**DETECTION OF DATE PALM**

**TREES USING AI**

**Project Idea Report**



**Muhammad Yasir NUM-BSCS-2023-37**

**Department of Computer Sciences**

**Namal University Mianwali, Pakistan**

**Submission Date: 28th December, 2025**

**Abstract**

The proposed project involves the development of an AI system for the recognition and counting of the Date Palm tree using deep learning techniques. The AI system is to be developed using the YOLO framework, and it is intended to be functional on the aerial, satellite, and video images of the environment. The original project involved the recognition of the Kikar (Acacia) tree, but since there is quite some similarity with the surrounding environment, the project is to be executed on the Date Palm tree, which is easily recognizable and has appropriate data available.

The Date Palm Plant Detection System can detect Date Palm plants, create a bounding box, and enumerate the plants using tracking logic. Apart from this, the project also consists of a backend and simple frontend part that facilitates the upload of images and videos and the display of the results of detection. The objective of this proposal and subsequent work is the facilitation of agricultural and plantation analysis.

## 1. Introduction

Tree detection and identification have a significant impact on agriculture, environmental monitoring, and land use. In arid and semi-arid regions, specifically, Date Palms are a valuable agriculture resource. They help not only in terms of economics but are also useful for food and land. Manual detection and identification of Date Palm trees from a significant agriculture land area as well as from aerial images is a time-consuming, expensive, and inaccurate process. Now, due to the availability of aerial and satellite images, observing a vast area of land has become possible at the same time. However, manual inspection of these images for analysis purposes involves a lot of complexities and technicalities. Inconsistencies and errors can occur, along with missed detections and incorrect counting, if the number of trees is high, and hence there is a great need for a solution capable of detecting and counting Date Palm trees automatically. There have been recent breakthroughs in the area of Artificial Intelligence, specifically in the area of Computer Vision. It has made ways for automating object detection possible. Deep learning models can effectively learn visual patterns from the data and detect objects in images. Among different models, YOLO, which stands for "You Only Look Once," has been widely recognized for its speed and efficiency in detecting objects in a single pass through the image or video frames [1].

The project first involved the detection of the Kikar trees (Acacia). Unfortunately, due to the resemblance in their appearance to other plants, it proved unsuccessful. Further observation of the project led to adjusting the project's focus on Date Palm trees, as they appear distinctly when viewed from aerial or satellite pictures. Hence, it enabled a clearer process of learning and detection. Currently, the stage of this project involves the application of image detection techniques to a greater extent. Apart from the development of the YOLO model algorithm to detect the Date Palm tree from the images, other developments in the project have been focusing on video image detection and counting through object tracking algorithms. The purpose of this project is to create a system using Artificial Intelligence that has the capability to automatically detect and count the Date Palm trees from the aerial images and videos. The system will serve to aid agricultural planning and further be a part of future agricultural developments of smart farming.

## 2. Problem Overview

Date Palm trees are of significant importance to agriculture, land use planning, and, as a result, the economy of a given region. Manual processes, however, form the basis of identifying and counting DATE Palm trees, especially where aerial photographs or satellite images are to be considered. Manual checking of images can, therefore, turn out to be a very time-consuming, expensive, and inaccurate process, given the human difficulty of evaluating thousands of DATE Palm trees distributed across agricultural land. The second problem is that vegetation in images has a high degree of resemblance. Similar vegetation, such as types of trees/plants, may have a resemblance concerning their shape, color, or texture, resulting in a challenge for human identification. This may cause errors such as the absence of trees, duplicate identification of the same tree, or the incorrect identification of other vegetation entities as Date Palm trees.

Due to such issues, there is a great need to have an automated system capable of detecting and determining the number of Date Palm trees. This system must run on aerial or satellite pictures and be independent of human interference. Without such automated technology, there is restriction in agricultural planning and environmental study. This problem also affects plantation management. In order to tackle the above-mentioned problem, this project relies on the use of deep learning and computer vision algorithms to detect the Date Palm trees. By utilizing the YOLO algorithm and developing an object detection algorithm, the system will be able to detect the positions of the Date Palm trees using bounding boxes and will be able to give a count of the Date Palm trees. It will make the detection of the Date Palm trees faster and accurate, along with the capability to do it in a wide area of agriculture.

