M.Sc. Thesis
Master of Science in Engineering

Department of Electrical and Computer Engineering
University of Thessaly

Design & implementation of a distributed data processing system for Wireless Sensor Networks

Author: Fotini Oikonomou

Supervisors

First	Second	Third
Associate Professor	Associate Professor	Associate Professor
Athanasios Korakis	Fotios Plessas	Gerasimos Potamianos

Volos 2019

M.Sc. Thesis
Master of Science in Engineering

Department of Electrical and Computer Engineering
University of Thessaly

Σχεδιασμός & υλοποίηση ενός κατανεμημένου συστήματος επεξεργασίας δεδομένων σε Ασύρματα Δίκτυα Αισθητήρων

Συγγραφέας: Οικονόμου Φωτεινή

Επιβλέποντες

Πρώτος	Δεύτερος	Τρίτος
Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής	Αναπληρωτής Καθηγητής
Κοράκης Αθανάσιος	Πλέσσας Φώτιος	Ποταμιάνος Γεράσιμος

Βόλος 2019

«Η άσκηση της Επιστήμης χρειάζεται σπάνιο κουράγιο…
Πιστεύω πως μοναδικός σκοπός της Επιστήμης είναι τούτος:
ν' αλαφρώσει τον μόχθο της ανθρώπινης ύπαρξης.
Αν οι επιστήμονες περιορισθούν να σωρεύουν γνώσεις πάνω σε γνώσεις,
μόνο και μόνο για τη χαρά της γνώσης,
η Επιστήμη δεν θα είναι πια παρά μια θλιβερή σακάτισσα.
Οι καινούργιες μηχανές σας δεν θα χρησιμεύουν παρά για καινούργια μαρτύρια.
Με τον καιρό, μπορεί ν' ανακαλύψετε ό,τι υπάρχει για ν' ανακαλυφθεί-κι ωστόσο,
η πρόοδός σας θα σας χωρίζει όλο και πιο πολύ απ' την Ανθρωπότητα.
Η άβυσσος ανάμεσα σ' εκείνην και σε σας μπορεί μια μέρα να γίνει
τόσο βαθιά που, στη χαρούμενη κραυγή σας για μια καινούργια κατάκτηση,
ν' αποκριθεί μια κραυγή φρίκης απ' όλη την οικουμένη»

Έργο του Bertolt Brecht, «Η ζωή του Γαλιλαίου»

Summary

The current thesis deals with a computer vision system that can be used in agriculture applications. We implemented and tested a vision system to recognize features especially for peaches, although it can be useful for others fruits, too. For this purpose, we used a camera, an accessible and affordable device, as a sensor to measure the variation of size, calculate the colour values and the number of fruits that can be detected on the brunches of the tree. The computer vision system is the back-end algorithms that run on server and take inputs from a camera ,which is connected on a breadboard that we call node. So, a node is these previous devices connected together, scattered across the field and creating a wireless sensors network.

Ultimately, the goal is to provide these results to agriculture consultants for advising the farmers about their farming experties and endure their farm based interests, so that they will experience the increase of their crops. Also, it's a useful map for people who are inexperience with computer based software relative to agriculture. Especially, peach cultivation need to optimize the productivity and protect crops. The vulnerability and the fragility of the peach crop with translates as the usual damage rating over 20% justifies the choice that we made for developing the system around the peach.

Computer vision is the ability to automatically understand any image or video based on visual elements and patterns. Technically, we used OpenCV library and Tensorflow framework. The images are captured in a BeagleBone Black breadboard and then are send over network to a server that makes image processing happen. Finally, the results are saved in a special format and databases.

The idea of this diploma thesis was developed under the guidance of my professor Dr Athanasios Korakis and my colleague Polychonis Symeonidis and the implementation of this project is one of many that take place in NITlab Laboratory.

Περίληψη

Στη παρούσα διπλωματική εργασία παρουσιάζεται ένα σύστημα μηχανικής όρασης το οποίο μπορεί να χρησιμοποιηθεί σε αγροτικές εφαρμογές. Σκοπός είναι να αξιοποιηθεί ένας αισθητήρας στον αγρό ο οποίος θα καταγράφει χρήσιμη πληροφορία από τον καρπό του φρούτου. Ενας τέτοιος αισθητήρας είναι η κάμερα η οποία θα λαμβάνει φωτογραφίες και μέσω του δικτύου ασύρματων αισθητήρων που θα εγκατασταθεί στον αγρό θα στέλνει τα δεδομένα σε ένα κεντρικό υπολογιστή για την επεξεργασία. Τα δεδομένα που θα μπορούμε να επεξεργαστούμε από μία φωτογραφία και βάσει αυτών θα μπορεί να αξιοποιηθεί καλύτερα η ανάπτυξη του δέντρου είναι, το μέγεθος του καρπού και η αυξομείωση του κατά την διάρκεια της ημέρας και η ωρίμανση του φρούτου από την καταγραφή των χρωματικών τιμών του καρπού.

Το σύστημα θα βασίζεται σε κάμερες που θα τοποθετηθούν πάνω στο δέντρο και θα προγραμματιστούν έτσι ώστε να καταγράφουν φωτογραφίες αυτόματα στην διάρκεια της μέρας. Επειτα, τα δεδομένα θα στέλνονται απευθείας σε μια κεντρική βάση δεδομένων και θα ταξινομούνται κατάλληλα ανάλογα το χωράφι, το δέντρο και την χρονική διάρκεια που λήφθηκε η φωτογραφία.

Ο στόχος είναι να συλλέξουμε δεδομένα και να τα δώσουμε ως πληροφορίες σε γεωπόνους ή άλλους ειδικούς συμβούλους ώστε να παρέχουν κατάλληλες οδηγίες στους αγρότες και να τους βοηθήσουν με τον καλύτερο δυνατό τρόπο να αυξήσουν την παραγωγή. Επίσης, αποτελεί έναν οδηγό για όσους επιθυμούν να ασχοληθούν με παρόμοιες εφαρμογές βλέποντας πως λειτουργεί ένα σύστημα οπτικής όρασης. Ο καρπός που επιλέχθηκε για να αναλυθεί περισσότερο είναι το ροδάκινο επειδή σε αυτού του είδους την καλλιέργεια παρατηρείται ιδιαίτερη ευαισθησία κατά την ανάπτυξη του.

Η μηχανική όραση είναι η ικανότητα να μπορούμε να κατανοούμε αυτό που εικονίζεται στην φωτογραφία και να αναγνωρίζουμε αυτόματα αντικείμενα και πληροφορίες. Από τεχνικής άποψης χρησιμοποιήθηκε η βιβλιοθήκη της OpenCV και το framework του Tensorflow. Οι εικόνες συλλέγονται από μία κάμερα που είναι συνδεδεμένη πάνω σε μία πλακέτα τύπου BeagleBone Black και στέλνονται μέσω δικτύου σε έναν server που αναλαμβάνει την επεξεργασίας της. Μετά την επεξεργασία εξάγονται τα χαρακτηριστικά και αποθηκεύονται τα αποτελέσματα σε ειδική μορφή στη βάση δεδομένων.

Η ιδέα, η καθοδήγηση και η επίβλεψη έγινε από τον καθηγητή κύριο Αθανάσιο Κοράκη και τον συμφοιτητή Πολυχρόνη Συμεωνίδη και η υλοποίηση αποτελεί ένα κομμάτι από τις γενικότερες δραστηριότητες του Εργαστηρίου Δικτύων του Πανεπιστημίου Θεσσαλίας, το NITlab.

Acknowledgements

With the completion of the present master theses I would like to express my sincere gratitude to my supervisors, Dr. Athanasios Korakis, Dr. Potamianos and Dr. Plessas for their guidance and support during my postgraduate studies. Moreover, special thanks goes to my colleague Polychronis for his critical help during this thesis.

Finally, I must express my very profound gratitude to my parents and life coaches **Konstantino** and **Vagia**, my beautiful sister **Vanesa** and my amazing friends for their support all of these years. This accomplishment would not have been possible without them.

Thank you.

Contents

List of Figures

Acronyms

ANN Artificial Neural Network.

BG Background.

CNNs Convolution Neural Networks.

ConvNets Convolution Neural Networks.

CPU Central Processing Unit.

FCs Fully Connected Layers.

FG Foreground.

FPN Feature Pyramid Network.

GPU Graphics Processing Unit.

ML Machine Learning.

MLP Multi Layer Perceptron.

PDE Partial Differential Equation.

ReLU Rectified Linear Unit.

ResNet Residual Networks.

RGB Red Green Blue.

ROI Region of Interest.

RPN Region Proposal Network.

WSN Wireless Sensor Network.

CHAPTER 1

Introduction

1.1 Image Processing in Precision Agriculture

Farmers need the productivity of the agriculture to be increased and build a management plan to save both money and time. Image processing holds an effective set of tools for the crops analysis and provide automated solutions with beneficial expert knowledge and advice. The advantages in image processing functions are that are combined with communication systems and it can transform the crop picture for farmers. The amount of image processing applications in precise agriculture is growing steadily with the availability of higher quality measurements coupled with modern algorithms and increased possibility to fuse multiple sources of information from sensors positioned in fields. Precision agriculture is one of many modern farming practices that make production more efficient. With precision agriculture, farmers work better, not harder.

The purpose of this thesis is to provide a precision agriculture application to farmers. They apply different techniques over the tree, as irrigation and pruning techniques, trying to find best conditions to grow trees and fruits as best as they can. For this reason, is a challenge to create a smart system which agriculture consultants will make experiments and take metrics for their projects. Some metrics that consultants want are size of fruit with accuracy in millimeters or the variation of size during the day. Moreover, they want to know colors in the area of fruits and how many of them are on branches. With these metrics, an agriculturalist obtain knowledge and use it to inform farmers about the best cultivation conditions.

Nowadays, there are some solutions that may be used in experiments. More specifically, there is an equipment that is applied over the fruit and measure the diameter. This tool is called fruit dendrometer and is a version for measurements on circular fruiting bodies. The fruit in the measuring frame is fixed without affecting its growth. First of all, a disadvantage for this method is the price which is quit expensive and its impossible to buy enough of them. Farmers need to buy the dendrometer equipment not only for fruits over one tree but for all trees across the fields. As well, it is not a practical equipment to install over branches.

The other significant metric is fruit's colors values. Color characterizes well their maturity and presence of defects. For a lot of products like peaches and others are important both the external and the fruit flesh color. A colorimeter can measure the absorbency of light waves. During color measurement the change in the intensity

(a) Dendrometer Equipment **(b)** Dendrometer Over Tree

Figure 1.1: Dendrometer.

of electromagnetic radiation in the visible wavelength region of the spectrum after transmitting or reflecting by an object or solution is measured.

Its important to measure color values during the production of fruit. Tools, such as colorimeters or spectrophotometer, demand to cut fruit and put it on device and then take measurements or it is demanded a person who must go in field and collect the data. Colors help farmers to know the phase of fruit growth. Furthermore, colors are affected from sun and water and it is possible to combine phase of growth with other parameters, such as irrigation and pruning techniques.

For all that, we recommend a system that contain cameras which are spread evenly over the field in a wireless sensor network (WSN). The WSN has the responsibility

(a) Colorimeter

Figure 1.2: Color Analyzer.

to capture images and send them in central server computer over the network. The server execute image processing task and extract the features that are classified in a database.

This method provides a solution where benefits outweigh disadvantages which are described in previous paragraph. First of all, is an accessible and affordable equipment that we can install it in big quantities in fields. Cameras which will be described analytical after, are small and flexible devices and we can apply them due to limited space availability on trees.

Color values are detected from camera's sensor. These sensors have small individual detectors, usually several million on a single sensor, which respond to light that falls upon them. We will program the camera to capture images automatically. We will send remotely a command to capture an image and then will send it back in the receiver. The advantage is that we do not need any extra device or a person to handle it but hole system is scheduled in an autonomous routine. Furthermore, we combine size and color features in one application with our camera node. During the day, size and color information are given to advisers for analysis. An additional advantage, is the wireless communications benefits that involves the transmission of information over a distance. A wireless network is a convenient and flexible system that ensures the remotely control of our devices where are spread in field. Finally, camera is a hardware which is connected easily over a breadboard which is the base of a WSN system.

To sum up, the quality of crops is a highly recommended concern of farmers. Automated quality analysis of food products is a process based on safety standards. Image processing is an accurate and reliable method for sorting and grading products as fruits, grains, bakery products etc, characterized by color, size and shape. We developed a monitoring system to advise farmers by combining these features.

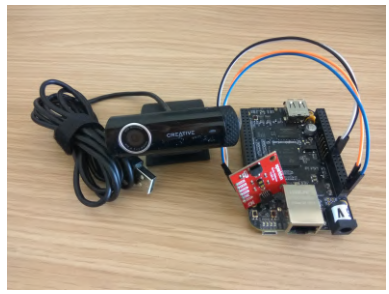


Figure 1.3: Our equipment: BeagleBone Black - Camera - Sensor.

