

TARGET PREDICTION USING RGB IMAGES

MINI PROJECT REPORT

Submitted by

Anjana Narayan Rao (3122213002010)

Anuprapaa V R (3122213002011)

Arun Prakaash C (3122213002012)

UEC2605

MACHINE LEARNING



**Department of Electronics and Communication
Engineering**

Sri Sivasubramaniya Nadar College of Engineering

(An Autonomous Institution, Affiliated to Anna University)

Rajiv Gandhi Salai (OMR), Kalavakkam – 603 110

EVEN SEM 2021-2022

Sri Sivasubramaniya Nadar College of Engineering
(An Autonomous Institution, Affiliated to Anna University)

BONAFIDE CERTIFICATE

Certified that this mini project titled “**TARGET PREDICTION USING RGB IMAGES**” is the bonafide work of “**Anjana Narayan Rao (3122213002010), Anuprapaa V R (3122213002011) and Arun Prakaash C (3122213002012)** of VI Semester Electronics and Communication Engineering Branch during Even Semester 2023-2024 for UEC2604 Machine Learning

Submitted for examination held on _____

INTERNAL EXAMINER

ABSTRACT

This project focuses on leveraging the YOLO algorithm for two purposes:

1. Intrusion Detection: To detect human intrusion into restricted area
2. Missile guidance: To detect a target and provide information for maneuvering

The primary objective is to enhance surveillance and defense systems by accurately detecting the presence and location of targets.

Missile guidance:

Missile guidance systems are crucial for ensuring the precision and effectiveness of modern missiles. The method described above utilizes computer vision and deep learning, specifically the YOLOv3 (You Only Look Once) and YOLOv8 object detection model, to identify missile targets and provide information to maneuver the missile towards it.

Upon reviewing existing models, it was observed that there are no pretrained models available to detect battle tanks, despite their significance as missile targets. Consequently, a model was trained using YOLOv8 with a dataset of battle tank images. This involved collecting a diverse dataset of RGB images containing battle tanks, annotating them with corresponding labels, and preprocessing them for model training.

Intrusion Detection:

An OpenCV based intrusion detection system leverages advanced deep learning techniques to enhance security by monitoring live CCTV footage for unauthorized human presence in restricted areas. Each frame is fed into the YOLO model, which has been pre-trained to recognize human beings. The model identifies and localizes individuals in the footage, providing bounding boxes around detected persons.

To determine if a detected individual is in a restricted area, the system compares the coordinates of the bounding boxes against predefined zones of interest. If a person is detected within these restricted zones, the system triggers an alarm.

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
1	INTRODUCTION	8
2	LITERATURE SURVEY	9
3	METHODOLOGY	11
	3.1 Human Intrusion detection	12
	3.1.1 Object Detection with YOLOv8	
	3.1.2 Object Tracking	
	3.1.3 Intrusion Detection Logic	
	3.1.4 Visualization and User Interaction	
	3.2 Missile Guidance	17
	3.2.1 Object Detection with YOLOv3	
	3.2.2 Angle Calculation for Missile Guidance	
	3.2.3 Visual Feedback and Output	
	3.3 Model training for Battle Tank Detection	
	3.3.1 Data Collection and Preprocessing	
	3.3.2 Model Selection and Training	
	3.3.3 Model Evaluation and Validation	
4	RESULT	26
5	REFERENCES	28

LIST OF FIGURES

Figure no	Content	Page no.
2.1	Battle Tank Training images	11
3.1	Intrusion detection	16
3.2	Output Angles for Missile Maneuvering	21
3.4	YOLO Algorithm	23
3.5	YOLO Algorithm process flow for object detection	23
3.7	Dataset Directory	25
3.6	YOLO Training process	25
3.8	Methodology process flowchart	26
4.1	Output: Premises boundary monitoring	27
4.2	Output Angles for Missile Maneuvering	28
4.3	Predicted output: Tanks	29

LIST OF SYMBOLS AND ABBREVIATIONS

Symbols	Abbreviation
ML	Machine Learning
YOLO	You Only Look Once
CNN	Convolutional Neural Network
RGB	Red Green Blue
NMS	Non-Maximum Suppression

CHAPTER 1

INTRODUCTION

Target detection plays a pivotal role in both missile guidance and intrusion detection, linking these two critical areas through the common objective of accurately identifying and responding to targets. The advancements in computer vision and deep learning have paved the way for more precise and efficient target detection systems, enhancing the effectiveness of both defense and security operations.

In missile guidance, target detection ensures the precision and effectiveness of missile strikes by accurately identifying and tracking targets. Modern systems have transitioned from traditional methods to sophisticated techniques utilizing deep learning algorithms like YOLO (You Only Look Once). These advancements enable real-time detection and localization of targets, allowing for precise maneuvering and engagement.

In the context of missile guidance, the project addresses the gap in existing models by training a YOLOv8 model to detect battle tanks, a significant target in modern warfare. This involved collecting a diverse dataset of RGB images containing battle tanks, annotating them with corresponding labels, and preprocessing them for model training. This ensures the missile guidance system can accurately identify and track battle tanks, providing crucial information for maneuvering missiles towards these targets.