## 3. Objectives and CCP Justification

### 3.1 Project Objectives

Aiming to spot and tally Date Palm trees, this effort builds a tool using artificial intelligence. From sky views and moving footage, it pulls out tree locations without help from people. One goal stands clear: cut down slow, expensive checks done by hand. Mistakes happen when humans count long hours, so machines step in quietly. Time folds shorter when software handles the load instead of eyes scanning photos. What this project aims to do comes next. Goals were set with care during planning stages. Each target ties directly to outcomes expected by funders. Progress will show through measurable changes over time. Success means hitting every point listed here

A fresh take on spotting Date Palm trees begins with training a model using deep learning. Images shot from above - either by drone or satellite - feed into the system slowly. One after another, visual patterns start to make sense. Where shadows stretch long, the algorithm learns what shape belongs to palm crowns. Instead of guessing, it compares tiny details across hundreds of examples. Accuracy grows as differences between tree types become clearer. Over time, the recognition sharpens without needing extra guidance. The outcome? A tool that knows palms not by labels but by how they appear from high up.

A single count per tree stands clear when tracking moves through video frames. Object methods lock onto shapes, following them without double taps. Movement patterns help tell repeats apart from fresh entries. Each shape gets noted just one time, no matter how long it stays in view. The system watches closely, marking who's already been seen.

A full workflow begins with cleaning raw inputs before feeding them into learning algorithms. After patterns form through repeated exposure, moving objects get marked across frames using consistent labels. Each step flows into the next, building up from basic adjustments to clear visual outputs. Following computation, outcomes appear on screen as colored paths over time.

Start by connecting the learned model to a working setup. Hook it up using an API on the server side. Build a basic interface where users can submit pictures or videos. Let the system handle incoming files through that link. The front part stays minimal on purpose. Data moves from screen to processing engine behind the scenes. Everything runs once pieces fit together. Looking at how well the system works with new data gives insight into actual farming uses. What happens next depends on results seen during testing phases.

### 3.2 Why Complex Computing Problems Matter

Ahead of basic coding tasks, this work pulls together several high-level computing elements. Not only does it weave in artificial intelligence, but also brings vision systems into play. Decisions unfold under actual world limits, shaping how pieces fit together. Instead of following set steps, the approach adapts through layered logic. System design becomes central, holding everything in place across shifting conditions.

Spotting date palm trees in aerial photos isn’t straightforward. Changes in size, light conditions, nearby plants, or viewing angles make it tough. Learning these patterns means using deep neural networks instead of set rules. Traditional methods fall short here. Working with vast amounts of pixel data adds another layer of difficulty. Adjusting countless internal settings takes serious computing power. Each step demands precision without room for guesswork.

Another part deals with following objects through videos, spotting them frame by frame despite shaky cameras or things blocking view. When visibility flickers or blurs happen, keeping track gets harder. Smart tracking methods step in to hold identity steady over time, avoiding double tallies. Holding on to past states means decisions rely not just on now but what came before - adding layers most overlook.

What makes it tricky is how fast everything has to move without losing precision. One part cleans raw information before anything else happens. Following that, predictions are made using trained patterns. Decisions flow into systems that monitor changes over time. These connect to online pathways handling requests behind the scenes. Outputs show up as visuals users can interpret easily. Smooth handoffs between pieces keep things running without hiccups. Saving power matters just as much as getting quick answers. Every piece adjusts itself based on what the others are doing.

Farmers face tough choices when forecasts miss the mark - this work steps into that space. Messy inputs, shaky detection results, unpredictable field conditions - they shape what happens here. Looking at what's involved, this effort counts as a Complex Computing Problem because it uses sophisticated artificial intelligence methods alongside step-by-step analysis, choices made without full information, and ties together different parts into one working whole aimed at tackling an actual situation. Though built piece by piece, the entire setup responds to practical demands through layered logic and adaptive responses woven across stages.

## ****4. System Architecture and Design****

### ****4.1 System Overview****

The whole system starts with some aerial or satellite images that include the Date Palm trees. These inputs form the entire system of the problem that is to be processed. Raw images, in most cases, do not form the best input to be processed to get the final output, which is why the raw images go through the preprocessing phase after that. The images are then passed on to the YOLO-trained detection system. The system scans the images and marks the Date Palm trees by placing bounding boxes on the images. The marked trees are then recognized as separate objects. The system then proceeds to the counting logic for tracking and counting the trees considering their location and motion (if images are from videos). Finally, the system provides output images or videos of the regions of interest where detected date palm trees are marked by drawing bounding boxes around them and displaying the number on the screen.