1.2 Thesis Organization

In second chapter, we will present some technical aspects which are based on system development. First of all, we will describe the image processing procedure and computer vision aspects with goal to answer questions like the difference between them, how extract features and what kind of information is it. Then, image processing algorithms are combined with machine learning techniques to become more powerful and useful. Machine learning is a branch of Computer Science that can function and power sophisticated systems that rely on ML algorithms. A step below is an introduction to neural networks and then to deep learning models which also are neural networks and is a subset of machine learning that uses a model of computing that's very much inspired by the structure of the brain.

After, we will investigate object detection models which are deep learning models and detecting instances of a fruit class. It is an exceedingly powerful technique to recognize an object and focus in a part of an image and not the hole one. We will combine these parts of image with other techniques and algorithms to separate background and foreground. So, the other significant procedure that we will implement is background removal. In last paragraphs included information about the implementation environment.

Chapter 3 is an important topic which will be described implementation details about application structure, i.e code analysis of the system and the results after making experiments. At the beginning, we reported an introduction of the program and the way of thinking behind it. The paragraph two is named pre-process functionality and is the main task of our code in application. The results are bounding boxes which included fruits. The models that will be tested for this purpose are Faster R-CNN model and Mask R-CNN model.

Then, we applied algorithms for the segmentation of background and foreground in images. Before the end of this paragraph, we will combine methods to optimize the results. After all of them, the penultimate paragraph will give the final output data which we described at the beginning and are the size, colors and the number of fruits. At the end, we will present the results of experiments in real time conditions. We will suggest some ideas to filter images and avoid some unwanted frames.

In final Chapter, we will draw a conclusion about the hole procedure that will be described a detailed summary about advantages, disadvantages and difficulties. Especially, in chapter 4 will be presented the future work and some interesting ideas to extract new features which could be implemented in new versions of our system.

CHAPTER 2

Technical Aspects

2.1 Image Processing

2.1.1 Digital Image Processing

Image processing is a method to perform some operations on an image on it, in order to get an enhanced image or to extract some useful information from it. There are two types of methods used for image processing namely, analogue and digital image processing. Digital image processing means processing in digital image by means of a digital computer. Image processing follow three basic steps:

1. Import an image with image acquisition tools
2. Analyzing and manipulating image
3. Report which is based on analysis

What is an image?

An image is defined as a two-dimensional function, χ (x,y), where x and y are spatial coordinates, and the amplitude of χ at any pair of coordinates (x,y) is called the intensity of that image at that point. When x,y, and amplitude values of χ are finite, we call it a digital image. So, an image can be defined by a two-dimensional array specifically arranged in rows and columns. Digital Image is composed of a finite number of elements, each of which elements have a particular value at a particular location. These elements are referred to as picture elements, image elements, and pixels. A pixel is most widely used to denote the elements of a digital image.

$$\chi(x,y) = \begin{vmatrix} \chi(0,0) & \chi(0,1) & . & . & \chi(0,N-1) \\ \chi(1,0) & \chi(1,1) & . & . & \chi(1,N-1) \\ . & . & . & . & . \\ . & . & . & . & . \\ \chi(M-1,0) & \chi(M-1,1) & . & . & \chi(M-1,N-1) \end{vmatrix}.$$

Table 2.1: Images are represented in rows and columns.

The image is a two dimensional array and it results from signals. In physical world, any quantity measurable through time over space or any higher dimension can be taken as a signal. A signal is a mathematical function, and it conveys some information. Signals can include sound, light, images, environmental signals such as temperature, humidity, and pollution, biological signals, telecommunication transmission signals, and many others. A signal can be one dimensional or two dimensional or higher dimensional signal. One dimensional signal is a signal that is measured over time and an example is a voice signal. The two dimensional signals are those that are measured over some other physical quantities. An example of two dimensional signal is the digital image.

Since anything that conveys information or broadcast a message in physical world between two observers is a signal. That includes speech or (human voice) or an image as a signal. Since when we speak , our voice is converted to a sound wave/signal and transformed with respect to the time to person we are speaking to. Not only this , but the way a digital camera works, as while acquiring an image from a digital camera involves transfer of a signal from one part of the system to the other.

Since capturing an image from a camera is a physical process. The sunlight is used as a source of energy. A sensor array is used for the acquisition of the image. So when the sunlight falls upon the object, then the amount of light reflected by that object is sensed by the sensors, and a continuous voltage signal is generated by the amount of sensed data. In order to create a digital image , we need to convert this data into a digital form. This involves sampling and quantization. The result of sampling and quantization results in an two dimensional array or matrix of numbers which are a digital image.

Types of an image

- The binary image contain only two pixel elements i.e 0 and 1,where 0 refers to black and 1 refers to white. This image is also known as Monochrome.

- The black and white image which consist of only black and white color.

- Images with 8 bit Color Format is the most famous image format.It has 256 different shades of colors in it and commonly known as gray scale Image. In this format, 0 stands for Black, and 255 stands for white, and 127 stands for gray.

- Images with 16 bit Color Format is a color image. It has 65,536 different colors in it. It is also known as High Color Format. In this format the distribution of color is not as same as gray scale image. A 16 bit format is actually divided into three further formats which are Red, Green and Blue, and is called RGB format.

2.2 Computer Vision

Computer vision is an interdisciplinary scientific field that deals with how computers can be made to gain high level understanding from digital images or vision. The terms of computer vision and image processing are used almost interchangeably in many contexts but they are not the same thing. An image processing algorithm can transform images in many ways such as smoothing, sharpening, changing the brightness and contrast, highlighting the edges and so on. Computer vision, on the other hand, focuses on making sense of what a machine sees. A computer vision system inputs an image and outputs task-specific knowledge, such as object labels and coordinates.

Differences	
Image Processing	Computer Vision
Image-To-Image *e.g* edge detection, compression	Image-To-Symbols *e.g* object tracking, face recognition

Table 2.2: Image Processing vs Computer Vision.

Computer vision and image processing work together in many cases. Many computer vision systems rely on image processing algorithms. For example, computer vision systems rarely use raw imaging data that comes directly from a sensor. Instead, they use images that are processed by an image signal processor. The opposite is also possible. Image processing algorithms and methods use computer vision to enhance images. For examples, face filters where use a lot of application use computer vision techniques to detect faces and then apply smoothing filters such as bilateral filter selectively.

A key similar characteristic of two of them is the use of machine learning. We can build model that recognize objects on pictures. Also many advanced image processing methods use machine learning models to transform images to accomplish a variety of tasks, such as tuning an image for optimal perceptual image quality. Furthermore, there are models where use pixel to pixel transformations and the line between these two fields is blurred. These models doing some sort of image processing. Also these transformations involves image understanding trying to understand what's in the input.

2.3 Artificial Intelligent

Artificial Intelligent is a branch of computer science that aims to create intelligent machines. Machines can often act and react like humans only if they have abundant information relating to the world. Artificial intelligence must have access to objects,

categories, properties and relations between all of them to implement knowledge engineering. Initiating common sense, reasoning and problem-solving power in machines is a difficult and tedious task. Machine learning is also a core part of AI. Machines can often react like humans only if they have abundant information relating to world. Artificial intelligence must have access to objects, categories, properties and relations between all of them to implement knowledge engineering. Initiating common sense, reasoning and problem-solving power in machines is a difficult and tedious task.

Particular applications of AI include expert systems, speech recognition and machine vision. The AI systems are more generic (rather than specific), have the ability to "think" and are more flexible. Many tools are used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics. The AI field draws upon computer science, mathematics, psychology, linguistics, philosophy, neuro-science, artificial psychology and many others.

AI can be categorized as either weak or strong. Weak AI, also known as narrow AI, is an AI system that is designed and trained for a particular task. Virtual personal assistants, such as Apple's Siri, are a form of weak AI. Strong AI, also known as artificial general intelligence, is an AI system with generalized human cognitive abilities. When presented with an unfamiliar task, a strong AI system is able to find a solution without human intervention.

2.4 Machine Learning

Machine Learning (ML) can be explained as automating and improving the learning process of computers based on their experiences without being actually programmed i.e. without any human assistance. The process starts with feeding good quality data and then training our machines(computers) by building machine learning models using the data and different algorithms. The choice of algorithms depends on what type of data do we have and what kind of task we are trying to automate.

Most machine learning problems belong to one of the following three main categories:

In supervised learning, each data point is labeled or associated with a category or value of interest. An example of a categorical label is assigning an image as either a cat or dog. An example of a value label is the sale price associated with a used car. The goal of supervised learning is to study many labeled examples like these (called training data) in order to make predictions about future data points (called test data). These predictions come in two flavors, such as identifying new photos with the correct animal (called a classification problem) or assigning accurate sale prices to other used cars (called a regression problem).

In unsupervised learning, data points have no labels associated with them. Instead, the goal of an unsupervised learning algorithm is to organize the data in some way

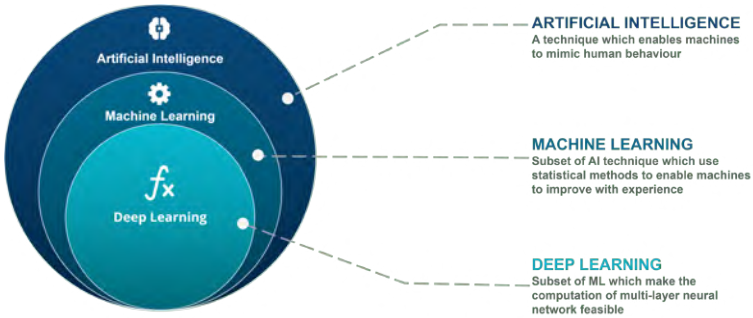


Figure 2.1: Artificial Intelligent vs Machine Learning vs Deep Learning, source: edureka.

or to describe its structure. This can mean grouping them into clusters or finding different ways of looking at complex data so that they appear simpler.

In reinforcement learning, the algorithm gets to choose an action in response to each data point. It is a common approach in robotics, where the set of sensor readings at one point in time is a data point and the algorithm must choose the robot's next action. It's also a natural fit for Internet of Things applications, where the learning algorithm receives a reward signal at a short time into the future, indicating how good the decision was. Based on this, the algorithm modifies its strategy in order to achieve the highest reward.

Differences between AI and ML

ML and AI are related, but they aren't the same, and they aren't necessarily suited to the same tasks. You can take your business to the next level by knowing when to choose ML or AI. There is a misconception that Artificial Intelligence is a system, but it is not a system .AI is implemented in the system. There can be so many definition of AI, one definition can be "It is the study of how to train the computers so that computers can do things which at present human can do better."Therefore It is an intelligence where we want to add all the capabilities to machine that human contain.

Machine learning, a fundamental concept of AI research since the field's inception, is the study of computer algorithms that improve automatically through experience. The mathematical analysis of machine learning algorithms and their performance is a branch of theoretical computer science known as a computational learning theory.

Machine Learning is the learning in which machine can learn by its own without being explicitly programmed. It is an application of AI that provide system the ability to automatically learn and improve from experience. Machine Learning is the learning in which machine can learn by its own without being explicitly programmed. It is an application of AI that provide system the ability to automatically learn and

improve from experience.

To conclude, I referred a general difference between of AI over ML are the below. The goal of AI is to simulate natural intelligence to solve complex problem, even thought the goal of machine learning is to to learn from data on certain task to maximize the performance of machine on this task.

2.5 Neural Networks

An Artificial Neural Network (ANN) is a computational model that is inspired by the way biological neural networks in the human brain process information. Artificial Neural Networks have generated a lot of excitement in Machine Learning research and industry, thanks to many breakthrough results in speech recognition, computer vision and text processing.

A Single Neuron

The basic unit of computation in a neural network is the neuron, often called a node or unit. It receives input from some other nodes, or from an external source and computes an output. Each input has an associated weight, which is assigned on the basis of its relative importance to other inputs. The node applies a function f to the weighted sum of its inputs.

Feedforward Neural Network

The feedforward neural network was the first and simplest type of artificial neural network. It contains multiple neurons (nodes) arranged in layers. Nodes from

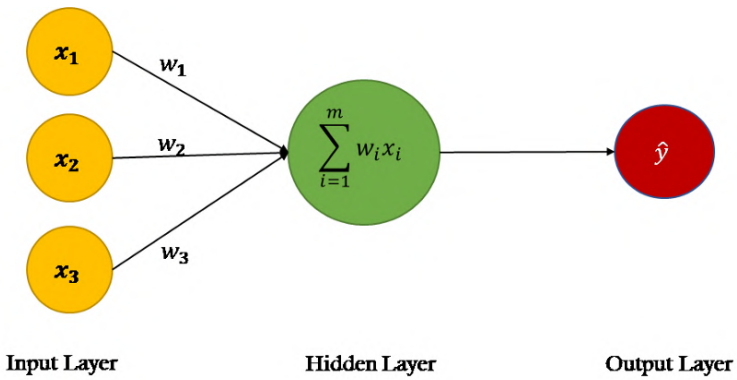


Figure 2.2: Basic Unit of a Artificial Neural Network — Artificial Neuron, source: towardsdatascience.

adjacent layers have connections of edges between them. All these connections have weights associated with them.

A feedforward neural network can consist of thee types of nodes:

Input Nodes: The inputs nodes provide information from outside world to the network and are together referred to as the "Input Layer". No computation is performed in any of the Input Nodes and they pass on information to the hidden nodes.

