



Ms. NATASHA ROBIN

Indoor Anomaly Detection in Smart City Environments

MSc Robotics
Thesis

MSc
Academic Year: 2024 - 2025

Supervisor: Dr Gilbert Tang, Dr Boyu Kuang
August 2025



School of Aerospace, Transport and Manufacturing
MSc Robotics

MSc

Academic Year 2024 - 2025

Ms. NATASHA ROBIN

Indoor Anomaly Detection in Smart City Environments

Supervisor: Dr Gilbert Tang, Dr Boyu Kuang
August, 2025

This thesis is submitted in partial fulfilment of the requirements for
the degree of MSc Robotics

© Cranfield University 2025. All rights reserved. No part of this
publication may be reproduced without the written permission of the
copyright owner.

Academic integrity declaration

I declare that:

- the thesis submitted has been written by me alone.
- the thesis submitted has not been previously submitted to this university or any other.
- that all content, including primary and/or secondary data, is true to the best of my knowledge.
- that all quotations and references have been duly acknowledged according to the requirements of academic research.

I understand that to knowingly submit work in violation of the above statement will be considered by examiners as academic misconduct.

Abstract

Hotel corridors often have unattended items like cutlery and utensils left on the corridors, which is unappealing and a trip hazard. Going around the hotels and clearing these items is time-consuming and labour-intensive. Automating this process can reduce workload. This thesis presents a multi-stage machine vision system to detect utensils such as cups, plates, glasses, trays, and cutlery items, including spoons, forks, and butterknives. The system integrates YOLO for object detection with ConvNeXt and ViT as classification models. The system also incorporates the Segment Anything Model (SAM) for improved object isolation. Custom datasets were created for the training and testing of the system in various environments. Different systems were created by using the above-mentioned models, and rigorous testing was conducted to evaluate all the systems' robustness under diverse conditions using precision, recall, and F1-score metrics, with the best-performing system achieving an F1 score of 0.7161 in the corridor setting. The system was then integrated with a robot, enabling real-time object detection. This work contributes to a robust, adaptable system for automated monitoring in the hospitality sector.

Keywords:

Object detection, Utensil and Cutlery Detection, Hotel Service Robot, Multi-model fusion, Machine vision systems

Acknowledgements

I would like to extend my deepest gratitude to Dr Gilbert Tang, Dr Boyu Kuang, my supervisors, for their guidance, encouragement and support throughout the completion of this assignment. Their feedback and suggestions have been invaluable, their mentorship has greatly enhanced the quality of the work. I would like to thank lab staffs as well, who provided technical support throughout the project. I would also like to thank Swift Robotics, especially Mr Ian Pulford and Mr Sairaj Dillikar, for providing the robot and helping with the integration. I am also grateful to Cranfield University and the faculty of the MSc Robotics program for providing the resources, tools, and conducive learning environment necessary to undertake this project. The university's commitment to fostering practical, hands-on learning has enabled me to translate theoretical concepts into real-world applications, further deepening my knowledge in this field.

Table of Contents

Academic integrity declaration.....	i
Abstract	ii
Acknowledgements	iii
Table of Contents	iv
List of Figures.....	vi
List of Tables	x
List of Equations.....	xii
List of Abbreviations	xiii
1 INTRODUCTION.....	14
1.1 Background and Motivation	14
1.2 Problem Statement	15
1.3 Scope Of Work.....	15
1.3.1 Aim	15
1.3.2 Objectives	15
1.3.3 Limitations	16
1.4 Technological Contribution	16
1.5 Thesis Structure.....	17
2 LITERATURE REVIEW	18
2.1 Evolution of object detection	18
2.2 Detection of anomalies	19
2.2.1 Medical domain	19
2.2.2 Environmental hazards.....	19
2.2.3 Surveillance.....	19
2.3 Detection of objects	21
2.3.1 YOLO based detections	21
2.3.2 Two-stage detection architecture	22
2.3.3 Fusion approach	22
2.4 Detection of falls	23
2.4.1 Sensor-based.....	24
2.4.2 Vison based	24
2.5 Summary	25
3 METHODOLOGY	27
3.1 Dataset configuration	27
3.1.1 YOLOv11 Dataset	27
3.1.2 Non-SAM (segment anything model) crop dataset.....	29
3.1.3 SAM (segment anything model) crop dataset	31
3.1.4 Datasets for testing the systems	34
3.2 Framework design	50
3.3 System architecture	52
3.3.1 Object detection	52

3.3.2 Segmentation	52
3.3.3 Pre-Processing.....	54
3.3.4 Classifiers.....	54
3.3.5 Fusion Logic.....	55
3.3.6 IoU-Based Post-Filtering	55
4 EXPERIMENT DESIGN	57
4.1.1 Base Model Selection.....	57
4.1.2 YOLO-Classifer Combination Testing.....	57
4.1.3 SAM Integration and Testing.....	57
4.1.4 Testing of the Model in Different Heights and Distances.....	58
4.1.5 Real-world robotic integration.....	59
5 RESULTS AND DISCUSSIONS.....	60
5.1 Base Model Selection	61
5.2 YOLO-classifier combination testing	65
5.2.1 Curves of ConvNeXt trained on non-SAM dataset	66
5.2.2 Curves of ViT trained on non-SAM dataset	68
5.3 SAM integration and testing.....	69
5.3.1 Comparison of models on sunny environment	70
5.3.2 Comparison of models on gloomy environment	71
5.3.3 Comparison of models on shady environment	72
5.3.4 Comparison of models on corridor environment.....	72
5.4 Testing of the model in different heights and distances	78
5.4.1 Height 60 cm	78
5.4.2 Height 110cm	78
5.4.3 Height 170 cm	79
5.5 Real-world robotic integration	79
5.5.1 Testing the robot-integrated system in different environments.....	80
5.5.2 Testing the robot-integrated system on different tilt angles	91
6 CONCLUSION AND FUTURE WORK	100
References	102
Appendices.....	108
Appendix A Ethical approval letter	108
Appendix B SYSTEM CODES AND DIAGRAMS	109
Appendix C Gantt chart.....	141
Appendix D Risk assessment	142

List of Figures

Figure 1 Utensils left on the corridors	14
Figure 2 Comparison of YOLOv3	20
Figure 3 Distribution of objects in the train set of YOLOv11 Datasets	28
Figure 4 Distribution of objects in the test set of YOLOv11 Datasets	28
Figure 5 Distribution of objects in the validation set of YOLOv11 Datasets	29
Figure 6 Annotation of classes of YOLOv11 Datasets	29
Figure 7 Distribution of objects in the train set of Non-SAM Datasets	30
Figure 8 Distribution of objects in the test set of Non-SAM Datasets	30
Figure 9 Distribution of objects in the validation set of Non-SAM Datasets	31
Figure 10 Classes of Non-SAM Datasets	31
Figure 11 Distribution of objects in the train set of SAM Datasets	32
Figure 12 Distribution of objects in the test set of SAM Datasets	32
Figure 13 Distribution of objects in the validation set of SAM Datasets	33
Figure 14 Classes of SAM Datasets	33
Figure 15 Images in datasets for comparing systems	36
Figure 16 Dataset images in sunny environment	37
Figure 17 Dataset images in gloomy environment	39
Figure 18 Dataset images in a shade environment	40
Figure 19 Dataset images in a corridor environment	42
Figure 20 Dataset images for different heights and distances	45
Figure 21 Detection frames from dark background	46
Figure 22 Original frames from dark background	47
Figure 23 Detection frames from white background	47
Figure 24 Original frames from white background	48
Figure 25 Detection frames from corridor	48
Figure 26 Original frames from corridor	49
Figure 27 Detection frames for different tilt angles	49
Figure 28 Original frames for different tilt angles	50

Figure 29 Block diagram for the system framework	51
Figure 30 Example of full mask and submasks	53
Figure 31 Block Diagram of the final system architecture	56
Figure 32 Validation and training curve of YOLOv11-XL.....	62
Figure 33 Precision and Recall curve of YOLOv11-XL.....	63
Figure 34 Confusion matrix for YOLOv11-XL.....	64
Figure 35 Training and validation curve for ConvNeXt trained on Non-SAM (Segment Anything Model) Crop Dataset	66
Figure 36 precision-recall curve for ConvNeXt trained on non-SAM (segment anything model) crop dataset	67
Figure 37 Confusion matrix for ConvNeXt trained on non-SAM (Segment Anything Model) Crop Dataset.....	67
Figure 38 Training and validation curve for ViT trained on non-SAM (Segment Anything Model) Crop Dataset.....	68
Figure 39 Precision-recall curve for ViT trained on NON-SAM (SEGMENT ANYTHING MODEL) CROP DATASET	68
Figure 40 Confusion matrix for ViT trained on NON-SAM (SEGMENT ANYTHING MODEL) CROP DATASET	69
Figure 41 Training and validation curve for ConvNeXt trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET	75
Figure 42 Precision-Recall curve for ConvNeXt trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET	75
Figure 43 Confusion matrix for ConvNeXt trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET	76
Figure 44 Training and validation curve for ViT trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET	76
Figure 45 Precision-Recall curve for ViT trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET	77
Figure 46 Confusion matrix for ViT trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET	77
Figure 47 Object detection screen.....	80
Figure 48 Confusion matrix of 2-frame logic.....	87
Figure 49 Confusion matrix of 5-frame logic.....	87
Figure 50 Box and whiskers plot of 5-frame and 2-frame logic.....	89

Figure 51 Robot detection frame	90
Figure 52 The stopped robot	90
Figure 53 Camera position	92
Figure 54 Confusion matrix of tilt 20°	93
Figure 55 Confusion matrix of tilt 30°	94
Figure 56 Confusion matrix of tilt 40°	95
Figure 57 Confusion matrix of tilt 50°	96
Figure 58 Confusion matrix of tilt 60°	97
Figure 59 Box and whiskers plot for different tilt angles	98
Figure B-1 Confusion matrix tilt 20° , Test 1	126
Figure B-2 Confusion matrix tilt 20° , Test 2	127
Figure B-3 Confusion matrix tilt 20° , Test 3	127
Figure B-4 Confusion matrix tilt 20° , Test 4	128
Figure B-5 Confusion matrix tilt 20° , Test 5	128
Figure B-6 Confusion matrix tilt 30° , Test 1	129
Figure B-7 Confusion matrix tilt 30° , Test 2	130
Figure B-8 Confusion matrix tilt 30° , Test 3	130
Figure B-9 Confusion matrix tilt 30° , Test 4	131
Figure B-10 Confusion matrix tilt 30° , Test 5	131
Figure B-11 Confusion matrix tilt 40° , Test 1	132
Figure B-12 Confusion matrix tilt 40° , Test 2	133
Figure B-13 Confusion matrix tilt 40° , Test 3	133
Figure B-14 Confusion matrix tilt 40° , Test 4	134
Figure B-15 Confusion matrix tilt 40° , Test 5	134
Figure B-16 Confusion matrix tilt 50° , Test 1	135
Figure B-17 Confusion matrix tilt 50° , Test 2	135
Figure B-18 Confusion matrix tilt 50° , Test 3	136
Figure B-19 Confusion matrix tilt 50° , Test 4	137

Figure B-20 Confusion matrix tilt 50° , Test 5	137
Figure B-21 Confusion matrix tilt 60° , Test 1	138
Figure B-22 Confusion matrix tilt 60° , Test 2	138
Figure B-23 Confusion matrix tilt 60° , Test 3	139
Figure B-24 Confusion matrix tilt 60° , Test 4	139
Figure B-25 Confusion matrix tilt 60° , Test 5	140
Figure C-1 Gantt chart.....	141

List of Tables

Table 1 Comparison of the pan-sharpening method	21
Table 2 Results on the VOD dataset	23
Table 3 Result on Flow dataset	23
Table 4 Detecting falls at three locations	24
Table 5 Number of images across all test, validation, and train datasets	34
Table 6 Distribution of classes in datasets for comparing systems	35
Table 7 Dataset distribution in sunny environment	37
Table 8 Dataset distribution in gloomy environment	38
Table 9 Dataset distribution in a shade environment	40
Table 10 Dataset distribution in a corridor environment	41
Table 11 Data Distribution of different heights and distances	43
Table 12 Comparison of YOLOv11 Models	61
Table 13 Comparison between YOLOv11 and Classifier models	65
Table 14 Comparison of the models in sunny environment	70
Table 15 Comparison of the models in gloomy environment	71
Table 16 Comparison of the models in shady environment	72
Table 17 Comparison of the models in corridor environment	72
Table 18 YOLOv11-XL in different categories	73
Table 19 Comparison of the model with YOLOv11-XL	74
Table 20 Comparison of height 60cm and different distances	78
Table 21 Comparison of height 110cm and different distances	78
Table 22 Comparison of height 170cm and different distances	79
Table 23 Test results for robot integration	81
Table 24 Number of detected objects in the Dark floor setting	81
Table 25 Number of original objects in the Dark floor setting	82
Table 26 Number of detected objects in the corridor setting	83
Table 27 Number of original objects in the corridor setting	83
Table 28 Number of detected objects in the white floor setting	84

Table 29 Number of original objects in the white floor setting	84
Table 30 Comparative table for 2-frame and 5-frame logic	85
Table 31 Number of detected objects in 5-frame logic	85
Table 32 Number of original objects in the 5-frame logic	86
Table 33 Test-wise table for 2-frame logic	88
Table 34 Test-wise table for 5-frame logic	88
Table 35 Comparison of different tilt angles	91
Table 36 Comparative table for tilt 20°	93
Table 37 Comparative table for tilt 30°	94
Table 38 Comparative table for tilt 40°	95
Table 39 Comparative table for tilt 50°	96
Table 40 Comparative table for tilt 60°	97

List of Equations

Equation 1 60

Equation 2 60

Equation 3 60

List of Abbreviations

YOLO	You Only Look Once
ConvNeXt	Convolutional Network Next
ViT	Vision Transformer
SAM	Segment Anything Model

1 INTRODUCTION

1.1 Background and Motivation

In recent years, machine vision has become an integral part of surveillance and operational safety and is widely implemented. Hospitals, hotels, and restaurants increasingly rely on machine vision systems for continuous monitoring to increase safety by hazard detection, error reduction and to perform repetitive tasks[1]. Integration of machine vision and artificial intelligence has allowed robots and automated systems to perform tasks such as cleaning, manufacturing, delivery etc. By incorporating object detection and classification techniques into these systems, they can be used for a variety of applications, such as safety monitoring and hazard prevention such as unattended or misplaced items may pose significant safety risks. For instance, an unattended bag might raise a security concern[2], while items kept in corridors can lead to slip hazards[3]. Figure 1 shows a common hazard left on hotel corridors, which is both unappealing and a trip hazard[4].



Figure 1 Utensils left on the corridors

1.2 Problem Statement

As discussed above, hotel corridors often have unattended items scattered around, especially utensils and cutlery like plates, spoons, forks, and trays. According to the National Floor Safety Institute (NFSI), the most common hotel-related accidents are slips, trips and falls from unattended objects in corridors, slippery floors and uneven surfaces[5]. Similarly, in hospitals, obstructions such as trolleys and items left in the corridors could lead to slips and trips [6]. As noted in the previous section, relying solely on manual monitoring for the detection and clearance of these items is inefficient, error-prone, and resource-intensive. Hence, there is a need for an automated system architecture for detecting these hazards. The system should be a generic vision-based model that can be deployed on mobile service robots to enhance safety, reduce human workload, and ensure timely hazard detection in hotel environments

1.3 Scope Of Work

The scope of this project is to develop a machine vision model to detect anomalies in the corridors of hotels and to integrate the system onto a robot.

The project will focus on the visual detection and on the precision, recall and F1 scores of the models, evaluating different combinations of object detection models and the implementation of segmentation and to determine the best model. This best-performing model will then be integrated with the robot for functional validation.

1.3.1 Aim

The aim of the project will be to develop and deploy a machine vision system onto a mobile robot to detect unattended items in the categories of plates, cups, glasses, trays, spoons, forks and butter knives.

1.3.2 Objectives

To conduct a literature review on the existing anomaly detection architectures for the detection of unattended objects and hazards

- To create a custom dataset reflecting different indoor corridor conditions with the specified objects.
- To develop a two-stage anomaly detection architecture for the detection of cluttered and overlapping objects listed above.
- To implement and evaluate the performance of the system on a mobile robot by making the mobile robot stop when anomalies are detected.

1.3.3 Limitations

The project will have the following limitations:

- The detected objects will be restricted to plates, cups, glasses, trays, spoons, forks and butter knives.
- Robot manipulation tasks of lifting or removing detected objects will be out of scope.
- The project won't involve deployment of path planning or SLAM (Simultaneous Localisation and Mapping) techniques for navigation and will rely on the existing navigation capabilities of the mobile robot.

1.4 Technological Contribution

This project addresses the challenge of detecting unattended items (plates, cups, glasses, trays, spoons, forks and butter knives) in cluttered. The objects are often stacked, partially obstructed and placed in narrow passages with varying illumination (day/night cycles and artificial lighting) often observed in hotel corridors. The main contribution of this thesis is:

Custom Dataset Creation: Custom datasets were created for different environmental conditions and the systems are trained and tested on these to address these challenges.

Anomaly Detection Framework: A novel two-stage anomaly detection framework is proposed, which combines object detection and segmentation models to accurately identify unattended items (plates, cups, glasses, trays, spoons, forks and butter knives) in corridors.

Comparative Evaluation: A comparative analysis with precision, recall and F1 is done for the system for different datasets to ensure the system is robust.

Robotic Integration: The final system is then integrated with a robot to ensure the adaptability and integrability with robotic systems.

1.5 Thesis Structure

In this Thesis, the first chapter above discussed the aim and objectives that are to be achieved at the end of the thesis. The second chapter contains a comprehensive literature review on various object detection methods. The third chapter describes the different datasets created and the general framework of the system designed. The fourth chapter gives a clear in-depth explanation about the system developed, its architecture and the different components of the system. The fifth chapter presents the results of the thesis and provides an explanation of each result. The last chapter discusses the conclusion and future work that could be done to further enhance the system.

2 LITERATURE REVIEW

The study will focus on visual anomaly detection in structured indoor environments, specifically the corridors of buildings. The aim is to detect various items such as cutlery, plates, trays, etc., placed in the corridors. The literature review focuses on a wide spectrum of anomalies and the methods used to detect them and understanding these methods better to help design our system. It is organised into 5 sections: (i) evolution of object detection, (ii) anomaly detection across domains, (iii) object detection architectures, (iv) fall detection approaches, and (v) a summary highlighting research gaps. It reviews methods for the detection of objects and falls to help understand the various methods that can be transferred and applied to the scope of the thesis.