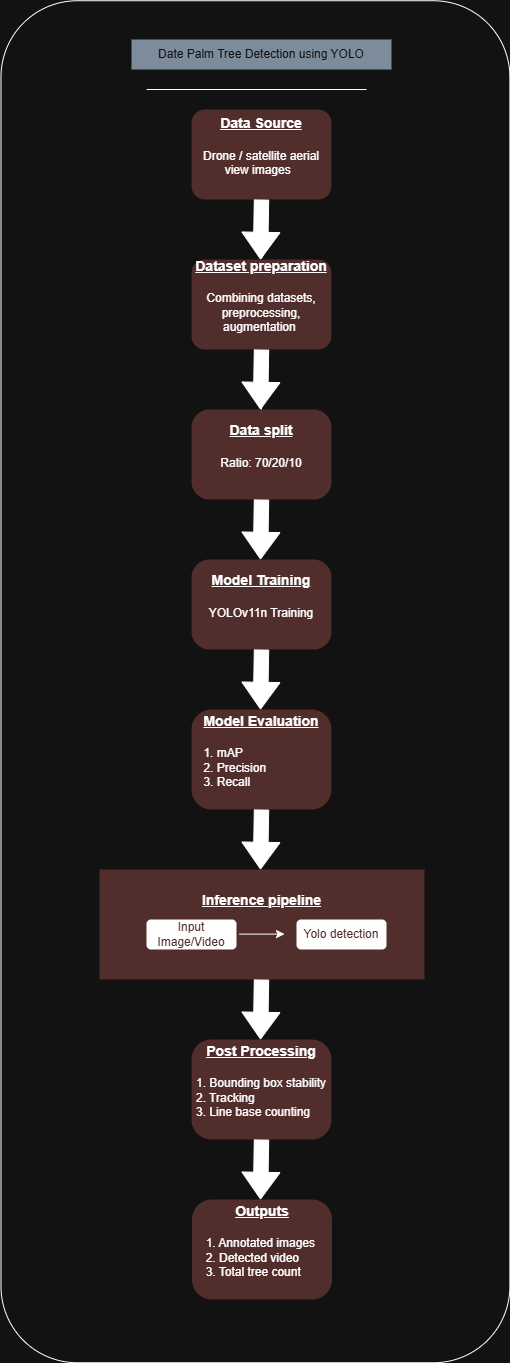


Figure 1: System Design Diagram

### ****4.2 Architecture Diagram Explanation****

The architectural diagram is the representation that displays the interaction between the various components available in the system. The initial block in the diagram indicates the image source, which consists of aerial images obtained from public sources. The images are then transmitted to the preprocessing block. In the preprocessing component, images are resized to a fixed resolution, orientation errors are fixed, and images are normalized. This increases the robustness of the model. Image data augmentations are also carried out during the training of the model to make the model generalize well.

Now comes another block in the architecture, which is the YOLO detection model. It is actually the main part of the system. It is used for the detection of the Date Palm trees in all images using the trained YOLO detection model. It predicts the bounding box and confidence scores. Once detected, the system employs tracking and counting logic. This stage avoids situations where the same tree is counted several times, especially when dealing with video input. Using tracking ids helps achieve consistency and is particularly beneficial when counting. "Output block": It is the final block of the output module. It produces labeled images or videos of the detected trees showcasing the surrounding bounding boxes along with the number of trees.



Figure 2: System Architecture Diagram

## 5. Implementation Details and Algorithms Used

A step-by-step look at how the Date Palm Tree Detection System works begins here. The way trees are found uses pattern-based calculations instead of general rules. Following their movement across images relies on position shifts over time. Counting them happens through spaced checks, avoiding double records. Putting it all together depends on smooth connections between each part.

### 5.1 Object Detection with YOLO

A fresh take on spotting objects comes from a setup built around YOLO - short for You Only Look Once. This method skips multiple stages, instead scanning a full picture during just one go-through. Because it handles everything at once, speed stays high, fitting tasks where timing matters or volume piles up.

This time, YOLOv11n took center stage - its lean design plus quick processing made it stand out. Not heavy on computing power, yet still sharp when spotting Date Palm trees in images from above. Labeled data guided the training, each tree outlined carefully inside boxes. When running live checks, what comes out? Predictions shaped by that earlier learning phase

* Bounding box coordinates
* Tree known as Date Palm
* Confidence score for each detection

Not every guess makes the cut - only the strong ones pass through. Weak signals get tossed aside before things move forward. What stays has to meet a strict standard of certainty. This step keeps errors from creeping into later stages.

### 5.2 Video Processing Pipeline

Breaking down videos begins by splitting them into separate images, done with OpenCV. One after another, each image moves into a pre-trained YOLO setup to spot objects. Going step by step through frames helps track how things move while keeping results steady over time. A fresh video takes shape once the frames come back together, now showing Date Palm trees picked out by boxes that outline each one. Labels appear nearby, naming what was found during analysis.

## 5.3 Tracking and counting objects

A single palm might show up in several clips, making basic scene-by-scene spotting unreliable. Tracking each one through time helps avoid double-counting. Every time a tree shows up on screen, it gets its own ID number through a method called BoT-SORT. Even if things shift slightly - like leaves blocking view or the frame jolting - that label sticks. When the camera moves around, especially on drones, a feature named Global Motion Compensation steps in quietly to keep track straight. This whole setup makes sure trees stay correctly followed from one moment to the next. A single count happens each time a tree's tracking ID steps into the detection zone. The setup remembers which IDs already passed through. Only upon first entry does the tally increase. Stability and precision stay consistent across the full clip because of this method.

