer):

  @classmethod

  def build_evaluator(cls, cfg, dataset_name, output_folder=None):



    if output_folder is None:

        os.makedirs("coco_eval", exist_ok=True)

        output_folder = "coco_eval"
    return COCOEvaluator(dataset_name, cfg, False, output_folder)

torch.cuda.empty_cache()

os.makedirs(cfg.OUTPUT_DIR,exist_ok=True)

trainer = CocoTrainer(cfg)

trainer.resume_or_load(resume=False)

trainer.train()

import json

import matplotlib.pyplot as plt

experiment_folder = '.'

def load_json_arr(json_path):

    lines = []

    with open(json_path, 'r') as f:

        for line in f:

            lines.append(json.loads(line))

    return lines

experiment_metrics = load_json_arr(experiment_folder + '/metrics.json')

plt.plot(

    [x['iteration'] for x in experiment_metrics],

    [x['total_loss'] for x in experiment_metrics])

plt.plot(

    [x['iteration'] for x in experiment_metrics if 'validation_loss' in x],
    [x['validation_loss'] for x in experiment_metrics if 'validation_loss' in x])
plt.legend(['total_loss'], loc='upper left')
plt.show()
after_train_cfg = cfg.clone()
after_train_cfg.MODEL.WEIGHTS = '/content/output/model_final.pth'
after_train_cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.2
im = cv2.imread('/content/tomato_frames/frame10.jpg')
predictor = DefaultPredictor(after_train_cfg)
outputs = predictor(im)
print(outputs["instances"].pred_classes)
print(outputs["instances"].pred_boxes.tensor.tolist())
# We can use `Visualizer` to draw the predictions on the image.
v = Visualizer(im[:, :, ::-1], MetadataCatalog.get(cfg.DATASETS.TRAIN[0]), scale=0.5)
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
! rm -rf tomato_frames
!rm frame*.jpg
all_outputs = []
all_pred_boxes = []
for idx, file in enumerate(glob.glob('/content/yield_est_frames/*.jpg')):
  im = cv2.imread(file)
  output = predictor(im)

  all_outputs.append(output)

  all_pred_boxes += outputs["instances"].pred_boxes.tensor.tolist()

print(all_pred_boxes)

total_yield = 0.0 # in grams

all_area = []

for pred_box in all_pred_boxes:

  x_diff = pred_box[2] - pred_box[0]

  y_diff = pred_box[3] - pred_box[1]



  area = x_diff * y_diff

  all_area.append(area)

  if area > 0 and area < 500:

    total_yield += 70

    continue

  elif area < 1100:

    total_yield += 120

    continue

  else:

    total_yield += 150

print('Total yield is {0}'.format(total_yield/1000))
```

## 6.3 Frame Extraction

```
# Load video frames for prediction of yield

!mkdir tomato_frames

import cv2

vidcap = cv2.VideoCapture('/content/drive/My Drive/Tomato crop.h264')

success,image = vidcap.read()

count = 0

success = True

while success:

  success,image = vidcap.read()

   cv2.imwrite("tomato_frames/frame%d.jpg" % count, image)      # save frame as JPEG
file

  #print('Read a new frame: ', success)

  count += 1



print(count)

#Take frames that are used for yield prediction

import glob

import os

import shutil
```

```
fps = vidcap.get(cv2.CAP_PROP_FPS)

!mkdir yield_est_frames

for idx, file in enumerate(glob.glob('/content/tomato_frames/*.jpg')):

  filename = os.path.basename(file)

  frame_no = int(filename[5:-4])

  if frame_no % fps == 0:

    shutil.copy(file,'yield_est_frames/'+filename)
```

# CHAPTER 7

# RESULTS

## 7.1 Testing

Software Testing is necessary because mistakes are made unexpectedly. Some of those mistakes are unimportant, but some of them are expensive or dangerous. It is necessary to check everything and anything that can be produced because things can always go wrong.

Software testing can be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements as guided by its design and development and meets the user requirements effectively and efficiently with handling all the exceptional and boundary cases.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy and usability. It mainly aims at measuring specification, functionality and performance of a software program or application.

### 7.1.1 Types of Software Testing

Software Testing can be broadly classified into two types:

**Manual Testing**

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

**Automation Testing**

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.

Apart from regression testing, automation testing is also used to test the application from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.

### 7.1.2 Techniques of Software Testing

Software techniques can be majorly classified into two categories:

**Black Box Testing**

The technique of testing in which the tester doesn't have access to the source code of the software and is conducted at the software interface without concerning with the internal logical structure of the software is known as black box testing.

**White-Box Testing**

The technique of testing in which the tester is aware of the internal workings of the product, has access to its source code and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing.

### 7.1.3 Levels of Software Testing

Software level testing can be majorly classified into 4 levels:

**Unit Testing**

A level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.

**Integration Testing**

A level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

**System Testing**

A level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.

**Acceptance Testing**

A level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

### 7.1.4 Manual Testing with Google Colab

The manual testing of the  application was done on Google Colab running on Windows 10, with access to the internet via WiFi, with permissions granted being storage. The System is required to have Google Chrome services enabled to access Firebase services.

Table 7.1 : Manual Testing with Google Colab

| Test Case # | Description | Expected Result | Actual Result | Status |
|---|---|---|---|---|
| 1 | Start Google Colab and load project | Project  Loaded Successfully | Project Loaded Successfully | Pass |
| 2 | Load Train Images | Train Images Loaded Successfully | Train Images Loaded Successfully | Pass |

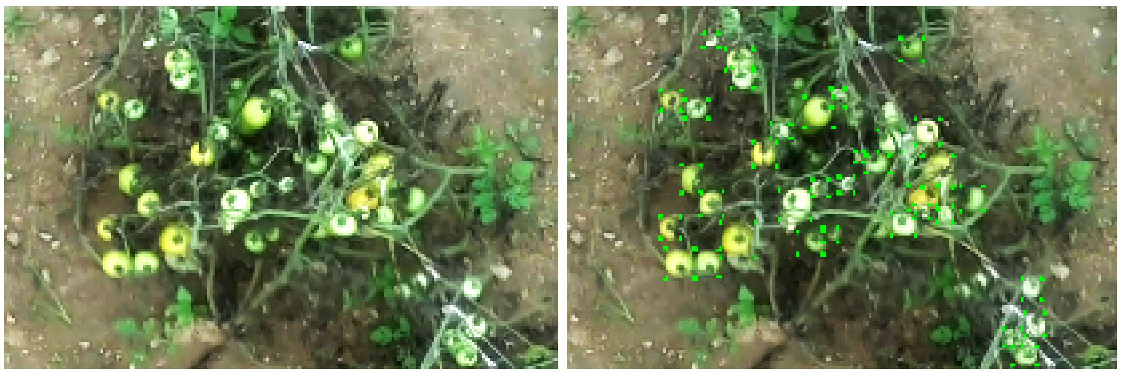
| | | | | |
|---|---|---|---|---|
| 3 | Run the Model Training Program | Model Trained Successfully | Model Trained Successfully | Pass |
| 4 | Test the Program with Custom Testing | Bounding Boxes labelled correctly | Bounding Boxes Labelled Correctly | Pass |
| 5 | Automated Testing<br><br>Test the Program with Pre Labelled Images and Measure Accuracy | Accuracy >=80% | Accuracy = 89% | Pass |
| 6 | Predict the Yield | Variable but should be close | Matches the expected yield | Pass |

## 7.2 Result

The results of the system show the most significant of all the components that are involved are the accuracy and response time concerning the search results. It is directly dependent on the accuracy of the keyphrase extraction module, which in turn depends on the accuracy of the Neural net model. For our testing, Our custom dataset has been. The dataset is a collection of aerial pictures of tomato fields and their corresponding labellings.

The testing starts with annotating an image of the frame of the video dataset, then Iterating through the dataset through our proposed system. Multiple outputs are generated for every single single. During this testing, our system performance is estimated with respect to the number of tomatoes mislabelled, the number of objects(Tomatoes) mislabeled by the NN model, and then estimate the accuracy detection done by the model. It is important to stress that the clarity of tomatoes labelled directly affects the quality of detection. Hence, accuracy in the results generated for the sample is always beneficial for good detection.

Depicted below is an example of how the model detects tomatoes on aerial images.



Normal Aerial view Detection of Tomatoes

Figure 7.2.1 : Aerial View of Tomato Detection

The measured performance of the prediction is based on the size of the bounding boxes. And accuracy of the FPN which is created by RetinaNET.

Table 7.2 : Prediction Performance

| **Total Boxes Predicted** | **Error** | **Accuracy** |
|---|---|---|
| 12148 | 1339 | 88.978% |

Depicted below is the minimization of Loss(error) over 2000 iterations.

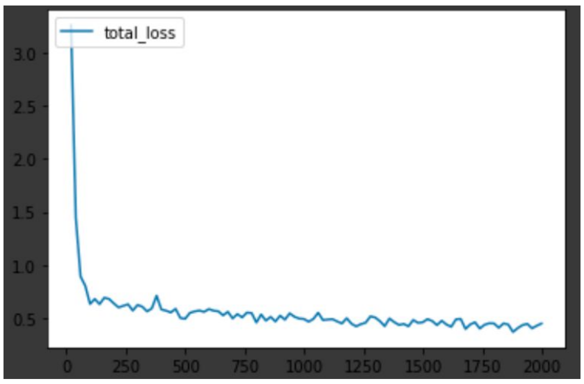
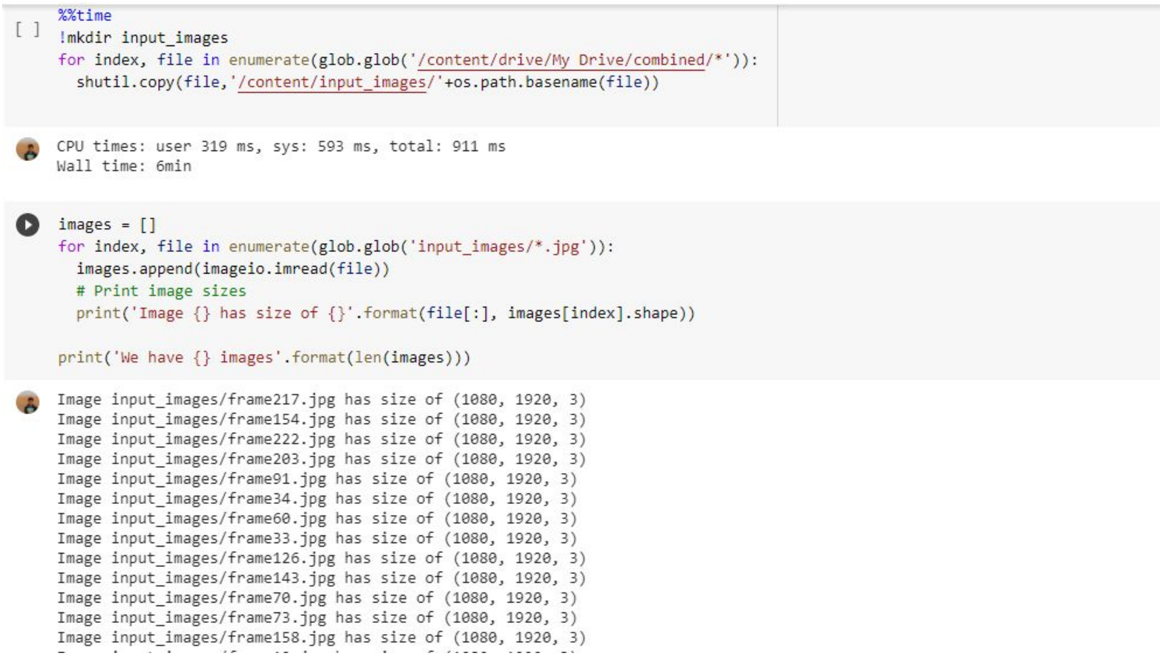


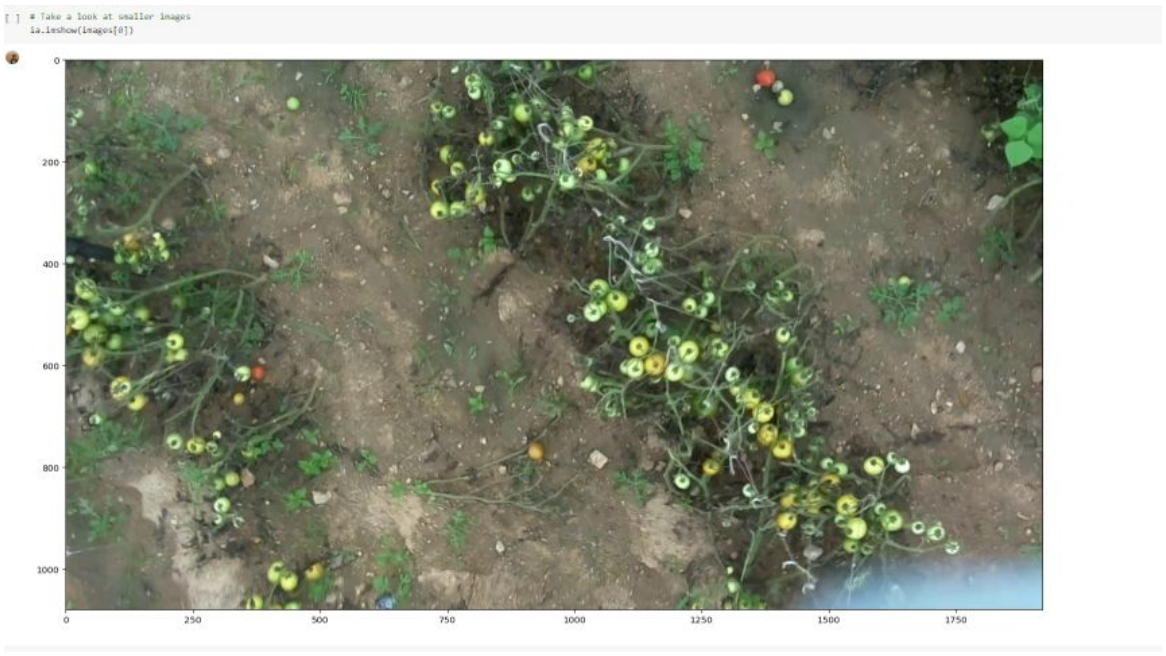
Figure 7.2.2: Minimization of total loss over 2000 iterations.

## 7.3 Snapshots



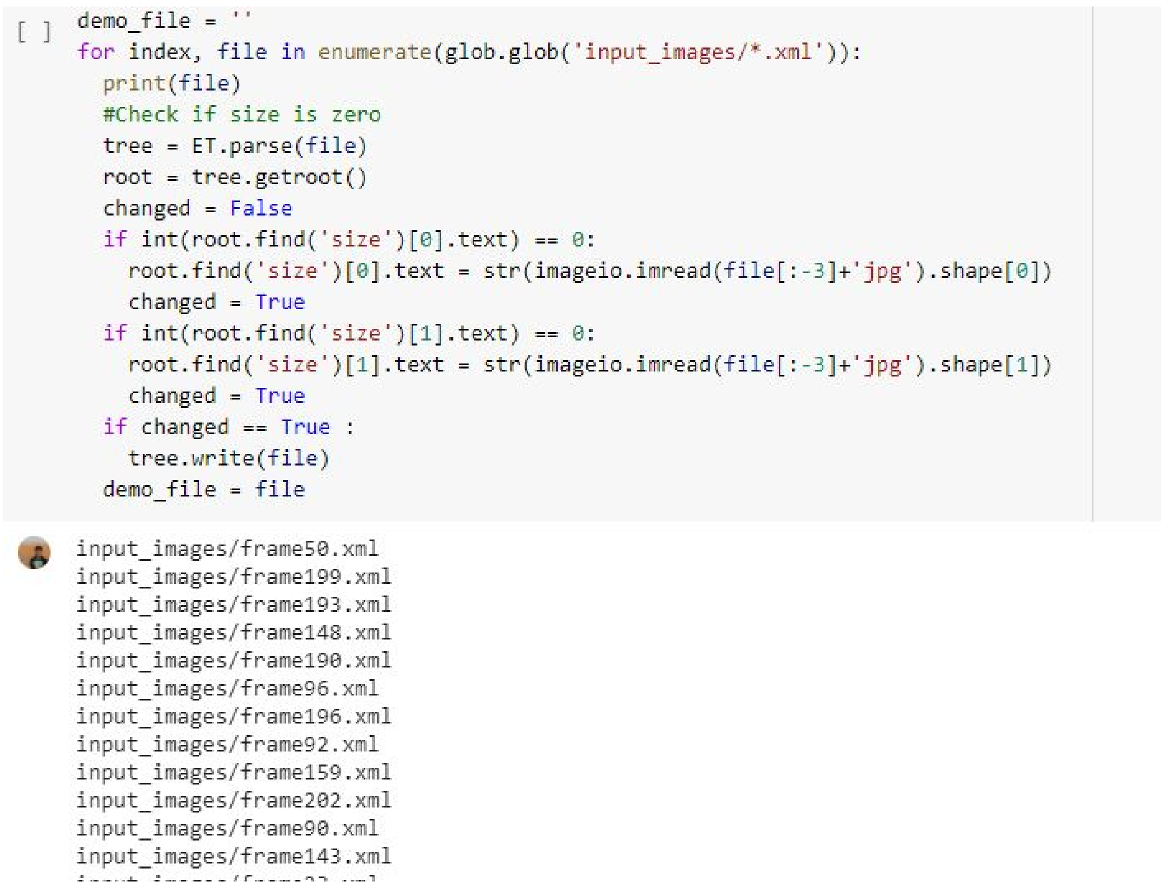
```
[ ] %%time
    !mkdir input_images
    for index, file in enumerate(glob.glob('/content/drive/My Drive/combined/*')):
      shutil.copy(file,'/content/input_images/'+os.path.basename(file))

    CPU times: user 319 ms, sys: 593 ms, total: 911 ms
    Wall time: 6min

    images = []
    for index, file in enumerate(glob.glob('input_images/*.jpg')):
      images.append(imageio.imread(file))
      # Print image sizes
      print('Image {} has size of {}'.format(file[:], images[index].shape))

    print('We have {} images'.format(len(images)))

    Image input_images/frame217.jpg has size of (1080, 1920, 3)
    Image input_images/frame154.jpg has size of (1080, 1920, 3)
    Image input_images/frame222.jpg has size of (1080, 1920, 3)
    Image input_images/frame203.jpg has size of (1080, 1920, 3)
    Image input_images/frame91.jpg has size of (1080, 1920, 3)
    Image input_images/frame34.jpg has size of (1080, 1920, 3)
    Image input_images/frame60.jpg has size of (1080, 1920, 3)
    Image input_images/frame33.jpg has size of (1080, 1920, 3)
    Image input_images/frame126.jpg has size of (1080, 1920, 3)
    Image input_images/frame143.jpg has size of (1080, 1920, 3)
    Image input_images/frame70.jpg has size of (1080, 1920, 3)
    Image input_images/frame73.jpg has size of (1080, 1920, 3)
    Image input_images/frame158.jpg has size of (1080, 1920, 3)
```

Figure 7.3.1 : Loading the Resources