2.1 Evolution of object detection

Various anomaly detection methods are developed over the years. Initially, classical computer vision algorithms like Gaussian Mixture Model (GMM), Principal Component Analysis (PCA) etc. were used to detect the anomalies, but they didn't perform well in dynamic environments hence models like SVM and ensemble methods were introduced to improve accuracy. The comparative study of machine learning between models like SVM and ensemble to CNN is noted in where the CNN-based model achieved the highest precision making it the most effective among the tested models for detecting abandoned bags[7]. With the rise of deep learning, convolution neural network began to outperform the previous models, becoming more robust and accurate. At the same time the object detection in computer vision has improved from classical algorithms such as Canny Edge Detector, Shi-Tomasi Corner Detector, and Otsu Thresholding to deep learning models like YOLO (You Only Look Once). The comparative study of these in [8] shows that YOLO significantly outperformed the others. Thus, it can be noted that over the years the computer vision has significantly evolved with advancements leading to deep learning methods like CNN and object detection models like YOLO.

2.2 Detection of anomalies

Anomaly detection plays a crucial role in safety and security in daily lives in shopping malls, in hospitals, etc. that need to be detected for safety purposes. Anomalies refer to something unusual which deviates from the usual situation the various anomalies are unidentified objects, loitering, running, human falls, and many more listed in [9].

2.2.1 Medical domain

In the health care domain anomaly detection can be used to detect anomalies in X-rays as seen in [10] where a new GMMAD method is proposed. It uses a pre-trained ResNet to extract features at different levels, and feature fusion is used to merge these features, after which GMM is deployed, which assigns higher anomaly scores to abnormal samples, thereby enabling anomaly detection.

2.2.2 Environmental hazards

Anomalies can also be environmental hazards like fire. In [11] a voice-controlled robot is used for fire detection, where voice is captured through a microphone, converted into text, and matched to a set of predefined commands and uses an Infrared Sensor (IR) to detect fire and also runs through a fire detection algorithm to reduce false alarms, and an alert is generated if a fire is detected. Gas leak is another environmental hazard and fixed gas sensors might not have enough coverage. A mobile TurtleBot 3 system equipped with four-directional MQ-2 sensors was developed to autonomously navigate through indoor spaces and detect LPG leaks in [12]. A mobile robot equipped with a microphone array and LiDAR was deployed in an industrial plant environment to detect leak-like sounds. The system successfully localised the leak source with an error margin of about 0.5 m compared to the LiDAR reference[13].

2.2.3 Surveillance

In context of surveillance the human trajectory detection is used to detect any suspicious activities such as entry to unauthorised areas. In [14] human trajectories in indoor spaces are analysed using a TENSIO model (a combination

of Transformer encoder and SOM), detecting unusual movements of people like entering restricted areas. While [15] uses a pre-trained YOLO object detector based on CNN (Darknet-53), which is YOLOv3, to achieve a successful monitoring system to detect suspicious activity. The results from this shown in the Figure 2.

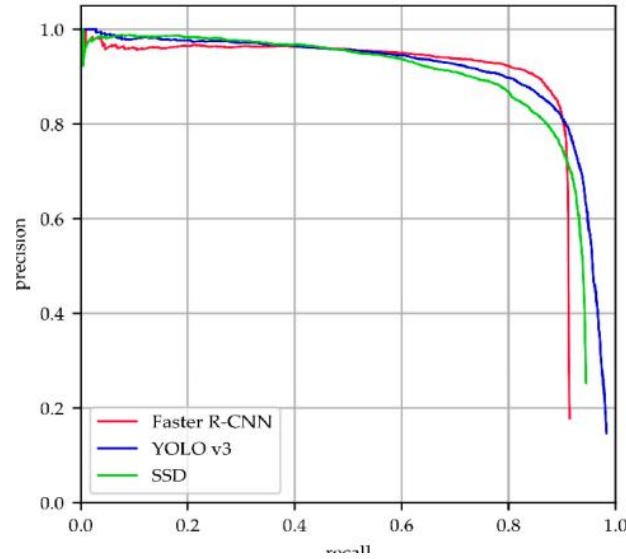


Figure 2 Comparison of YOLOv3

Hence, it can be inferred that YOLO perform better among the compared models. Further in [16], videos are converted into still images, then people are separated from the background, and GMM (Gaussian Mixture Model) is integrated to detect humans. Features like speed, measurement, and object movement are extracted, and then LSTM (Long Short-Term Memory) networks are trained using these features. As anomaly detection heavily relies on videos, protection of individual's privacy is a crucial point. A privacy-preserving framework for anomaly detection using Mask-RCNN ,2D Convolutional Autoencoder (CONV-AE) and Spatio-Temporal Autoencoder (ST-AE) is explained in [17].

Together, these studies show progress in surveillance, but none address cluttered, object-overlapping environments like indoor corridors, which remain an open challenge. They also do not address constrained environments, such as corridors with varying lighting conditions, which can affect the detection performance of a system.

2.3 Detection of objects

2.3.1 YOLO based detections

YOLOv8, trained on the COCO dataset, is used for real-time object detection in [18]. YOLOv8 was also used to train AMPharmy a hospital delivery robot in [19] to detect objects and uses the size of the object in the database and the camera feed to detect the distance to the object. These lack fine tuning and may not be able to perform well in cluttered or object specific scenarios that require fine details unless trained. Hence CNN-based model inspired by YOLO was used to train for object detection, with pan-sharpened images as input is seen in [20]. The proposed system was then compared with Traditional object recognition using sliding window and feature-based methods, and CNN-based object detection without pan-sharpening preprocessing and this can be seen in Table 1. The pan sharpening introduces high resolution images for detection that requires finer spatial detail, but this might not work for dynamic environments, due to misalignment and high computational cost.

Table 1 Comparison of the pan-sharpening method

METHOD	PRECISION (%)	RECALL (%)	MAP (%)
Sliding Window + Feature-based	78.9	85.2	82.1
Standalone CNN	88.3	89.9	88.9
Proposed Method	92.1	94.5	93.2

To address tracking challenges in dynamic environments the YOLOv8 and ByteTrack algorithms are combined in [21] for the detection and tracking of small objects in driving scenarios, these were trained on the SODA-D dataset. In [22] a dataset created for abandoned objects on the road, the images are first sent to a UNet-based encoder-decoder and then to an area-aware module, which enhances the segmentation process by highlighting the areas with the objects. Another approach with YOLO models is also implemented to detect the unidentified objects on the road and then compared with the YOLO models used as a baseline.

2.3.2 Two-stage detection architecture

More structured two-stage architectures are proposed in [23] and [24]. In [23], the preliminary stage uses a Gaussian Mixture Model (GMM) to identify background static objects, and the second stage (Recognizer) consists of a Convolutional Neural Network (CNN) trained on datasets containing various types of bags to verify whether the detected object is a bag. Similarly, in [24] where the first stage, Static Object Detection (SOD), involves eliminating the background and subtracting frames five frames apart to detect stationary objects. The second stage uses the Cascade of Convolutional Neural Networks (CCNN) to classify the detected objects as bags or not. This cascade includes two CNNs: the first CNN detects luggage items, and the second CNN distinguishes between abandoned (unattended) and attended bags. These stages improve the accuracy for classification and detection but also add computational complexity. The subtraction of the background is a good enhancement that can be inculcated into the framework for the thesis.

2.3.3 Fusion approach

The multi-modal fusion approach in [25] uses different types of input data, such as LiDAR, radar, images, and audio and employs deep learning models like YOLO and Faster R-CNN to independently process each of the above. Later, the outputs of these models are then combined through model-level fusion to enhance object detection accuracy. Unlike single modal approaches, this integrates multiple sensors to improve detection and accuracy, but this may introduce latency to the real-time systems due to sensor integration and data processing. A similar approach is seen in [26] where LiDAR, mmWave radar, and camera data are combined to form a system named LRVFNet to detect 2D objects in driving environments. The proposed system was then tested on VOD dataset and the Flow dataset which showed that it outperforms the current state-of-the-art detection systems, as seen in Table 2 and Table 3.

Table 2 Results on the VOD dataset

Modality	Method	AP^{35}	AP^{50}
Image	Yolov4 [45]	31.06	29.78
	Yolov7 [20]	83.74	80.65
Image + Radar	RISFNet [8]	66.78	45.55
	LRVFNet (Ours)	88.30	85.03
Image+LiDAR+Radar	RISFNet + LiDAR	76.72	54.23
	LRVFNet (Ours)	89.37	86.50

Table 3 Result on Flow dataset

Modality	Method	AP^{35}	AP^{50}
Image	Faster-RCNN [13]	77.35	57.58
	Yolov4 [45]	78.46	57.04
	EfficientDet [15]	78.62	58.52
	FCOS [49]	68.71	58.56
Image + Radar	CRF-Net [6]	79.63	57.74
	Li et al. [35]	85.28	64.64
	RISFNet [8]	90.05	75.09
	LRVFNet (Ours)	91.12	78.46

From the two tables above, it can be noted that the average precision (AP) of LRVFNet outperforms the other models specified in the tables.

While all these studies demonstrate the versatility of deep learning, most of them concentrate on broad, outdoor-based surveillance applications. These contexts often involve well-separated objects, where detecting them is easier. This research, however, will be looking into the detection of plates, cups, trays, and cutlery, which will be stacked on top of each other, causing overlapping of objects. These studies also highlight the effectiveness of YOLO it still lacks the implementation in cluttered object identification with the implementation of segmentation.

2.4 Detection of falls

While most fall detection methods rely on vision-based systems, other sensor modalities have also been explored. Below, we examine both vision-based and sensor-based approaches to detect falls.

2.4.1 Sensor-based

In [27] falls are detected using LiDAR sensor data and these are used to train the SO-Net, which classifies the events into falling, lying down, standing, and sitting, and detects falls when the event is classified under falling. This gives an advantage of detecting falls in low light due to the use of LiDAR, but these lack the visual details available in vision-based systems, which are required for complex situations. Hence, vision-based systems are more favoured in complex situations like falls, which require the detection of facial expression and features to identify the person and to analyse the situation. LiDAR is integrated with a camera in [28] and an application called “Fallert” was developed, which calculates the distance between a person's head and the speed at which the head is moving towards the ground for the detection of falls. The system was able to detect falls with a precision of 90% as seen in Table 4.

Table 4 Detecting falls at three locations

	A Fall Is Identified	A Fall Is Not Identified	Precision
Fall on The Sofa	0	20	100 %
Fall on The Bed	0	20	100 %
Perform sit-ups on The Ground	0	16	100 %
Fall on The Ground	19	2	90 %

2.4.2 Vision based

In, Madsen et al. [29] a dataset was created consisting of normal day-to-day activities and of people falling and adopted MIL (Multiple Instance Learning) approach with an improved loss function, it was also determined that 3DResNet performed better feature extraction in general, but zero-padded TimeSformer achieved the highest accuracy in detecting falls. The implementation of the YOLOv7-Pose model can be seen in [30] where the video input is divided into frames, and noise and Gaussian filters are applied. Then, polarisation and solarisation are applied to the images, which are then sent to the YOLOv7 model.

This model determines the position of the person's shoulder and body relative to the feet, and falls can be detected if there is any change in the relative positions.

In contrast, a more lightweight vision-based system is proposed in [31] where a private dataset was created and used to fine-tune a pre-trained SSD MobileNet-V2 model, which had initially been trained on the COCO dataset. It was observed that reducing the number of training epochs helped achieve a balance in detecting complete falls and semi-falls. This suggests that even small models can be trained to achieve reliable results.

While the primary focus of our study is to detect cups, trays, cutlery, and similar items, the system can also be integrated with fall detection. It can take inspiration from the above approaches, such as using MobileNet as a classifier, as it did achieved reliable results in [31] and creating a custom dataset. These are also not tested in cluttered environments or with different lighting conditions, limiting their reliability in real-world settings.

2.5 Summary

The reviewed literature highlights a wide range of anomalies and detection methods, but there are many research gaps that can be identified. Most of the systems are tailored for outdoor environments or surveillance and typically focus on motion-based hazards. Most of the anomalies are well-separated, making them easier to detect, and there are only a few studies handling cluttered indoor environments with overlapping objects. Hence, there is a lack of detection of clustered anomalies in structured indoor settings. This may be due to the fact that most commonly available datasets are with minimal overlapping and focusses on few objects. While YOLO-based models have proven their effectiveness, segmentation has been rarely used, and multi-stage ensemble approach implemented with YOLO models to improve their accuracy remains underexplored in cluttered indoor environments. Cluttered environments have overlapping objects of same or similar features which makes it difficult to detect using single stage models while using multistage pipeline is more complex and computationally intensive and may introduce some delays. But as the scope of this research project is to detect unattended objects on mobile robots it doesn't

require immediate response as some delay is still acceptable while detecting the objects.

To address these gaps, this study will focus on building on the existing YOLOv8 segmentation to categorise and count the objects detected—that is, plates, cups, cutlery, etc. These objects are to be detected in the corridors of buildings, and when they are kept stacked on top of each other in a cluttered way.

3 METHODOLOGY

In this section, a detailed overview of the different the datasets created for the various stages of this thesis is given. Following this, the general system framework and the process it was conceptualised.

3.1 Dataset configuration

The dataset used in this project is specifically constructed for the target objects which are plates, cups, glasses, trays, spoons, forks and butter knives. This is because a dataset of the target objects under various real-world conditions was necessary for the thesis, which wasn't publicly available. Separate datasets were created to train the different models and test various strategies.

3.1.1 YOLOv11 Dataset

Initially, a dataset was created to train the YOLOv11(You Only Look Once) model, which involved different images containing all the specified objects in different instances. Each image was manually annotated by creating bounding boxes around the respective objects as seen in Figure 1, where each colour of the bounding box represents a specific class. This dataset was then used to train the YOLOv11 model. The images are taken in different environments, which are:

1. Corridors in front of the robotics teaching lab
2. On white tables
3. On a light grey tiled floor

The images are split into three sets: test, train, and validation. This is for the purpose of evaluating the model's performance effectively during and after training. The number of objects annotated per class can be seen in the images below. For the images, it can be observed that the smaller images, like fork, spoon, and butter knives, are represented less compared to the other classes. This is due to the fact that these items are small and are prone to being obstructed by other objects during detection. Priority was given to the bigger and most visible objects like cups, trays, plates and glasses. This enables their detection even at

a distance, while less priority was given to spoons, forks, and butter knives, which are less visible and less differentiable as all look similar at one end and are difficult to differentiate and annotate in a cluttered environment. The following Figures 3 to 5 show the distribution of objects in each class. Figure 6 shows the annotation of the objects.