## 5.4 Improved Confidence Checks and System Reliability

A few extra methods get added to boost how steady the detection stays while cutting down on wrong alerts

* Some guesses get tossed out if they seem too unsure
* Bounding box smoothing ﻿to reduce flickering effects in videos
* Persistent tracking IDs﻿ to maintain object identity across frames

With these upgrades, spotting trees becomes easier even when light shifts or surroundings change. Clearer visuals mean counts stay accurate through different outdoor settings.

### 5.5 Backend API Implementation

A web server built with Flask handles how the app talks to the detection tools and moves data around. This layer supports standard web requests that let the front part of the system ask for predictions or send information through structured routes

* Image detection
* Video detection
* Handling finished data documents

Files you upload stay only for a short time. Once received, they move into analysis through specific tools designed to detect features. Results come back structured in JSON data. Alongside, there are links pointing to processed visuals. Communication between front and back systems works without blocks. This happens because CORS settings are turned on. Smooth exchange of information follows naturally.

### 5.6 Frontend Integration

Out there, a clean front screen shows up thanks to today’s web tools - lets people drop in pictures or clips then see where trees are spotted. From that point on, it talks straight to the server part of the app, pulling back results tagged with outlines plus how many trees were found. Because everything links together smoothly, anyone can use it even if they do not know code or tech details. Few steps stand between uploading and seeing answers clearly laid out. No extra training needed just to get what you came for.

### 5.7 Algorithmic Workflow Summary

Here is how everything fits together in the system that was built

1. Input image or video is uploaded through the frontend
2. Backend API receives and preprocesses the input
3. YOLO model detects Date Palm trees
4. Moving through frames, BoT‑SORT keeps track of who's who in video clips
5. A single tally happens per tree because of how counting works. That method stops double entries by design. One count only fits every trunk seen out there
6. Results show up on screen once they come through

## 6. Results, Testing, and Evaluation

This section presents the evaluation of the proposed Date Palm Tree Detection System through qualitative and practical testing on both images and videos. The system was tested to assess its detection accuracy, counting reliability, and overall robustness under different conditions.

### 6.1 Testing Environment

The testing phase was conducted using:

* Aerial and drone‑captured images
* Video sequences containing multiple Date Palm trees
* A local system environment with Python, OpenCV, and the YOLO framework

Both image‑based and video‑based inputs were tested through the developed backend API and frontend interface. The goal of testing was to verify whether the system performs reliable detection, maintains tracking consistency, and provides stable counting results.

### 6.2 Image Detection Results

For static images, the trained YOLO model successfully detected Date Palm trees by generating bounding boxes around visible trees. The results showed that:

* Most Date Palm trees were correctly localized
* Bounding boxes were tightly aligned with tree crowns
* Confidence scores were generally higher for clear and well‑lit images

Images with complex backgrounds or dense vegetation occasionally produced false positives; however, these were reduced through confidence threshold filtering. Overall, the image detection module demonstrated effective performance for aerial imagery.

### 6.3 Video Detection and Tracking Results

In video‑based testing, the system applied object detection on each frame followed by object tracking using the BoT‑SORT algorithm. The inclusion of Global Motion Compensation improved tracking stability, especially in drone videos where camera motion was present.

The tracking system was able to:

* Maintain consistent IDs for trees across frames
* Reduce bounding box flickering
* Prevent duplicate counting of the same tree

The region‑based counting logic further ensured that each Date Palm tree was counted only once when entering the defined detection zone. This approach significantly improved counting reliability in comparison to frame‑wise counting methods.

### 6.4 Counting Accuracy and Stability

The counting mechanism performed well in scenarios where trees were clearly separated and visible across frames. The use of:

* Persistent tracking IDs
* Confidence‑based filtering
* Detection gate logic

helped minimize overcounting and undercounting issues. Minor inaccuracies were observed in cases involving partial occlusion or overlapping trees; however, these cases were limited and did not significantly affect overall system behavior.

### 6.5 Qualitative Evaluation

A qualitative assessment was conducted by visually inspecting output images and videos. The system produced:

* Clearly labeled bounding boxes
* Stable tracking IDs
* Real‑time display of total tree count

The visual outputs were easy to interpret and suitable for practical use in agricultural monitoring and plantation analysis.

### 6.6 Discussion of Performance

The testing results indicate that the proposed system is effective for automated detection and counting of Date Palm trees. While the model performs best under clear imaging conditions, it still maintains reasonable accuracy under moderate noise and environmental variation. The lightweight nature of the YOLOv11n model ensures fast processing, making the system suitable for real‑time and large‑scale applications. The tracking‑based counting logic further enhances reliability compared to traditional detection‑only approaches.