```
[ ] # Take a look at smaller images
    ia.imshow(images[0])
```

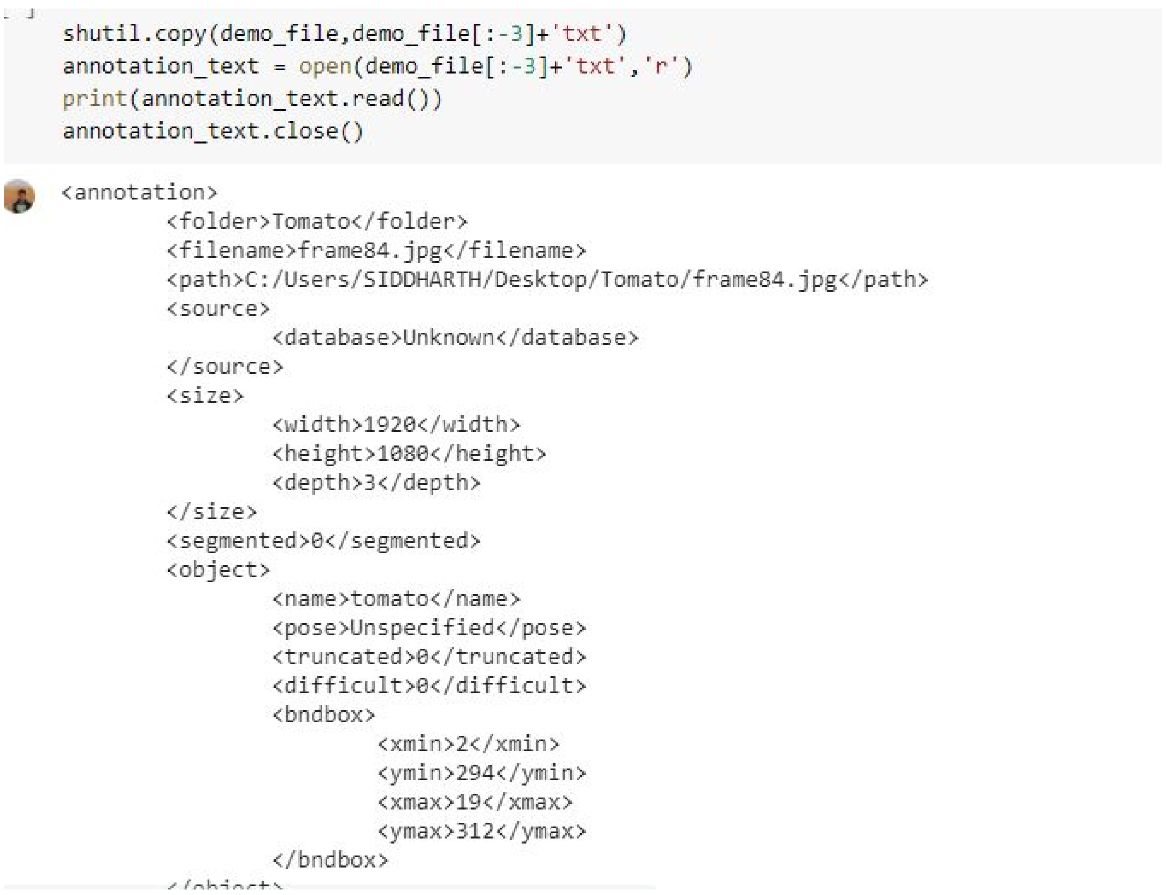
Figure 7.3.2 : A Sample Image From the Resource



```
demo_file = ''
for index, file in enumerate(glob.glob('input_images/*.xml')):
  print(file)
  #Check if size is zero
  tree = ET.parse(file)
  root = tree.getroot()
  changed = False
  if int(root.find('size')[0].text) == 0:
    root.find('size')[0].text = str(imageio.imread(file[:-3]+'jpg').shape[0])
    changed = True
  if int(root.find('size')[1].text) == 0:
    root.find('size')[1].text = str(imageio.imread(file[:-3]+'jpg').shape[1])
    changed = True
  if changed == True :
    tree.write(file)
  demo_file = file
```

```
input_images/frame50.xml
input_images/frame199.xml
input_images/frame193.xml
input_images/frame148.xml
input_images/frame190.xml
input_images/frame96.xml
input_images/frame196.xml
input_images/frame92.xml
input_images/frame159.xml
input_images/frame202.xml
input_images/frame90.xml
input_images/frame143.xml
```

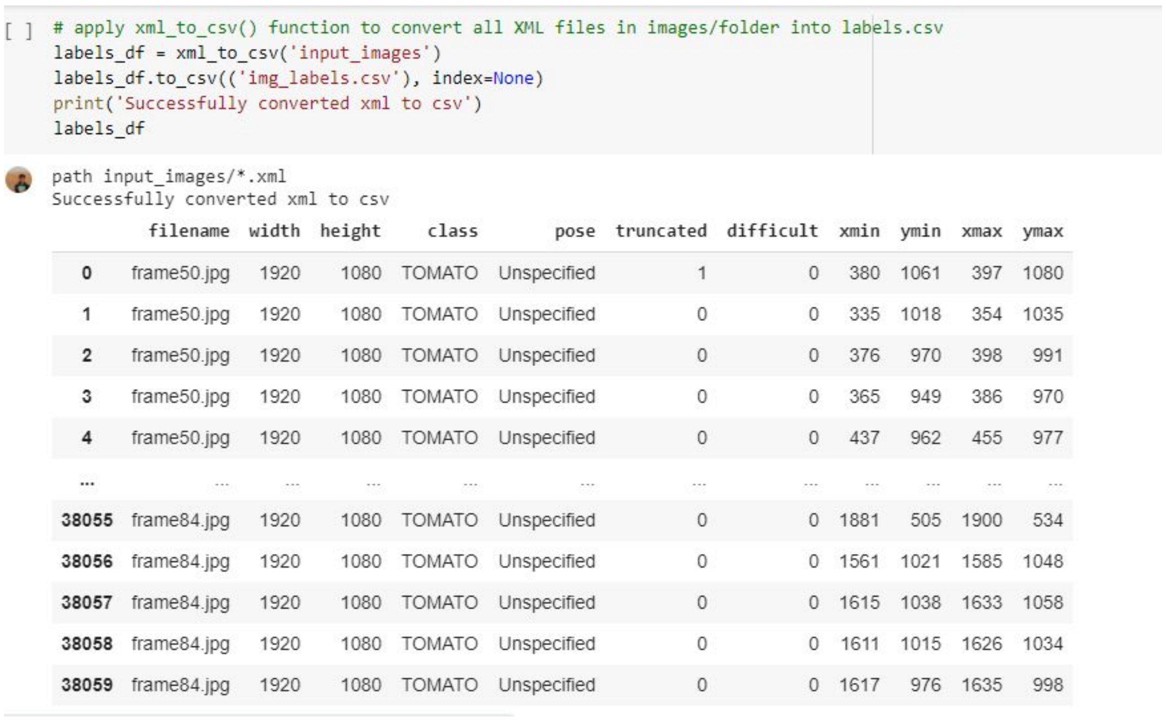
Figure7.3.3 : Importing Bounding Boxes to Train Model



```
shutil.copy(demo_file,demo_file[:-3]+'txt')
annotation_text = open(demo_file[:-3]+'txt','r')
print(annotation_text.read())
annotation_text.close()
```

```
<annotation>
        <folder>Tomato</folder>
        <filename>frame84.jpg</filename>
        <path>C:/Users/SIDDHARTH/Desktop/Tomato/frame84.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>1920</width>
                <height>1080</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>tomato</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>2</xmin>
                        <ymin>294</ymin>
                        <xmax>19</xmax>
                        <ymax>312</ymax>
                </bndbox>
```

Figure 7.3.4 : A Sample File from the Bounding Boxes File

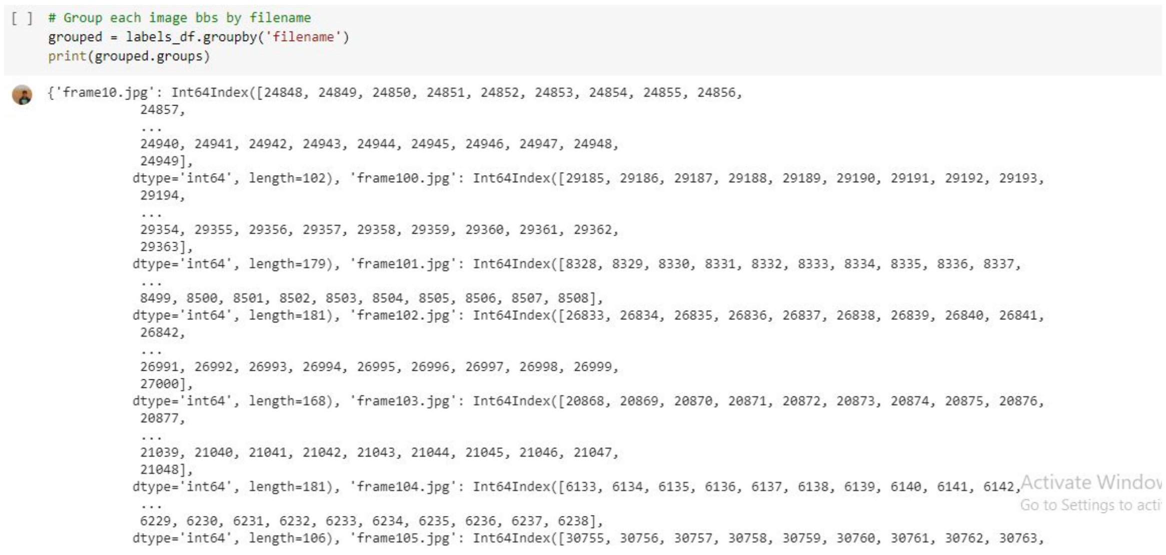