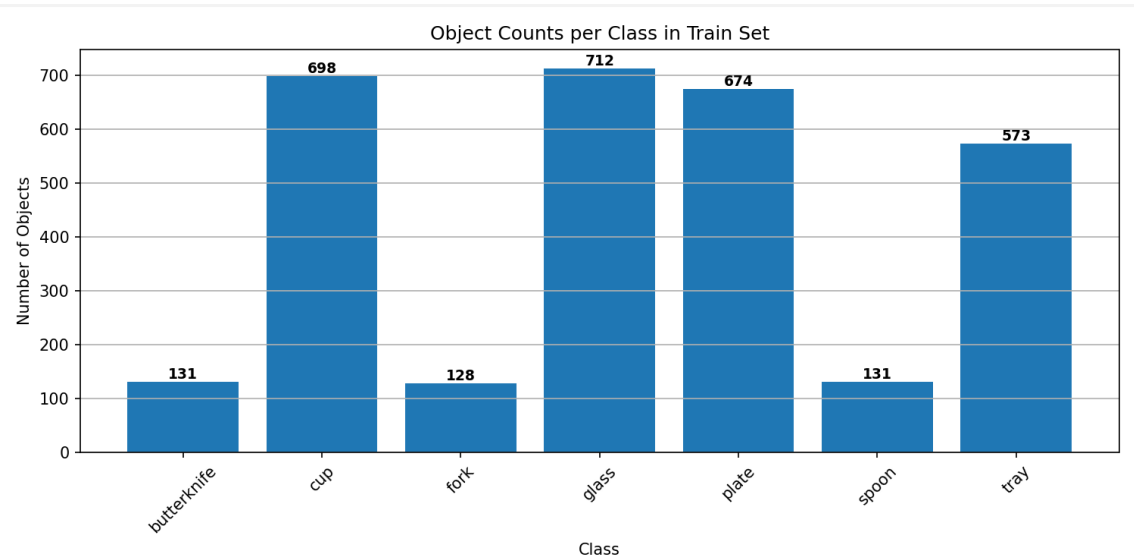


Figure 3 Distribution of objects in the train set of YOLOv11 Datasets

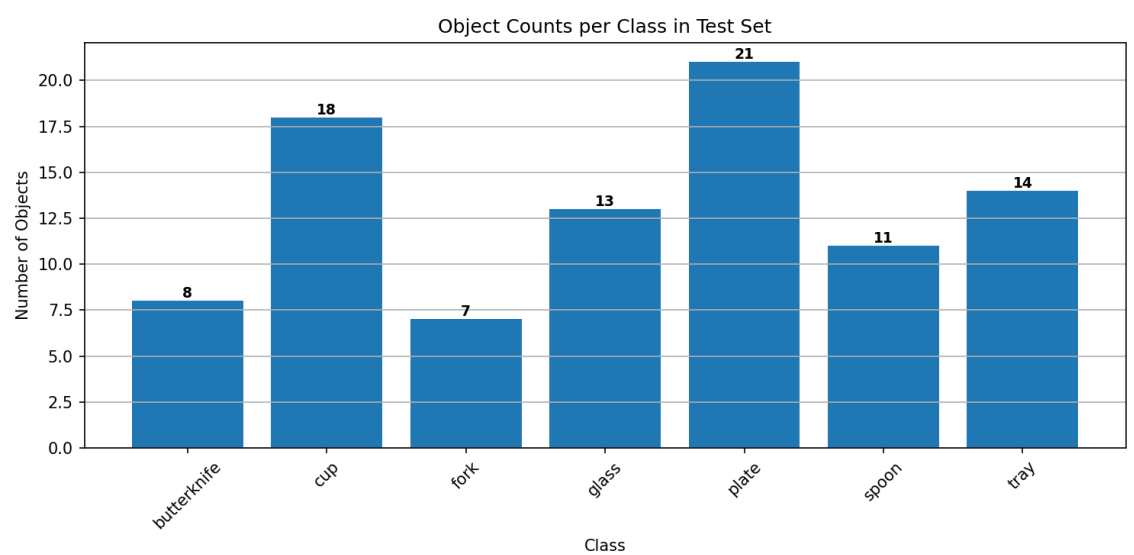


Figure 4 Distribution of objects in the test set of YOLOv11 Datasets

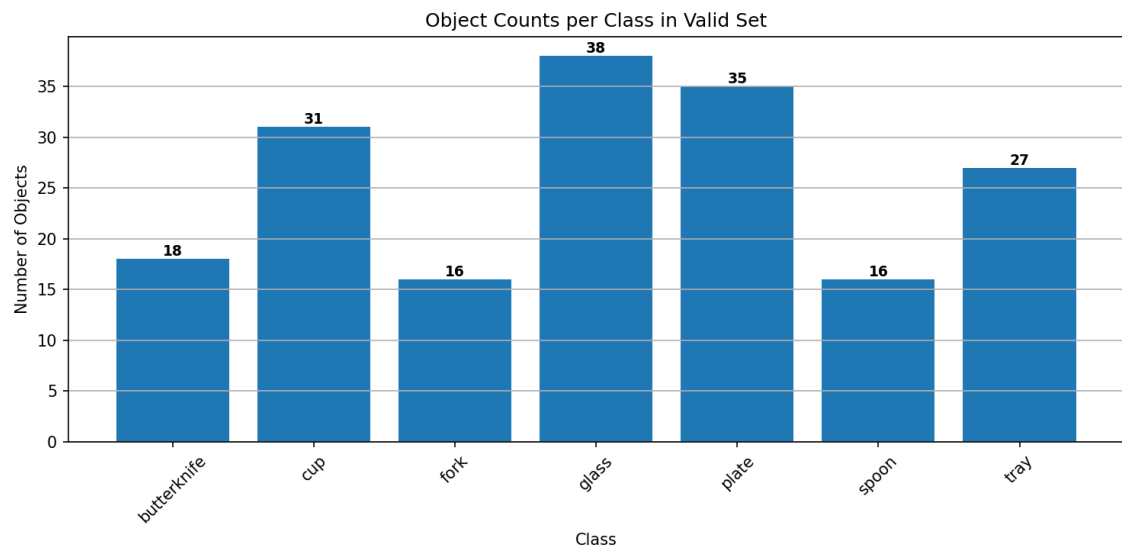


Figure 5 Distribution of objects in the validation set of YOLOv11 Datasets

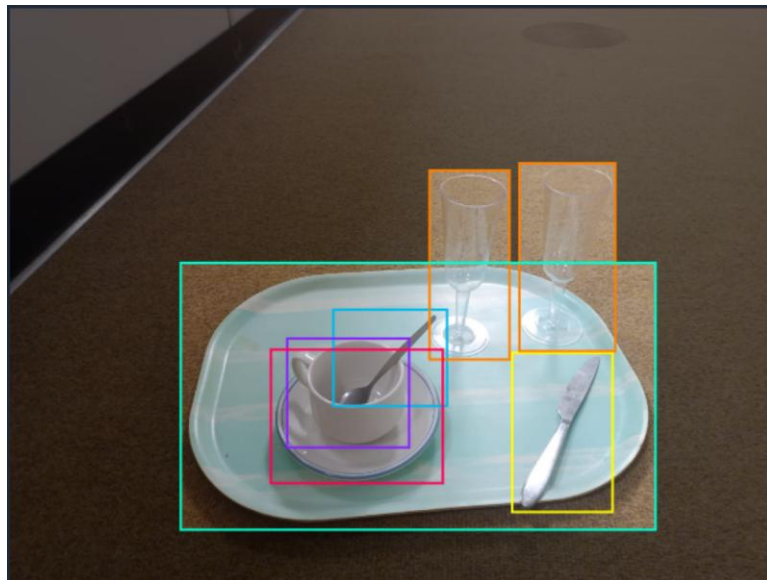


Figure 6 Annotation of classes of YOLOv11 Datasets

3.1.2 Non-SAM (segment anything model) crop dataset

The bounding boxes from the above dataset were then cropped and stored in their respective classes. After cropping the objects from the YOLO dataset from all test, train and validation sets and combining them into one folder, and created

a new dataset by splitting the folder created into train, validation, and test folders. The number of images per class can be seen in Figures 7-9. This cropped dataset was then used to train ConvNeXt (Convolutional Network Next) and ViT(Vision Transformer) models and used in the frameworks where the models were getting detection from YOLOv11 directly without the implementation of a SAM (segment anything model). A few extra images were added, SAM (the smaller datasets like butterknife, spoon and fork to improve the training on the models. Images from this dataset are depicted in Figure 10.



Figure 7 Distribution of objects in the train set of Non-SAM Datasets

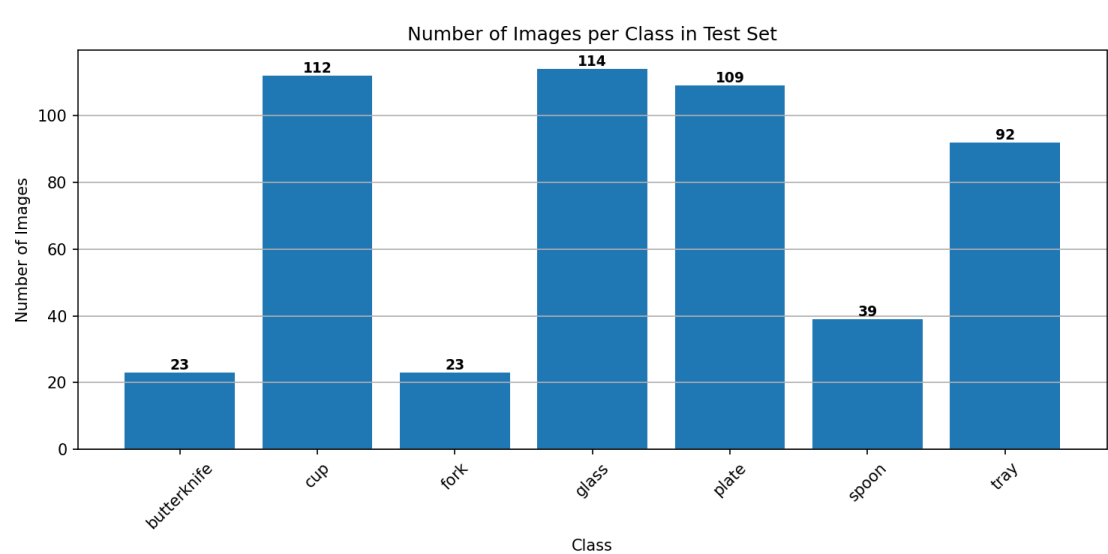


Figure 8 Distribution of objects in the test set of Non-SAM Datasets

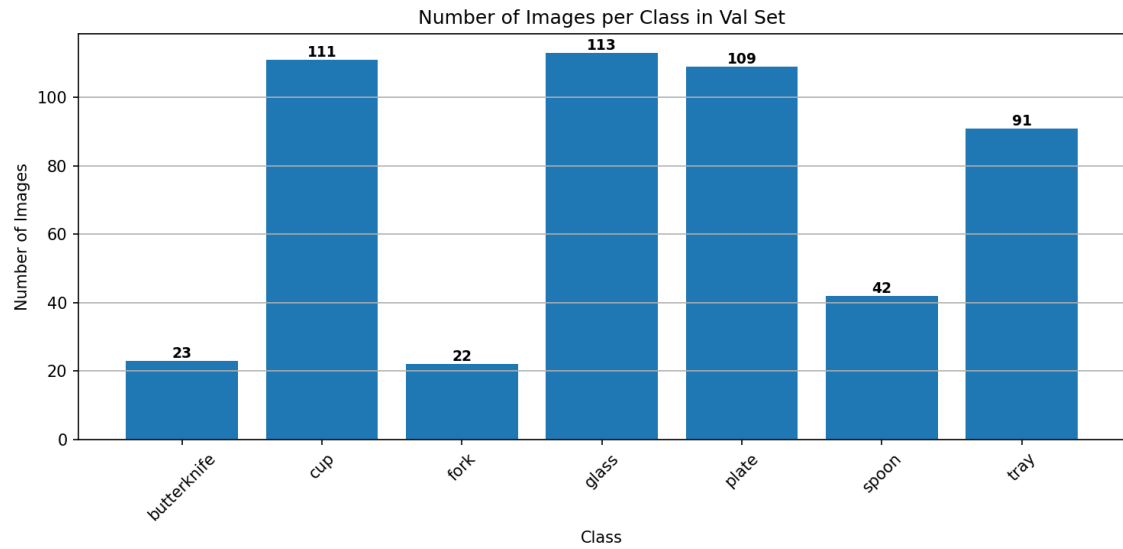


Figure 9 Distribution of objects in the validation set of Non-SAM Datasets



Figure 10 Classes of Non-SAM Datasets

3.1.3 SAM (segment anything model) crop dataset

To improve object isolation, the images from YOLO dataset was passed through YOLOv11 and then through the Segment Anything Model (SAM), which isolated the objects from the background. These were stored in their respective classes and further separated into test, train, and validation folders. By using YOLO to generate bounding boxes, the dataset better aligns with the kind of data that the system would encounter during the implementation. YOLO's bounding boxes are

looser, giving SAM more flexibility for object segmentation. The distribution of the datasets can be seen in Figures 11-13. This dataset was then used to train another set of ConvNeXt and ViT models. These models were then used in frameworks where SAM was applied before classification. The example of the images in this dataset can be seen in Figure 14.

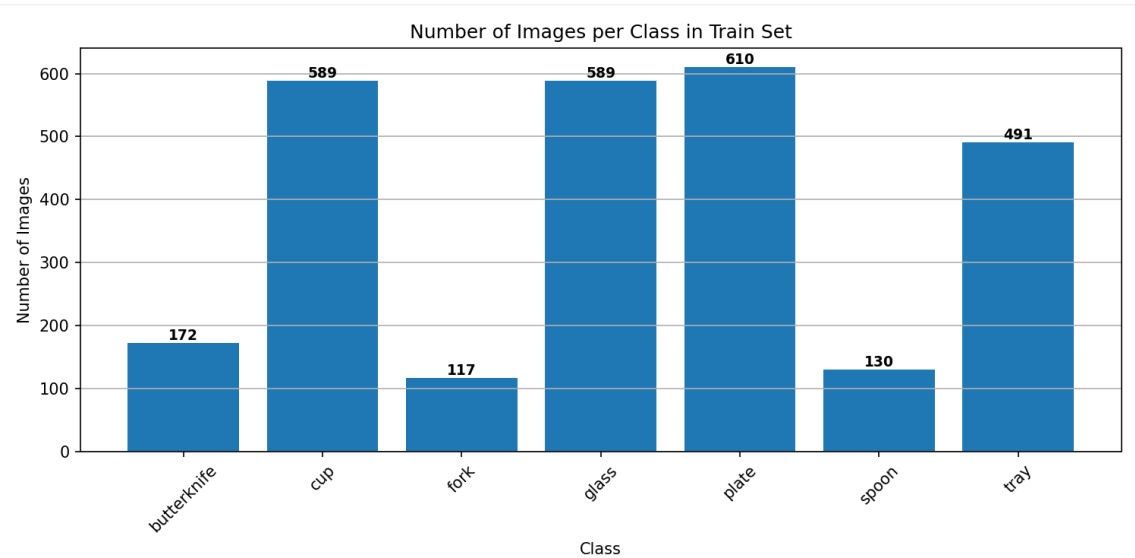


Figure 11 Distribution of objects in the train set of SAM Datasets

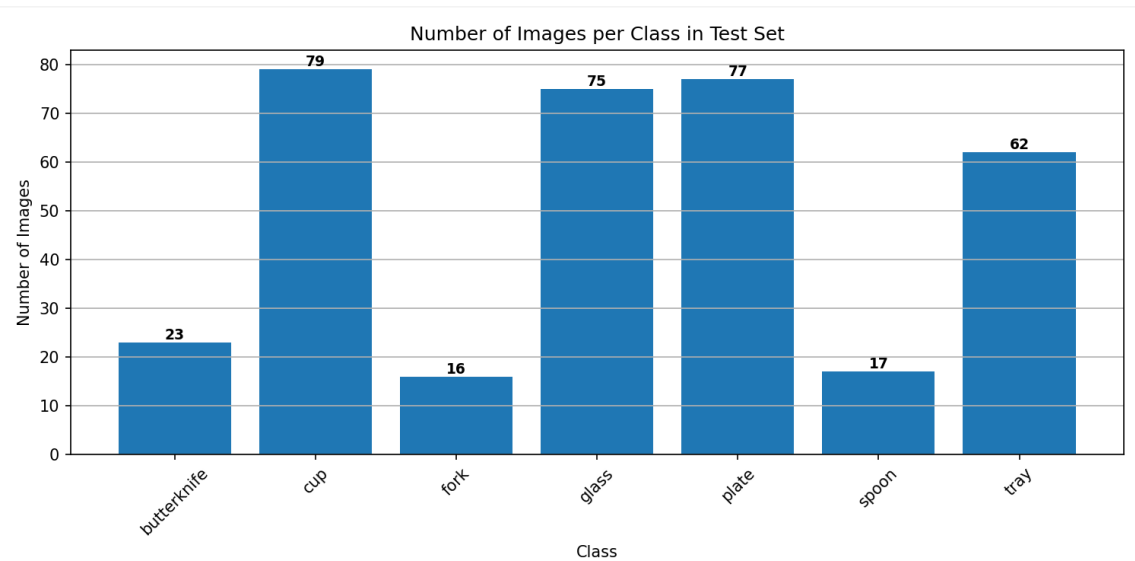


Figure 12 Distribution of objects in the test set of SAM Datasets

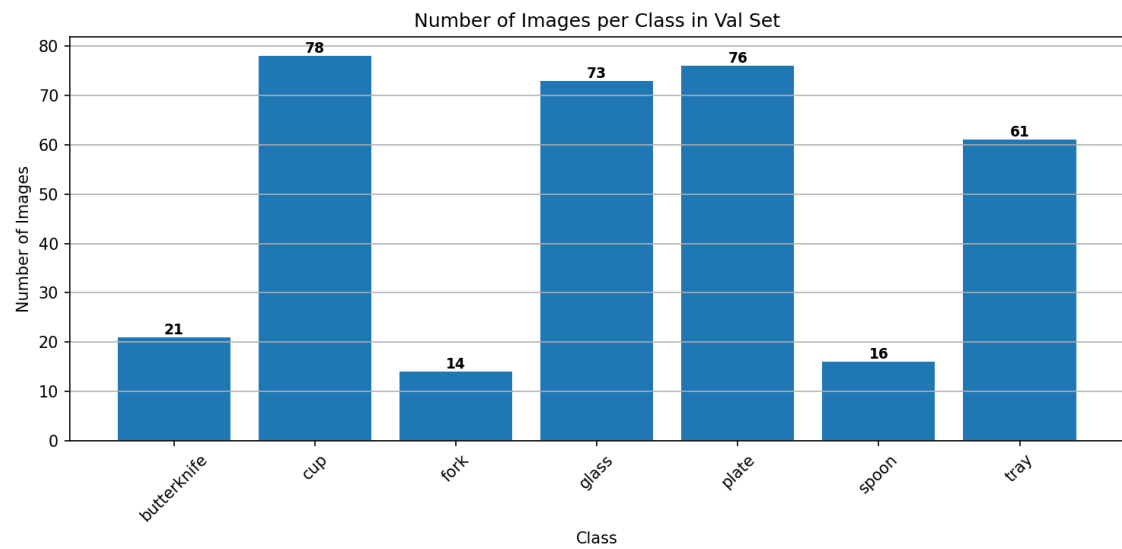


Figure 13 Distribution of objects in the validation set of SAM Datasets



Figure 14 Classes of SAM Datasets

The Table 5 shows the total number of images across all test, validation, and train datasets.

Table 5 Number of images across all test, validation, and train datasets

Class	YOLOv11	Without SAM	SAM
Butterknife	157	201	216
Cup	747	741	746
Fork	162	220	147
Glass	763	760	737
Plate	730	734	763
Spoon	158	297	163
Tray	614	608	614

The increase in objects in classes like butterknife, fork, and spoon in the "Without SAM Dataset" is because extra cropped images were added to it to increase the image numbers of these categories for better classification of the same. The slight decrease in the other classes (cups, plates, trays and glasses) in "Without SAM Dataset" compared to the YOLO is because some bounding boxes might be towards the edges of the images, resulting in invalid crops and thus might not be saved. The further inconsistencies seen in the "With SAM" dataset could be that when it is passing through YOLO, some small or overlapping bounding boxes might be created for the same object, resulting in duplicate images or missed images, depending on SAM.

3.1.4 Datasets for testing the systems

To assess the systems working in various lighting, background, heights and camera tilts, different datasets were further created as ground truths to test the working of the entire system. These are listed and explained below.

3.1.4.1 Datasets For Comparing Systems

The dataset was created for comparing different models and model combinations. This involved a total of 29 images taken under the following conditions:

- On corridor
- On white tables

The dataset included the following distribution of classes as seen in Table 6.

Table 6 Distribution of classes in datasets for comparing systems

Class	Number of items
Butterknife	14
Cup	34
Fork	13
Glass	26
Plate	35
Spoon	18
Tray	29

Few examples of the images in the dataset can be seen in Figure 15.



Figure 15 Images in datasets for comparing systems

3.1.4.2 DATASETS FOR DIFFERENT ENVIRONMENTS

Different data sets for different environments, which were sunny, under shade on a sunny day, gloomy day and Corridor, each consisting of 40 images

3.1.4.2.1 SUNNY

These consist of images taken under sunrays in 4 different buildings. These objects were placed under the following conditions:

1. On the floor directly under the sun's rays
2. On the floor in a skylit area
3. On a table under a skylight
4. In front of a window with sunlight falling directly on the objects

The class-wise dataset distribution is shown in Table 7 below.

Table 7 Dataset distribution in sunny environment

Class	Number of items
Butterknife	27
Cup	49
Fork	36
Glass	62
Plate	57
Spoon	31
Tray	56

A few images collected can be seen in Figure 16.



Figure 16 Dataset images in sunny environment

3.1.4.2.2 GLOOMY

This dataset includes images of objects placed in locations that typically receive direct sunlight indoors but were taken when the sun was not present (gloomy

conditions). The data was collected from 3 different buildings, and under the following conditions

1. On the floor next to a long window
2. On the tables next to a window
3. On the floor next to a high window

The class-wise dataset distribution is shown in the Table 8 below.