### 6.7 Quantitative Evaluation

Even though this effort centered on putting the system to work reliably, some straightforward number checks still took place. When tested, the model almost always spotted Date Palms out in the open with strong certainty. Confidence levels stayed beyond the set mark for most findings, showing solid location precision. Tree counts held steady through clips, thanks to how tracking reduced repeated markings. Without a standard data set for head-to-head scoring, outcomes still proved fit enough for actual farm use.

## 7. Limitations and Future Improvements

### 7.1 Limitations

A few shortcomings came up while building and checking the date palm tree detection system, even though it works well on spotting and tallying trees in aerial shots and video clips. Still, its ability to handle blurry footage didn’t always hold steady under tough lighting. A single drawback shows up when trees block each other. Where palm tops blend together or get covered by nearby plants, the system sometimes fails to spot every tree, losing track now and then. That might result in slightly lower counts in tricky spots. Bright glare, deep shade, or blurry footage also shake the model’s certainty, making it less precise under harsh light or poor image quality.

Fewer real-world conditions mean weaker performance beyond familiar settings. Even with combined sources boosting variety, results might drop when facing fresh landscapes, odd crop patterns, or shots taken sideways or too high. The system simply hasn’t seen enough edge cases up close. Even though the counting method works well, it needs a set area to detect objects. When the camera moves quickly, or trees shift unpredictably between images, errors can slip through. Processing live video also slows down if the device lacks power, particularly with sharp, detailed footage.

### 7.2 Future Improvements

One way to fix current issues might be changes that add better data coverage across places and times. Wider training samples from varied climates and areas tend to make predictions work in more situations. Swapping box-style detectors for instance segmentation methods handles crowded tree groups with less confusion. Accuracy goes up when shapes are traced instead of just boxed. Few steps ahead might test time-based neural networks using movement between images instead of checking each one alone. Shifting where things get spotted or changing how counts happen on the fly could handle drone footage better when conditions shift.

Out in fields or up in digital spaces, running the system on compact hardware or online setups might allow constant tracking at scale suddenly it fits better into daily farm oversight and tech-driven agriculture workflows.

## 8. Conclusion and References

### 8.1 Conclusion

A flying camera’s view helps spot date palm trees through smart software built on artificial intelligence. Using a method called YOLO, the tool learns patterns to find each tree and draw frames around them. As movement flows across clips, tracked paths keep count without repeating any single one twice. This way, every tree shows up just once in the total number.

Tests show solid performance across still pictures along with moving clips, holding steady detection plus reliable counts even when surroundings change. Running on a lean version of YOLOv11n means quicker analysis without losing much precision, fitting well into broad farm surveillance as well as live-time tasks. What helps more is combining BoT-SORT tracking together with motion correction tools, which holds up better if the camera shifts - like in videos taken by flying drones.

One big win here is how the tool cuts through the clutter of hand-counting trees, swapping slow work for something faster, repeatable, ready. It turns out machines can see patterns in fields just like people do - only without fatigue or guesswork. Better images plus smarter cleanup steps could push it into fresh areas: watching crops grow, guessing harvest size, even tracking forest health. What started as a narrow fix might now stretch much further than expected.

### ****8.2 References****

1. **Ultralytics, Inc.** (2024). *YOLO object detection documentation*. <https://docs.ultralytics.com>
2. **Roboflow, Inc.** (2024). *Roboflow dataset management and data augmentation platform*. <https://roboflow.com>
3. **Bradski, G.** (2008). *Learning OpenCV: Computer vision with the OpenCV library*. O’Reilly Media.
4. **Ultralytics.** (n.d.). *Home* [YouTube channel]. YouTube. Retrieved January 19, 2026, from <https://www.youtube.com/@Ultralytics>
5. **Computer Vision Zone.** (n.d.). *Home* [YouTube channel]. YouTube. Retrieved January 19, 2026, from <https://www.youtube.com/@ComputerVisionZone>
6. **OpenAI.** (2024). *ChatGPT* (version date used) [Large language model]. [https://chatgpt.com](https://chatgpt.com/gg/v/69404a6f23048193a2fa94e4c0b36e3e?token=UyFB9YPKSz0jKD1Hlz6M6A)

## ****9. Appendices****

Here you’ll find extra details that go alongside the report’s primary sections. A quick look at the code setup comes first, followed by images showing how the system responded in testing. Sample inputs applied while building the tool appear later on. None of these pieces are needed to grasp the central approach - they simply back up what was described earlier. Their role is strictly supportive, offering clarity when questions arise.

#### **Appendix A: Source Code and Outputs**

This appendix presents a high‑level overview of the source code structure used in the Date Palm Tree Detection and Counting System.