```
[ ] # apply xml_to_csv() function to convert all XML files in images/folder into labels.csv
    labels_df = xml_to_csv('input_images')
    labels_df.to_csv(('img_labels.csv'), index=None)
    print('Successfully converted xml to csv')
    labels_df
```

```
path input_images/*.xml
Successfully converted xml to csv
```

| | filename | width | height | class | pose | truncated | difficult | xmin | ymin | xmax | ymax |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | frame50.jpg | 1920 | 1080 | TOMATO | Unspecified | 1 | 0 | 380 | 1061 | 397 | 1080 |
| **1** | frame50.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 335 | 1018 | 354 | 1035 |
| **2** | frame50.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 376 | 970 | 398 | 991 |
| **3** | frame50.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 365 | 949 | 386 | 970 |
| **4** | frame50.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 437 | 962 | 455 | 977 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **38055** | frame84.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 1881 | 505 | 1900 | 534 |
| **38056** | frame84.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 1561 | 1021 | 1585 | 1048 |
| **38057** | frame84.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 1615 | 1038 | 1633 | 1058 |
| **38058** | frame84.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 1611 | 1015 | 1626 | 1034 |
| **38059** | frame84.jpg | 1920 | 1080 | TOMATO | Unspecified | 0 | 0 | 1617 | 976 | 1635 | 998 |

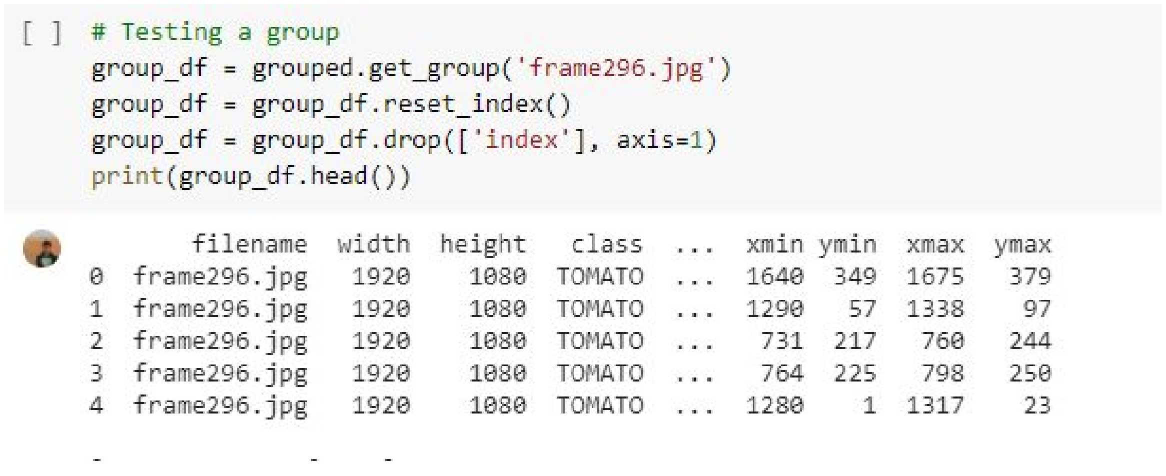
Figure 7.3.5 : Transforming the Bounding Boxes to a Convenient Format



```
[ ] # Group each image bbs by filename
    grouped = labels_df.groupby('filename')
    print(grouped.groups)
```

```
{'frame10.jpg': Int64Index([24848, 24849, 24850, 24851, 24852, 24853, 24854, 24855, 24856,
             24857,
             ...
             24940, 24941, 24942, 24943, 24944, 24945, 24946, 24947, 24948,
             24949],
            dtype='int64', length=102), 'frame100.jpg': Int64Index([29185, 29186, 29187, 29188, 29189, 29190, 29191, 29192, 29193,
             29194,
             ...
             29354, 29355, 29356, 29357, 29358, 29359, 29360, 29361, 29362,
             29363],
            dtype='int64', length=179), 'frame101.jpg': Int64Index([8328, 8329, 8330, 8331, 8332, 8333, 8334, 8335, 8336, 8337,
             ...
             8499, 8500, 8501, 8502, 8503, 8504, 8505, 8506, 8507, 8508],
            dtype='int64', length=181), 'frame102.jpg': Int64Index([26833, 26834, 26835, 26836, 26837, 26838, 26839, 26840, 26841,
             26842,
             ...
             26991, 26992, 26993, 26994, 26995, 26996, 26997, 26998, 26999,
             27000],
            dtype='int64', length=168), 'frame103.jpg': Int64Index([20868, 20869, 20870, 20871, 20872, 20873, 20874, 20875, 20876,
             20877,
             ...
             21039, 21040, 21041, 21042, 21043, 21044, 21045, 21046, 21047,
             21048],
            dtype='int64', length=181), 'frame104.jpg': Int64Index([6133, 6134, 6135, 6136, 6137, 6138, 6139, 6140, 6141, 6142,
             ...
             6229, 6230, 6231, 6232, 6233, 6234, 6235, 6236, 6237, 6238],
            dtype='int64', length=106), 'frame105.jpg': Int64Index([30755, 30756, 30757, 30758, 30759, 30760, 30761, 30762, 30763,
```

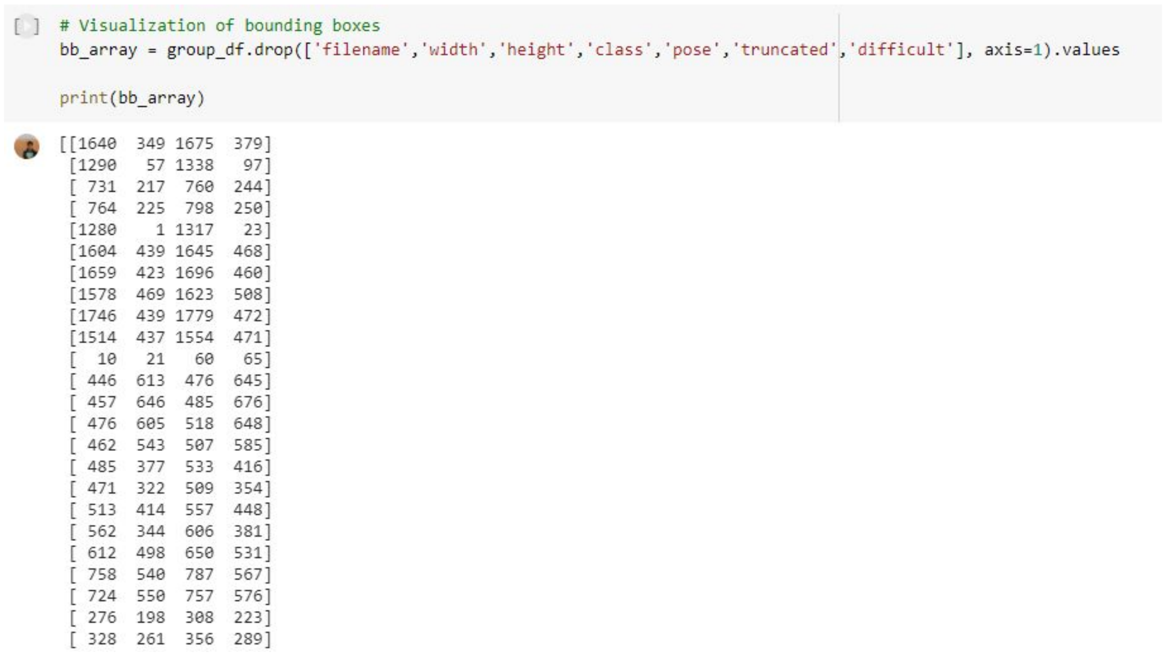
Figure 7.3.6 : Training Model with Boxes and Image Files



```
[ ] # Testing a group
    group_df = grouped.get_group('frame296.jpg')
    group_df = group_df.reset_index()
    group_df = group_df.drop(['index'], axis=1)
    print(group_df.head())
```

```
         filename  width  height   class  ...  xmin  ymin  xmax  ymax
0  frame296.jpg   1920    1080  TOMATO  ...  1640   349  1675   379
1  frame296.jpg   1920    1080  TOMATO  ...  1290    57  1338    97
2  frame296.jpg   1920    1080  TOMATO  ...   731   217   760   244
3  frame296.jpg   1920    1080  TOMATO  ...   764   225   798   250
4  frame296.jpg   1920    1080  TOMATO  ...  1280     1  1317    23
```

Figure 7.3.7 : Tomatoes Labelled for Each Frame



```
[ ] # Visualization of bounding boxes
    bb_array = group_df.drop(['filename','width','height','class','pose','truncated','difficult'], axis=1).values

    print(bb_array)
```

```
[[1640  349 1675  379]
 [1290   57 1338   97]
 [ 731  217  760  244]
 [ 764  225  798  250]
 [1280    1 1317   23]
 [1604  439 1645  468]
 [1659  423 1696  460]
 [1578  469 1623  508]
 [1746  439 1779  472]
 [1514  437 1554  471]
 [  10   21   60   65]
 [ 446  613  476  645]
 [ 457  646  485  676]
 [ 476  605  518  648]
 [ 462  543  507  585]
 [ 485  377  533  416]
 [ 471  322  509  354]
 [ 513  414  557  448]
 [ 562  344  606  381]
 [ 612  498  650  531]
 [ 758  540  787  567]
 [ 724  550  757  576]
 [ 276  198  308  223]
 [ 328  261  356  289]
```

Figure 7.3.8 : Bounding Boxes Prediction