Table 8 Dataset distribution in gloomy environment

Class	Number of items
Butterknife	27
Cup	55
Fork	28
Glass	72
Plate	55
Spoon	29
Tray	61

A few images from the dataset are shown in Figure 17.



Figure 17 Dataset images in gloomy environment

3.1.4.2.3 SHADE

The data were collected on a sunny day under shaded conditions. These were taken from three different locations under the following scenarios:

1. On a table placed under an extended shed-like roof of a building, with sunlight on three sides
2. On a table positioned next to a building and within its shade
3. On a table under a tent-like structure, surrounded by sunlight on all four sides

The class-wise dataset distribution is shown in the Table 9 below.

Table 9 Dataset distribution in a shade environment

Class	Number of items
Butterknife	34
Cup	64
Fork	38
Glass	85
Plate	50
Spoon	36
Tray	75

The images from the dataset is seen in Figure 18.



Figure 18 Dataset images in a shade environment

3.1.4.2.4 CORRIDOR

These images were collected from carpeted corridors of three different buildings, mimicking a hotel setting. The lighting conditions were varied and are as listed below:

1. White light
2. Yellow light
3. Natural light

The class-wise dataset distribution is shown in the Table 10 below.

Table 10 Dataset distribution in a corridor environment

Class	Number of items
Butterknife	35
Cup	73
Fork	38
Glass	96
Plate	67
Spoon	29
Tray	69

A few images collected from the dataset is seen in Figure 19 below.



Figure 19 Dataset images in a corridor environment

3.1.4.3 DATASETS FOR DIFFERENT HEIGHTS AND DISTANCES

Further, different datasets were created for testing the system at various heights and distances. The configurations were designed by considering the minimum field of view and the largest object dimension, which is 30 cm × 40 cm. Therefore, a minimum gap of 50 cm was maintained between each tested distance to ensure there was no overlap between test areas, especially when the tray is placed at the edge of a measured distance. This 50 cm accounts for the tray length and provides an extra margin for separation. The distances and heights taken into consideration are as follows:

Height 60cm with distances:

- Distance 140cm (minimum distance of field of vision at height 60cm)
- Distance 190cm
- Distance 240 cm

Height 110cm

- Distance 190cm (minimum distance of field of vision at height 60cm)
- Distance 240cm

- Distance 290 cm

Height 170cm

- Distance 240cm (minimum distance of field of vision at height 60cm)
- Distance 290cm
- Distance 340 cm

The number of items per class is as seen in Table 11 below.

Table 11 Data Distribution of different heights and distances

Height and Distance	Number of items per Class						
	Butterknife	Cup	Fork	Glass	Plate	Spoon	Tray
Height 60cm Distance 140cm	2	8	4	12	8	2	8
Height 60cm Distance 190cm	5	10	4	10	8	5	10
Height 60cm Distance 240cm	5	10	5	14	10	4	10
Height 110cm Distance 190cm	1	10	5	14	10	5	10

Height 110cm Distance 240cm	2	10	5	14	10	2	10
Height 110cm Distance 290cm	1	10	1	9	7	4	10
Height 170cm Distance 240cm	5	7	4	9	10	4	9
Height 170cm Distance 290cm	5	6	5	9	10	5	10
Height 170cm Distance 340cm:	1	10	2	7	10	4	8

The images from the dataset can be seen in Figure 20.

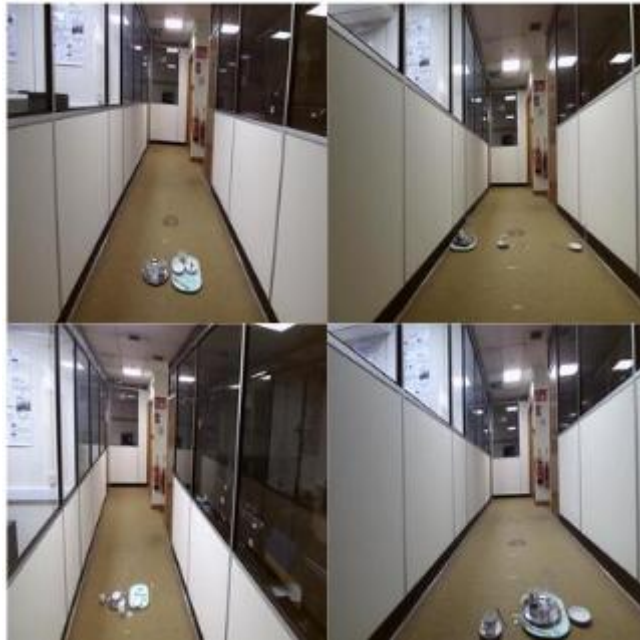


Figure 20 Dataset images for different heights and distances

3.1.4.4 DATASETS FOR ROBOT INTEGRATION

Datasets were further created to test the system after integration with the robot. These datasets were collected by running the robot-integrated system in different scenarios. The images were saved frame-wise, both

1. Original (unprocessed) frames
2. Detection frames, annotated with predicted bounding boxes and class labels.

Along with this, a text file with bounding box coordinates and class labels in YOLO format. It has to be noted that only the frames where any detections were present for the suggested number of frames are used to make the datasets. The frames where no objects or detections were present are discarded.

These original frames were then manually annotated and saved to form the ground truth. Hence, four datasets were created under the following conditions:

1. On dark backgrounds (dark grey floor)
2. On light backgrounds (white floor)

3. In corridor environments
4. At different tilt angles of the camera

The tests were conducted 5 times per condition. For dark and white backgrounds, the detection algorithm was made if the object is found in 2 frames the robot stops and the algorithm pauses. While for corridor 2-frames logic was adopted along with a 5-frame logic (if the same object is found for 5-frames) this is further explained in sections below. For different tilts the 5-frame logic was implemented. The number of images per class for original and detected folders will be explained in later sections.

3.1.4.4.1 DARK BACKGROUND

Here, the items were placed on a dark grey floor. A total of 5 tests were conducted and each test video was split into frames and these images were stored in the above-mentioned folders in folders. A few images from detection folder is shown in Figure 21 and images from the original folder is seen in Figure 22.



Figure 21 Detection frames from dark background



Figure 22 Original frames from dark background

3.1.4.4.2 WHITE BACKGROUND

Similarly, 5 tests were conducted when objects were placed on the White floor. A few images stored in the detection folder are shown in the Figure 23 and images stored in the original folder are shown in Figure 24.



Figure 23 Detection frames from white background



Figure 24 Original frames from white background

3.1.4.4.3 CORRIDOR

Again, 5 tests were conducted and images saved into the folders. The Images stored in the detection folder are shown in the Figure 25 and images stored in the original folder are shown in Figure 26.

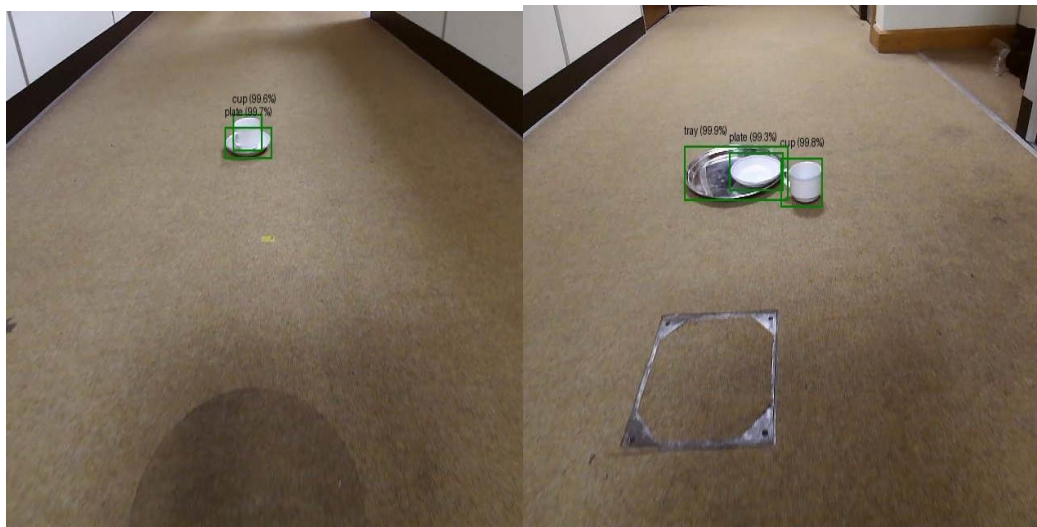


Figure 25 Detection frames from corridor



Figure 26 Original frames from corridor

3.1.4.4.4 TILT ANGLES

Again, 5 tests were conducted for each tilt angle, which are 20°, 30°, 40°, 50° and 60°. The angles were measured with respect to the horizontal axis. The outputs were stored in the folders mentioned above. The images were obtained from 2 different conditions, one in a grey flooring and one on carpet in a corridor condition. The images stored in both the detection and original folders.

The images from the detection folder are seen in the Figure 27 and the original folder is seen in Figure 28.

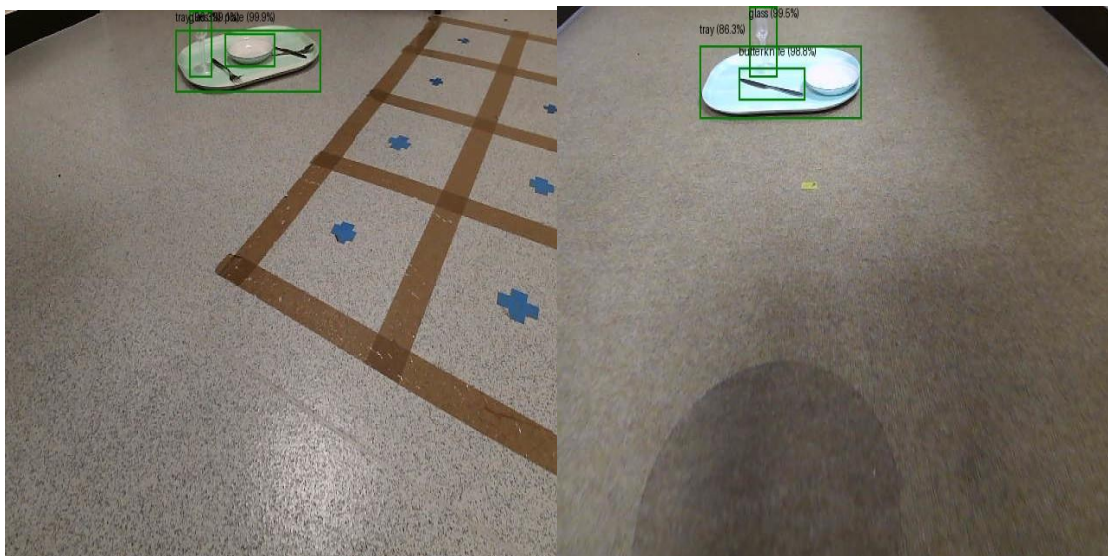


Figure 27 Detection frames for different tilt angles



Figure 28 Original frames for different tilt angles

3.2 Framework design

The design of the system framework was done to achieve accurate and reliable object detection. The initial idea was to adopt a 2-stage framework architecture inspired by the work of Sidyakin et al.[23] The primary framework involved the use of a base detection system capable of identifying and localising objects, followed by a secondary classifier system to verify the detections. But here there was a critical limitation, which was the possibility of both the systems predicting wrong classes or the base model predicting correct classes while the secondary classifier overwriting it to a wrong class. Hence, a design modification was introduced to add another secondary classifier system to the existing one. Hence, the two secondary classifier systems could give predictions, this enabled to increase confidence in the final decision by relying on agreement across multiple models.

As the system was developed, more attention was given to improving the prediction accuracies. It was noted that the bounding boxes from the base model, though effective for localisation lacked isolation of objects from the background. Hence, a segmentation module was introduced to extract object-specific regions, enabling the classifiers to focus only on the objects. After introducing segmentation, a choice had to be taken whether to use full mask or to use sub

mask. Hence, a comparative analysis was done for all implementations using SAM and the system architecture.

Further, another major requirement was to determine the model output that should be chosen as the final output, and hence a fusion logic was developed to choose the final output. This is further explained in sections below. The general framework of the system is as given in Figure 29.

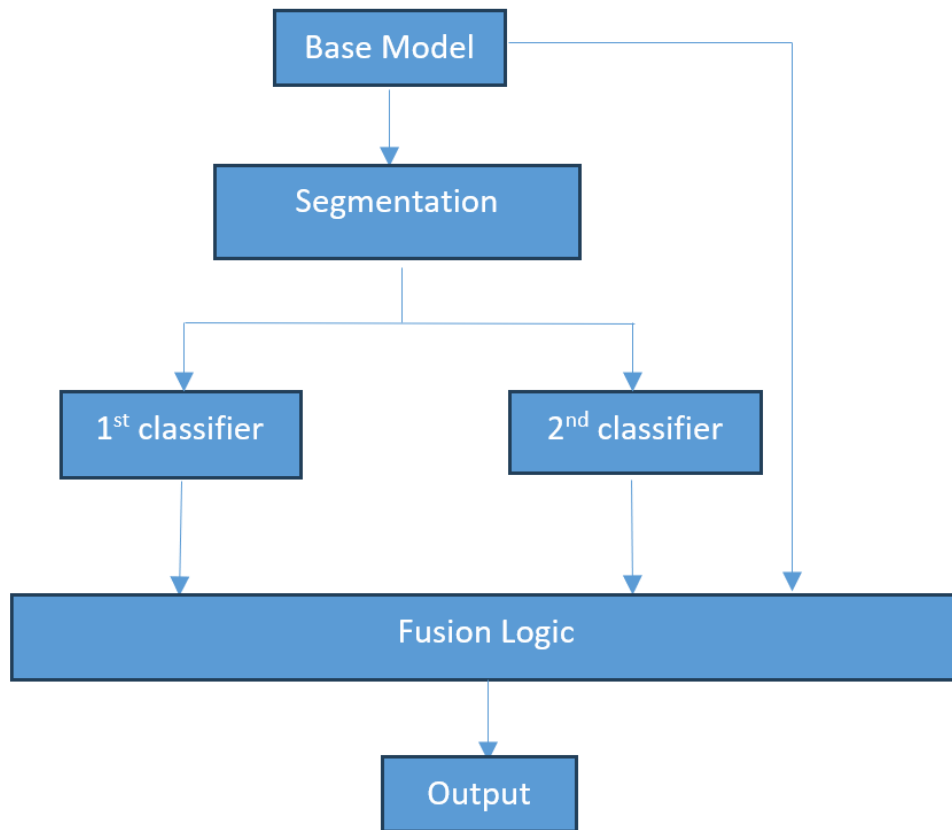


Figure 29 Block diagram for the system framework

As shown in Figure 29, the base model performs object detection and generates bounding boxes, which are then passed to a segmentation module to remove the background. The segmented outputs are forwarded to two independent classifiers that assign the detected objects to their respective classes. Finally, a

fusion module receives predictions from YOLO and the two classifiers, integrating the results to determine the final output.

The final system was developed through continuous testing and improvement. It started with a basic 2-stage detection and verification system and developed into a multi-modal segmentation-based system.

3.3 System architecture

The proposed model architecture is a multi-stage machine vision system which involves object detection, preprocessing, segmentation and classifiers to give a robust output. Each stage of the system is explained in detail below, in sequential order.

3.3.1 Object detection

The base model of the System architecture is YOLOV11-XI, created by Ultralytics[32]. It performs real-time object detections and produces bounding boxes with labels and confidence scores. Here, only detections above a confidence of 60% are forwarded for segmentation to SAM. This percentage level balances filters out incorrect detections while retaining true positives. This ensures that incorrect or low-confidence detections are not passed on to the later stages. The system processes one bounding box at a time, ensuring each object is evaluated through the entire system, and a loop iterates over each detected bounding box. This ensures that each prediction and fusion logic is applied to the same object.

3.3.2 Segmentation

The SAM model created by Meta AI is used for segmentation. SAM is capable of producing segmentation masks without any training or fine-tuning. Each bounding box above 60% confidence is processed by SAM, and the full mask with the highest confidence is retained for further processing. The full mask represents the entire silhouette of the object with the background removed. This full mask is directly given to the ViT model, and the same full mask is split into

multiple submasks using connected components analysis via the cv2.connectedComponents function from the OpenCV library. The submask separates the connected components from the object, breaking into parts, eg separating overlapping cutleries from each other, separating different parts of the plate if there are overlapping objects on it etc, as seen in Figure 30. Connected components analysis creates multiple submasks for each full mask. All these submasks are submitted to ConvNeXt.

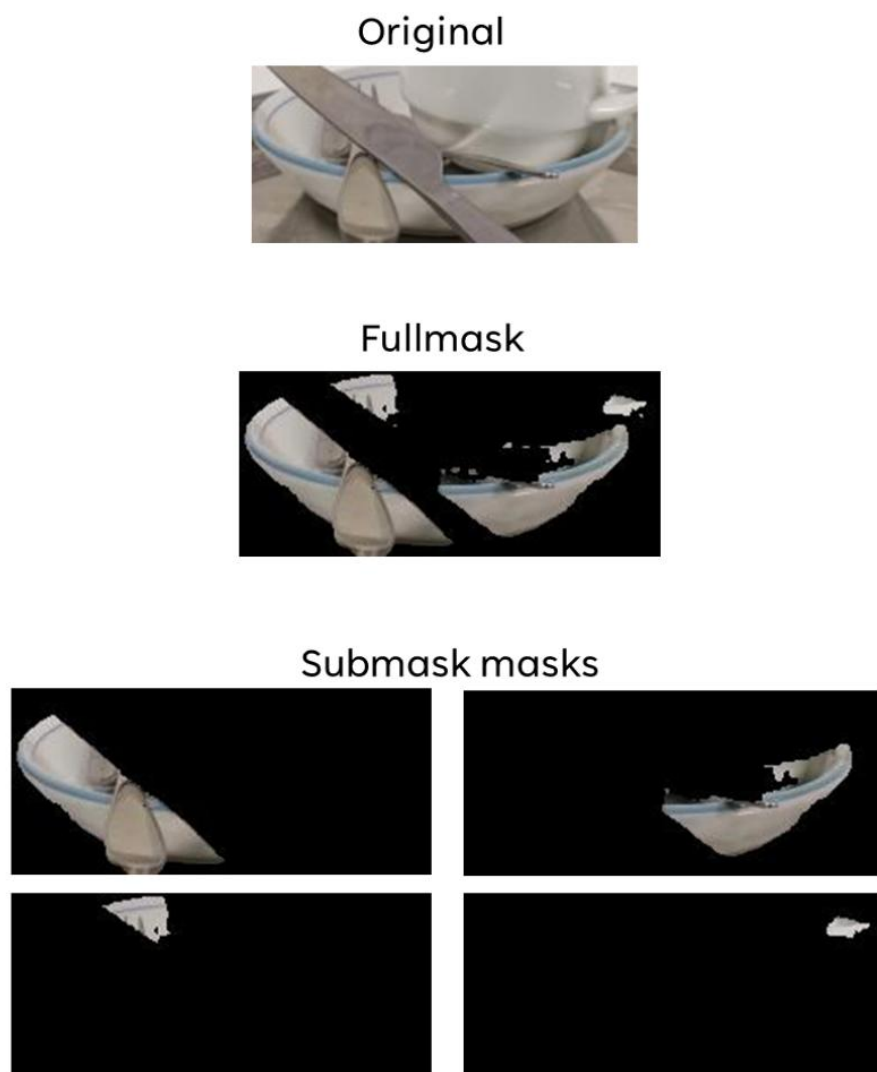


Figure 30 Example of full mask and submasks

3.3.2.1 TRAY EXCEPTIONS

If the YOLO model detects an object as a tray, submasks are not created, and the full mask is directly submitted to ConvNeXt. This is because trays are large, uniform objects and don't require splitting.