Hidden Nodes: The Hidden Nodes have no direct connection with the outside world. They perform computations and transfer information from the input nodes to the output nodes. A collection of hidden nodes forms a "Hidden Layer". While a feedforward network will only have a single input layer and a single output layer, it can have a zero or multiple Hidden Layers.

Output Nodes: The Output nodes are collectively referred to as the "Output Layer" and are responsible for computations and transferring information from the network to the outside world.

In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes and to the output nodes. There no cycles or loops in the network. Two examples of feedforward networks are:
Single Layer Perceptron: This is simplest feedforward neural network and does not contain any hidden layer.
Multi Layer Perceptron: A Multi Layer Perceptron has one or more hidden layers.

Multi Layer Perceptron (MLP)
A Multi Layer Perceptron contains one or more hidden layers apart from input

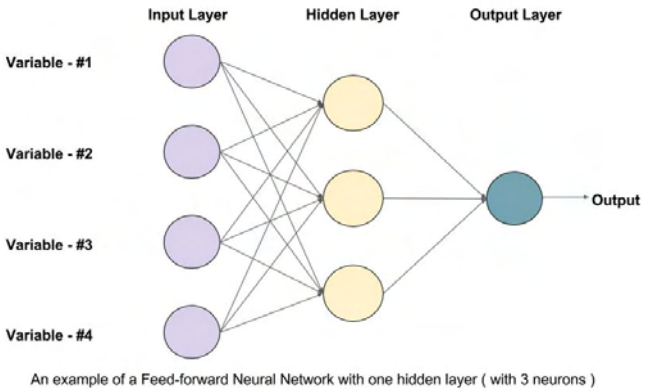


Figure 2.3: Feedforward Neural Network, source: learnopencv.

and output layer. While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non-linear functions.

Training a MLP: The Back - Propagation Algorithm

The process by which a Multi Layer Perceptron learns is called the Back Propagation algorithm. Backward Propagation of Errors, often abbreviated as BackProp is one of the several ways in which an artificial neural network can be trained. It is a supervised training sheme, which means, it learns from labeled training data. There is a supervisor to guide its learning. BackProp is like "learning from mistakes". The supervisor corrects the ANN whenever it makes mistakes.

An ANN consists of nodes in different layers, input layer, intermediate hidden layers and the output layer. The connections between nodes of adjacent layers have "weights" associated with them. The goal of learning is to assign correct weights for these edges. Given an input vector these weights determine what the output vector is. In supervised learning, the training set is labeled. This means, for some given inputs we know the desired/expected output (label).

2.6 Deep Learning

2.6.1 Introduction

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans, learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

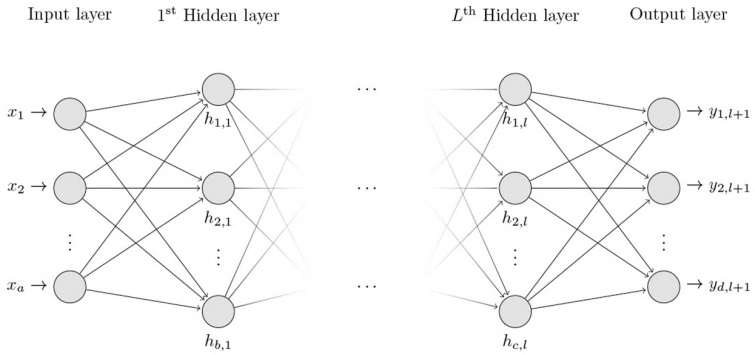


Figure 2.4: A Multi Layer Perceptron, source: journalsplos.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state of the art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

What's the Difference Between Machine Learning and Deep Learning?

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the image. With a deep learning workflow, relevant features are automatically extracted from images. In addition, deep learning performs end-to-end learning, where a network is given raw data and a task to perform, such as classification, and it learns how to do this automatically. Another key difference is deep learning algorithms scale with data, whereas shallow learning converges. Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network. A key advantage of deep learning networks is that they often continue to improve as the size of your data increases.

2.6.2 Convolutional Neural Networks

Convolution Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. ConvNets have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self driving cars. ConvNets are an important tool for most machine learning practitioners today. CNNs is used at vision, audio, and even natural language processing applications which researchers and engineers have built amazing applications using CNNs.

The LeNet Architecture

Lenet was one of the very first convolutional neural networks which help on Deep Learning. This pioneering work by Yann LeCun was named LeNet5 after many previous successful recognition tasks such as reading zip codes, digits, etc. There have

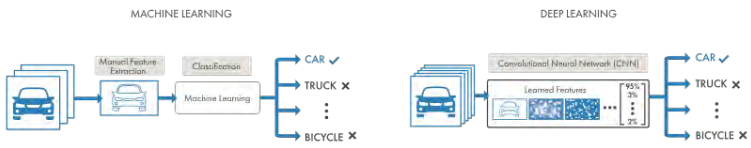


Figure 2.5: Comparing a machine learning approach to categorizing vehicles (left) with deep learning (right), source: mathworks.

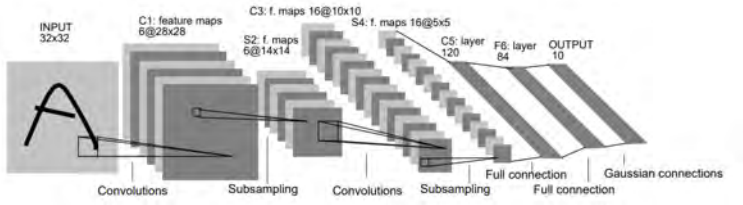


Figure 2.6: Original Image published in 1998, Architecture of LeNet-5, a Convolutional Neural Network for digits recognition.

been several new architectures proposed in the recent years which are improvements over the LeNet, but they all use the main concepts from the LeNet.

There are four main operations in the ConvNet. These operations are the basic building blocks of every Convolutional Neural Network.

- Convolution
- Non Linearity (ReLU)
- Pooling or Sub Sampling
- Classification (Fully Connected Layer)

1. Convolution:

ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the spatial relationship between pixels by learning image features using small squares of input data.

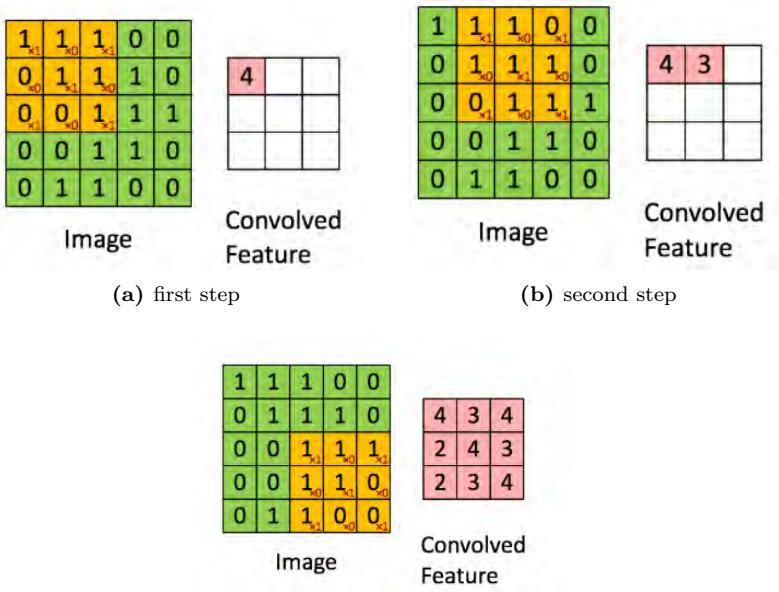
To produce a convolved feature, we slide the 3x3 Matrix over the image array (5x5 matrix) and for every position, compute element wise multiplication (between the two matrices) and add the multiplication outputs to get the final integer which forms a single element of the output matrix.

The 3x3 martix is called a filter or kernel or feature detector and the matrix formed by sliding the filter over the image and computing the dot product is called Convolved Feature or Action Map or Feature Map. That filters acts as feature detectors from original input image.

A CNN learns the values of these filters on its own during the training process, although we still need to specify parameters such number of filters, filter size, architecture of the network etc before the training process. The more number of filter we have, the more image features get extracted and the better our network becomes at recognizing patterns in unseen images.

The size of the Feature Map (Convolved Feature) is controlled by three parameters:

- Depth: Depth corresponds to the number of filters we use for the convolution operation.



(a) first step **(b)** second step

(c) last step

Figure 2.7: Convolution operation, the output matrix is called Convolved Feature or Feature Map.

- Stride: Stride is the number of pixels by which we slide our filter matrix over input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.
- Zero-padding: It is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is called wide convolution, and not using it wound be a narrow convolution.

2. Non Linearity (ReLU):

An addition operation called ReLU has been used after every convolution operation. ReLU stands for Rectified Linear Unit and is non-linear operation. ReLU is an element operation where applied per pixel and replaces all negative pixel values in the feature map by zero. Other non linear functions such as tanh or sigmoid can also be used instead of ReLU, but ReLU has been found to perform better in most situations.

3. Pooling Layer:

The Pooling layer can be seen between Convolution layers in a CNN architecture. This layer basically reduces the number of parameters and computation in the network, controlling overfitting by progressively reducing the spatial size of the network.

Spatial Pooling, also called subsampling or downsampling, reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types, Max, Average, Sum etc. In the network, pooling operation is applied separately to each feature map. The function of Pooling is to progressively reduce the spatial size of the input representation.

4. Fully Connected Layer:

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer, other classifiers like SVM can also be used. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer.

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

2.7 Object Detection

2.7.1 Introduction

Computer vision is an interdisciplinary field that has been gaining huge amounts of traction in the recent years (since CNN) and self-driving cars have taken center stage. Another integral part of computer vision is object detection. Object detection aids in pose estimation, vehicle detection, surveillance etc. The difference between object detection algorithms and classification algorithms is that in detection algorithms, we try to draw a bounding box around the object of interest to locate it within the image. Also, it could be draw not only one bounding box in an object detection case, there could be many bounding boxes representing different objects of interest within the image and you would not know how many beforehand.

2.7.2 Faster R-CNN model

Faster R-CNN has two networks, the Region Proposal Network (RPN) for generating region proposals and a network using these proposals to detect objects. The main different here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the object detection network. Briefly, RPN ranks region boxes, called anchors, and proposes

the ones most likely containing objects.

Region Proposal Network

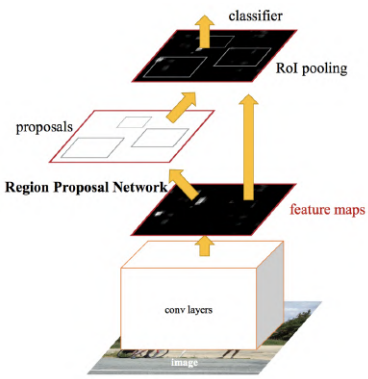
The output of a region proposal network (RPN) is a bunch of boxes/proposals that will be examined by a classifier and regressor to eventually check the occurrence of objects. To be more precise, RPN predicts the possibility of an anchor being background or foreground, and refine the anchor. Anchors play an important role in Faster R-CNN. An anchor is a box. In the default configuration of Faster R-CNN, there are 9 anchors at a position of an image.

The Classifier of Background and Foreground

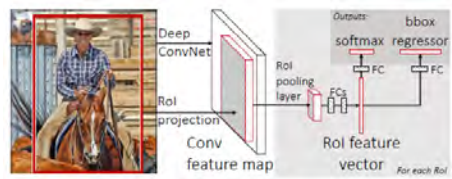
The first step of training a classifier is make a training dataset. The training data is the anchors we get from the above process and the ground-truth boxes. The problem which need to solve is how use the ground-truth boxes to label the anchors. The basic idea here is that we want to label the anchors having the higher overlaps with ground-truth boxes as foreground, the ones with lower overlaps as background. Apparently, it needs some tweaks and compromise to separate foreground and background. Now we have labels for the anchors.

Another thing you may pay attention to is receptive field if you want to re-use a trained network as the CNNs in the process. Make sure the receptive fields of every position on the feature map cover all the anchors it represents. Otherwise the feature vectors of anchors won't have enough information to make predictions.

In the architecture of Overfeat, it only uses non-overlapping convolutional and pooling filters to make sure every position in the feature map cover its own receptive



(a) Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.



(b) Fast R-CNN

Figure 2.8: Faster R-CNN.

field without overlapping others. In Faster R-CNN, receptive fields of different anchors often overlap each other, as you can from the above graph. It leaves the RPN to be position-aware.

The Regressor of Bounding Box

If you follow the process of labelling anchors, you can also pick out the anchors based on the similar criteria for the regressor to refine. One point here is that anchors labelled as background shouldn't included in the regression, as we don't have ground-truth boxes for them. The depth of feature map is 32 (9 anchors x 4 positions). The paper uses smooth-L1 loss on the position (x ,y) of top-left the box, and the logarithm of the heights and widths, which is as the same as in Fast R-CNN.