```
[ ] # pass the array of bounding boxes coordinates to the imgaug library
    image = imageio.imread('input_images/frame296.jpg')
    bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)
    #display the image and draw bounding boxes
    ia.imshow(bbs.draw_on_image(image, size=2))
```

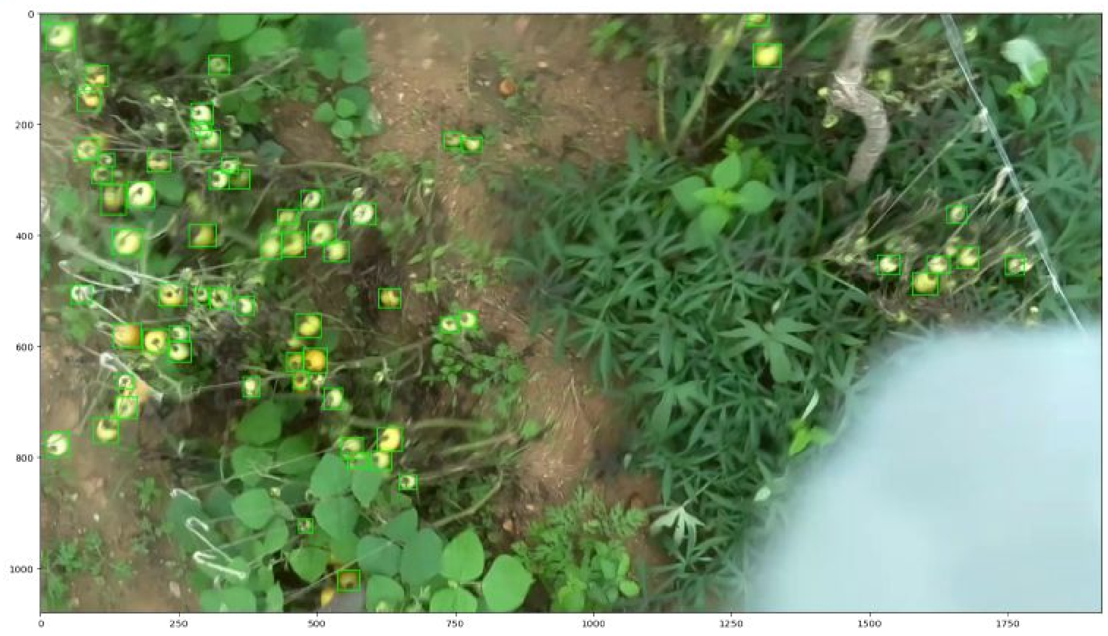


Figure 7.3.9 : Visualization of Bounding Boxes

```
[ ] # visualize the resized image with bb to make sure resizing worked
    grouped = resized_images_df.groupby('filename')
    group_df = grouped.get_group('frame296.jpg')
    group_df = group_df.reset_index()
    group_df = group_df.drop(['index'], axis=1)
    bb_array = group_df.drop(['filename', 'width', 'height', 'class', 'pose', 'truncated', 'difficult'], axis=1).values
    image = imageio.imread('resized_images/frame296.jpg')
    bbs = BoundingBoxesOnImage.from_xyxy_array(bb_array, shape=image.shape)
    ia.imshow(bbs.draw_on_image(image, size=2))
```

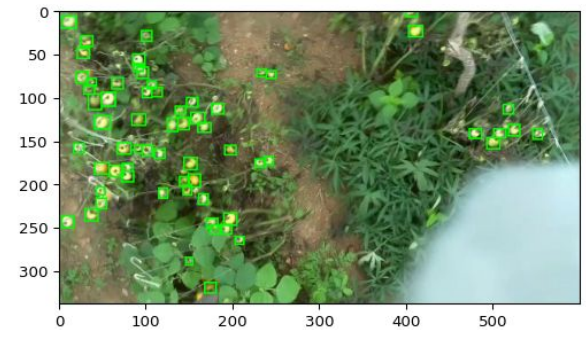
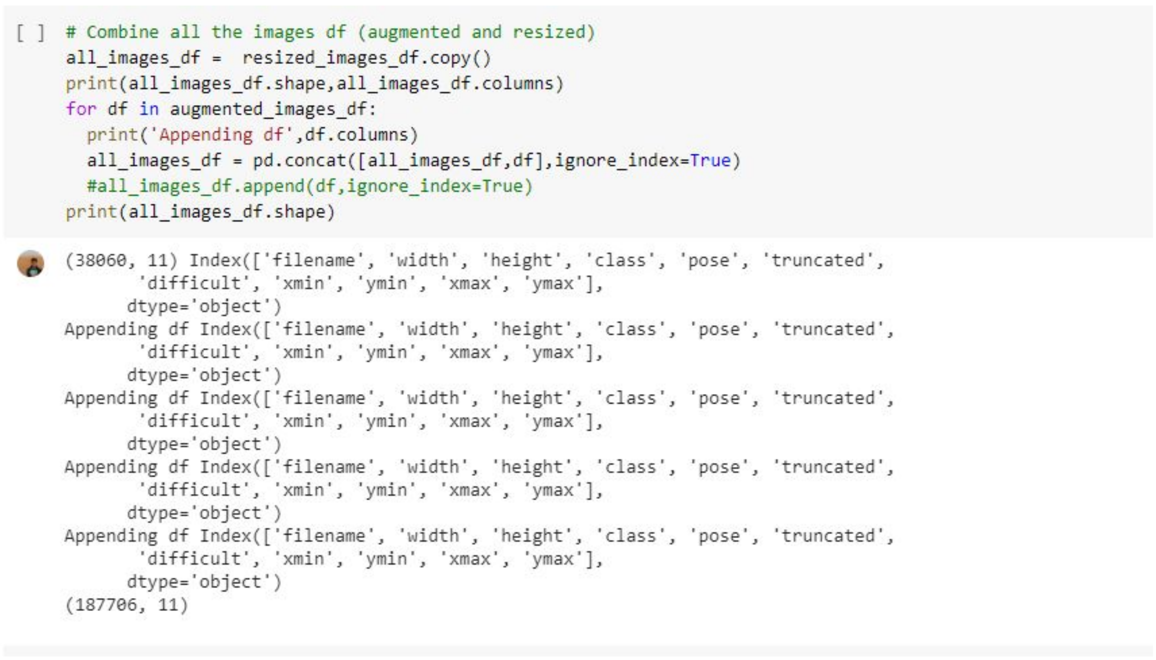


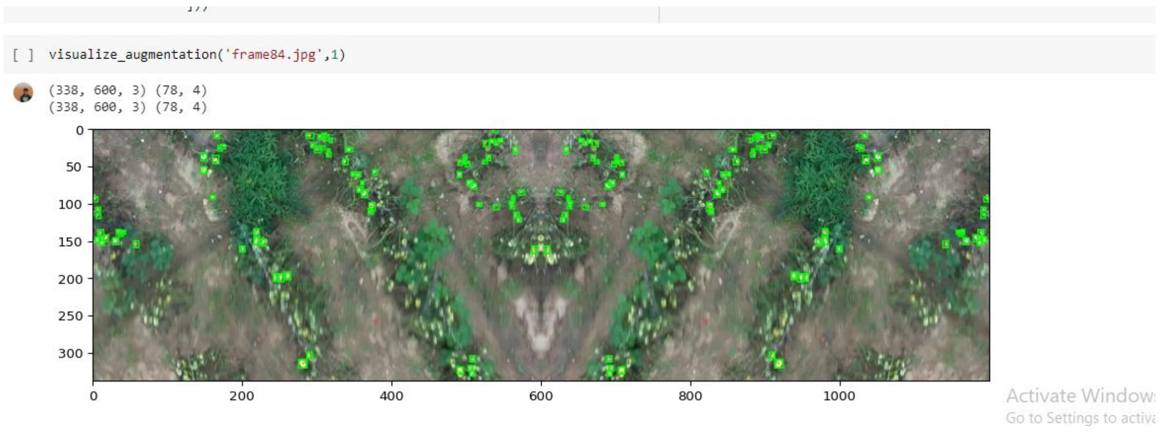
Figure 7.3.10 : Resizing the Image



```
[ ] # Combine all the images df (augmented and resized)
    all_images_df =  resized_images_df.copy()
    print(all_images_df.shape,all_images_df.columns)
    for df in augmented_images_df:
      print('Appending df',df.columns)
      all_images_df = pd.concat([all_images_df,df],ignore_index=True)
      #all_images_df.append(df,ignore_index=True)
    print(all_images_df.shape)
```