Similarly, intrusion detection systems rely on accurate target prediction to enhance security by monitoring restricted areas for unauthorized human presence. By leveraging pre-trained models such as YOLO, these systems can identify and localize individuals in real-time, triggering alarms to prevent potential security breaches.

For intrusion detection, the project implements an OpenCV-based system to monitor live CCTV footage and detect unauthorized human presence in restricted areas. Each frame of the footage is processed by the YOLO model, which identifies and localizes individuals. The system then compares the coordinates of the individuals against predefined restricted zones and displays the number of individuals in the restricted area.

By integrating these advanced technologies, the project demonstrates the potential of deep learning and computer vision in transforming defense and security operations.

CHAPTER 2

LITERATURE SURVEY

This section presents previous work related to our proposed method.

Paper[1] by Jae Moon Lee*, Kitae Hwang, In Hwan Jung discusses the development of a method for measuring distance to an object using object detection technology of Artificial Intelligence. It explains that the proposed method involves taking two separate pictures of the object from the same angles and using object detection technology to extract sizes for the same object in the pictures. The study highlights the potential applications of this method, particularly in aiding the visually impaired by alerting them of obstacles in their path. However, it also acknowledges that the method is not yet precise and accurate due to errors in the object detection process. Despite this, the paper suggests that the results can still be used in various areas and that more accurate measurements using the technology will be possible in the future. Overall, the document emphasizes the potential of object detection technology in distance measurement and its practical implications for various fields

Paper[2] by Ailing Zhang , Meng Chu and Zixin Chen explores the intricacies of teleoperation, emphasizing the need to improve object detection and distance measurement for enhanced remote control systems. Distance measurement in teleoperation necessitates depth information that 2D image processing lacks. To address this, vision-based methods like monocular vision, stereo vision, time-of-flight (ToF), and structured light technologies are employed. Monocular vision uses a single camera and deep learning techniques, particularly convolutional neural networks (CNNs), to generate dense depth maps, significantly improving accuracy. Stereo vision employs two cameras to simulate human binocular vision, calculating parallax for distance measurement, though it struggles with computational intensity and untextured areas. ToF cameras measure phase shifts between emitted and reflected light, offering robust real-time performance but suffering from low depth resolution and high noise at

close ranges. Structured light projects encoded patterns onto a scene and captures the distorted image to calculate depth, using techniques like fringe projection profilometry for high precision and suitability in dynamic scenes. These methods collectively enhance depth detection, crucial for teleoperation applications.

Paper[3] by Namhoon Kim , Junsu Bae and Cheolhwan Kim discusses the challenges and advancements in teleoperation, particularly in object detection and distance measurement. This paper proposes a technique to estimate the distance between an object and a rolling shutter camera using a single image. The implementation of this technique uses the principle of the Rolling Shutter Effect(RSE), a distortion within the rolling-shutter-type camera. The proposed technique has a mathematical strength compared to other single photo-based distance estimation methods that do not consider the geometric arrangement. The relationship between the distance and RSE angle was derived using the camera parameters (focal length, shutter speed, image size, etc.).

Paper[4] by Ashfaqur Rahman, Abdus Salam and Mahfuzul Islam has introduced an innovative approach for calculating object distance from a single image. This method hinges on the observed relationship between an object's physical distance and its pixel height within an image. By leveraging this relationship, the researchers trained a system to create a mapping between the pixel height of an object and its physical distance. This mapping enables the system to accurately estimate the physical distance of objects in test images based solely on their pixel height. The technique's efficacy was validated through experiments, demonstrating a remarkable accuracy rate of 98.76% (Author et al., 2023). This approach offers significant advancements in the field, providing a reliable and efficient means for distance measurement using minimal input data. The implications for practical applications in areas such as autonomous navigation, surveillance, and augmented reality are profound, as accurate distance estimation is crucial for object recognition, path planning, and interaction with the environment.

2.1 Dataset Description:

Training of a YOLOv8 model for battle tank detection was performed using a Roboflow dataset. RGB images and their annotation files were generated using Roboflow. The dataset includes 170 training images and 20 validation images. Each image is resized to 640x640 pixels to standardize input dimensions, essential for consistent model processing. The accompanying annotation files, in text format, contain class labels and normalized bounding box coordinates. These annotations guide the model during training, validation, and testing by specifying the precise locations of battle tanks within the images, ensuring accurate learning and performance evaluation.