3.3.3 Pre-Processing

Before forwarding the masks to classifiers, the masks are pre-processed.

- Colour space conversion: All input frames are converted from BGR (OpenCV format) to RGB. This conversion is done as ViT and ConvNeXt expect RGB while OpenCV uses BGR. OpenCV is primarily used in this system for tasks such as drawing bounding boxes, applying segmentation masks, and saving annotated images and videos.
- Background removal: Pixels outside the masks are set to zero to isolate the objects
- Resizing: Objects are resized to 224x224 pixels to match the size the models are trained.
- Tensor conversion: Images are first handled as NumPy arrays for tasks like segmentation and masking, because tools like OpenCV and SAM work with NumPy. Images are transformed into PyTorch tensors scaled to [0, 1], as PyTorch models like ViT and ConvNeXt expect tensors. This preprocessing ensures compatibility with ViT and ConvNeXt models.

These pre-processing are done ensures compatibility with ViT and ConvNeXt models and smooth working of the integrated model.

3.3.4 Classifiers

3.3.4.1 ViT

The Vision Transformer (ViT) is a transformer-based image classification model developed by Google Research. The ViT model is trained on the "SAM (SEGMENT ANYTHING MODEL) CROP DATASET" explained section [3.1.3](#).

The Vision Transformer (ViT) receives the full mask crop and produces a single class prediction with a confidence score.

3.3.4.2 ConvNeXt

ConvNeXt-Tiny is a lightweight convolutional neural network model proposed by Facebook AI (Meta). ConvNeXt, like ViT model is trained on “[SAM \(SEGMENT ANYTHING MODEL\) CROP DATASET](#)”. The ConvNeXt receives multiple submasks and provides its predictions and confidence scores. Among all submask predictions, the one with the highest confidence is selected as the final ConvNeXt output.

3.3.5 Fusion Logic

Fusion logic is used for determining the final predictions. This fusion logic is designed in a way that it receives the predictions from all three models and if two or all models agree on the predictions, then the best confidence score prediction is taken as output.

If all models disagree then a weighted confidence scheme is applied where the weights are as follows:

- YOLO: weight 3
- ConvNeXt: weight 2
- ViT: weight 1

The final confidence scores of all models are multiplied with their respective weights and the one with the highest score is taken as the final output. The objects above a confidence score of 75% confidence are displayed in final output. This ensures that only high-confidence and accurate detections are shown.

3.3.6 IoU-Based Post-Filtering

To eliminate duplicate or overlapping detections, the system applies Intersection over Union (IoU) filtering. Here, detected bounding boxes that overlap more than 0.9 (90%), i.e. $\text{IoU} > 0.9$, are considered duplicate detections. Among the overlapping detections, only the box with the highest confidence score is retained.

The system architecture block diagram is shown in Figure 31.

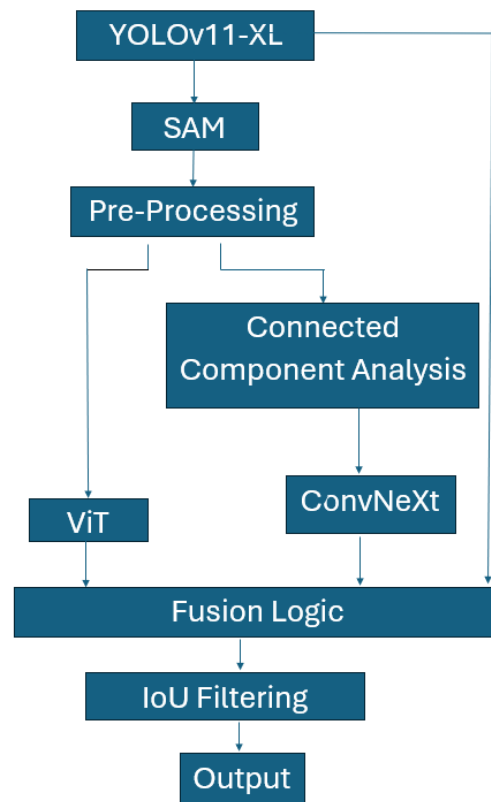


Figure 31 Block Diagram of the final system architecture

4 EXPERIMENT DESIGN

Careful design of the experimental stages was essential for selecting and validating the system to ensure that the system is robust and performs in diverse conditions.

4.1.1 Base Model Selection

The first experimental stage involved testing different YOLOv11 models, specifically YOLOv11-L and YOLOv11-XL. YOLOv11 is selected as the base model due to its status as the latest and most advanced version among available YOLO models, and YOLOv11-L and YOLOv11-XL were the 2 powerful models.

The goal of the stage was to compare the performance of the two models on the same datasets to see which one worked better and serve as a reference baseline that future model combinations should aim to improve upon.

4.1.2 YOLO-Classifier Combination Testing

Further, the combination of YOLO is tested with different classifier models. Both YOLOv11-L and YOLOv11-XL were tested with all classifier models to determine which YOLO model performed better when combined with classifiers. The classifier models selected were MobileNetV3, EfficientNet-B3, ConvNeXt, and Vision Transformer (ViT). MobileNetV3 was tested because it was lightweight and efficient and hence used as the first model. EfficientNet-B3 was selected as it uses effective scaling methods to improve classification. ConvNeXt is a convolution network that paid attention to details and hence was chosen for testing. ViT (Vision Transformer) was selected as it could understand relationships between objects and hence was good at classification in cluttered environments.

4.1.3 SAM Integration and Testing

Following the classifier testing and after identifying the two best classifier combinations, both classifiers were combined together along with the YOLO model. Further, SAM was integrated into the pipeline. This involved combinations of the two SAM methods, which were:

- Full mask: representing the whole object
- Submask: multiple smaller regions of the object

The SAM models were applied before each classifier, and the following combinations were tested:

- No SAM
- 1 SAM full mask and no SAM
- 1 no SAM and 1 full mask
- 2 SAM full masks
- 1 SAM full mask and 1 submask
- 1 submask and 1 full mask
- 2 SAM submasks

It has to be noted that where SAM full mask and SAM submasks are used before a classifier, the particular classifier used will be trained on the “[3.1.3. SAM \(SEGMENT ANYTHING MODEL\) CROP DATASET](#)”. Whenever SAM isn’t used before a classifier, the classifier used will be trained on the “[3.1.2. NON-SAM \(SEGMENT ANYTHING MODEL\) CROP DATASET](#)”.

To ensure that the model would work under different conditions, all systems were tested in the following environments:

1. Bright sunlight (outdoor sunny conditions)
2. Shaded sunlight (partial shadows outdoors)
3. Gloomy outdoor conditions
4. Indoor corridor settings with artificial lighting

The system that consistently performed better across all conditions was selected as the final model. These evaluations were done to ensure that the selected system is reliable in real-world scenarios.

4.1.4 Testing of the Model in Different Heights and Distances

To test the robustness of the final system selected above, the heights and distances were varied. This tested the model's ability to handle scale variation,

perspective distortion, and resolution loss. The models were tested on the [“3.1.4.3 DATASETS FOR DIFFERENT HEIGHTS AND DISTANCES”](#) discussed earlier.

4.1.5 Real-world robotic integration

In the final stage, the system is tested with integration with the robot, which is the Swift robot platform. The robot's live camera feed was recorded, capturing both the original frames and the corresponding detection outputs. The original frames could be later manually labelled and made into ground truth, which could then be compared with the detected frames to calculate the precision, recall, and F1 score. The robot integration is tested on the following datasets explained before in section [3.1.4.4](#):

1. [On a dark background \(dark grey flooring where objects are placed\)](#)
2. [On a white background \(white tiles where objects are placed\)](#)
3. [On corridor settings](#)
4. [On different tilt angles of the cameras.](#)

The results of the same are explained in section [6 “RESULTS AND DISCUSSIONS”](#). Each experimental phase was designed to select the most optimised system and is rigorously tested to ensure that the system is capable of real-time deployment. The final model that was selected, was ensured for its accuracy, consistency, and adaptability.

5 RESULTS AND DISCUSSIONS

To evaluate the performance of the models, three metrics were considered, which are:

- Precision
- Recall
- F1

This is because the datasets used are unbalanced, and these are apt for unbalanced datasets [33].

Precision measures the proportion of correctly identified objects (true positives) out of all objects predicted by the model. Recall measures the proportion of correctly identified objects out of all actual objects present in the image (true positives + false negatives).

F1 measures the mean of precision and recall. It provides a single measure that balances both metrics. The formulas for these are as follows:

$$precision = \frac{TP}{TP + FP} \quad \text{Equation 1}$$

$$recall = \frac{TP}{TP + FN} \quad \text{Equation 2}$$

$$F1 = 2 * \frac{precision * recall}{precision + recall} \quad \text{Equation 3}$$

Where,

TP: True positives (correctly identified objects)

FP: False positives (wrongly detected objects)

FN: False negatives (detected object which is not present)

5.1 Base Model Selection

Initially, YOLOv11-XL and YOLOv11-L models were taken and trained on the [“YOLOv11 DATASET”](#), discussed earlier in [Section 3.1.1](#), consisting of labelled images, representing classes such as spoon, fork, glass, tray, etc. Early stopping was applied with a patience of 10 epochs, monitoring the validation loss to prevent overfitting and to stop training once it starts converging.

After this, the first test conducted was as explained in section [4.2.1 “BASE MODEL SELECTION”](#) to select the YOLO models. Here, the datasets for comparing systems explained in section [3.1.4.1](#) were used to run the YOLOv11. The results obtained are given in Table 12.

Table 12 Comparison of YOLOv11 Models

Model	Precision	Recall	F1
YOLOv11 L	0.6492	0.7337	0.6889
YOLOv11 XL	0.7251	0.9053	0.8053

It can be noticed that YOLO-XL performs significantly better than YOLOv11-L in all metrics. A precision of 0.7251 and a recall of 0.9053 indicate that YOLOv11-XL predicted the actual objects right and fewer false positives. The F1 score, which is the

- YOLOv11-XL outperformed YOLOv11-L in all metrics. A precision of 0.7251 and a recall of 0.9053 indicate that YOLOv11-XL not only predicts fewer false positives but also detects a larger proportion of actual objects.
- The F1-score of 0.8053 for YOLOv11-XL shows a strong balance between precision and recall, making it a suitable choice for deployment in downstream tasks.

The training and validation curves for YOLOv11-XL is as given in Figure 32.

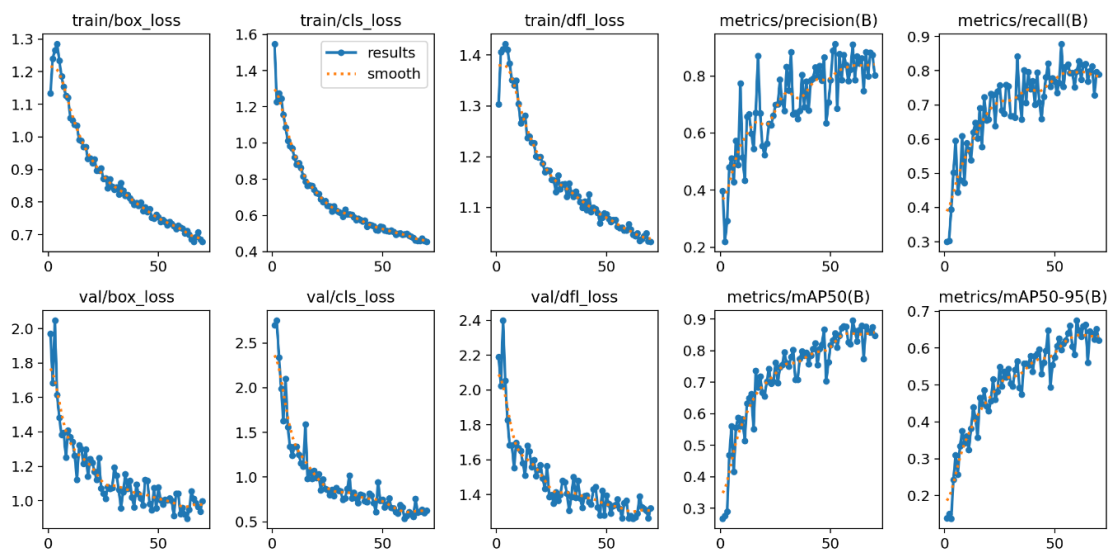


Figure 32 Validation and training curve of YOLOv11-XL

From Figure 32 the following can be notice:

Row 1: Training metrics

- *train/box_loss*: The box loss drops from ~1.3 to below 0.7, showing that the model is learning to locate objects accurately.
- *train/cls_loss*: The classification loss decreases from ~1.5 to ~0.5, hence model is improving in object recognition
- *train/dfl_loss*: The distribution focal loss falls steadily, indicating better bounding box precision.
- *precision*: Precision increases from ~0.2 to ~0.85, showing fewer false positives being detected over time.
- *recall*: Recall rises from ~0.3 to ~0.75, meaning the model is detecting more objects accurately.

Row 2: Validation metrics

- *val/box_loss*: Validation box loss follows training closely, suggesting good generalisation.

- *val/cls_loss*: Validation classification loss also decreases but with slightly more variation, showing that the system faces difficulties in separating the classes.
- *val/df_l_loss*: Validation focal loss decreases steadily, confirming stable learning.
- *mAP50*: mAP at IoU 0.5 improves from ~0.3 to >0.8, showing strong detection in low overlap thresholds.
- *mAP50-95*: The stricter mAP@[.50:.95] rises to ~0.65, showing good accuracy even under high overlap thresholds.

Overall it can be seen that training loss is steadily decreasing, showing effective learning and the validation loss closely following the training loss indicates that there is no overfitting.

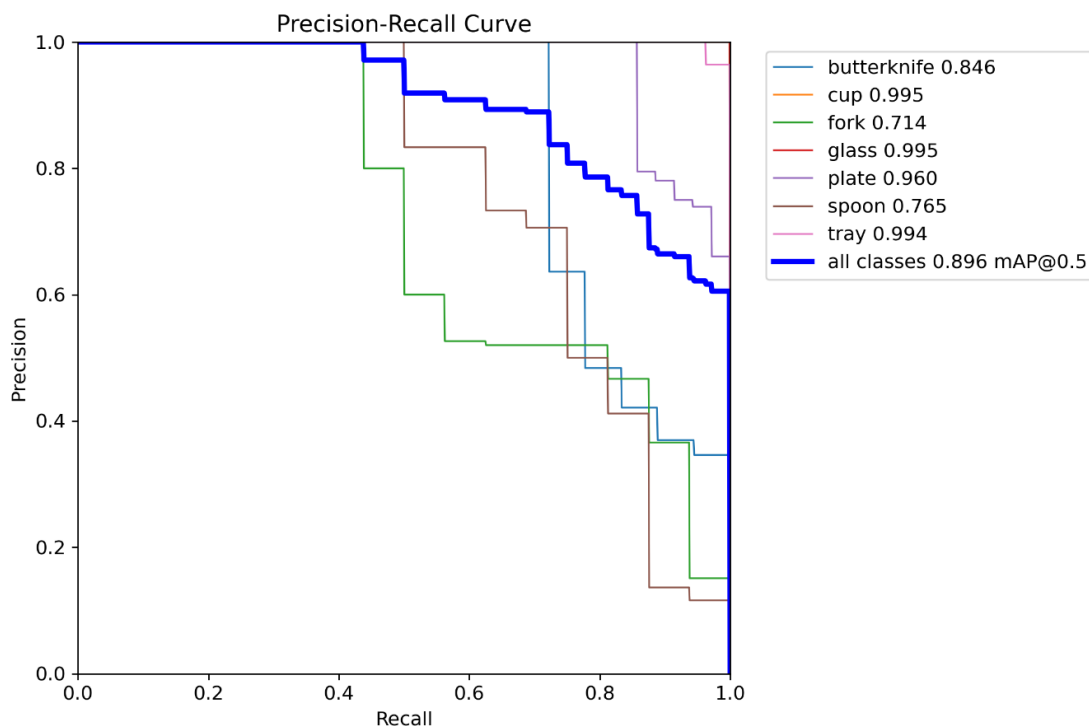


Figure 33 Precision and Recall curve of YOLOv11-XL

This Precision–Recall curve of YOLOv11-XL in Figure 33 shows per-class performance, where most classes, such as cup, glass, and tray, achieve very high precision and recall, while fork and spoon have lower scores. The thick blue line represents overall precision and recall across all classes and stays near the top, showing consistently high precision across most recall values, with a slight decline at higher recall because achieving maximum recall often requires accepting more false positives.

The confusion matrix for YOLOv11-XL is as given in Figure 34.