```
(38060, 11) Index(['filename', 'width', 'height', 'class', 'pose', 'truncated',
       'difficult', 'xmin', 'ymin', 'xmax', 'ymax'],
      dtype='object')
Appending df Index(['filename', 'width', 'height', 'class', 'pose', 'truncated',
       'difficult', 'xmin', 'ymin', 'xmax', 'ymax'],
      dtype='object')
Appending df Index(['filename', 'width', 'height', 'class', 'pose', 'truncated',
       'difficult', 'xmin', 'ymin', 'xmax', 'ymax'],
      dtype='object')
Appending df Index(['filename', 'width', 'height', 'class', 'pose', 'truncated',
       'difficult', 'xmin', 'ymin', 'xmax', 'ymax'],
      dtype='object')
Appending df Index(['filename', 'width', 'height', 'class', 'pose', 'truncated',
       'difficult', 'xmin', 'ymin', 'xmax', 'ymax'],
      dtype='object')
(187706, 11)
```

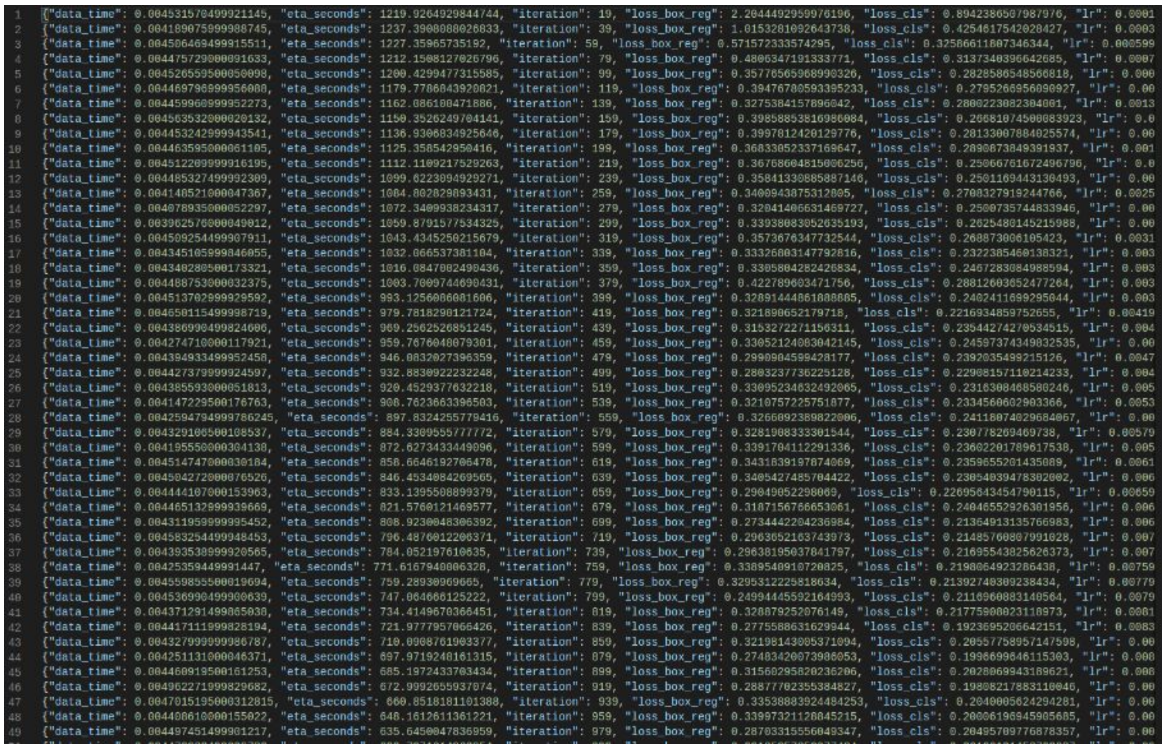
Figure 7.3.11 : Combining Images for the Full Field View



```
[ ] visualize_augmentation('frame84.jpg',1)
```

```
(338, 600, 3) (78, 4)
(338, 600, 3) (78, 4)
```

Figure 7.3.12 : Visualization of the Combined Image