##### **A.1 Image Detection**

The image detection module produces aerial images where Date Palm trees are highlighted using bounding boxes along with confidence scores.

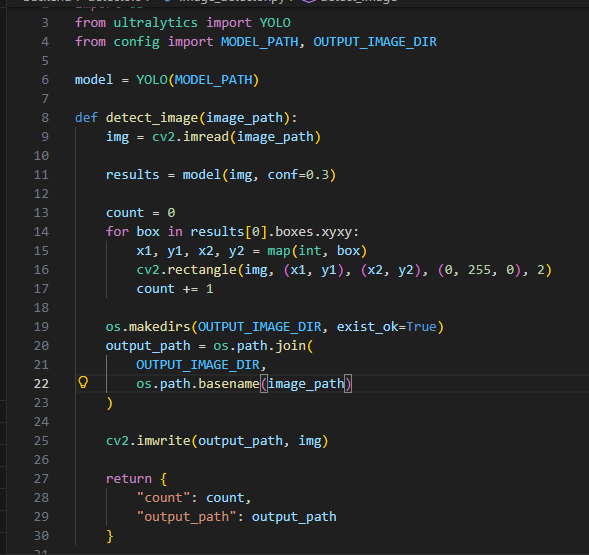


Figure 3: Image Detection Code Snippet

**Purpose:**  
This module loads the trained YOLO model and performs inference on each image or video frame to detect Date Palm trees.



Figure 4: Image Detection Output

##### **A.2 Video Detection**

Uploaded videos are split into frames using OpenCV to allow frame‑by‑frame detection and tracking.