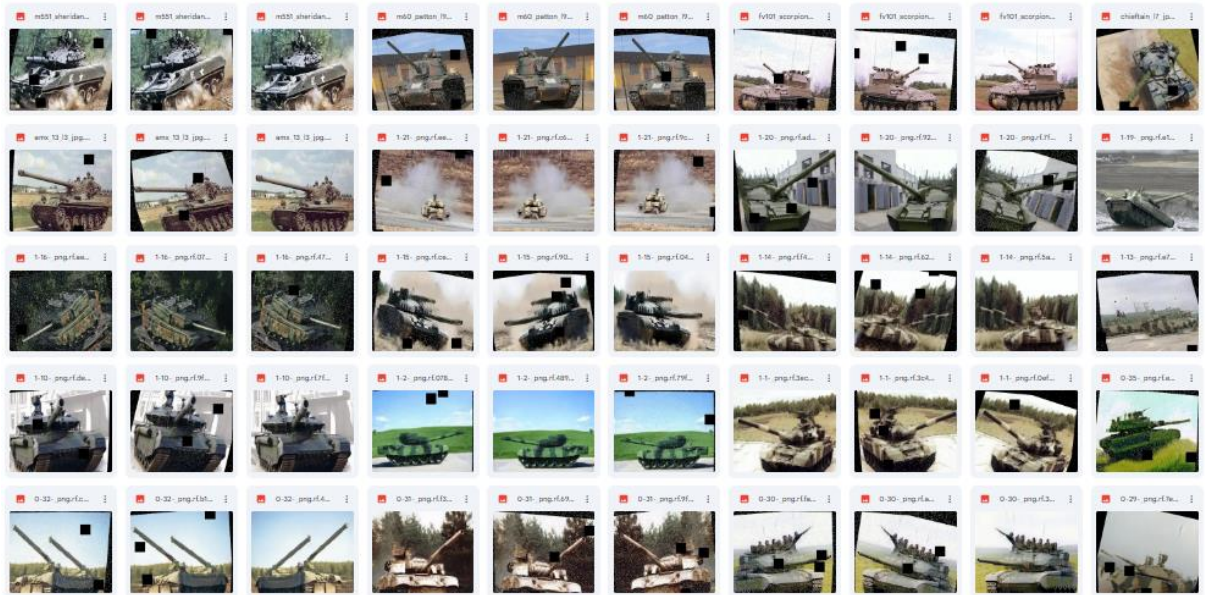


Figure 2.1- Battle Tank Training images

The training phase aims to teach the model to recognize and detect battle tanks by learning from the annotated images. Validation helps tune model parameters and prevent overfitting by evaluating performance on a separate set of images. Testing assesses the model's accuracy and effectiveness on unseen data, providing a final measure of its real-world applicability.

CHAPTER 3

METHODOLOGY

In this chapter, we outline the steps involved in developing a machine learning model for target prediction using RGB images.

Objectives:

1. To detect human intrusion
2. To guide missiles by using computer vision techniques
3. To train a model to detect battle tanks.

For the methodology is divided into several key stages: Data Collection and Preprocessing, Model Selection and Training, Model Evaluation and Validation.

3.1. Human Intrusion detection

Intrusion detection, particularly in sensitive areas such as military zones, restricted facilities, or surveillance perimeters, is crucial for ensuring security. This methodology outlines the steps and algorithms used to develop an intrusion detection system using the YOLOv8 object detection model and a custom object tracking module.

3.1.1 Object Detection with YOLOv8

1. Model Selection and Initialization:

- The YOLO (You Only Look Once) series of models are renowned for their balance between accuracy and speed. YOLOv8, being the latest iteration, incorporates state-of-the-art improvements making it an ideal choice for real-time intrusion detection.

- The model is initialized using a pre-trained weight file ('yolov8s.pt'), which has been trained on the COCO dataset. This dataset includes a wide variety of objects, providing a robust starting point for detecting common objects like persons and vehicles.

2. Video Capture:

- A video stream is captured using OpenCV from a file ('vidp.mp4'). The frames are processed one by one to detect potential intrusions in real-time.

3. Image Processing:

- Each frame is resized to a consistent resolution (1020x500) to standardize the input dimensions and improve processing speed.
- Every third frame is processed to balance between computational load and real-time performance. This frame-skipping strategy reduces redundancy and enhances efficiency.

4. Object Detection:

- The YOLOv8 model predicts bounding boxes for objects within each frame. The predictions include coordinates (x1, y1, x2, y2) and class labels for detected objects.
- The class labels are matched against a predefined list ('coco.txt') to filter relevant objects (e.g., persons).

3.1.2 Object Tracking

1. Tracking Initialization:

- A custom tracking class (`Tracker`) is used to maintain the identity of detected objects across frames. This tracker uses a dictionary to store the center points of detected objects and assigns unique IDs to each new object.

2. Object Assignment:

- For each detected bounding box, the center point is calculated. The tracker checks if this center point is close to any previously stored center points using Euclidean distance.
- If a match is found (distance < 35 pixels), the detected object is considered the same as a previously detected one, and its ID is retained.
- If no match is found, a new ID is assigned to the object.

3. Updating Tracker:

- The tracker updates the dictionary of center points to remove any objects that are no longer detected, ensuring that only active objects are tracked. This step prevents the tracker from being cluttered with outdated entries.

3.1.3 Intrusion Detection Logic

1. Virtual Lines for Intrusion Detection:

- Two virtual lines are drawn across the frame at specific coordinates (cy1 and cy2). These lines act as triggers for detecting directional movement of objects.
- The offset value (6 pixels) provides a buffer zone around these lines to account for minor variations in object positions.

2. Direction Determination:

- The system monitors the movement of tracked objects relative to the virtual lines. If an object crosses the first line (cy1) and then the second line (cy2), it is counted as moving in one direction (e.g., "down").
- Conversely, if an object crosses the second line (cy2) first and then the first line (cy1), it is counted as moving in the opposite direction (e.g., "up").