```
{"data_time": 0.004531570499921145, "eta_seconds": 1219.9264929844744, "iteration": 19, "loss_box_reg": 2.2044492959976196, "loss_cls": 0.894238650787976, "lr": 0.0001
{"data_time": 0.004189075999988745, "eta_seconds": 1237.3908088026833, "iteration": 39, "loss_box_reg": 1.0153281092643738, "loss_cls": 0.4254617542028427, "lr": 0.0003
{"data_time": 0.004506469499915511, "eta_seconds": 1227.35965735192, "iteration": 59, "loss_box_reg": 0.571572333574295, "loss_cls": 0.3258661180734634, "lr": 0.000599
{"data_time": 0.004475729000091633, "eta_seconds": 1212.1508127026796, "iteration": 79, "loss_box_reg": 0.4806347191333771, "loss_cls": 0.3137340396642685, "lr": 0.0007
{"data_time": 0.004526559500050098, "eta_seconds": 1200.4299477315585, "iteration": 99, "loss_box_reg": 0.3577656596899326, "loss_cls": 0.2828586548566818, "lr": 0.000
{"data_time": 0.004469796999956008, "eta_seconds": 1179.7786843920021, "iteration": 119, "loss_box_reg": 0.3947678059339523, "loss_cls": 0.2795266956090927, "lr": 0.00
{"data_time": 0.004459960999952273, "eta_seconds": 1162.086100471886, "iteration": 139, "loss_box_reg": 0.327538415789604, "loss_cls": 0.2800223082304001, "lr": 0.0013
{"data_time": 0.004563532000020132, "eta_seconds": 1150.3526249704141, "iteration": 159, "loss_box_reg": 0.39858853816986084, "loss_cls": 0.26681074500083923, "lr": 0.0
{"data_time": 0.004453242999943541, "eta_seconds": 1136.9306834925646, "iteration": 179, "loss_box_reg": 0.3997812420129776, "loss_cls": 0.2813300788402557, "lr": 0.00
{"data_time": 0.004463595000061105, "eta_seconds": 1125.358542950416, "iteration": 199, "loss_box_reg": 0.3683305233716964, "loss_cls": 0.2890873849391937, "lr": 0.001
{"data_time": 0.004512209999916195, "eta_seconds": 1112.1109217529263, "iteration": 219, "loss_box_reg": 0.36768604815006256, "loss_cls": 0.25066761672496796, "lr": 0.0
{"data_time": 0.004485327499992309, "eta_seconds": 1099.6223094929271, "iteration": 239, "loss_box_reg": 0.35841330885887146, "loss_cls": 0.2501169443130493, "lr": 0.00
{"data_time": 0.004148521000047367, "eta_seconds": 1084.802829893431, "iteration": 259, "loss_box_reg": 0.3400943875312805, "loss_cls": 0.2708327919244766, "lr": 0.0025
{"data_time": 0.004078935000052297, "eta_seconds": 1072.3409938234317, "iteration": 279, "loss_box_reg": 0.32041406631469727, "loss_cls": 0.2500735744833946, "lr": 0.00
{"data_time": 0.003962576000049012, "eta_seconds": 1059.8791577534325, "iteration": 299, "loss_box_reg": 0.33938083052635193, "loss_cls": 0.2625480145215988, "lr": 0.00
{"data_time": 0.004509254499907911, "eta_seconds": 1043.4345250215679, "iteration": 319, "loss_box_reg": 0.3573676347732544, "loss_cls": 0.268873006105423, "lr": 0.0031
{"data_time": 0.004345105999846055, "eta_seconds": 1032.066537381104, "iteration": 339, "loss_box_reg": 0.3332680314779216, "loss_cls": 0.2322385460138321, "lr": 0.003
{"data_time": 0.004340280500173321, "eta_seconds": 1016.0847002490436, "iteration": 359, "loss_box_reg": 0.3305804282426834, "loss_cls": 0.2467283084988594, "lr": 0.003
{"data_time": 0.004488753000032375, "eta_seconds": 1003.7009744690431, "iteration": 379, "loss_box_reg": 0.422789603471756, "loss_cls": 0.2881260365247726, "lr": 0.003
{"data_time": 0.004513702999929592, "eta_seconds": 993.1256086081606, "iteration": 399, "loss_box_reg": 0.3289144486188805, "loss_cls": 0.2402411699295044, "lr": 0.003
{"data_time": 0.004650115499998719, "eta_seconds": 979.7818290121724, "iteration": 419, "loss_box_reg": 0.321890652179718, "loss_cls": 0.2216934859752655, "lr": 0.00419
{"data_time": 0.004386990499824606, "eta_seconds": 969.2562526851245, "iteration": 439, "loss_box_reg": 0.3153272271156311, "loss_cls": 0.2354427427053451, "lr": 0.004
{"data_time": 0.004274710000117921, "eta_seconds": 959.7676048079301, "iteration": 459, "loss_box_reg": 0.33052124083042145, "loss_cls": 0.24597374349832535, "lr": 0.00
{"data_time": 0.004394933499952458, "eta_seconds": 946.0832027396359, "iteration": 479, "loss_box_reg": 0.2990904599428177, "loss_cls": 0.2392035499215126, "lr": 0.0047
{"data_time": 0.004427379999924597, "eta_seconds": 932.8830922232248, "iteration": 499, "loss_box_reg": 0.2803237736225128, "loss_cls": 0.2290815711021423, "lr": 0.004
{"data_time": 0.004385593000051813, "eta_seconds": 920.4529377632218, "iteration": 519, "loss_box_reg": 0.33095234632492065, "loss_cls": 0.2316308468580246, "lr": 0.005
{"data_time": 0.004147229500176763, "eta_seconds": 908.7623663396503, "iteration": 539, "loss_box_reg": 0.3210757225751877, "loss_cls": 0.2334560602903366, "lr": 0.0053
{"data_time": 0.0042594794999786245, "eta_seconds": 897.8324255779416, "iteration": 559, "loss_box_reg": 0.32660923898220006, "loss_cls": 0.24118074029684067, "lr": 0.00
{"data_time": 0.004329106500108537, "eta_seconds": 884.3309555777772, "iteration": 579, "loss_box_reg": 0.3281908333301544, "loss_cls": 0.230778269469738, "lr": 0.00579
{"data_time": 0.004195550000304138, "eta_seconds": 872.6273433449096, "iteration": 599, "loss_box_reg": 0.3391704112291336, "loss_cls": 0.2360220178961753, "lr": 0.005
{"data_time": 0.004514747000030184, "eta_seconds": 858.6646192706478, "iteration": 619, "loss_box_reg": 0.3431839197874069, "loss_cls": 0.2359655201435089, "lr": 0.0061
{"data_time": 0.004504272000076526, "eta_seconds": 846.4534084269565, "iteration": 639, "loss_box_reg": 0.3405427485704422, "loss_cls": 0.23054039478302002, "lr": 0.006
{"data_time": 0.004444107000153963, "eta_seconds": 833.1395508899379, "iteration": 659, "loss_box_reg": 0.29049052298069, "loss_cls": 0.22695643454790115, "lr": 0.00659
{"data_time": 0.004465132999939669, "eta_seconds": 821.5760121469577, "iteration": 679, "loss_box_reg": 0.3187156766653061, "loss_cls": 0.24046552926301956, "lr": 0.006
{"data_time": 0.004311959999995452, "eta_seconds": 808.9230048306392, "iteration": 699, "loss_box_reg": 0.2734442204236984, "loss_cls": 0.21364913135766983, "lr": 0.006
{"data_time": 0.004583254499948453, "eta_seconds": 796.4876012206371, "iteration": 719, "loss_box_reg": 0.2963652163743973, "loss_cls": 0.21485760807991028, "lr": 0.007
{"data_time": 0.004393538999920565, "eta_seconds": 784.052197610635, "iteration": 739, "loss_box_reg": 0.29638195037841797, "loss_cls": 0.21695543825626373, "lr": 0.007
{"data_time": 0.00425359449991447, "eta_seconds": 771.6167940006328, "iteration": 759, "loss_box_reg": 0.3389540910720825, "loss_cls": 0.2198064923286438, "lr": 0.00759
{"data_time": 0.004559855500019694, "eta_seconds": 759.28930969665, "iteration": 779, "loss_box_reg": 0.3295312225818634, "loss_cls": 0.21392740309238434, "lr": 0.00779
{"data_time": 0.004536990499900639, "eta_seconds": 747.064666125222, "iteration": 799, "loss_box_reg": 0.24994445592164993, "loss_cls": 0.2116960883140564, "lr": 0.0079
{"data_time": 0.004371291499865038, "eta_seconds": 734.4149670366451, "iteration": 819, "loss_box_reg": 0.328879252076149, "loss_cls": 0.21775900023110973, "lr": 0.0081
{"data_time": 0.004417111999828194, "eta_seconds": 721.9777957066426, "iteration": 839, "loss_box_reg": 0.2775588631629944, "loss_cls": 0.1923695206642151, "lr": 0.0083
{"data_time": 0.004327999999986787, "eta_seconds": 710.0908761903377, "iteration": 859, "loss_box_reg": 0.32198143005371094, "loss_cls": 0.20557758957147598, "lr": 0.00
{"data_time": 0.004251131000046371, "eta_seconds": 697.9719248161315, "iteration": 879, "loss_box_reg": 0.27483420073986053, "loss_cls": 0.1998699646115303, "lr": 0.008
{"data_time": 0.004460919500161253, "eta_seconds": 685.1972433703434, "iteration": 899, "loss_box_reg": 0.31560295820236206, "loss_cls": 0.2028069943189621, "lr": 0.008
{"data_time": 0.004962271999829682, "eta_seconds": 672.9992655937074, "iteration": 919, "loss_box_reg": 0.28877702355384827, "loss_cls": 0.19808217883110046, "lr": 0.00
{"data_time": 0.0047015195000312815, "eta_seconds": 660.8518181101388, "iteration": 939, "loss_box_reg": 0.33538883924484253, "loss_cls": 0.2040005624294281, "lr": 0.00
{"data_time": 0.004408610000155022, "eta_seconds": 648.1612611361221, "iteration": 959, "loss_box_reg": 0.33997321128845215, "loss_cls": 0.20006196945905685, "lr": 0.00
{"data_time": 0.004497451499901217, "eta_seconds": 635.6450047836959, "iteration": 979, "loss_box_reg": 0.28703315556049347, "loss_cls": 0.20495709776878357, "lr": 0.00
```

Figure 7.3.13 : Time Elapsed and Loss for Running the Test Cases.

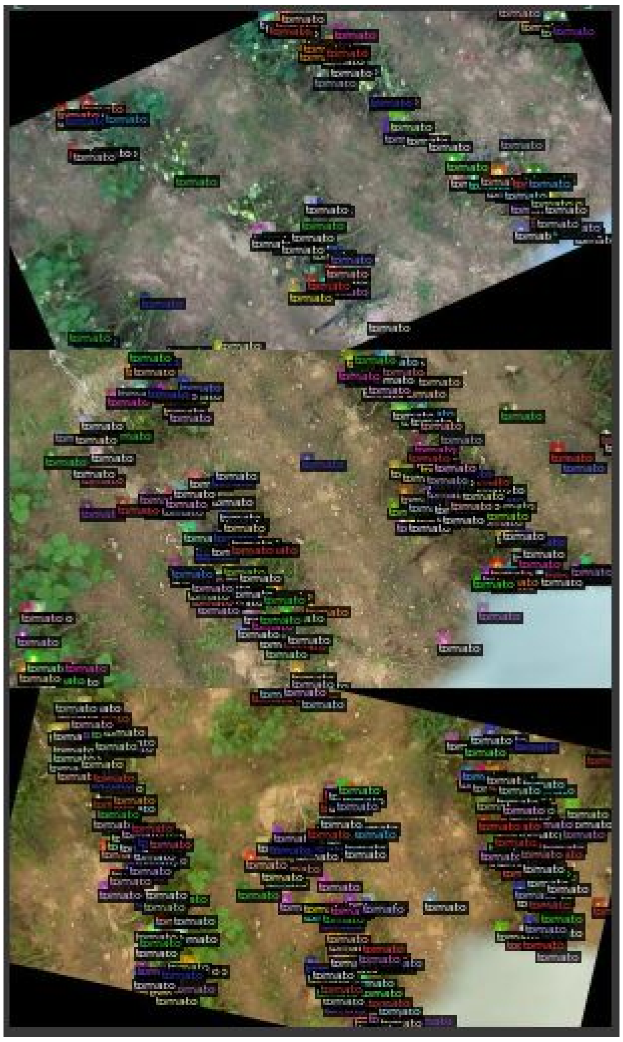


Figure 7.3.14 : Tomatoes Labelled in COCO form After Prediction

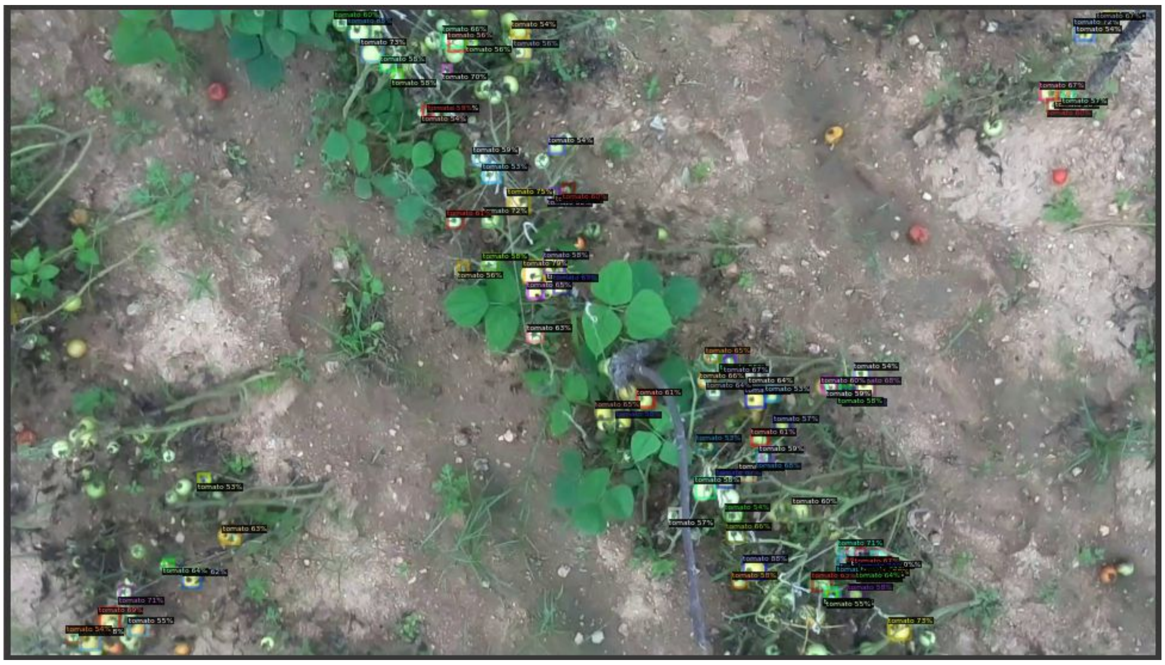
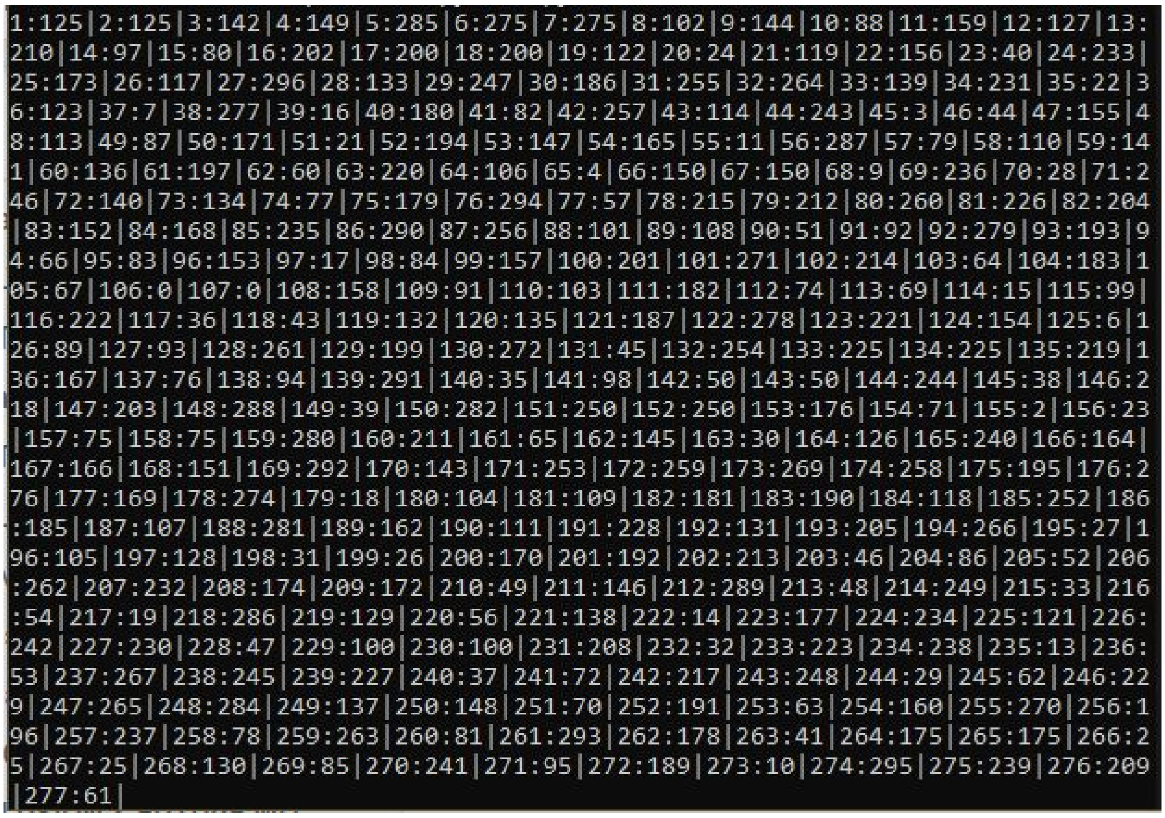


Figure 7.3.15 : Tomatoes Labelled in a Particular Frame.