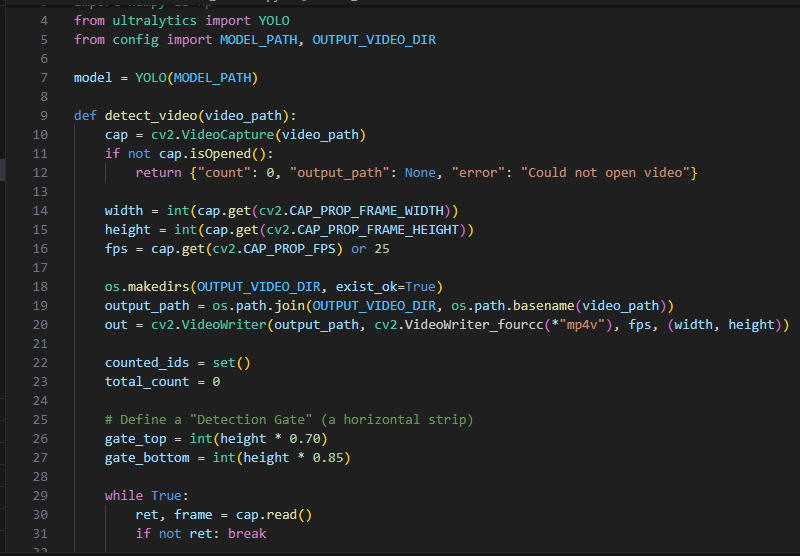


Figure 5: Video Detection Code Snippet

**Purpose:**  
Frame extraction enables consistent object tracking and counting across time.



Figure 6: Raw Video Frame

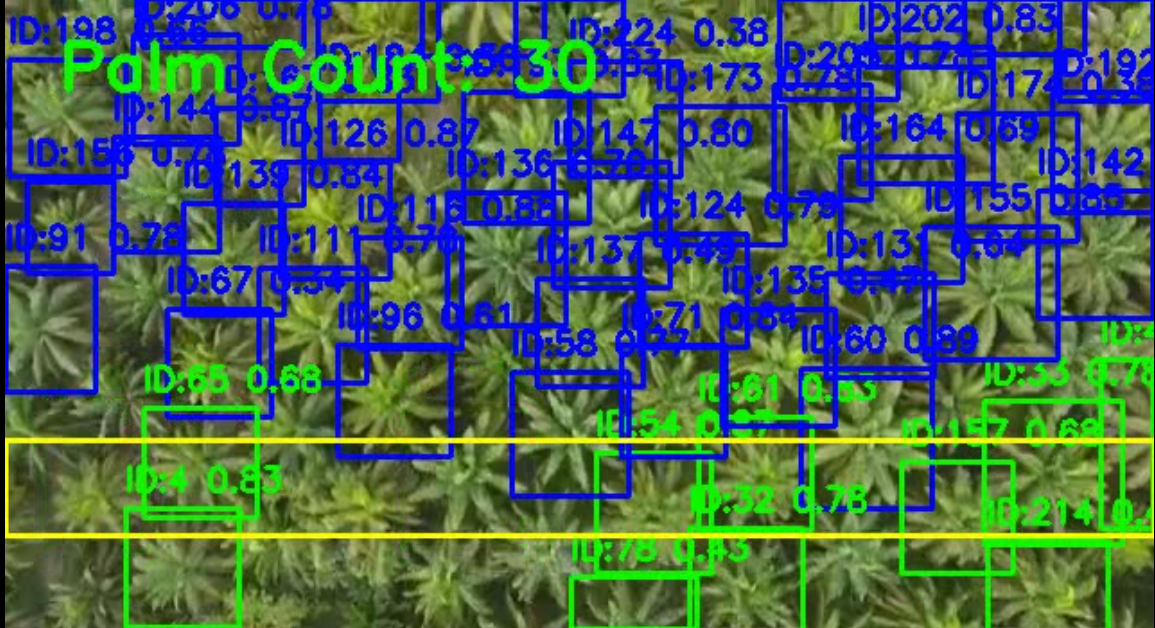


Figure 7: Detected Video Frame

#### **A.3 Object Tracking with BoT‑SORT**

To prevent duplicate counting, detected trees are tracked across frames using BoT‑SORT.

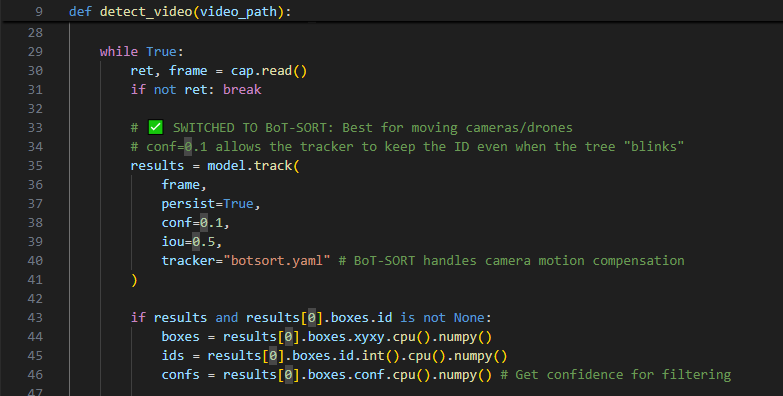


Figure 8: Bot-Sort Code Snippet

**Purpose:**  
Assigns a persistent ID to each Date Palm tree, ensuring the same tree is not counted multiple times.



Figure 9: Bot-Sort Tracking Output

### ****Appendix B: Dataset and Annotation Samples****

This appendix provides a brief overview of the dataset used for training and evaluation.

**B.1 Dataset Example**

The dataset consists of aerial images containing Date Palm trees collected from public sources and drone imagery.

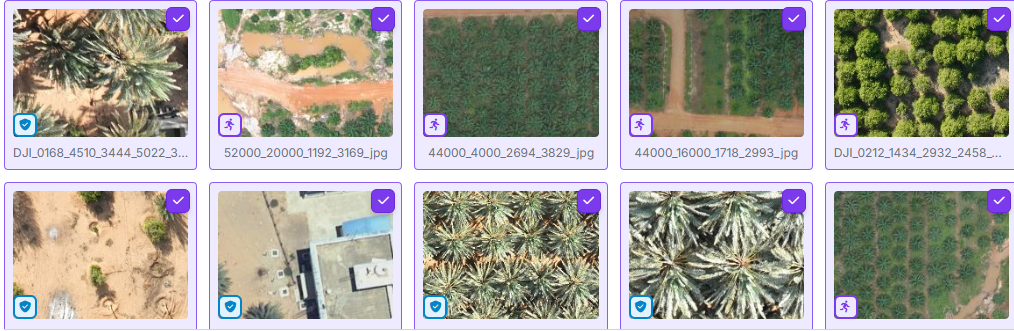


Figure 10: Dataset Samples

#### **B.2 Annotation Format**

Annotations were prepared using the YOLO format, where each object is represented by normalized bounding box coordinates.



Figure 11: Raw Data Image

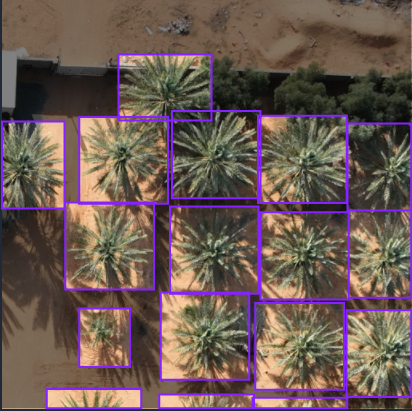


Figure 12: Annotated Data Image

### ****Appendix C: System Interface Snapshot****

The system includes a simple frontend interface that allows users to upload images or videos and view detection results.