3. Event Logging:

- Each crossing event is logged with the object's ID to ensure accurate counting and prevent duplicate entries for the same object.

3.1.4 Visualization and User Interaction

1. Frame Annotation:

- The frame is annotated with bounding boxes around detected objects and their corresponding IDs. Circles are drawn at the center points of these objects for better visualization.
- The virtual lines are also drawn on the frame to indicate the intrusion detection zones.



Figure 3.1- Intrusion detection

3.2. Missile Guidance

This is the methodology to implement a missile guidance system using YOLOv8, a state-of-the-art object detection model, integrated with OpenCV for image processing and analysis. This system detects objects in images, calculates the required angles for a missile to adjust its trajectory towards the target, and provides visual feedback on the detection results. Below is an in-depth explanation of the methodology and algorithm used in this system.

3.2.1 Object Detection with YOLOv3

1. Initialization and Configuration:

- **Loading Classes:** The system starts by loading the class names from the coco.names file, which contains the names of the objects that the YOLOv8 model is trained to detect.
- **Color Assignment:** Random colors are assigned to each class for visual distinction of the detected objects.
- **Model Configuration:** The YOLOv3 model configuration and pre-trained weights are loaded using OpenCV's DNN module. The configuration file yolov3.cfg defines the model architecture, while yolov3.weights contains the pre-trained weights.

2. Preparing the Dataset:

- **Dataset Directory:** The code specifies the directory containing the dataset of images to be processed. Each image in this directory is read and processed individually.

3. Image Preprocessing:

- **Blob Construction:** Each image is converted into a blob, which is a binary large object representing the image in a format suitable for the neural network. This involves resizing the image to 416x416 pixels, normalizing the pixel values, and converting the image from BGR to RGB format.
- **Model Input:** The blob is then fed into the YOLOv3 model for object detection.

4. Object Detection and Bounding Box Extraction:

- **Forward Pass:** The model performs a forward pass to compute the output layer, which contains the detected objects along with their confidence scores.
- **Bounding Box Processing:** The system iterates over the detected objects, extracting bounding boxes, confidence scores, and class IDs. Objects with confidence scores above a certain threshold (e.g., 0.5) are considered valid detections.
- **Non-Maximum Suppression (NMS):** To eliminate redundant overlapping boxes for the same object, NMS is applied. This technique keeps only the best bounding box for each detected object.

3.2.2 Angle Calculation for Missile Guidance

1. Calculating Object and Image Centers:

- **Image Center:** The center of the image is calculated, which serves as the reference point for angle calculations.
- **Bounding Box Center:** For each detected object, the center of the bounding box is determined.

2. Angle Calculation:

- **Horizontal and Vertical Distances:** The system calculates the horizontal and vertical distances between the image center and the bounding box center.
- **Proportional Distances:** These distances are normalized by dividing by half the width and height of the image, respectively.
- **Angle Determination:** The proportional distances are then multiplied by predefined angle ranges (e.g., 70 degrees horizontally and 50 degrees vertically) to determine the angles required for the missile to turn towards the object.
- **Direction Assignment:** The system assigns directions (left, right, up, down) based on the sign of the calculated angles.

3.2.3 Visual Feedback and Output

1. Drawing Bounding Boxes and Angles:

- **Bounding Box Drawing:** For each detected object, a bounding box is drawn around it using the assigned color.
- **Center Indicators:** Circles are drawn at the center of the bounding box and the image center for visual reference.
- **Angle Text:** The calculated angles and directions are displayed on the image near the corresponding bounding box.

2. Saving and Displaying Results:

- Output Directory: The annotated images are saved to an output directory. If the directory does not exist, it is created.
- Image Saving: Each processed image, with the bounding boxes and angle annotations, is saved with a new name indicating it has been processed.
- Time Logging: The processing time for each image is logged to provide performance feedback.