```
1:125|2:125|3:142|4:149|5:285|6:275|7:275|8:102|9:144|10:88|11:159|12:127|13:
210|14:97|15:80|16:202|17:200|18:200|19:122|20:24|21:119|22:156|23:40|24:233|
25:173|26:117|27:296|28:133|29:247|30:186|31:255|32:264|33:139|34:231|35:22|3
6:123|37:7|38:277|39:16|40:180|41:82|42:257|43:114|44:243|45:3|46:44|47:155|4
8:113|49:87|50:171|51:21|52:194|53:147|54:165|55:11|56:287|57:79|58:110|59:14
1|60:136|61:197|62:60|63:220|64:106|65:4|66:150|67:150|68:9|69:236|70:28|71:2
46|72:140|73:134|74:77|75:179|76:294|77:57|78:215|79:212|80:260|81:226|82:204
|83:152|84:168|85:235|86:290|87:256|88:101|89:108|90:51|91:92|92:279|93:193|9
4:66|95:83|96:153|97:17|98:84|99:157|100:201|101:271|102:214|103:64|104:183|1
05:67|106:0|107:0|108:158|109:91|110:103|111:182|112:74|113:69|114:15|115:99|
116:222|117:36|118:43|119:132|120:135|121:187|122:278|123:221|124:154|125:6|1
26:89|127:93|128:261|129:199|130:272|131:45|132:254|133:225|134:225|135:219|1
36:167|137:76|138:94|139:291|140:35|141:98|142:50|143:50|144:244|145:38|146:2
18|147:203|148:288|149:39|150:282|151:250|152:250|153:176|154:71|155:2|156:23
|157:75|158:75|159:280|160:211|161:65|162:145|163:30|164:126|165:240|166:164|
167:166|168:151|169:292|170:143|171:253|172:259|173:269|174:258|175:195|176:2
76|177:169|178:274|179:18|180:104|181:109|182:181|183:190|184:118|185:252|186
:185|187:107|188:281|189:162|190:111|191:228|192:131|193:205|194:266|195:27|1
96:105|197:128|198:31|199:26|200:170|201:192|202:213|203:46|204:86|205:52|206
:262|207:232|208:174|209:172|210:49|211:146|212:289|213:48|214:249|215:33|216
:54|217:19|218:286|219:129|220:56|221:138|222:14|223:177|224:234|225:121|226:
242|227:230|228:47|229:100|230:100|231:208|232:32|233:223|234:238|235:13|236:
53|237:267|238:245|239:227|240:37|241:72|242:217|243:248|244:29|245:62|246:22
9|247:265|248:284|249:137|250:148|251:70|252:191|253:63|254:160|255:270|256:1
96|257:237|258:78|259:263|260:81|261:293|262:178|263:41|264:175|265:175|266:2
5|267:25|268:130|269:85|270:241|271:95|272:189|273:10|274:295|275:239|276:209
|277:61|
```

Figure 3.16: Tomatoes Detected in Each Frame

# CHAPTER 8

# APPLICATIONS

The result of this project will increase productivity and yield estimation and other factors like the grades of the tomatoes. This will drastically reduce the manual efforts made to estimate the marketable yields for the current growing season. This project can also identify the quality of tomatoes during the growing system which helps the farmers identify the need for special care for the low grade tomatoes which can be a tomato of better quality with intensive care. With the growth of technology in all the fields today, having accurate yield predictions of the tomato can be useful to sell tomatoes when the demand is high and tomato prices are rising, moreover it would help the farmers to estimate the financial requirements of the next growing season. The farmers can easily make appropriate deals if they know the yield of each grade and size of tomato.

Video would greatly improve this and reduce the time taken. Also it will not require much intervention as it is with images. Also video can capture parts that may have been missed in images so a more accurate estimation can be made by the use of videos instead of images as the same tomato is captured in more than one frame and through different angles. Also the state of farmers is very poor in our country as the government fails to provide proper facilities to aid the farmer. This system will allow the farmer to remotely monitor the crops and help them prosper. Farmers can check everything and give special care or attention to special areas of the field which would boost the yield and bring them out of their current state.

To summarize, following are the applications of the app that are resultant of this project.

- Predict the yield of the tomato crop.
- Interpolation of the video captured by the UAV using context aware synthesis.
- Identification of the different stages of the tomato crop growing as well of the defective ones.
- Suggestions to the farmer to make changes based on the color of the

tomatoes in different stages of the tomato growing, so that the farmer can give special attention to specific parts of the field.

- Helps in the detection of suckering in tomato plants without actually visiting each and every part of the field. Removal of suckering would help tomatoes grow fuller and receive more sunshine.
- Provide specific yields for specific grades of tomatoes which would help the farmer make an estimate of the earnings made from the growing season.

CHAPTER 9

# CONCLUSION & FUTURE WORK

The proposed system presents a computationally efficient system designed for prediction of yield of tomatoes. This helps farmers remove the tedious, labor intensive, costly and time consuming task of manual data collections. Remote sensing is the potential candidate to overcome the problems of manual data collections. The remotely captured images/videos processed by computer vision algorithms provide meaningful and actionable information.

The use of technology in agriculture has led to the increase in productivity and decrease in losses for farmers. Thus this project aims to develop a method that helps farmers predict the yield of tomatoes, which would help them plan ahead in a better manner. This project going into the future can become instrumental in efficient agricultural practices. With the growth of technology in all the fields today, having accurate yield predictions of the tomato can be useful to sell tomatoes when the demand is high and tomato prices are rising, moreover it would help the farmers to estimate the financial requirements of the next growing season. The farmers can easily make appropriate deals if they know the yield of each grade and size of tomato.

## 9.1 Future Work

This project could be further extended in the following areas:

- It could be used for the detection and prediction of any agricultural crop. The model could be trained with the respective dataset of the crop.
- It could further be used for detecting diseases in fruits,vegetables or any crop. Detecting diseases at an earlier stage will help farmers to contain infection in crops.
- It can be used to detect and identify various stages of fruit or vegetable, which can be a boon in the field of agriculture as it will help farmers to identify a high quality crop at an early stage.

# REFERENCES

[1]Senthilnath, J, Dokania, Akanksha, Kandukuri, Manasa, Ramesh, KN, Anand, Gautham, Omkar, SN, 2016. Detection of tomatoes using spectral-spatial methods in remotely sensed RGB images captured by UAV. Biosyst. Eng. 146, 16–32.

[2]Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

[3]Haotian Zhang, Gaoang Wang, Zhichao Lei, Jenq-Neng Hwan, 2019. Eye in the Sky: Drone-Based Object Tracking and 3D Localization. In Proceedings of the 27th ACM International Conference on Multimedia, 899-907.

[4]Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar,´ Facebook AI Research (FAIR), 2017. Focal Loss for Dense Object Detection. The IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2980-2988.

[5]Deng, L., Mao, Z., Li, X., Hu, Z., Duan, F. and Yan, Y., 2018. UAV-based multispectral remote sensing for precision agriculture: A comparison between different cameras. *ISPRS journal of photogrammetry and remote sensing*, *146*, pp.124-136.

[6]Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[7]Wong, A., Shafiee, M.J., Li, F. and Chwyl, B., 2018, May. Tiny ssd: A tiny single-shot detection deep convolutional neural network for real-time embedded object detection. In *2018 15th Conference on Computer and Robot Vision (CRV)* (pp. 95-101). IEEE.

[8]]Cai, Z. and Vasconcelos, N., 2018. Cascade r-cnn: Delving into high quality object detection. In Proceedings of the IEEE conference on computer vision and pattern.