Detection Network

Except the RPN, the remaining part is similar to the Fast R-CNN. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI, softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

ROI Pooling

After RPN, we get proposed regions with different sizes. Different sized regions means different sized CNN feature maps. It's not easy to make an efficient structure to work on features with different sizes. Region of Interest Pooling can simplify the problem by reducing the feature maps into the same size. Unlike Max-Pooling which has a fix size, ROI Pooling splits the input feature map into a fixed number (e.g k) of roughly equal regions, and then apply Max-Pooling on every region. Therefore the output of ROI Pooling is always k regardless the size of input. With the fixed ROI Pooling outputs as inputs, we have lots of choices for the architecture of the final classifier and regressor.

2.7.3 Mask R-CNN model

What is Mask R-CNN?

Mask R-CNN is a regional convolutional neural network and is a two stage framework. The first stage scans the image and generates proposals, areas likely to contain an object. And the other stage classifies the proposals and generates bounding boxes and masks. Mask R-CNN introduced to extend its predecessor Faster R-CNN which we described in previous paragraph. Faster R-CNN is a popular framework for object detection, and Mask R-CNN extends it with instance segmentation, among other things.

What is R-CNN?

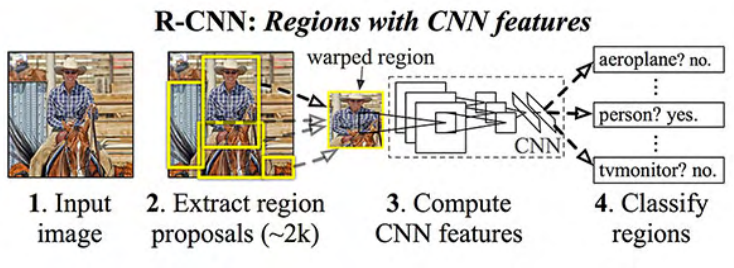


Figure 2.9: Instance Segmentation with Mask R-CNN.

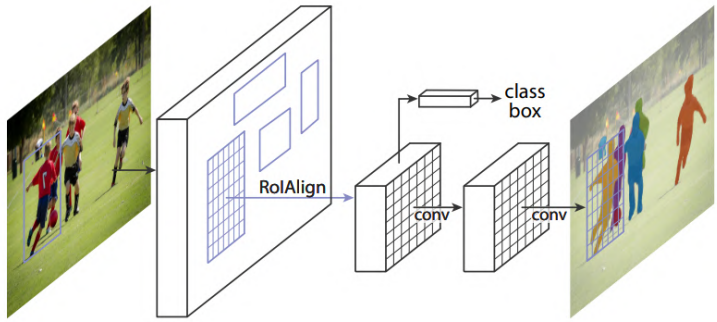


Figure 2.10: Mask R-CNN.

The R-CNN algorithm is a four-step process:
Step 1: Input an image to the network.
Step 2: Extract region proposals (i.e., regions of an image that potentially contain objects) using an algorithm such as Selective Search.
Step 3: Use transfer learning, specifically feature extraction, to compute features for each proposal (which is effectively an ROI) using the pre-trained CNN.
Step 4: Classify each proposal using the extracted features with a Support Vector Machine (SVM).

To improve upon the original R-CNN, Girshick et al. published the Fast R-CNN algorithm. Similar to the original R-CNN, Fast R-CNN still utilizes Selective Search to obtain region proposals; however, the novel contribution from the paper was Region of Interest (ROI) Pooling module. ROI Pooling works by extracting a fixed-size window from the feature map and using these features to obtain the final class label and bounding box. The Faster R-CNN paper introduced the Region Proposal Network (RPN) that bakes region proposal directly into the architecture, alleviating the need for the Selective Search algorithm.

At a high level, Mask R-CNN consists of these modules:

Backbone

This is a standard convolutional neural network (typically, ResNet50 or ResNet101) that serves as a feature extractor. The early layers detect low level features (edges and corners), and later layers successively detect higher level features (car, person, sky). Passing through the backbone network, the image is converted from 1024x1024px x 3 (RGB) to a feature map of shape 32x32x2048. This feature map becomes the input for the following stages.

Feature Pyramid Network

While the backbone described above works great, it can be improved upon. The Feature Pyramid Network (FPN) was introduced by the same authors of Mask R-CNN as an extension that can better represent objects at multiple scales. FPN improves the standard feature extraction pyramid by adding a second pyramid that takes the high level features from the first pyramid and passes them down to lower layers. By doing so, it allows features at every level to have access to both, lower and higher level features.

Region Proposal Network (RPN):

The RPN is a lightweight neural network that scans the image in a sliding-window fashion and finds areas that contain objects. The regions that the RPN scans over are called anchors. Which are boxes distributed over the image area, as show on the left. This is a simplified view, though. In practice, there are about 200K anchors of different sizes and aspect ratios, and they overlap to cover as much of the image as possible.

The RPN scan many anchors pretty fast. The sliding window is handled by the convolutional nature of the RPN, which allows it to scan all regions in parallel (on a GPU). Further, the RPN does not scan over the image directly (even though we draw the anchors on the image for illustration). Instead, the RPN scans over the backbone feature map. This allows the RPN to reuse the extracted features efficiently and avoid duplicate calculations. With these optimizations, the RPN runs in about 10 ms according to the Faster RCNN paper that introduced it. In Mask RCNN we typically use larger images and more anchors, so it might take a bit longer.

The RPN generates two outputs for each anchor:

Anchor Class: One of two classes: foreground or background. The FG class implies that there is likely an object in that box.

Bounding Box Refinement: A foreground anchor (also called positive anchor) might not be centered perfectly over the object. So the RPN estimates a delta (% change in x, y, width, height) to refine the anchor box to fit the object better.

Using the RPN predictions, we pick the top anchors that are likely to contain objects and refine their location and size. If several anchors overlap too much, we keep the one with the highest foreground score and discard the rest (referred to as Non-max Suppression). After that we have the final proposals (regions of interest)

that we pass to the next stage.

ROI Classifier & Bounding Box Regressor:
This stage runs on the regions of interest (ROIs) proposed by the RPN. And just like the RPN, it generates two outputs for each ROI:

Class: The class of the object in the ROI. Unlike the RPN, which has two classes (FG/BG), this network is deeper and has the capacity to classify regions to specific classes (person, car, chair, ...etc.). It can also generate a background class, which causes the ROI to be discarded.

Bounding Box Refinement: Very similar to how it's done in the RPN, and its purpose is to further refine the location and size of the bounding box to encapsulate the object.

ROI Pooling: Classifiers don't handle variable input size very well. They typically require a fixed input size. But, due to the bounding box refinement step in the RPN, the ROI boxes can have different sizes. That's where ROI Pooling comes into play. ROI pooling refers to cropping a part of a feature map and resizing it to a fixed size. It's similar in principle to cropping part of an image and then resizing it (but there are differences in implementation details). The authors of Mask R-CNN suggest a method they named ROIAlign, in which they sample the feature map at different points and apply a bilinear interpolation.

Segmentation Masks
If you stop at the end of the last section then you have a Faster R-CNN framework for object detection. The mask network is the addition that the Mask R-CNN paper introduced. The mask branch is a convolutional network that takes the positive regions selected by the ROI classifier and generates masks for them. The generated masks are low resolution: 28x28 pixels. But they are soft masks, represented by float numbers, so they hold more details than binary masks. The small mask size helps keep the mask branch light. During training, we scale down the ground-truth masks to 28x28 to compute the loss, and during inferencing we scale up the predicted masks to the size of the ROI bounding box and that gives us the final masks, one per object.

2.8 Background removal

2.8.1 Introduction

Background removal is a task that is quite easy to do manually, or semi manually (Photoshop, and even Power Point has such tools) if you use some kind of a "marker" and edge detection. However, fully automated background removal is quite a challenging task. This process of image background removal services is a simple enough process and as the name suggests, it largely depends on removing the isolated central image's background from the object. The background is removed but the central

image of the product is retained.

Semantic Segmentation

When studding deep learning and computer vision tasks which resemble ours, it is easy to see that one best option is to try semantic segmentation task. Semantic segmentation is a well known computer vision task, one of the top three, along with classification and object detection. Mask R-CNN is a semantic segmentation model which we described in previous paragraph and we will show the results later. We want this architect to remove background. The segmentation is actually a classification task, in the sense of classifying every pixel to a class. Unlike image classification or detection, segmentation model really shows some "understanding" of the images, in not only saying "there is a cat in this image", but pointing where and what is the cat, on a pixel level.

In this section, we will describe an other algorithm with better performance than Semantic Segmentation. It is not fully automated as Mask R-CNN but we will suggest how to be automatic. This algorithm is called GrabCut and satisfies our criteria.

2.8.2 GrabCut Algorithm

How GrabCut algorithm works?

GrabCut is a segmentation technique that uses graph cuts to perform segmentation. Like most segmentation techniques GrabCut uses information encapsulated in the image. Most segmentation techniques make use of either edge information or region information in the image. GrabCut makes use of both edge and region information. This information is used to create an energy function which, when minimized, produces the best segmentation.

In order to perform segmentation a graph is built, where nodes in the graph represent pixels in the image. In addition two special nodes are also created. These are the Sink and Source nodes. Every pixel node in the graph is connected to the Source and Sink node. The Source node represents the foreground of the image, and the Sink node the background. In order to segment the image the Source and Sink nodes must be separated.

The energy function is incorporated into the graph as weights between pixel nodes and weights between pixel and Source or Sink nodes. Weights between pixel nodes are determined by edge information in the image. Thus a strong indication of an edge between two pixels (a large difference in pixel color) results in a very small weight between two pixel nodes. The region information determines the weights between pixels nodes and the Source and Sink nodes. These weights are calculated by determining the probability of the pixel node being part of the background or foreground region.

In order for foreground and background regions to be created, some pixels in the image need to be labeled before segmentation as either foreground or background.

This is referred to as the clue marking stage. Any pixels that are labeled during this stage are set as hard constraints. This means that during the segmentation process, hard labeled pixels cannot change their labeling. A Min-cut/Max-Flow algorithm is used to segment the graph. This algorithm determines the minimum cost cut that will separate the Source and Sink nodes. The cost of the cut is determined by the sum of all the weights of the links that are cut. Once the Source and Sink nodes are separated, all pixel nodes connected to the Source node become part of the foreground, and the rest become part of the background.

The figure below shows the process of labelling an image, creating the graph, and then segmenting the graph to produce a segmentation of the image.

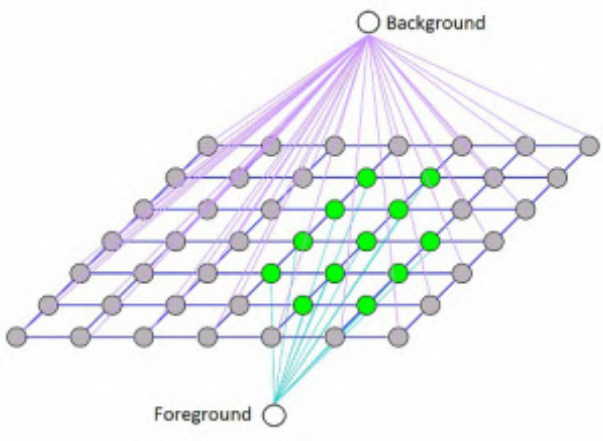


Figure 2.11: Segmenting Graph in GrabCut, source: opencv.

2.9 Implementation environment

2.9.1 Tensorflow framework

TensorFlow is an open-source software library. TensorFlow was originally developed by researchers and engineers working on the Google Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

TensorFlow is basically a software library for numerical computation using data flow graphs where:
1.Nodes in the graph represent mathematical operations.
2.Edges in the graph represent the multidimensional data arrays (called tensors) communicated between them.

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

TensorFlow allows developers to create dataflow graphs—structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor.

Also, it provides all of this for the programmer by way of the Python language. Python is easy to learn and work with, and provides convenient ways to express how high-level abstractions can be coupled together. Nodes and tensors in TensorFlow are Python objects, and TensorFlow applications are themselves Python applications.

2.9.2 OpenCV library

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. Real-life problems require you to use many blocks together to achieve the desired result. So, you just need to understand what modules and functions to use to get what you want.

CHAPTER 3

Implementation and Results

3.1 Introduction

The code was written in python and was needed to install Tensorflow API to run deep learning detect objects models. Then, we implemented algorithms by OpenCV library to extract features such as size of fruit, the color and the number of fruits over the tree. The system was separated in two parts. The mainly part is the pre-process functionality and it's the initially state of the system. This part is the object detection model which the image will be cropped in bounding boxes, each of them will include the object of interest. The second part is the post-process functionality that all new images from the previous part will input in the functions which extract the features of size and colors. The number of items will compute by the number of objects in the first part.

3.2 Pre-process functionality

3.2.1 Object Detection with Faster R-CNN model

We used Tensorflow's Object Detection API to train an object detection classifier for multiple objects. We chose to train our own convolution neural network which it's consist of three kind of fruits but it could be larger with more classes or smaller with only one class, and it's depend on the use of application. In ours system, we need only one class that is peaches but we configured it with three classes to see how robust is the model. At the end of this pre-process step, we will create a program that can identify and draw boxes around specific objects in pictures.