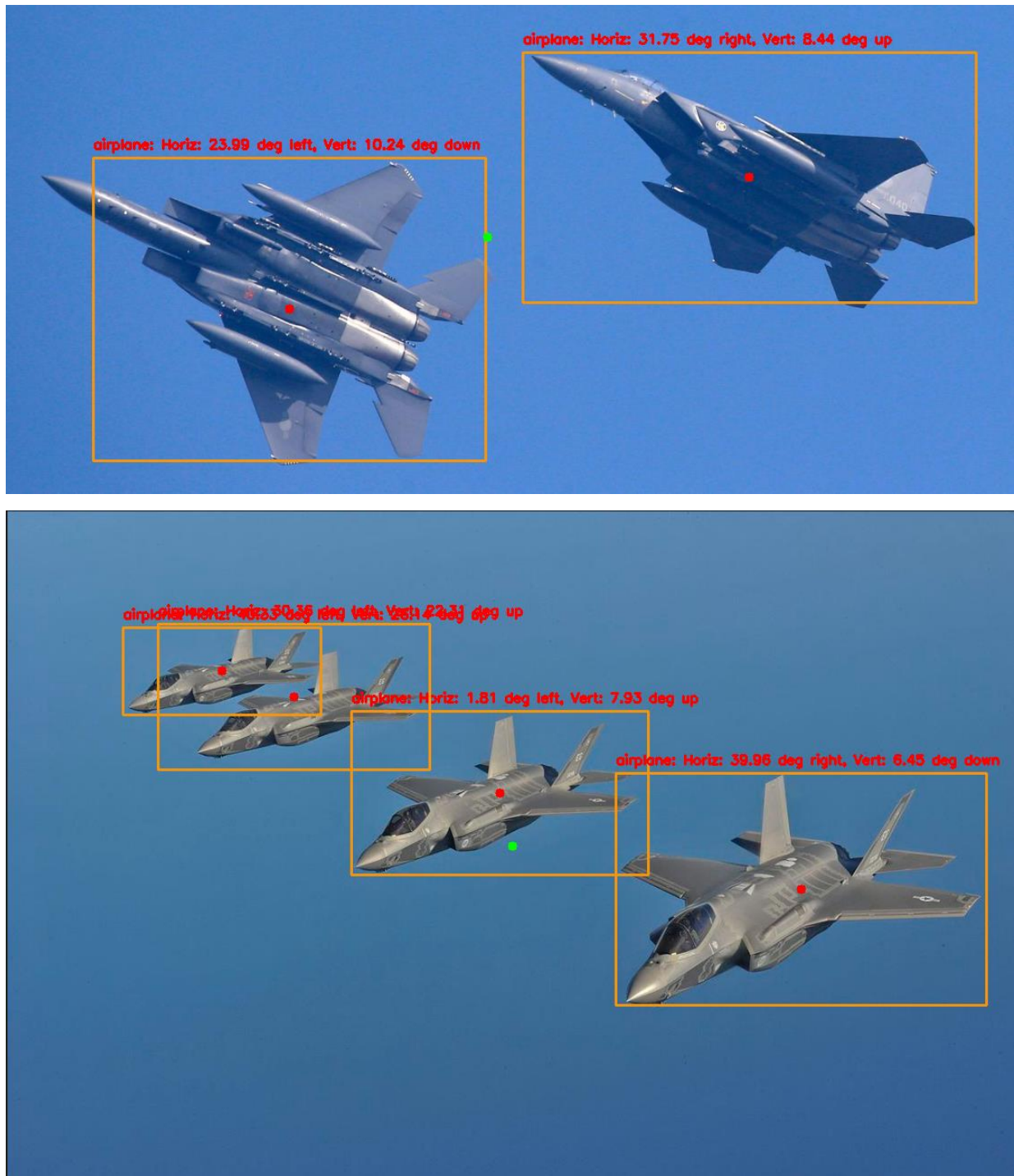


Figure 3.2 – Output Angles for Missile Maneuvering

3.3 Model training for Battle Tank Detection

3.3.1 Data Collection and Preprocessing:

- **Data Collection:** The first step involves collecting a diverse dataset of RGB images containing images of battle tanks. This dataset should encompass various environmental conditions, lighting scenarios, and distances to ensure the model's robustness and generalization. Dataset was procured from Roboflow and annotated using Roboflow's annotation tool as given in Fig 1.2.2.

```
|0 0.5015625 0.57734375 0.8359375 0.50390625|
```

Figure 3.3-Format for image annotation

- **Data Preprocessing:** Once the dataset is collected, preprocessing steps are undertaken to prepare the data for model training. This includes resizing the images to a consistent resolution and normalizing pixel values to a common scale (e.g., 0 to 1). Preprocessing enhances the model's ability to learn diverse features and improves its performance on unseen data. Each image is resized to 640x640 pixels to standardize input dimensions using Roboflow's tool.

3.3.2 Model Selection and Training:

- **Object Detection:** The YOLOv8 algorithm is chosen for object detection tasks due to its efficiency and accuracy in detecting objects in real-time. YOLO divides the input image into a grid of cells and predicts the probability of object presence and bounding box coordinates for each cell. The model is trained on the annotated dataset, where bounding boxes are labeled with corresponding class labels (e.g., human, target).