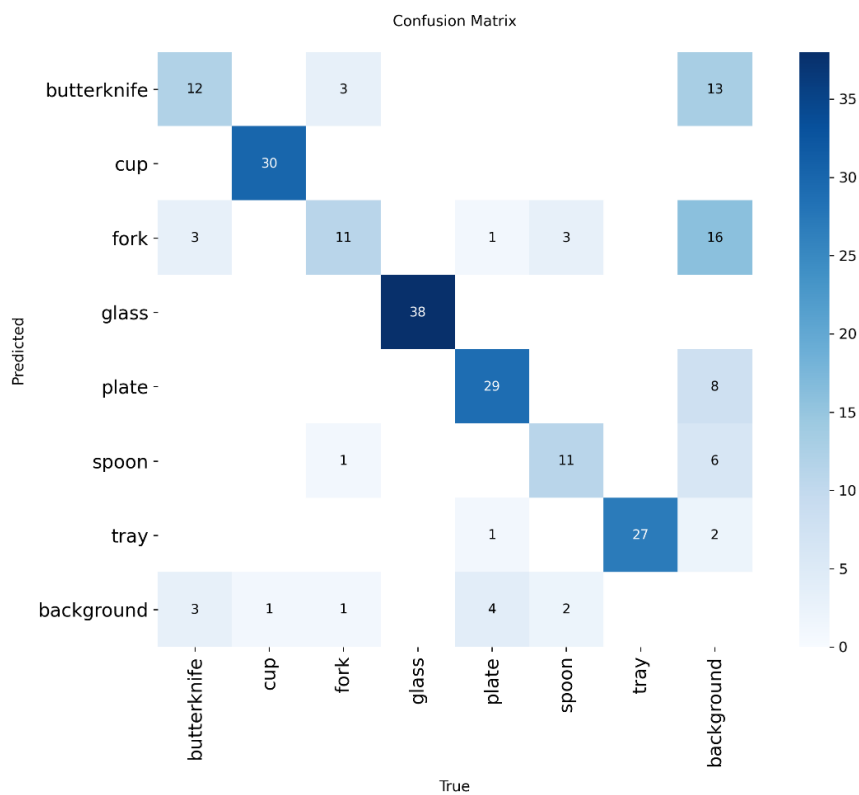


Figure 34 Confusion matrix for YOLOv11-XL

The class Background seen in the confusion matrix is implicitly created by YOLO models and isn't present in classes. It represents regions or detections that do not belong to any of the specified object categories. It can also be noted that most of the objects are identified correctly, except fork, spoon, and butter knife, which are often seen confused with one another. This could be because they look

similar to each other on one end and are smaller than other objects, and can be easily obstructed by the other items.

5.2 YOLO-classifier combination testing

The classifier model selection was conducted using four models: EfficientNet, MobileNet, ConvNeXt, and ViT. Table 13 below shows the combination of YOLOv11-L and YOLOv11-XL with the classifiers. YOLOv11-L is tested to ensure that it does not outperform YOLOv11-XL when combined with the classifiers. All the models were trained on the same images, which is the “NON-SAM (SEGMENT ANYTHING MODEL) CROP DATASET” explained in section [3.1.2](#).

Table 13 Comparison between YOLOv11 and Classifier models

Model	Precision	Recall	F1
YOLOv11-L + EfficientNet	0.6492	0.7337	0.6889
YOLOv11-L + MobileNet	0.6597	0.7456	0.7000
YOLOv11-L + ConvNeXt	0.6440	0.7278	0.6833
YOLOv11-L + ViT	0.6597	0.7456	0.7000
YOLOv11-XL + EfficientNet	0.7109	0.8876	0.7895
YOLOv11-XL + MobileNet	0.6967	0.8698	0.7737
YOLOv11-XL + ConvNeXt	0.7251	0.9053	0.8053
YOLOv11-XL+ ViT	0.7204	0.8994	0.8000

It can be noticed that YOLOv11-XL with ConvNeXt and YOLOv11-XL with ViT perform the best among all the other combinations in all the metrics, and these are considered for further system formulation.

The early stops were implemented for all classifiers, but for classifiers, the patience was made to ensure that overtraining does not happen. Patience of two was chosen for the classifier as they are smaller models and converge faster. The validation and training loss curves, precision-recall curves and confusion matrix for the selected models, which are ConvNeXt and ViT trained on NON-SAM (SEGMENT ANYTHING MODEL) CROP DATASET, are seen in the Figures 35-40.

5.2.1 Curves of ConvNeXt trained on non-SAM dataset

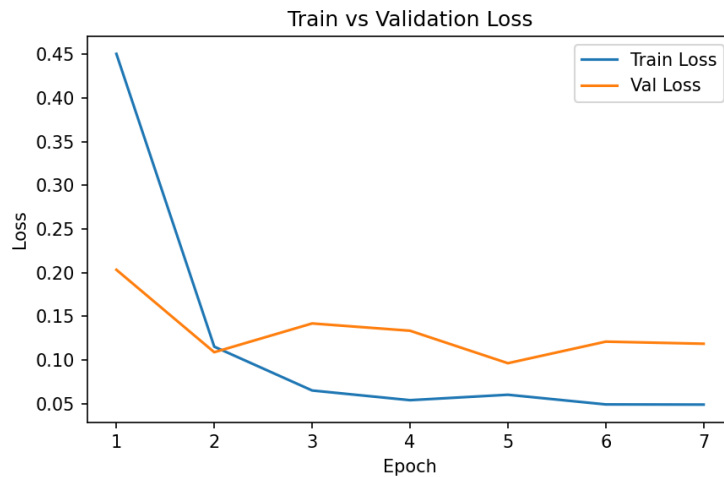


Figure 35 Training and validation curve for ConvNeXt trained on Non-SAM (Segment Anything Model) Crop Dataset

From the curve, it can be noticed that the training loss (blue) steadily decreases while the validation loss also decreases; later, it fluctuates as the system is actively learning, but towards the end, the validation loss stabilises. Hence, there are no signs of overfitting.

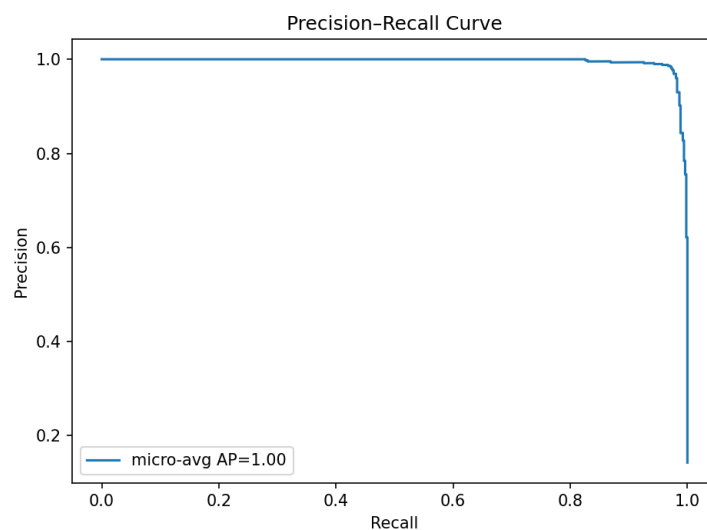


Figure 36 precision-recall curve for ConvNeXt trained on non-SAM (segment anything model) crop dataset

As seen from the precision-recall curve, the system achieves near-perfect precision and recall. The dip at the end of the Precision–Recall curve is because to achieve high recall, the system also introduces more false positives and hence the precision decreases.

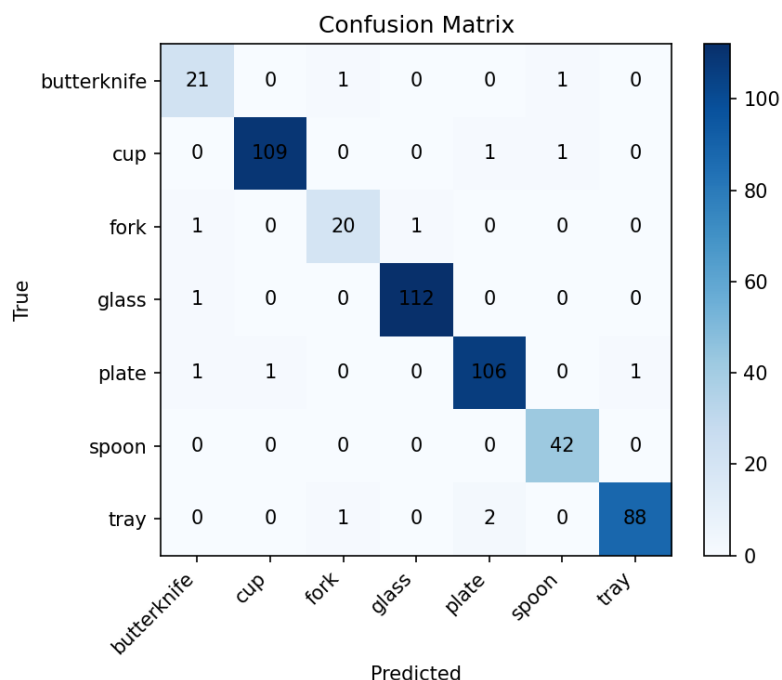


Figure 37 Confusion matrix for ConvNeXt trained on non-SAM (Segment Anything Model) Crop Dataset

The confusion matrix for ConvNeXt has minor misclassifications—e.g., "plate" is occasionally confused with "cup" or "tray", and "fork" is sometimes mistaken for "butterknife" or "glass".

5.2.2 Curves of ViT trained on non-SAM dataset

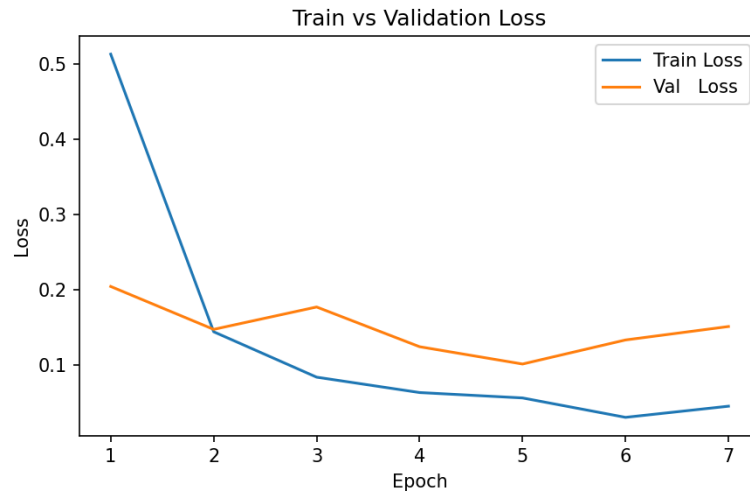


Figure 38 Training and validation curve for ViT trained on non-SAM (Segment Anything Model) Crop Dataset

A very similar trend to that obtained in ConvNeXt is seen in ViT. Here also, the training loss constantly reduces while validation loss fluctuates and further starts increasing, and it can be seen that the early stop prevented it from overfitting.

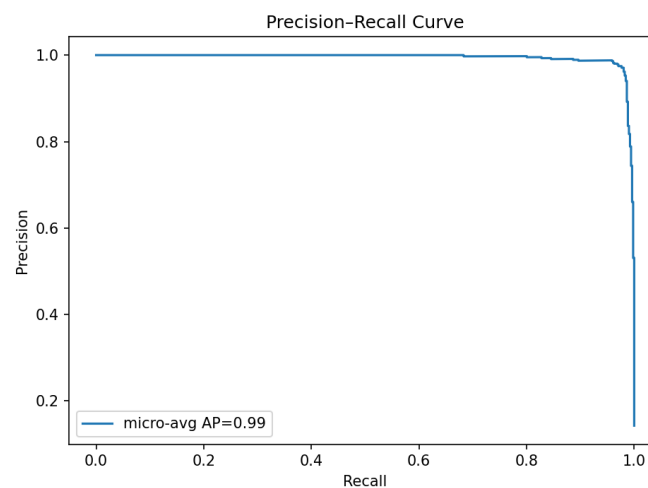


Figure 39 Precision-recall curve for ViT trained on NON-SAM (SEGMENT ANYTHING MODEL) CROP DATASET

The precision-recall seen in Figure 36 shows that the system has achieved good precision and recall and is very similar to the curve obtained for ConvNeXt.

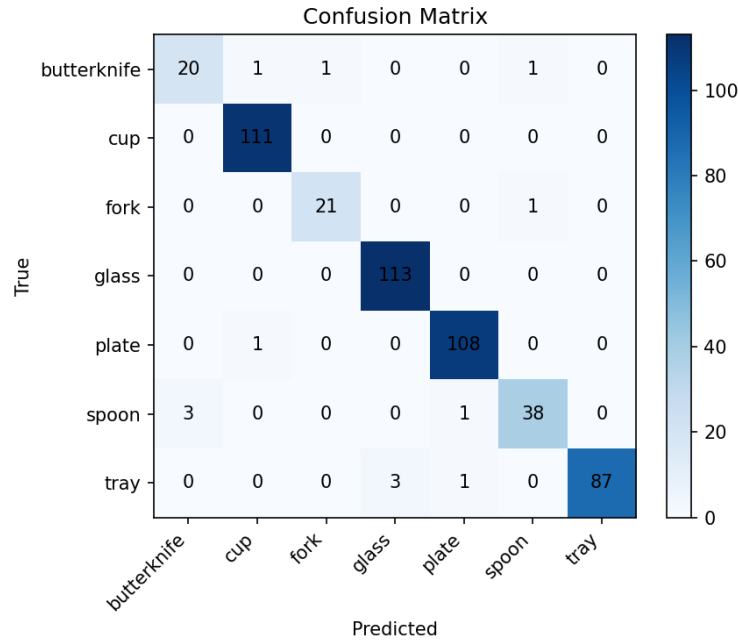


Figure 40 Confusion matrix for ViT trained on NON-SAM (SEGMENT ANYTHING MODEL) CROP DATASET

The confusion matrix in Figure 37 shows that most classes are predicted correctly, while some minor confusions between similar-looking classes like spoons, forks, and butter knives are present.

5.3 SAM integration and testing

The SAM model was decided to be added for further enhancement. All the following combinations were tried and tested under different conditions listed below:

- YOLOv11-XL + ConvNeXt + ViT
- YOLOv11-XL + SAM_ConvNeXt + ViT
- YOLOv11-XL + ConvNeXt + SAM_ViT
- YOLOv11-XL + SAM_ConvNeXt + SAM_ViT
- YOLOv11-XL + Submask_ConvNeXt + SAM_ViT
- YOLOv11-XL + SAM_ConvNeXt + Submask_ViT
- YOLOv11-XL + Submask_ConvNeXt + Submask_ViT

It had to be noted that SAM_ConvNeXt and SAM_ViT is when SAM full mask is used for the classifiers, and Submask_ConvNeXt and Submask_ViT is when classifiers use submasks. When nothing is mentioned before classifiers, it means that no SAM is used. And whenever SAM is used with a classifier, the classifier will be trained on the [SAM Crop dataset](#), and whenever SAM isn't used, the classifier will be trained on the [Non-SAM Crop dataset](#).

All models are tested in different environments, which are sunny, gloomy, shady, and corridor conditions mentioned in section [3.1.4.2](#). All the models were arranged in descending order of F1-score for each environment as given below in Tables 14-17.

5.3.1 Comparison of models on sunny environment

Table 14 Comparison of the models in sunny environment

Model	Precision	Recall	F1
YOLOv11-XL + ConvNeXt + sam_ViT	0.4461	0.7547	0.5607
YOLOv11-XL + submask_ConvNeXt + sam_ViT	0.4538	0.7264	0.5586
YOLOv11-XL + submask_ConvNeXt + submask_ViT	0.4538	0.7264	0.5586
YOLOv11-XL + sam_ConvNeXt + submask_ViT	0.4519	0.7233	0.5562
YOLOv11-XL + ConvNeXt + ViT	0.4424	0.7484	0.5561

YOLOv11-XL + sam_ConvNeXt + ViT	0.4312	0.7279	0.5421
YOLOv11-XL + sam_ConvNeXt + sam_ViT	0.4312	0.7296	0.5421

5.3.2 Comparison of models on gloomy environment

Table 15 Comparison of the models in gloomy environment

Model	Precision	Recall	F1
YOLOv11-XL + submask_ConvNeXt + sam_ViT	0.5185	0.8135	0.6333
YOLOv11-XL + sam_ConvNeXt + submask_ViT	0.5185	0.8135	0.6333
YOLOv11-XL + submask_ConvNeXt + submask_ViT	0.5185	0.8135	0.6333
YOLOv11-XL + ConvNeXt + sam_ViT	0.5065	0.8318	0.6296
YOLOv11-XL + sam_ConvNeXt + sam_ViT	0.5009	0.8226	0.6227
YOLOv11-XL + ConvNeXt + ViT	0.4991	0.8196	0.6204
YOLOv11-XL + sam_ConvNeXt + ViT	0.4935	0.8104	0.6134

5.3.3 Comparison of models on shady environment

Table 16 Comparison of the models in shady environment

Model	Precision	Recall	F1
YOLOv11-XL + submask_ConvNeXt + sam_ViT	0.4356	0.788	0.5610
YOLOv11-XL + sam_ConvNeXt + submask_ViT	0.4342	0.7853	0.5592
YOLOv11-XL + submask_ConvNeXt + submask_ViT	0.4327	0.7827	0.5573
YOLOv11-XL + ConvNeXt + sam_ViT	0.4217	0.8037	0.5532
YOLOv11-XL + ConvNeXt + ViT	0.419	0.7984	0.5495
YOLOv11-XL + sam_ConvNeXt + ViT	0.4162	0.7932	0.5459
YOLOv11-XL + sam_ConvNeXt + sam_ViT	0.4121	0.7853	0.5405

5.3.4 Comparison of models on corridor environment

Table 17 Comparison of the models in corridor environment

Model	Precision	Recall	F1
YOLOv11-XL + submask_ConvNeXt + sam_ViT	0.6776	0.7592	0.7161
YOLOv11-XL + sam_ConvNeXt + submask_ViT	0.6776	0.7592	0.7161

YOLOv11-XL + submask_ConvNeXt + submask_ViT	0.6776	0.7592	0.7161
YOLOv11-XL + ConvNeXt + sam_ViT	0.6639	0.7764	0.7157
YOLOv11-XL + ConvNeXt + ViT	0.6555	0.7666	0.7067
YOLOv11-XL + sam_ConvNeXt + ViT	0.6534	0.7641	0.7044
YOLOv11-XL + sam_ConvNeXt + sam_ViT	0.6513	0.7617	0.7022

It can be noted that in both corridor and gloomy conditions, the top three models achieved similar F1-scores. However, in the shady and sunny conditions, YOLOv11-XL + Submask_ConvNeXt + SAM_ViT (highlighted in yellow) performs better than all the other models, making it consistently at the top in all conditions. This indicates that it is the most robust and reliable model in varying real-world conditions. Hence, the YOLOv11-XL + Submask_ConvNeXt + SAM_ViT model is selected as the final model.