We used Tensorflow-GPU version which allows our computer to use the video card that provides extra processing power. Tensorflow-GPU, instead of regular Tensorflow-CPU, reduces training time in much better performance. Tensorflow provides several object detection models, which is pre-trained classifiers with specific neural network architectures, in it's model zoo. Model zoo is a collection of detection models pre-trained on existing datasets, such as COCO dataset, the Kitti dataset, the Open

Images dataset, the iNaturalist Species Detection dataset etc. These models can be useful for out of the box inference if someone interests for categories in those datasets. Also, datasets can be used for initializing a model when training a novel dataset. The categories in these models are not satisfied our goals and we created a new customized dataset with the new class of peaches.

There are models, such as the SSD-MobileNet model, which have an architecture that allows faster detection but with less accuracy, while on the other hand, there are some models, such as the Faster R-CNN model, which gives slower detection but more accuracy. In our system, we chose to re-train a detector with Faster R-CNN Inception V2 model because it worked considerably better and we did not need speed in outputs results or real time efficiency. The goal is to take results and update the platform in same periods of times and this assumption satisfies the needs of our system. To sum up, the selection of model deals with criteria such as speed, accuracy and the computational power of the device.

Create our own Dataset

As it was referred previously, we decided to create our own dataset and train a new classifier. The first step was to gather and label pictures. Tensorflow needs hundreds of images to train a good detection classifier. The training of a robust classifier was to gather images in variation backgrounds scenes and lighting conditions. For better results was incorporated in classifier some images which the desired object is partially obscured and overlapped with something else, or was only halfway in the picture.

For our classifier, we used different search engines to download a variety of images for fruits, such as google and flickr. An other option was to use a camera and capture a lot of images with different backgrounds and conditions of lightning. Finally, we chose the search engines because of the large variety. Then, we categorized images depending on the size because we wanted those that size was approximately 200KB each and their resolution was not more 1280x720. Larger images are more complicated and it takes longer time to train the classifier. After all, we separated images in two directories, the test and train directory that the test had 20% of images and the other had 80%.

With all the pictures in our collection, we needed to label the desired objects in every picture. We used LabelImg tool to do that but there are a lot of labeling tools to do this job with different front-end environment and different output format files. The LabelImg is an easy to use graphical image annotation tool for labeling images with analytical instructions. It is written in python and uses Qt for it's graphical interface. Annotations are saved as XML files in PASCAL VOC format used by ImageNet. With the images labeled, the next step was to generate data that served as input data to the Tensorflow training model. At this point was generated TFRecords a special file format of Tensorflow.

When training begins each step of it reports the loss. It will start high and get lower and lower as training progresses. For the training in Faster R-CNN Inception V2 model, it started at about 3.0 and quickly dropped below to 0.8. Good results

are achieved when model train until loss consistently drops below 0.05, which it will take about 40000 steps or two or three hours and depending on how powerful is GPU. The loss numbers will be different if use other model.

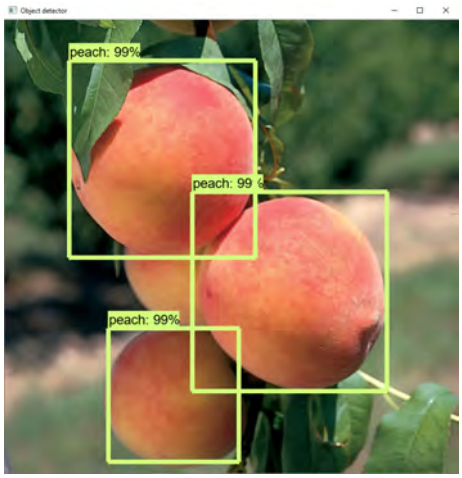
Figure 3.1: TensorBoard.

Tensorflow has TensorBoard where we can view the progress of the training job. The TensorBoard page provides information and graphs that show how the training is progressing. One important graph is Loss graph, which shows the overall loss pf classifier over time. The training routine periodically saves checkpoints about every five minutes. Tensorflow give us to control the process and stop the process whenever we want. Also we can terminate training and start it later, and it will restart from the last saved checkpoint. The checkpoint at the highest number of steps will be used to generate the frozen inference graph.

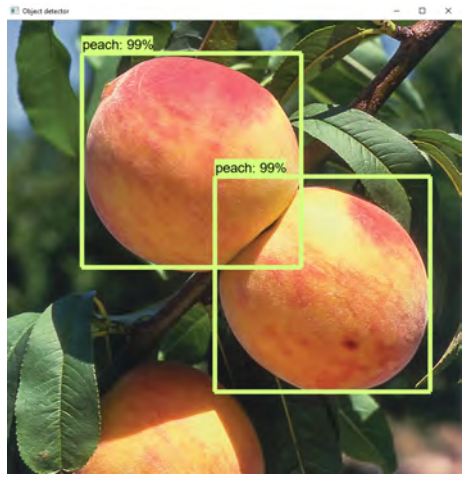
When the train is over, the last step is to generate inference graph.It is a file that end it in .pb file. With that file we can use the object detection classifier.

CPU Processor	Intel Core i5-8300H CPU @ 2.30GHz
Graphic Card	4095MB NVIDIA GeForce GTX 1050
OS System	Windows 10
Tensorflow Version	Tensorflow 1.9
Python Version	Python 3.6.8 in Anaconda environment
Average Training Time	5 hours

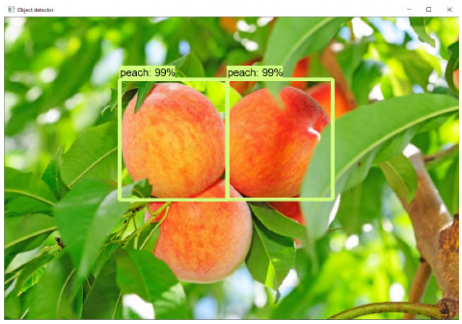
Table 3.1: Information about training.



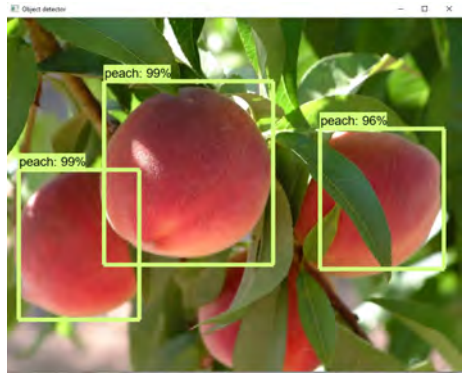
(a) experiment 1



(b) experiment 2



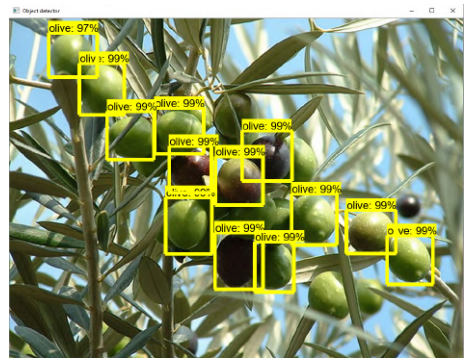
(c) experiment 3



(d) experiment 4

Figure 3.2: Object Detection - Peach Class.

(a) experiment 1

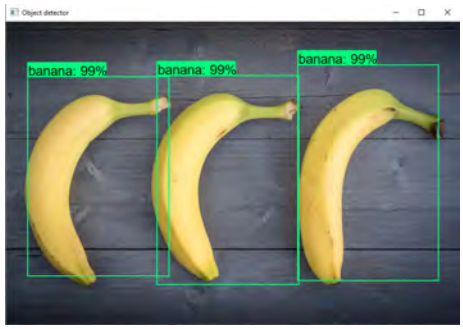


(b) experiment 2

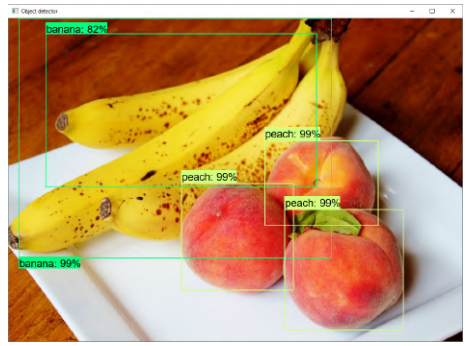
(c) experiment 3

(d) experiment 4

Figure 3.3: Object Detection - Olive Class.



(a) experiment 1



(b) experiment 2

Figure 3.4: Object Detection - Banana Class.

3.2.2 Object Detection with Mask R-CNN model

Mask R-CNN model is another technique to do object detection. This model is suitable to both Object Detection and Segmentation. The model generates bounding boxes and segmentation masks for each instance of an object in the image. It's based on Feature Pyramid Network (FPN) and a ResNet101 backbone. On these experiments, we modified Matterport's implementation of Mask R-CNN deep neural network for object instance segmentation. The model configured to detect fruits in images with varying size. On the figures below, we present bounding boxes refinement and mask generations. Anchor sorting and filtering visualizes the steps of Region Proposal Network and displays positive and negative anchors along with anchor box refinement. Bounding Boxes are final detection boxes and the refinement applied to them. Instance Segmentation is the task of identifying object outlines at the pixel level. Consider the following:
Classification: Find if there is a peach class in image.
Semantic Segmentation: Find all peach pixels.
Object Detection: Find every item on image and covered it with bounding box.
Instance Segmentation: Find the pixels than corresponding on items.

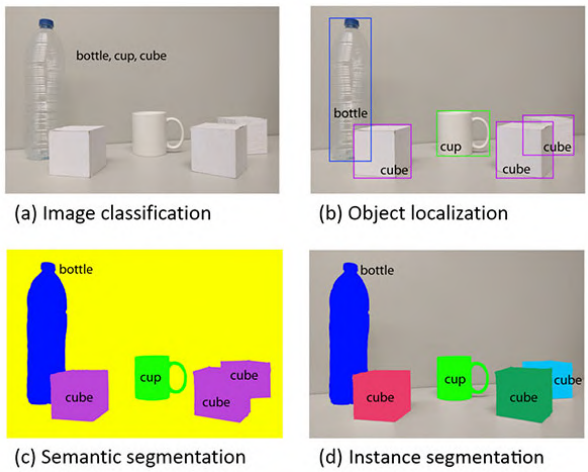


Figure 3.5: Instance Segmentation with Mask R-CNN.

The first step was to find some public dataset which contain classes with fruits but it was not achieved. Then we build a dataset from scratch. At this point, we made similar steps, like previous model, which we used search engines and annotations tools to build new dataset. We created a dataset with approximately 200 images and split in two folders, the train folder and valid folder.

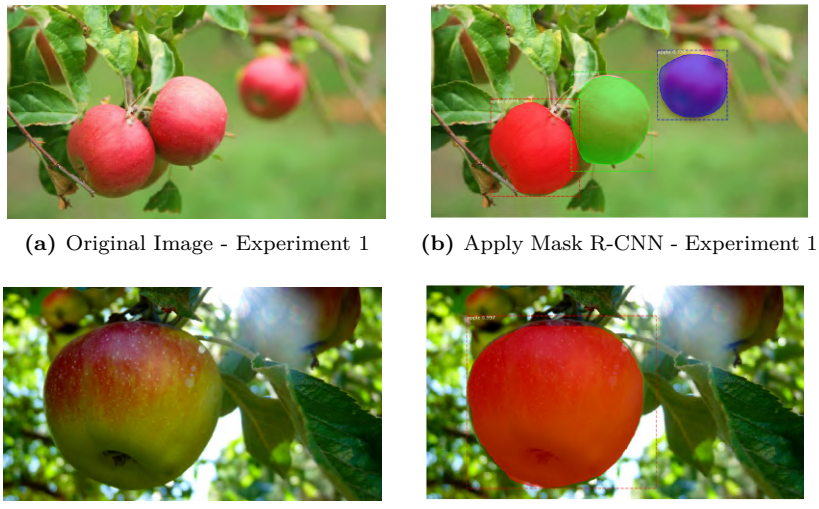
When an application needs a new dataset from scratch, there is an approach which

is useful to follow. We want to reduce training requirements and don't train million images. The way is transfer learning. Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. So, instead of training a model from scratch, the training start with a weights file that's been trained on the COCO dataset. Although, the COCO dataset does not contain a peach class but it contains a lot of other classes and images, the number of them is 120K images, so the trained weights have already learned a lot of features common in natural images and it is very helpful.

There are a lot of tools to annotate images. In our experiments was used VGG image annotator that is called VIA. VIA is a simple manual annotation open source software for image, audio and video. It runs in a web browser and does not require any installation or setup, it's a single HTML file. There is not a universally accepted format to store segmentation masks. In that case, the data are stored as polygon points in a json file. After all of them, the training could be start. The time that needs for hole processing depends on power of GPU and the training started with pre-trained weights of COCO.

Accuracy

For better results, we should reconstruct the dataset with much more images and better resolution. But after the experiments, we assume that the results was not an effective solution because we need more accuracy and the Mask R-CNN model is not provide the accuracy that we want.



(a) Original Image - Experiment 1 **(b)** Apply Mask R-CNN - Experiment 1

(c) Original Image - Experiment 2 **(d)** Apply Mask R-CNN - Experiment 2

Figure 3.6: Object Detection with Mask R-CNN - Apple Class.