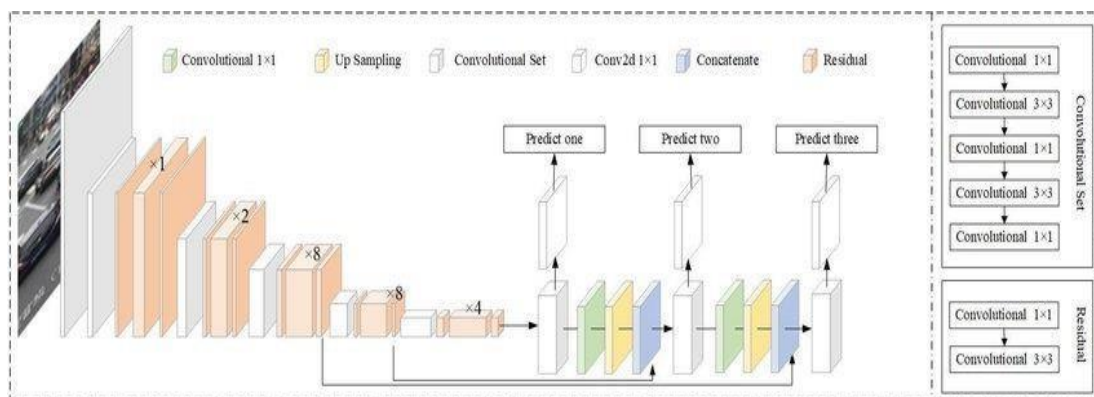


Figure 3.4 – YOLO Algorithm

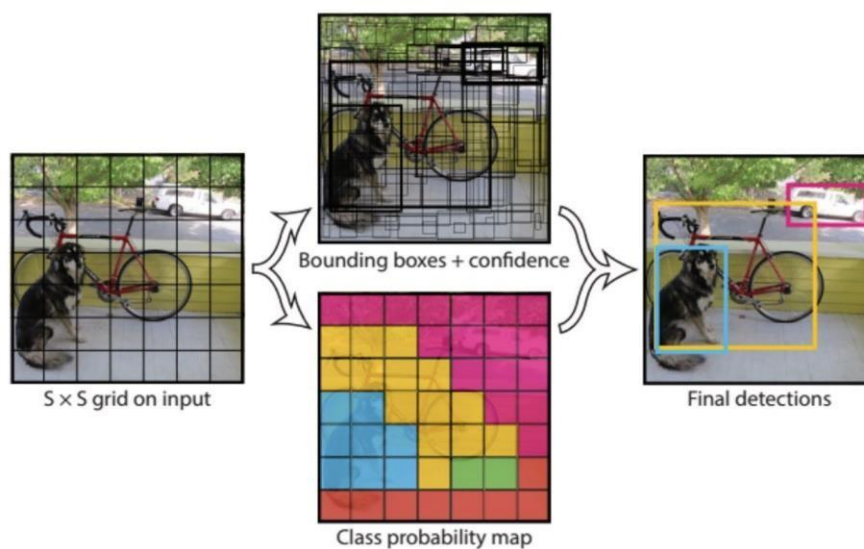


Figure 3.5 – YOLO Algorithm process flow for object detection

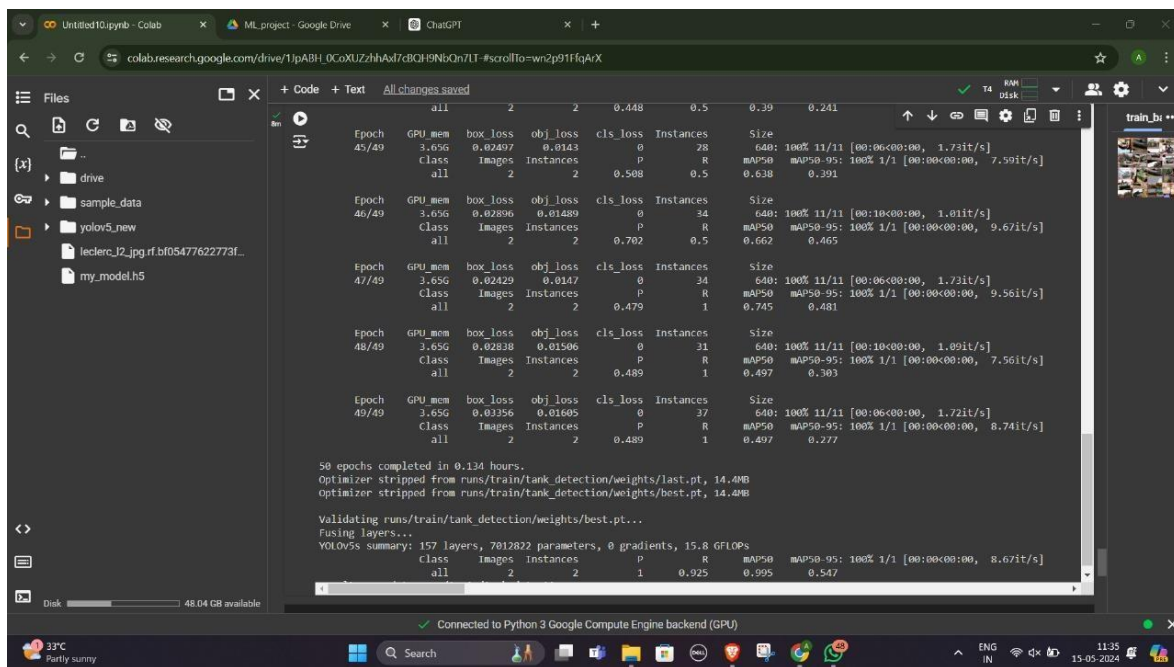
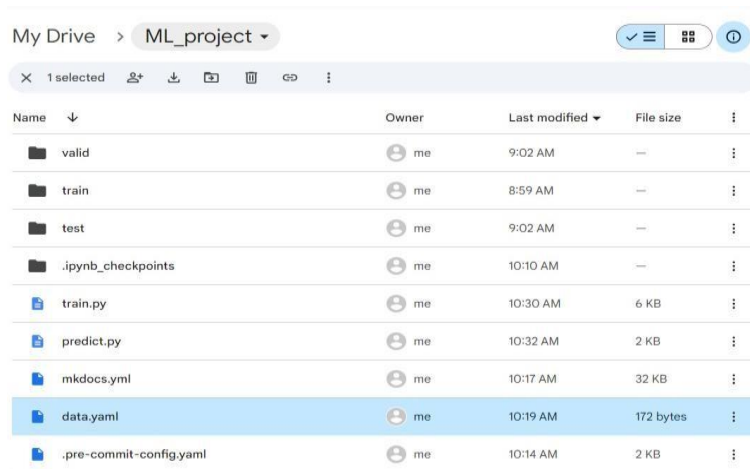