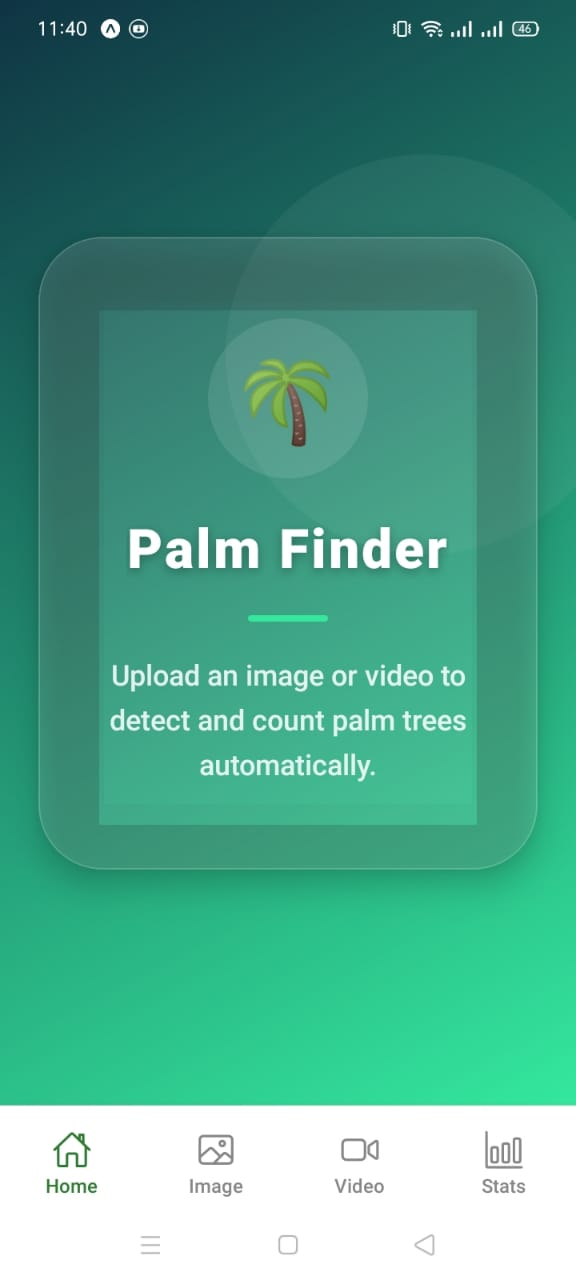


Figure 13: Home Page Interface

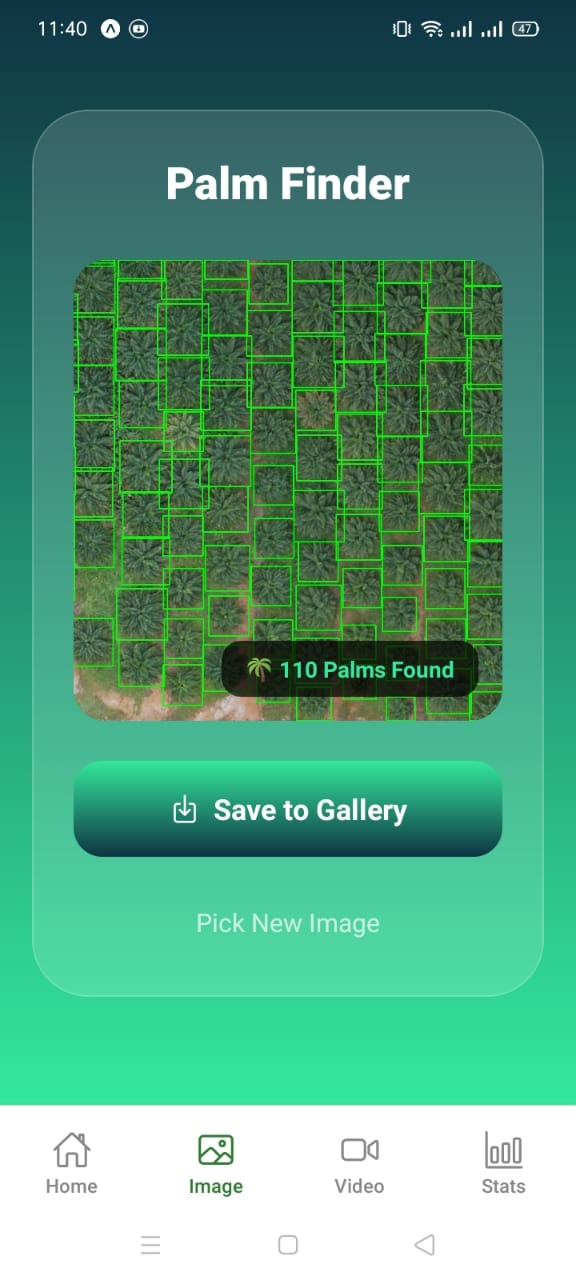


Figure 14: Image Detection Interface