(a) Original Image - Experiment 1

(b) Apply Mask R-CNN - Experiment 1

(c) Original Image - Experiment 2

(d) Apply Mask R-CNN - Experiment 2

(e) Original Image - Experiment 3

(f) Apply Mask R-CNN - Experiment 3

Figure 3.7: Object Detection with Mask R-CNN - Peach Class.

3.2.3 Faster R-CNN and GrabCut

GrabCut is an innovative segmentation technique that uses both region and boundary information contained in an image in order to perform segmentation. GrabCut also performs image segmentation in a novel way by using graphs to store region and boundary information. We will apply GrabCut in bounding boxes that are the regions of interest, it is the outputs from Faster R-CNN deep neural network.

GrabCut algorithm makes foreground extraction for background. Foreground-background separation is a segmentation task, where the goal is to split the image into foreground and background. In semi-interactive settings, the user marks some pixels as "foreground", a few others as "background", and it's up to the algorithm to classify the rest of the pixels.

I would remind that our first try was Mask R-CNN but it was rejected because we did not have the accuracy that we want to have. So, our next try is GrabCut algorithm that was designed by Carsten Rother, Vladimir Kolmogorov & Andrew Blake from Microsoft Research Cambridge. We will make same properly customization on it with the goal to achieve better accuracy in background extraction.

GrabCut is considered as one of semi-automatic image segmentation techniques, since it needs user interaction to initialize segmentation process. Initially, we draw a rectangle around the foreground region (foreground region should be completely inside the rectangle). Then algorithm segments it iteratively to get the best result. In some cases, the segmentation won't be fine, like, it may have marked some foreground region as background and vice versa. In that case, user need to do fine touch-ups. Just give some strokes on the images where some faulty results are there. In other cases, user need to specify that this region should be foreground and he marked it background and correct it in next iteration. Then in the next iteration, we get better results.

To make the process automatic, we correspond the rectangle with a bounding box which is generated on deep neural network. In our case, the algorithm is performed very well because the object is always in rectangle and shape has well geometrical specifications. So, we modified the input of algorithm to operate automatic and the user interaction is avoided. So, we created a new fixed rectangle around the bounding box which was returned in previous steps. The new rectangle is approximately 10% or 15% smaller than bounding box. GrabCut knows that there are only background pixels out of rectangle and on the other hand, some pixels inside the rectangle are foreground pixels. Then it can classify pixels as background and foreground. To choose a suitable rectangle we need to know the resolution of images and then we can make assumptions.

(a) Original Image - Experiment 1

(b) Apply GrabCut - Experiment 1

(c) Original Image - Experiment 2

(d) Apply GrabCut - Experiment 2

(e) Original Image - Experiment 3

(f) Apply GrabCut - Experiment 3

Figure 3.8: GrabCut Algorithm.

3.3 Extract features and Results

3.3.1 Extract size

3.3.1.1 Measuring size with help of reference object

Digitally measuring the length and breadth of an object is an important challenge in image processing. For this case, we need to define a ratio that measures the number of pixels per a given metric. We needed to calibrate our system using a reference object in order to determine the size of an object. At this point, we will make some assumptions. The first is that we must know the dimensions of the reference object in terms of width or height in measurable unit such as millimeters. The other one is that we should be able to detect this reference object in an image. The reference object should be uniquely identifiable, either on the placement of the object such as an object placed in left side of the image or via appearances which like being a color. For this experiment, we chose a coin (one euro) that has width 23.25 mm, and the detection was been via placement, it will be the left-most side in images.

The pixels per metric is a number that we will calculate as:

$$Pixels\ per\ metric = \frac{pixel\ width}{real\ width\ in\ millimeters}$$

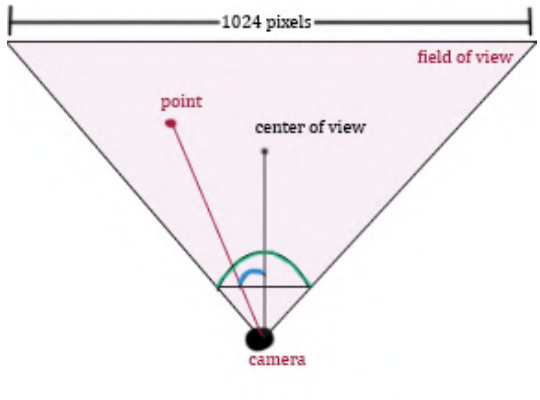


Figure 3.9: Angle between center of view and a point on frame.

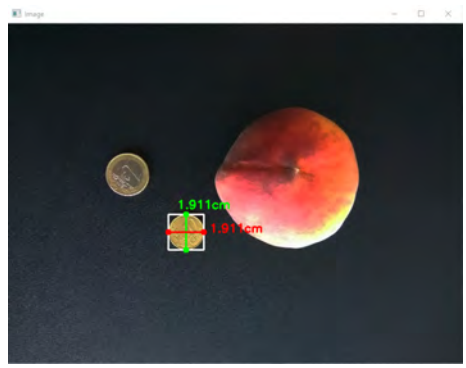
Using this ratio, we can compute the size of objects in an image. Briefly, the process steps are:
We load the image from disk, convert it to gray scale and smooth it using a Gaussian filter. Then, we perform edge detection along with dilatation and erosion to close any gaps between the edges. Sort objects from left to right contours and after that extract the reference object, it is the left most object. Finally. we draw bounding

boxes around the objects and measure the width and height for all of them. The dimensions is the Euclidean distance between the midpoint of bounding boxes. The real dimensions of the object is the divide of respective Euclidean distances with the pixel per metric value.

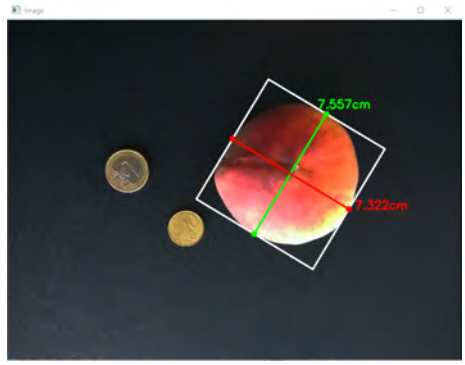
Disadvantages

In real time conditions, when we use camera to capture images there are some disadvantages that we must address. Firstly, when objects are not on center of view on camera then we will calculate measurements with errors. There are points that create angle between center of view and camera lens and this event affect measurement accuracy. In the dimensions of object will appear distorted. It is an error tolerance in millimeters. For example, we can compare Figure 3.10 with Figure 3.11 where in last is appeared error in one euro coin and others. Moreover, another problem when we capture images is that lens have the radial and tangential distortion.

(a) Reference Object

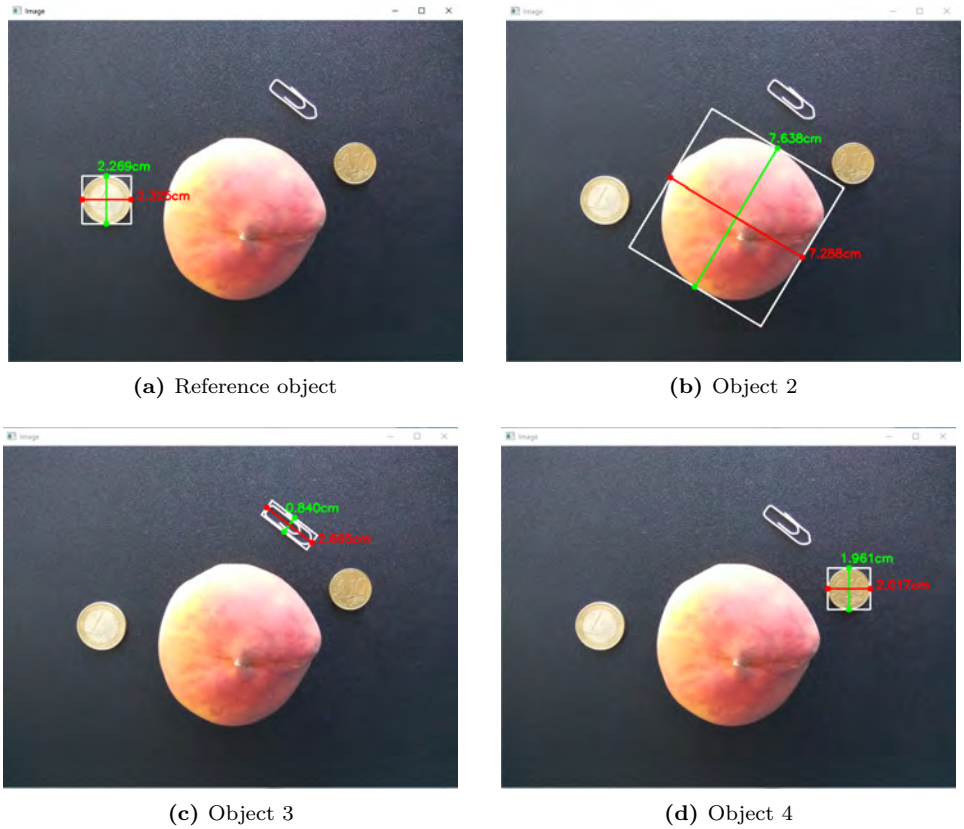


(b) Object 2



(c) Object 3

Figure 3.10: Measuring size with help of reference object.



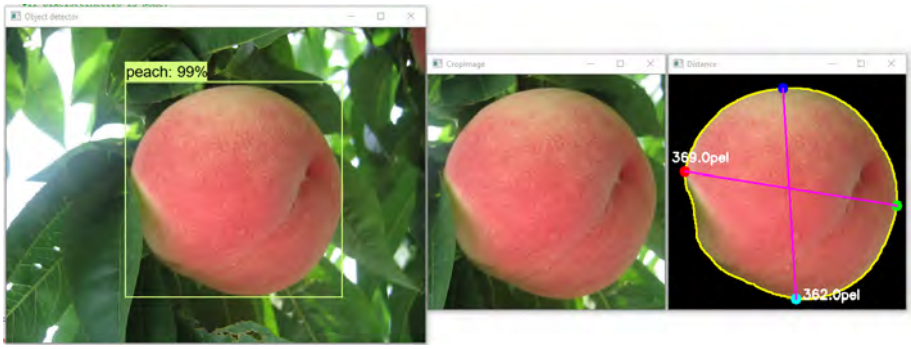
(a) Reference object **(b)** Object 2

(c) Object 3 **(d)** Object 4

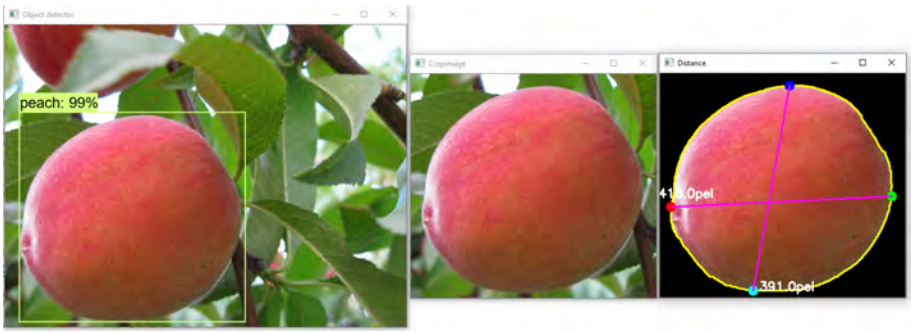
Figure 3.11: Disadvantages - Objects are not in center of view.

3.3.1.2 Calculate perimeter in pixels

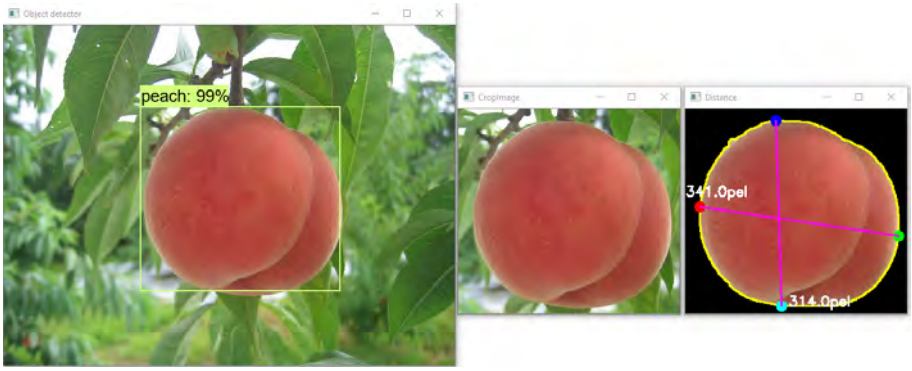
An other case, is to use an abstract metric to calculate the variation of size during the day and not exactly the real size. This new metric is perimeter in pixels. We will store perimeter value on a database and we will track these values and compare with previous results to extract variation of fruit. We can save the variation as the percentage of reduction or growth. For these experiments, we use again the deep learning power to do object detection and find regions of interest. As we mentioned in introduction, this is the pre-process functionality of our system that we run to initialize it. Then we use GrabCut algorithm to extract background and calculate perimeter in pixels with OpenCV functions. Furthermore, we calculate the Euclidean distances between the most upper pixel and most low pixel and most right with most left pixel correspondingly. The Euclidean distances is an extra information that we can use instead of perimeter.