The YOLOv11-XL model is also tested for the 4 conditions and the following Table 18 was obtained.

Table 18 YOLOv11-XL in different categories

Category	Precision	Recall	F1
Sunny	0.4145	0.7013	0.5210
Gloomy	0.4898	0.8043	0.6088
Shade	0.3860	0.7356	0.5063
Corridor	0.6324	0.7396	0.6818

The comparison table of YOLOv11-XL + Submask_ConvNeXt + SAM_ViT and YOLOv11-XL in all conditions can be seen in Table 19.

Table 19 Comparison of the model with YOLOv11-XL

Condition	YOLOv11-XL F1 Score	Model F1 Score	% Increase
Sunny	0.5210	0.5586	7.22%
Gloomy	0.6088	0.6333	4.02%
Shade	0.5063	0.5610	10.80%
Corridor	0.6818	0.7161	5.03%

It can be noticed that the model consistently outperforms YOLOv11-XL, with improvements ranging from 4.02% (Gloomy) to 10.80% (Shade). It can be noted that in high contrast situations like sunny, even though the objects are not visible clearly due to the glare of the light, improvements in the F1 score can be noticed.

Overall, these findings demonstrate that the proposed enhancements lead to improved detection consistency and adaptability across diverse real-world conditions, particularly in situations where standard YOLOv11-XL struggles.

It can be seen that the YOLOv11-XL + Submask_ConvNeXt + SAM_ViT performs better than just YOLOv11-XL in all conditions.

The training and validation loss graphs, precision-recall graph and confusion matrix for ConvNeXt and ViT trained on “SAM CROP DATASET” in section [3.1.3](#) is seen in Figures 41-46. For these SAM-based models, early stopping with a patience value of 2 was implemented to prevent overfitting.

5.3.4.1 Curves of convnext trained on sam crop dataset



Figure 41 Training and validation curve for ConvNeXt trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET

It can be seen that the training loss decreases steadily, while the validation loss fluctuates. It can be noted that early stopping successfully prevented overfitting as soon as the system began to show slight signs of overfitting.

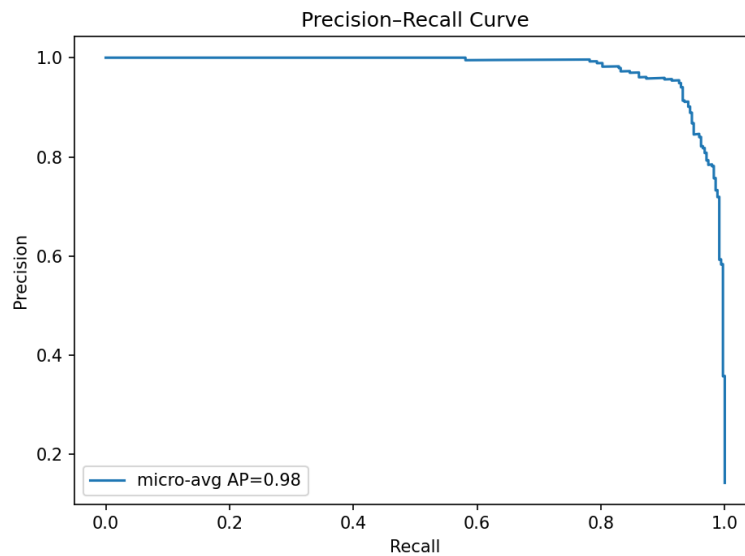


Figure 42 Precision-Recall curve for ConvNeXt trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET

This precision–recall graph shows excellent precision across almost the entire recall range.

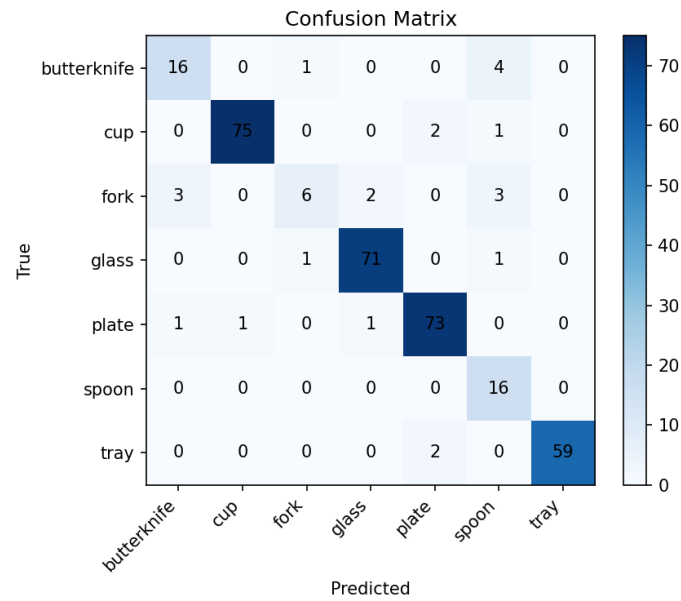


Figure 43 Confusion matrix for ConvNeXt trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET

From the confusion matrix it can be seen that while the classification for most classes is extremely good, there are slight misclassifications for butterknife, fork, and tray.

5.3.4.2 Curves of vit trained on sam dataset

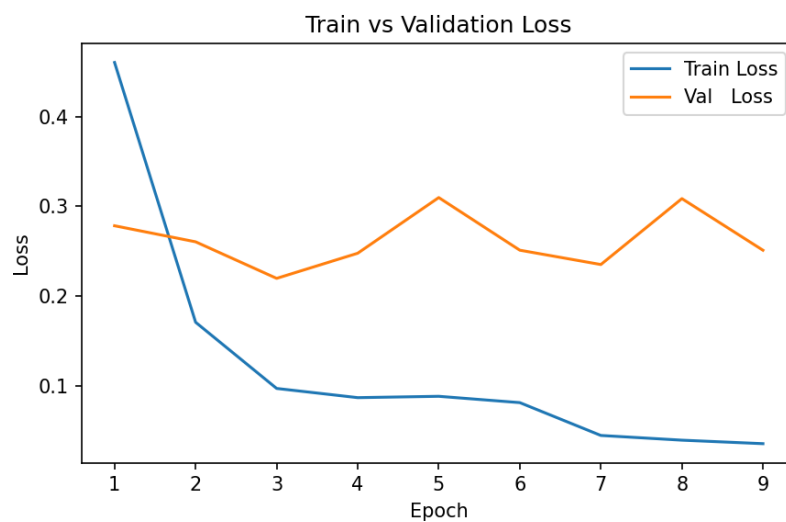


Figure 44 Training and validation curve for ViT trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET

The graph in Figure 44 shows that while the training loss decreases, the validation loss fluctuates, and while slight overfitting was seen in the beginning, as the validation loss further decreased and stabilised, the early stopping prevented any further overfitting.

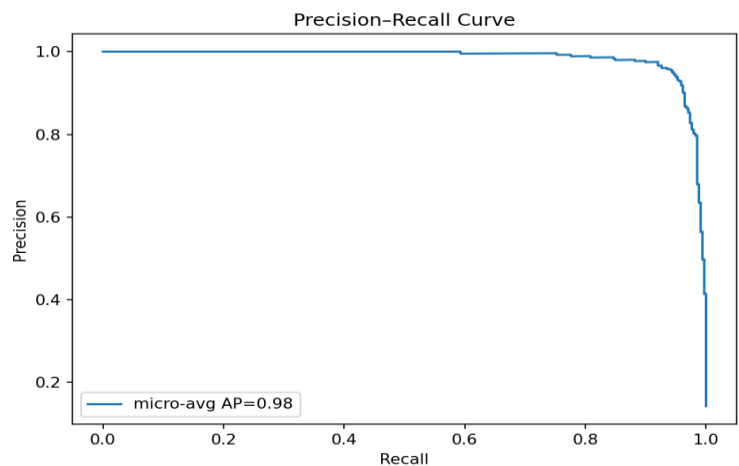


Figure 45 Precision-Recall curve for ViT trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET

The above graph shows that the precision remains high across almost the entire recall range, indicating that the system performs excellent classification throughout.

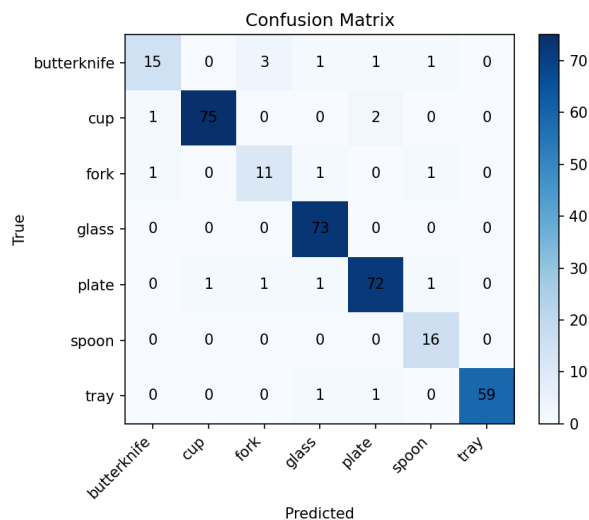


Figure 46 Confusion matrix for ViT trained on SAM (SEGMENT ANYTHING MODEL) CROP DATASET

From the confusion matrix in Figure 46, it can be noted that the system performs well for most classes, with only slight misclassifications observed in classes such as fork, plate, and tray.

5.4 Testing of the model in different heights and distances

As explained in section [4.2.4](#), the models were tested on different heights and distances and tested on the datasets explained in [3.1.4.3 “DATASETS FOR DIFFERENT HEIGHTS AND DISTANCES”](#), and the Tables 20-22 were obtained.

5.4.1 Height 60 cm

Table 20 Comparison of height 60cm and different distances

Distance	Precision	Recall	F1
140 cm	0.7121	0.8868	0.7899
190 cm	0.5085	0.5357	0.5217
240 cm	0.4762	0.3448	0.4000

5.4.2 Height 110cm

Table 21 Comparison of height 110cm and different distances

Distance	Precision	Recall	F1
190 cm	0.5373	0.6545	0.5902
240 cm	0.3333	0.3333	0.3333
290 cm	0.2000	0.1905	0.1951

5.4.3 Height 170 cm

Table 22 Comparison of height 170cm and different distances

Distance	Precision	Recall	F1
240 cm	0.5323	0.6875	0.6000
290 cm	0.3651	0.4600	0.4071
340 cm	0.0976	0.0952	0.0964

From the tables, it can be observed that

- At smaller heights (60 cm) and closer distances (140 cm), the model achieved highest performance with an F1-score of 0.7899, indicating optimal object detection and classification under such conditions.
- As the distance and height increased, that is at height 170 cm and distance 340 cm, there was a significant drop in performance, indicating difficulty in the detection and classification of objects.
- The performance consistently decreased as the height and distance increased, showing that it works better at moderate height and distance.

This trend is seen could be due to the fact that the system uses webcams, whose clarity is highly affected at greater distances and heights, and also that spoons, forks, and butterknives can be easily overlapped by larger objects, leading to missed detections at larger heights.

5.5 Real-world robotic integration

The system is further integrated with the robot (swift robot platform) using WSS connection. The code for which is given in Appendix B. Here, the detection system sends a stop signal when an object is detected to the WebSocket using FastAPI, which then forwards it to a robot using an active WebSocket connection. The robot, connected as the WebSocket client, receives the command to stop the

robot which is “cancel_goal” is this case. The Figure 47 below, which is the object detection screen of the system running the machine vision system, shows that the robot has successfully received the stop signal from the detection system, as indicated by the log message “Received: {“name”: “cancel_goal”, “args”: []}” in the terminal.

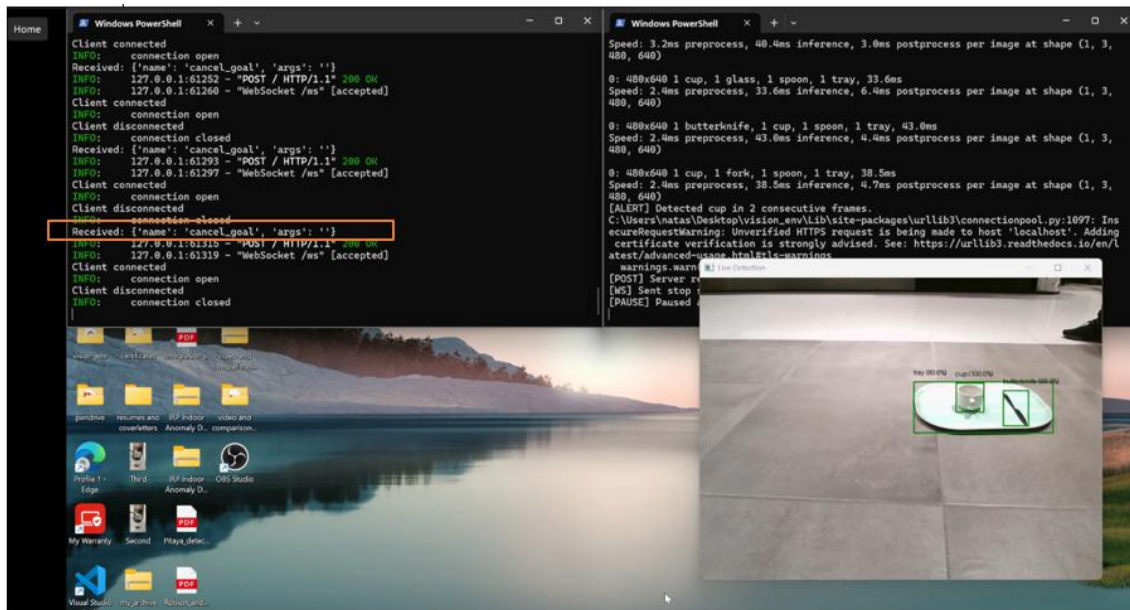


Figure 47 Object detection screen

While this could have been implemented through a simple master–slave serial connection, compatibility issues between the computer system and the robot required the use of this WebSocket-based approach instead. The robot used Ros 1 melodic which was supported by Ubuntu 18, and the computer system couldn't be integrated with Ubuntu 18 with an internet connection, as it showed internet drivers aren't found. This could be due to the fact that Ubuntu 18 has reached its end of life.

5.5.1 Testing the robot-integrated system in different environments

After the integration of the system with the robot, it was tested under different conditions, but the stop signal was sent if the same object was detected in two consecutive frames. This was to ensure that no false detection triggered the signal and made the robot stop. It has to be noted that for each condition, the system was run five times and its results stored and used for calculations.

The precision, recall and F1-score metrics were calculated using the ground truth annotations manually created from the original frames, and the corresponding detected frames stored by the system. The system was tested on dark background, white background and corridor settings and the dataset created was as discussed in [3.1.4.4 “DATASETS FOR ROBOT INTEGRATION”](#).

The test results obtained are given in Table 23.

Table 23 Test results for robot integration

Situation	Precision	Recall	F1
Dark floor	1.000	0.3636	0.5333
Corridor	0.6667	0.1905	0.2963
White floor	0.7083	0.2576	0.7083

From Table 23, it can be seen that in the dark floor setting, the system achieved perfect precision (1.000), indicating that all detected objects were correct with no false positives. However, the recall was relatively low (0.3636), which suggests that the majority of the objects weren't detected.

The number of detected objects for dark floor setting is given in Table 24.

Table 24 Number of detected objects in the Dark floor setting

Class	Number of items
Butterknife	2
Cup	2
Fork	0
Glass	0
Plate	4
Spoon	0

Tray	8
-------------	---

This can be compared to the original number of classes present in dark floor settings, which is shown in Table 25.

Table 25 Number of original objects in the Dark floor setting

Class	Number of items
Butterknife	6
Cup	12
Fork	0
Glass	2
Plate	14
Spoon	2
Tray	10

This shows that only a few objects were detected by the system.

Further, for the corridor setting, it can be seen that both precision (0.6667) and recall (0.1905) were lower, leading to the poorest F1-score (0.2963) among the three environments. This may be attributed to complex lighting and shadows present in corridor settings.

The number of detected objects in corridor is given in Table 26.

Table 26 Number of detected objects in the corridor setting

Class	Number of items
Butterknife	6
Cup	4
Fork	0
Glass	0
Plate	2
Spoon	0
Tray	0

Table 26 can be compared to the actual number of objects present, which is shown in Table 27.

Table 27 Number of original objects in the corridor setting

Class	Number of items
Butterknife	6
Cup	14
Fork	0
Glass	0
Plate	3
Spoon	0
Tray	10

This further proves that the system didn't perform well.

In the white floor environment, the model achieved a moderate precision of 0.7083 and a recall of 0.2576. The F1-score (0.3787) indicates that the system remained reasonably precise. The number of detected objects is listed in Table 28.

Table 28 Number of detected objects in the white floor setting

Class	Number of items
Butterknife	1
Cup	10
Fork	3
Glass	0
Plate	2
Spoon	0
Tray	8

The number of actual objects present are given in Table 29.

Table 29 Number of original objects in the white floor setting

Class	Number of items
Butterknife	3
Cup	18
Fork	4
Glass	19
Plate	0
Spoon	7
Tray	15

This shows that the system performed relatively better in the white background compared to the other 2 conditions.

As can be seen, the corridor setting has the least F1-score and hence the least performance. The number of frames that the object is present before stopping was increased from 2 to 5. There was a significant improvement in its performance when this was done. The comparative table is shown in Table 30.

Table 30 Comparative table for 2-frame and 5-frame logic

Number of frame where object is present	Precision	Recall	F1
5 frames	0.9459	0.5833	0.7216
2 frames	0.6667	0.1905	0.2963

All metrics have better values in 5-frame logic when compared to the 2-frame logic. Hence, the system performance can be improved by increasing the number of frames the object has to be present.

Table 31 shows the number of items that were detected in 5 frame logic.

Table 31 Number of detected objects in 5-frame logic

Class	Number of items
Butterknife	10
Cup	25
Fork	2
Glass	7
Plate	6

Spoon	5
Tray	19

This can be compared to the actual number of items that were present in 5 frame logic which is given in Table 32.

Table 32 Number of original objects in the 5-frame logic

Class	Number of items
Butterknife	10
Cup	30
Fork	10
Glass	15
Plate	20
Spoon	10
Tray	25

The number of detected and original objects for 2-frame is as given in Table 26 and Table 27.

The confusion matrix for the 2 Frame logic is as given in Figure 48.