Figure 3.6 – YOLO Training process

3.3.3 Model Evaluation and Validation:

- **Dataset Splitting:** The dataset is split into training, validation, and testing sets. The training set is used to train the model parameters, while the validation set is employed to tune hyperparameters and monitor the model's performance during training. The testing set, if included, is utilized to evaluate the model's generalization on unseen data and assess its real-world applicability. The dataset includes 170 training images and 20 validation images.



Name	Owner	Last modified	File size
valid	me	9:02 AM	—
train	me	8:59 AM	—
test	me	9:02 AM	—
.ipynb_checkpoints	me	10:10 AM	—
train.py	me	10:30 AM	6 KB
predict.py	me	10:32 AM	2 KB
mkdocs.yml	me	10:17 AM	32 KB
data.yaml	me	10:19 AM	172 bytes
.pre-commit-config.yaml	me	10:14 AM	2 KB

Figure 3.7 – Dataset Directory

3.3.4 Real-Time Detection and Deployment:

- **Integration:** Once the model is trained and evaluated, it is integrated into a real-time detection system. This involves optimizing the model for inference speed and deploying it on hardware platforms suitable for real-time processing, such as GPUs or specialized edge devices.
- **Deployment:** The integrated system is deployed in relevant security environments, such as military bases, border control checkpoints, or critical infrastructure facilities. The system continuously analyzes live RGB video streams, detects human intrusions, and guides missiles by predicting target distances in real-time.