(a) Experiment 1



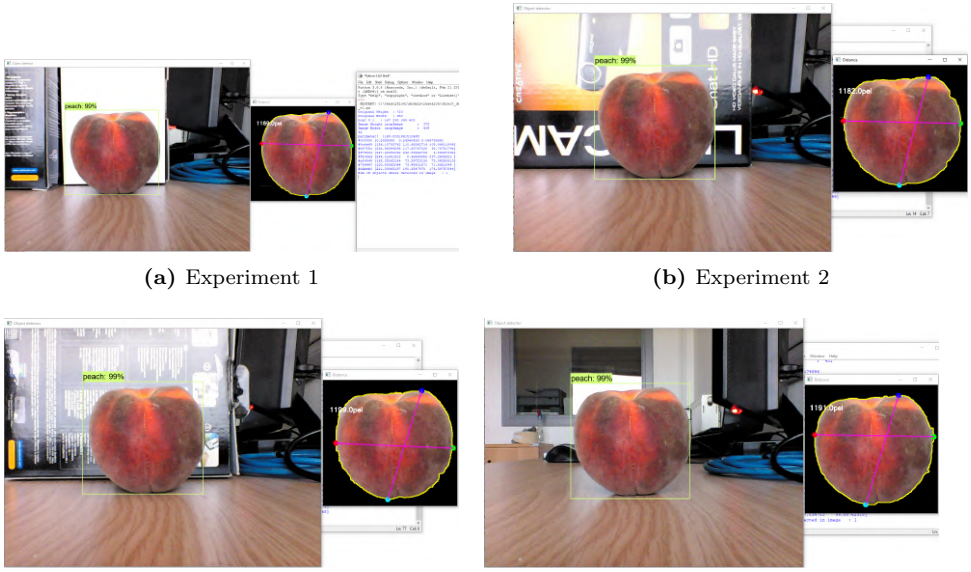
(b) Experiment 2



(c) Experiment 3

Figure 3.12: Yellow line is Perimeter in Pixels - Purple lines is Euclidean distances.

As well, we calculated perimeter in pixels with our camera in laboratory. This showed in Figure 3.13.



(a) Experiment 1 **(b)** Experiment 2

(c) Experiment 3 **(d)** Experiment 4

Figure 3.13: Calculate Perimeter in pixels with our camera.

3.3.1.3 Real size measurements with help of a distance sensor

Dimension inspection plays an important role and it required large amounts of time and effort to calculate it. These dimension inspections can be automated with image processing. In general, the dimensions of parts and products are measured with micro-gauges or calipers or checked with inspection jigs to ensure that there is no variation in accuracy.

Measurement using instruments such as micro-gauges or calipers inevitably involve errors due to individual differences among workers and measurement conditions. The result of checks with inspection jigs can be used for pass/fail judgment of whether the product is within the tolerance range or not. However, it does not show accurate measured values.

Other methods of dimension measurement include optical cooperators, GD&T and profile measurement systems, and 3D measurement systems. They share the problems of requiring manual operation, and being both time consuming and expensive. With image processing, various dimensions can be obtained from captured images. It is easy to measure the dimensions of various sections of parts and products based on this data and judge whether they are within tolerances. Another advantage is that we can measure angles or circle roundness simultaneously with the lengths of various sections, and save them as numerical data.

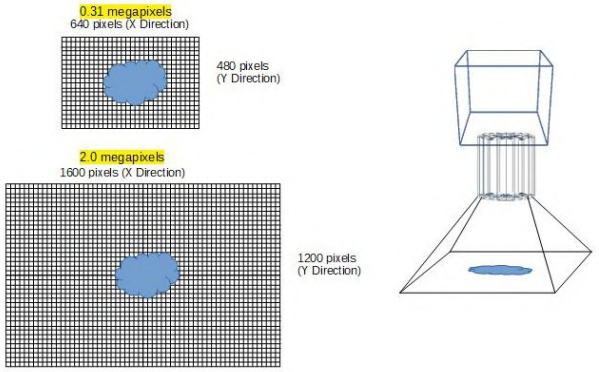


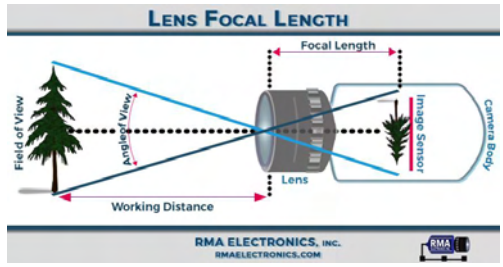
Figure 3.14: Resolution of camera.

Sensor

This SparkFun Distance Sensor Breakout utilizes the VL53L1X next generation ToF (Time of Flight) sensor module and it gives highly accurate measurements at long ranges for its size. These low power laser based time of flight sensors have great accuracy and sampling frequency, and this particular sensor has a wide range of detection, from 4 cm to 4 metres. It provide to us a Python library that makes it straightforward to use the time of flight sensor, providing methods for short, medium, and long range, and returning distances in millimeters (mm).

Measurement based on edge detection is generally used for dimension inspection using image processing. The image pickup device of a vision system consists of an array of pixels. In dimension measurement, we can calculate the dimension tolerance based on this number of pixels and the field of view. An important factor for the calculation is pixel resolution, which is the actual length that corresponds to a single pixel of the image pickup device.

(a) Distance Sensor



(b) Focal Length, source: rmaelectronics

Figure 3.15: Triangle Similarity.

How to calculate dimensions of an object

We used triangle similarity technique to determine dimensions. In order to determine the distance from camera to a known object we used a distance sensor and then we was gone utilize triangle similarity.

Firstly, we wanted to calibrate our system. We needed a marker or object with a known width. Then we placed this marker in some distance from camera. We took a picture of our object using our camera and then measured the apparent width in pixels. This allowed us to derive the perceived focal length of camera:

$$Focal\ Length\ = \frac{Object\ in\ pixels \times Distance\ (Sensor\ output)}{Object\ width\ in\ centimeter}$$

Our calibration object was a simple rectangle on a piece of paper. Then, when we will capture a new unknown object, we will solve the same equation with unknown part will be "Object width in centimeter". So, the equation is:

$$Object\ width\ in\ centimeter\ = \frac{Object\ in\ pixels \times Distance\ (Sensor\ output)}{Focal\ Length}$$

To sum up, this task utilized the triangle similarity, which requires us to know two parameters prior to applying algorithm:

1. The width (or height) in some distance measure, such as meters or centimeters, of the object we are using as a marker.

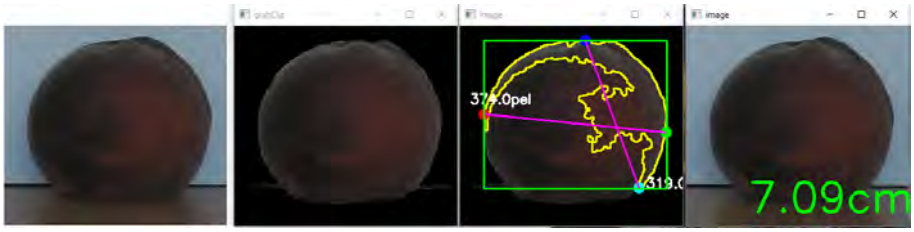
2. The distance (in meters or centimeters) of the camera to the marker.

Computer vision and image processing algorithms can then be used to automatically determine the perceived width/height of the object in pixels and complete the triangle similarity and give us our focal length.

Disadvantages

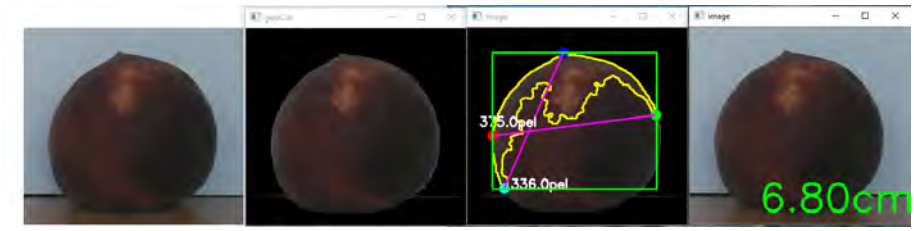
We will face up problems when we will calculate the real size for 3-D objects with 2-D cameras because we don't have depth dimension. Time of flight describes a variety of methods that measure the time that it takes for an object, particle or acoustic, electromagnetic or other wave to travel a distance through a medium.

The Time-of-Flight principle (ToF) is a method for measuring the distance between a sensor and an object, based on the time difference between the emission of a signal and its return to the sensor, after being reflected by an object. Sensor returns distance in millimeters. The problem is when we will remove camera from object and the reflected point is not the same, so the measurements have error in millimeters. This showed in Figure 3.18.



(a) Experiment 1 - Measure Diameter

(b) Experiment 1 - Real width

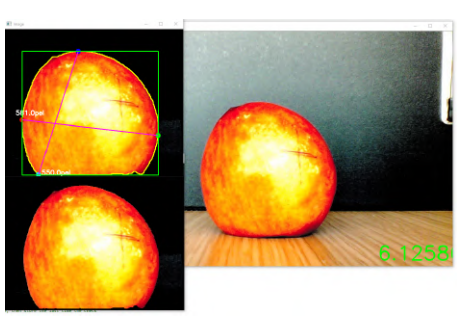


(c) Experiment 2 - Measure Diameter

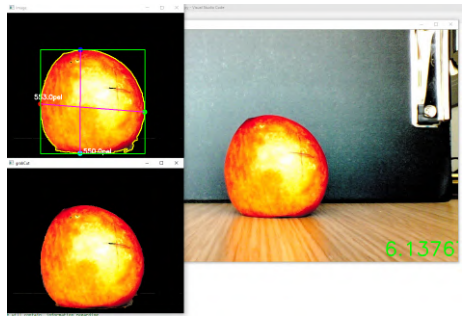
(d) Experiment 2 - Real width

Figure 3.16: Diameter of peach with Distance Sensor.

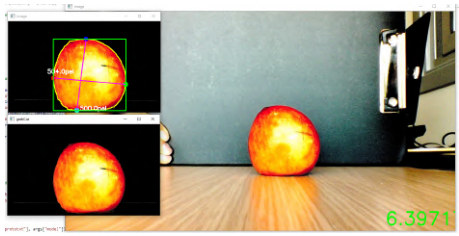
An other experiment was to move camera from object in different distances.



(a) Small Distance



(b) Medium Distance



(c) Large Distance

Figure 3.17: Diameter of peach with Distance Sensor at different distances.

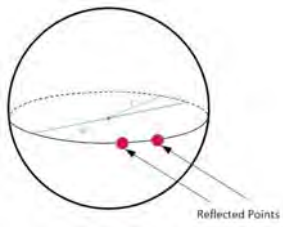


Figure 3.18: Reflected points in 3-D Objects.

Distance Sensor	SparkFun Distance Sensor Breakout - 4 Meter, VL53L1X (Qwiic)
Camera	Creative Live! Cam Chat HD
Breadboard	BeagleBone Black

Table 3.2: Information about camera node.

3.3.2 Extract Colors

3.3.2.1 Introduction

The other feature that we want to extract is colors of the fruit. We want to make a color identification to extract colors from a given image. For this purpose, we used a machine learning algorithm, K-means, to find most dominant colors over fruits. We used some necessary libraries that have important functions, such as sklearn for K-means algorithm and matplotlib for plotting graphs. K-Means expects flattened array as input during its fit method. Then, we can apply K-Means to first fit and then predict on the image to get the results. Finally, the cluster colors are identified an arranged in the correct order. We plotted the colors as a pie chart.

3.3.2.2 K-means algorithm

K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. Moreover, it is an extensively used technique for data cluster analysis. Firstly, we define a target number k, which refers to the number of centroids we need in the dataset. A centroid is the imaginary or real location representing the center of the cluster. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible. The 'means' in the K-means refers to averaging of the data, that is, finding the centroid.

In our case, we will be clustering the pixel intensities of a RGB image. Given a MxN size image, we thus have MxN pixels, each consisting of three components: Red, Green, and Blue respectively. We will treat these MxN pixels as our data points and cluster them using K-means. Pixels that belong to a given cluster will be more similar in color than pixels belonging to a separate cluster. One caveat of K-means is that we need to specify the number of clusters we want to generate ahead of time.

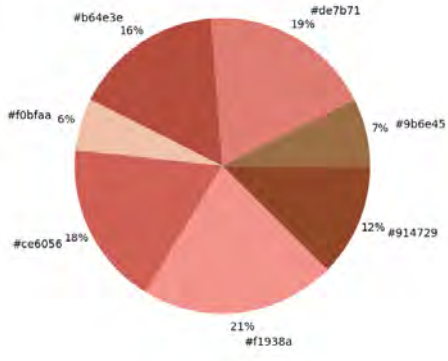
Implementation

We used the scikit-learn implementation of K-means to avoid re-implementing the algorithm, but there is also a K-means built into OpenCV. The K-means algorithm assigns each pixel in our image to the closest cluster. We grab the number of clusters and then created a histogram of the number of pixels assigned to each cluster.