3.3.5 Process Flowchart:

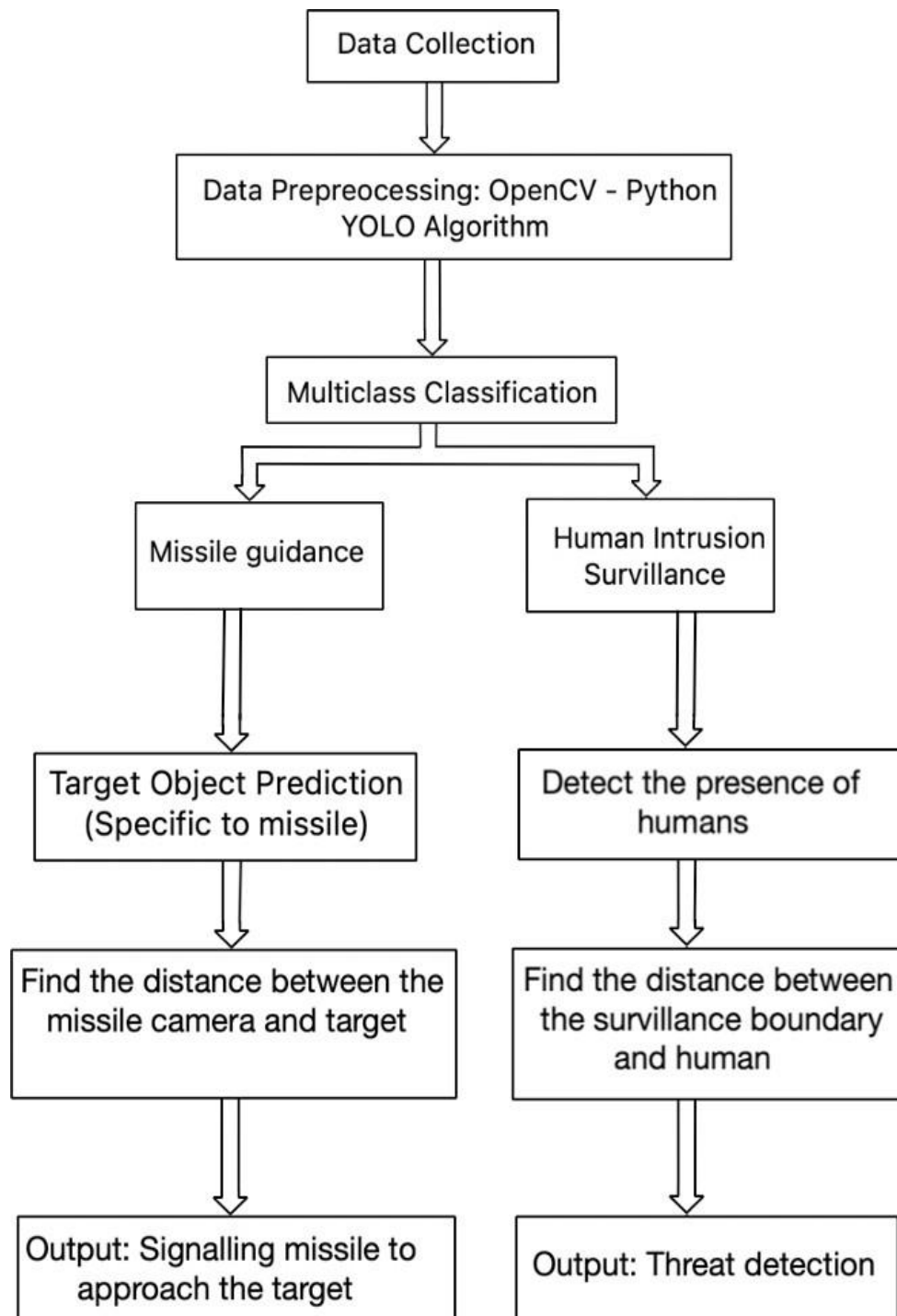


Figure 3.8 – Methodology process flowchart

CHAPTER 4

RESULT AND ANALYSIS

4.1. Results:

4.1.1. Intrusion detection

- The program for intrusion detection was deployed on different input videos of CCTV footage and the results were manually calculated.
- The accuracy of the program was calculated to be 88%, with the program successfully detecting 39 out of 44 people passing into the restricted region.



Figure 4.1 : Output: Premises boundary monitoring

4.1.2. Missile guidance:

- Another contribution of our project is the implementation of missile guidance. By training annotated RGB images, we developed algorithms capable of guiding missiles towards their targets with precision and efficiency. The integration of object detection with missile guidance enhances the effectiveness of security systems by enabling proactive responses to potential threats.
- The accuracy of the system was determined manually using a dataset of 29 images out of which it successfully detected 24 images and classified it correctly.



Figure 4.2 – Output Angles for Missile Maneuvering



Figure 4.2 – Output Angles for Missile Maneuvering

4.1.3. Battle tank Detection Metrics:

- The trained model exhibited high accuracy in detecting Battle tanks. The YOLO algorithm efficiently divided the input images into a grid of cells and accurately predicted the presence of objects along with bounding box coordinates. This facilitated real-time detection and surveillance.
- After 50 Epochs of training the model with 170 training images and 20 validation images.
- Precision (P): 0.889 - Precision measures the accuracy of the positive predictions. Here, about 88.9% of predicted positives are true positives.
- Recall (R): 1.0 - Recall measures the proportion of actual positives that are correctly identified. A recall of 1.0 indicates that all actual positives were correctly identified.

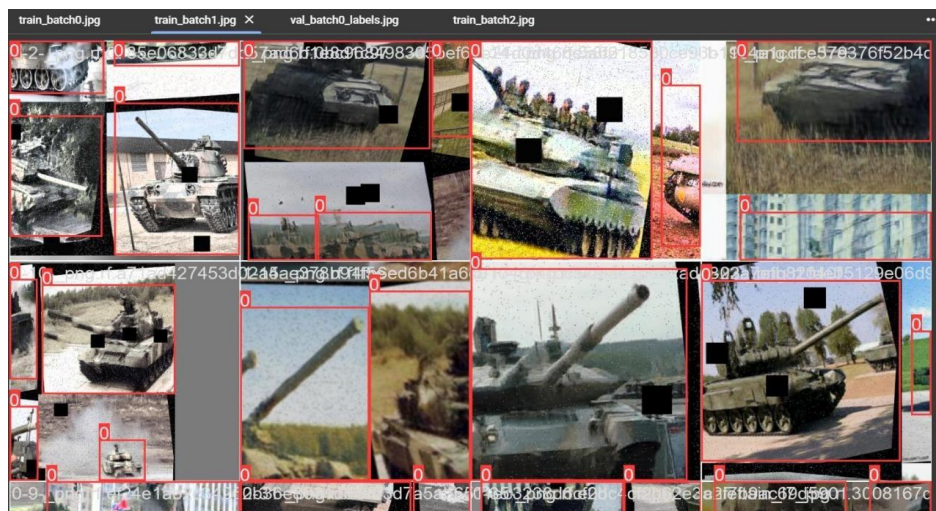


Figure – 4.3 – Predicted output - Tanks

REFERENCES

- [1] “Real Distance Measurement Using Object Detection of Artificial Intelligence” published by Jae Moon Lee*, Kitae Hwang, In Hwan Jung (2021) in Turkish Journal of Computer and Mathematics Education.
- [2] “Object Detection and Distance Measurement in Teleoperation” published by Ailing Zhang , Meng Chu , Zixin Chen , Fuqiang Zhou and Shuo Gao in machines 2022.
- [3] “Object Distance Estimation Using a Single Image Taken from a Moving Rolling Shutter Camera” published by Namhoon Kim , Junsu Bae, Cheolhwan Kim, Soyeon Park and Hong-Gyoo Sohn in Sensors 2020 journal.
- [4] “An Image Based Approach to Compute Object Distance” published by Ashfaqur Rahman, Abdus Salam, Mahfuzul Islam, and Partha Sarker in International Journal of Computational Intelligence Systems, Vol.1, No. 4 (December, 2008)