3.3.2.3 RGB values

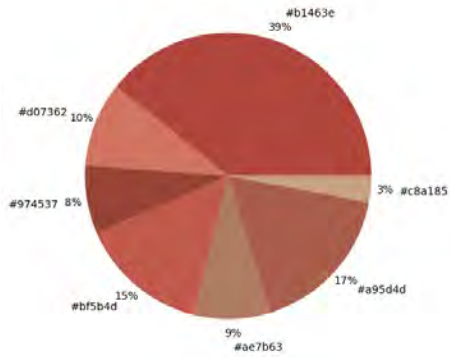
The RGB color model is an additive color model in which red, green and blue light are added together in various ways to reproduce a broad array of colors. We want to extract the top colors from the image and display them as a pie chart. K-Means algorithm creates clusters based on the supplied count of clusters. In our case, it was form clusters of colors and these clusters was been our top colors. Then, we fit and predict on the same image to extract the prediction into the variable labels.

(a) Original Image

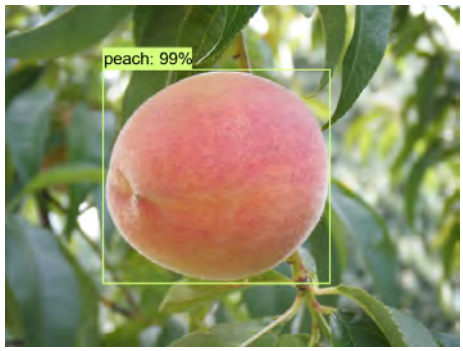


(b) Color pie

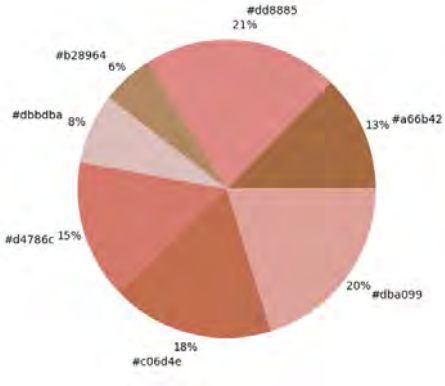
(c) Original Image



(d) Color Pie



(e) Original Image



(f) Color Pie

Figure 3.19: Colors in pie chart.

3.3.2.4 LAB values

The LAB color space goes about defining colors differently. Whereas RGB defines color by a combination of red, green, and blue values of different shades, LAB uses three different channels. They are Lightness, the A Channel, and the B Channel. Hence, Lightness, A Channel, and B Channel are shortened to L-A-B, LAB. Lightness represents the relative brightness of the pixels without regard to color. So, lightness is kind of like a grey scale image, where each pixel is defined by how close to white or black it falls on the scale.

Whereas the Lightness Channel defines the lightness of the pixels without regard to color, the A and B channels define color without regard to lightness. Color and lightness are addressed separately in LAB, not together as they are in RGB (more on this in a bit). We have two channels A and B, and the name of them does not actually mean anything. The A Channel is just a definition of color values based strictly on how much green on one hand, or magenta on the other, are contained therein. The very middle is actually gray, and the hues get progressively more green to one side and progressively more magenta to the other. The B Channel works the same way as A, except that it defines color by how much blue on one side, and yellow on the other, it contains.

Whereas RGB renders color by defining each color as some combination of red, green, and blue, LAB renders color by defining each color as some combination of green, magenta, blue, and yellow, with lightness addressed separately. However, while each color gets its own channel in RGB, colors share channels in LAB (two per channel). Firstly, we convert RGB values to L*a*b format. Then we create 1-D, 2-D and 3-D histogram which are 1 channel, 2 channel and 3 channel L*a*b color spaces.

Identifying Color Differences Using CIE L*a*b* Coordinates

Defined by the Commission Internationale de l'Eclairage (CIE), the L*a*b* color space was modeled after a color-opponent theory stating that two colors cannot be red and green at the same time or yellow and blue at the same time. Color difference can be defined as the numerical comparison of a sample's color to the standard. It indicates the differences in absolute color coordinates and is referred to as Delta (Δ). These formulas calculate the difference between two colors to identify inconsistencies and help users control the color of their products more effectively. As shown below, L* indicates lightness, a* is the red/green coordinate, and b* is the yellow/blue coordinate. Deltas for L* (ΔL*), a* (Δa*) and b* (Δb*) may be positive (+) or negative (-). The total difference, Delta E (ΔE*), however, is always positive.

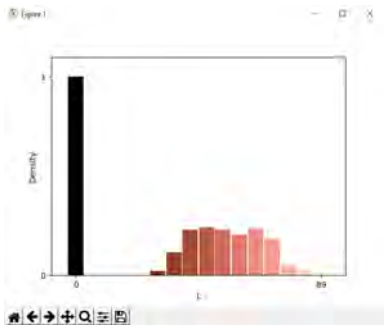
ΔL* (L* sample minus L* standard) = difference in lightness and darkness (+ = lighter, - = darker)

Δa* (a* sample minus a* standard) = difference in red and green (+ = redder, - = greener)

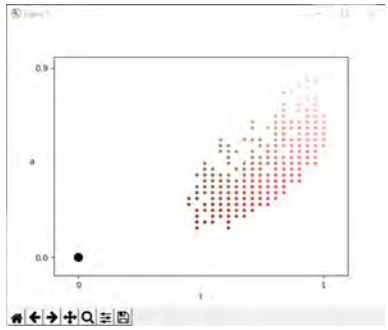
Δb* (b* sample minus b* standard) = difference in yellow and blue (+ = yellower, - = bluer)

(a) Original Image

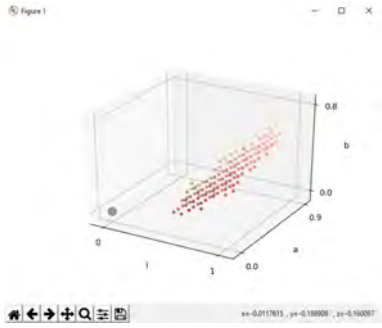
(b) GrabCut



(c) 1 channel L*a*b



(d) 2 channels L*a*b



(e) 3 channels L*a*b

Figure 3.20: Color Histogram.

3.3.3 Calculate the number of fruits

After all previous descriptions and all implemented functions can be perceived that the last feature is an easy procedure to calculate. It is a side effect information after object detection algorithms. The number of fruits is the number of bounding boxes that we processed before with Tensorflow framework. So, we can calculate the number of iterations of Tensorflow outputs.

3.4 Filter Function

In real time conditions, the different weather conditions creates difficulties that effect our system. One of the most common and difficult weather condition is wind. In that case, fruits will move in uniform directions accordingly with the power of wind. Specifically in Figure 3.21, fruits will move around of center of their gravity in all possible directions. If we want to make an effectiveness system, we need to solve two problems. Firstly, we want accuracy when we calculate the size of fruits. If fruit is not in center of view we will have errors in calculations, and this problem is described previously in Figure 3.9. The other problem is the 2-D analysis in 3-D objects.

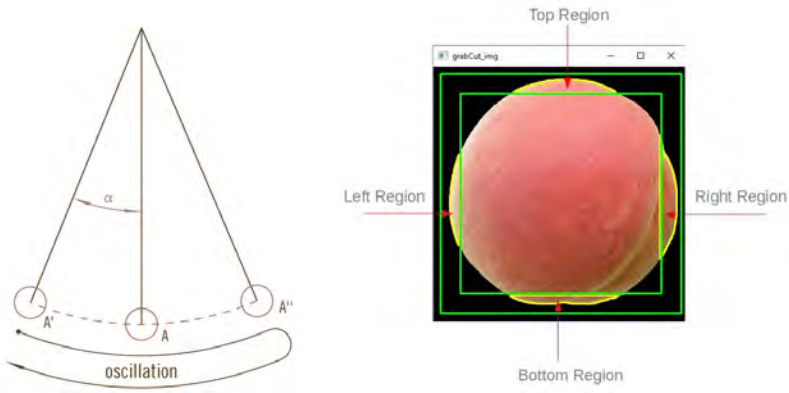
So, we will propose a function that it will run before all the other functions in our system. The function is a kind of filter that decide to reject or pass images. We will make some assumptions:

- When we want to measure the size of fruit during the day, we want to process fruit as much as possible in same coordinates that the fruit was in previous captures. We ensure more accuracy when fruit is in same center of view position and calculations are similar.
- The best of cases is the balance situation of fruit where the power of wind is almost zero. That case is our reference point.
- We install camera in fixed place on branches.
- After all of them, we capture an image and initialize our system. The image has the fruit in balance situation and the application return the coordinators of a bounding box. This bounding box is our reference bounding box.

Function of filter is:

We created a function that captures repeatedly same frames (during in seconds) of fruit that possibly had it in different positions because of wind. The function returns a boolean value that is a flag of PASS of FAIL. PASS is images that likes with reference bounding box and the other are fail. So, we accept fruits that are in reference bounding box. The steps are:

- Create the reference bounding box



(a) Oscillation **(b)** Regions and reference bounding box

Figure 3.21: Filter Function.

- Draw two smaller rectangles inside the reference bounding box and separate fruit in four regions, top, bottom, right and left. This showed in Figure 3.21.
- Calculate the number of not black pixels in each of the region.
- Comparison the values of previous step with a minimum and a maximum limit value. Then, we accepted an error tolerance.

We create two videos that makes clearly the functionality of filter at the links below:

Link to Video 1
Link to Video 2

To sum up, we implemented a filter that reject or pass images and it makes our system more accurate. We will choose the better image because the filter should be based on natural lows.

CHAPTER 4

Conclusion

In conclusion, we designed and implemented a distributed data processing system which is consisted of computer vision algorithms and techniques. It is smart system that we will install in trees and it can recognize useful features such as size and colors. This application is intended to farmers to inform them about health and growth of crops. We chose cameras as powerful sensor and we will create a Wireless System Network with them.

So, we made a first version of a computer vision system which automatically recognized fruits and then calculate the useful features. Camera is applied over the tree and it capture images that are send over the network in a central computer for processing. Camera could be used to take pictures automatically during the day. Then, the images will be classified in a database depending to the time, feature and tree. After all of them, we will run the main part of procedure which is computer vision algorithms. Computer vision algorithms are state of the art methods which are fast and accurate.

Particularly, the size of fruit is a complicated function that we achieved it with modern technology. Technically, we needed to separate background and foreground to find the regions of interest. It's important to crop original image in accurate coordinates to simplify our problem. After this step we continued with other geometrical calculation to extract the size. The segmentation of image is not a straight through procedure but we used state of the art technology which is based on artificial intelligent and more specific in machine learning and deep learning models.

With the advantage of separated images, we extended our problem to take fruit colors. We focused on fruit to save all color values from fruit area in an array. This values is needed to process again if we want to be useful. Firstly, all values was in RGB format and was stored in an array. The array had thousand records and we needed to classify color data. We used machine learning algorithms to print incidence rates of dominant colors. It's a practical format to understand the mature of fruits. Furthermore, we calculated colors in other formats such as L*a*b values for more effectiveness.

4.1 Future Work

We will want to create a more robust system which it will be honour obligations of a real conditions system. For example, in crops will have some animals that impend the field of view and system will decide wrong. Also, in real conditions we faced up with weather that it will make an unstable system.

Moreover, we want to recognize immature fruits. In that case, the problem became more complicated because of the similarity of background and foreground. We should be implement other effectiveness algorithms for image segmentation. Finally, one interesting feature is to recognized diseases on the fruit. We want to implement new algorithms which will recognize if a fruits are damaged or misshapen.

Bibliography

[Gir+] R Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation. arXiv 2014". In: *arXiv preprint arXiv:1311.2524* ().

[Gir15] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pages 1440–1448.

[Har15] Adam W Harley. "An interactive node-link visualization of convolutional neural networks". In: *International Symposium on Visual Computing*. Springer. 2015, pages 867–877.

[He+17] Kaiming He et al. "Mask r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pages 2961–2969.

[Hua+17] Jonathan Huang et al. "Speed/accuracy trade-offs for modern convolutional object detectors". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pages 7310–7311.

[Kan10] Saeideh Gorji Kandi. "Automatic defect detection and grading of single-color fruits using HSV (Hue, Saturation, Value) color space". In: *Journal of Life Science* 4.7 (2010), pages 39–45.

[LeC+98] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pages 2278–2324.

[Lee+09] Honglak Lee et al. "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations". In: *Proceedings of the 26th annual international conference on machine learning*. ACM. 2009, pages 609–616.

[LSD15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pages 3431–3440.

[NG15] Thien Huu Nguyen and Ralph Grishman. "Relation extraction: Perspective from convolutional neural networks". In: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. 2015, pages 39–48.

[PG17] Josh Patterson and Adam Gibson. *Deep learning: A practitioner's approach*. " O'Reilly Media, Inc.", 2017.

[Ren+15] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*. 2015, pages 91–99.

[RKB04] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. “Grabcut: Interactive foreground extraction using iterated graph cuts”. In: *ACM transactions on graphics (TOG)*. Volume 23. 3. ACM. 2004, pages 309–314.