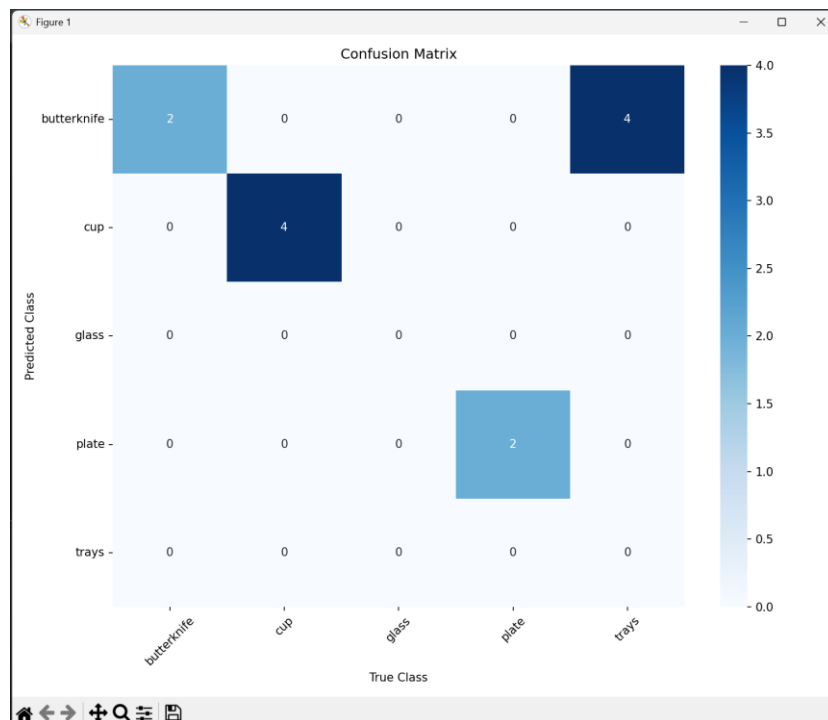


Figure 48 Confusion matrix of 2-frame logic

The confusion matrix for 5-frame logic is shown in Figure 49.

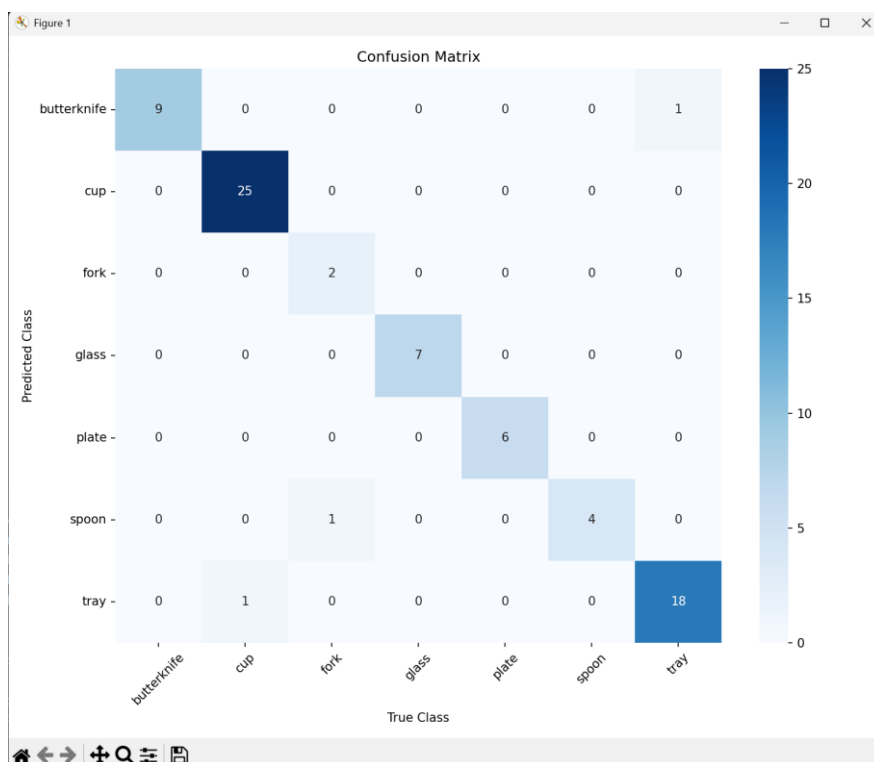


Figure 49 Confusion matrix of 5-frame logic

These confusion matrices depict the improvement in the detection in 5-frame logic clearly, as it predicts a wider number of classes and has significantly higher true positives.

Since only five tests were conducted, if one of the values is higher, it will increase the overall performance of the system. Hence, it is important to understand the overall trend, where most values lie, and identify whether any results are significantly higher or lower than the rest. For this, box and whiskers plot is used to spot patterns and understand how consistent the model is. To create the box and whiskers plot, each test has to be run separately for both logics. The precision, recall, F1, and IoU are calculated separately for each test to understand whether the 2-frame logic or the 5-frame logic performs better. The IoU metric was added to ensure that after the system was selected, tested, and integrated, the detected objects matched the location where the original object is present—i.e., they fell within the same bounding boxes as the manually annotated objects. This ensured the classification was accurate and well localised. Table 33 presents the results obtained for 2-frame logic.

Table 33 Test-wise table for 2-frame logic

Test Number	Precision	Recall	F1	IoU
1	0.000	0.000	0.0000	0.000
2	1.000	0.1429	0.2500	0.7462
3	1.000	1.000	1.000	0.8194
4	0.000	0.000	0.000	0.000
5	1.000	1.000	1.000	0.8349

The Table 34 is for the 5- frame logic.

Table 34 Test-wise table for 5-frame logic

Test Number	Precision	Recall	F1	IoU
--------------------	------------------	---------------	-----------	------------

1	1.000	0.8667	0.9286	0.8720
2	1.000	0.7333	0.8462	0.7823
3	0.9444	0.5667	0.7083	0.7972
4	0.9375	0.6000	0.7317	0.8335
5	0.8750	0.4000	0.5490	0.8478

A box and whiskers plot we created from the two tables above to understand the performance better. The box and whiskers plot are given in Figure 50.

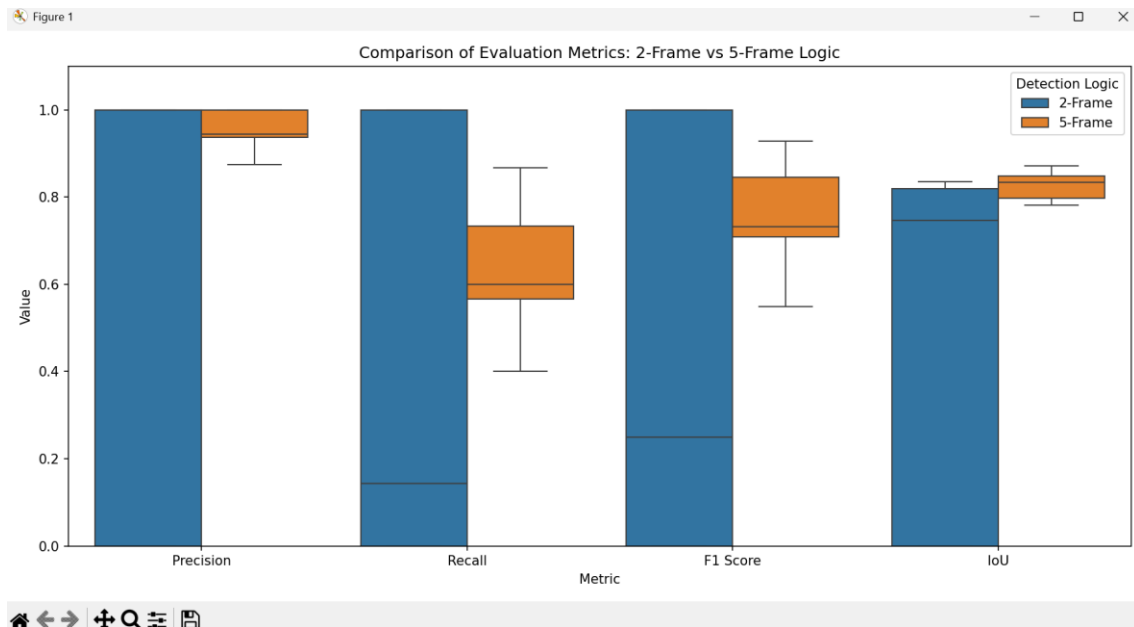


Figure 50 Box and whiskers plot of 5-frame and 2-frame logic

From the plot, it can be deduced that 5-frame logic has higher median values and a more consistent result range, while 2-frame logic has extreme values and thus lacks robustness. Hence, it further confirms that frame 5 logic performs better than frame 2.

Figure 51 shows the robot detection frame and the image of the stopped robot in front of the objects when the 5 frame is implemented.



Figure 51 Robot detection frame

Figure 52 shows the stopped robot before the obstacles after detecting it as shown in the detection frame in Figure 51.



Figure 52 The stopped robot

5.5.2 Testing the robot-integrated system on different tilt angles

The 5-frame logic is then tested for different tilt angles. The datasets created is explained in [section 3.1.4.4.4](#). Table 35 was created showing the precision, recall and F1 scores of each tilt angle.

Table 35 Comparison of different tilt angles

Tilt (degrees)	Distance between the robot and objects when the robot stops	Precision	Recall	F1
20	Between 60 to 70cm	0.9524	0.4444	0.6061
30	Between 50 to 60cm	0.9315	0.5913	0.7234
40	Between 30 to 50cm	0.9464	0.4274	0.5889
50	Between 10 to 30cm	0.9552	0.4571	0.6184
60	Between 0 to 10cm	0.9636	0.6400	0.7692

It should be noted that the tilt angles are measured from the horizontal axis, with higher values indicating a greater downward tilt. The camera is placed as seen in Figure 53 with the camera circled in red and its tilt varied as required.



Figure 53 Camera position

At lower tilts (more horizontal), some objects may appear smaller or partially occluded due to perspective. At higher tilts (camera pointing more downward), objects become more fully visible and centred in the frame, making them easier to detect.

It can be noted that

- At 20° tilt, the camera is nearly horizontal and hence the object is inside the field of vision at a greater distance itself and hence only the larger objects will be detected
- As the tilt increases, the camera points more downward and gets closer to the objects.
- At 60° tilt, the camera is nearly aiming down and hence the object is detected very close to the robot, and hence each object can be viewed better
- At 50°, the camera angle is neither top-down nor side-on, making object shapes and edges harder to distinguish and reducing detection accuracy.

It is hard to select which model performs best from the confusion matrices and hence a Box and whiskers plot must be created for the different tilt angles to understand its performance and hence precision, recall, F1, and IoU is calculated separately for each test in each tilt angle, and these are given in Tables 36-40 along with the overall confusion matrices for each tilt in Figures 54-58. The test respective confusion matrices are given in Appendix B.

5.5.2.1 Tilt 20°

Table 36 Comparative table for tilt 20°

Frame no	Precision	Recall	F1	IoU
1	1.000	0.2400	0.3871	0.8692
2	0.8333	0.3333	0.4762	0.7616
3	0.9444	0.6800	0.7907	0.8673
4	1.000	0.7000	0.8235	0.7990
5	1.000	0.2400	0.3871	0.6704

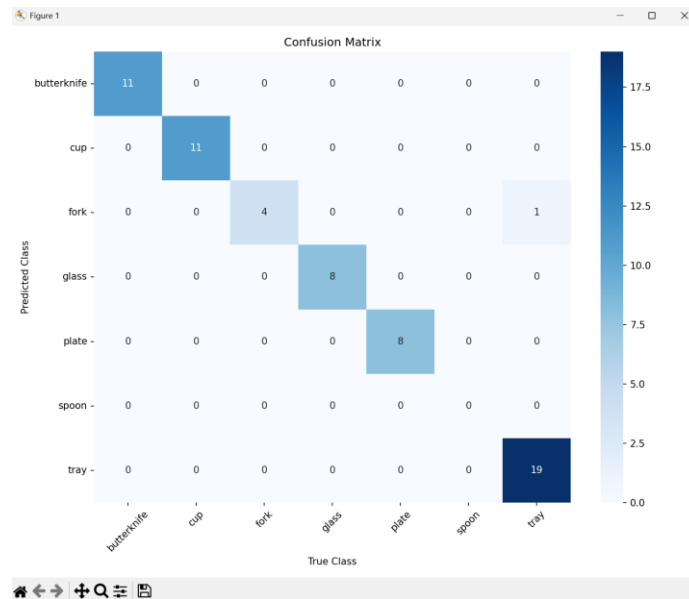


Figure 54 Confusion matrix of tilt 20°

The model accurately detected the 11 butterknives, 11 cups, 4 forks, 8 glasses, 8 plates and 19 trays accurately. The model also misclassified 1 tray as a fork.

5.5.2.2 Tilt 30°

Table 37 Comparative table for tilt 30°

Frame no	Precision	Recall	F1	IoU
1	1.000	0.8667	0.9286	0.8794
2	0.9091	0.7143	0.8000	0.7713
3	0.9412	0.5517	0.6957	0.7815
4	0.9375	0.6000	0.7317	0.8099
5	0.8750	0.4375	0.5833	0.7847

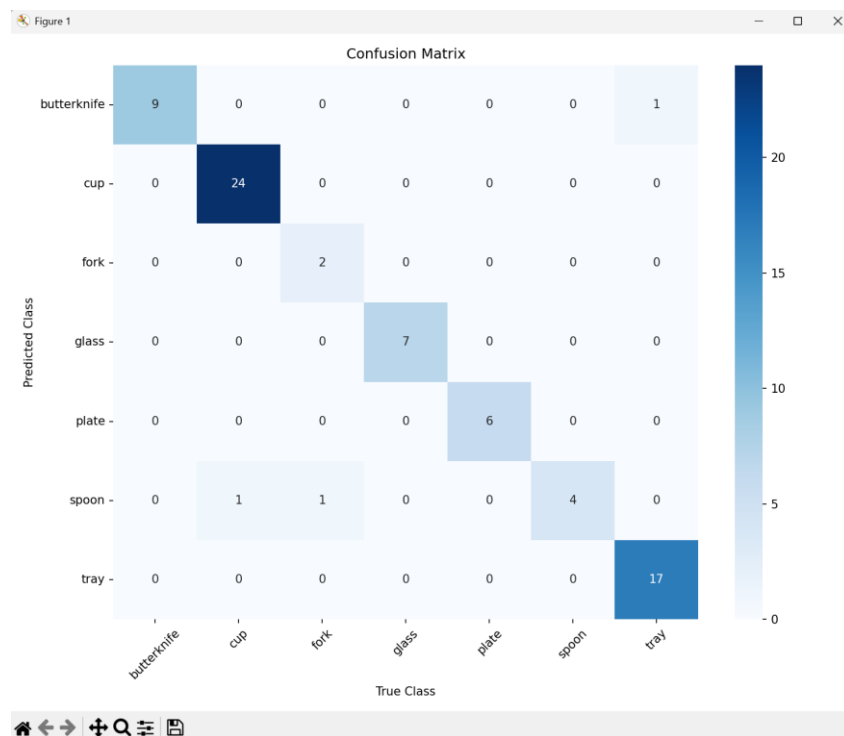


Figure 55 Confusion matrix of tilt 30°

The model accurately identified 9 butterknives, 24 cups, 2 forks, 7 glasses, 6 plates, 4 spoons and 17 trays. While it wrongly detected 1 tray as butterknife, 1 cup as spoon and 1 fork as spoon.

5.5.2.3 Tilt 40 °

Table 38 Comparative table for tilt 40°

Frame no	Precision	Recall	F1	IoU
1	0.8750	0.3500	0.5000	0.7867
2	1.000	0.4333	0.6047	0.7893
3	1.000	0.6800	0.8095	0.8186
4	1.000	0.2500	0.4000	0.9302
5	0.8462	0.3793	0.5238	0.8376

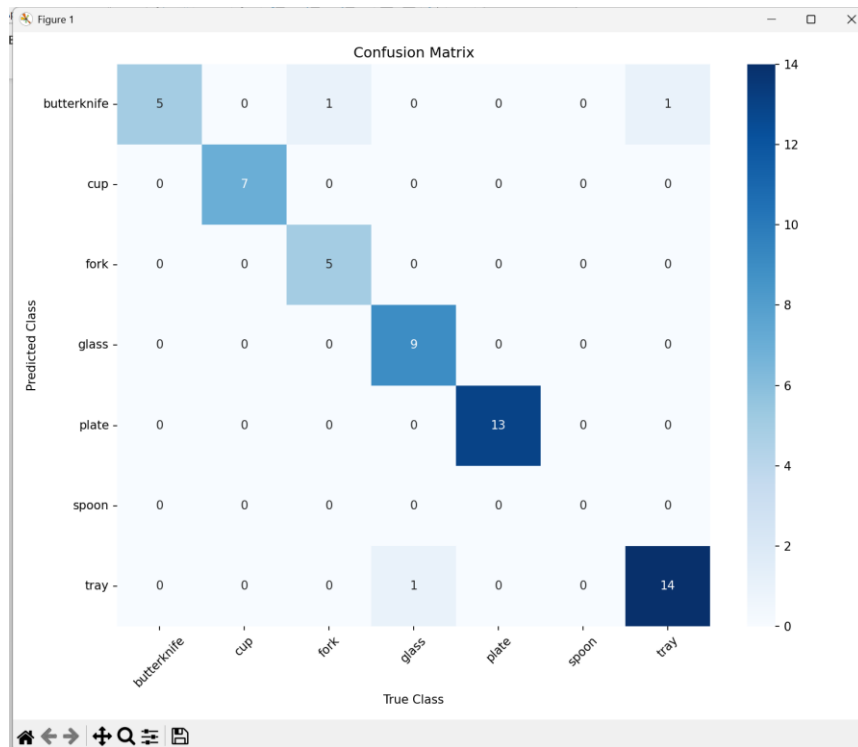


Figure 56 Confusion matrix of tilt 40°

The model classified 5 butterknife, 7 cup, 5 fork, 9 glasses, 13 plates, 14 trays accurately and misclassified 1 glass as tray, 1 fork as butterknife and 1 tray as butterknife

5.5.2.4 Tilt 50°

Table 39 Comparative table for tilt 50°

Frame no	Precision	Recall	F1	IoU
1	1.000	0.7600	0.8636	0.8242
2	1.000	0.4333	0.6047	0.8341
3	0.8333	0.3333	0.4762	0.8783
4	1.000	0.3600	0.5294	0.8179
5	0.9286	0.4333	0.5909	0.8227

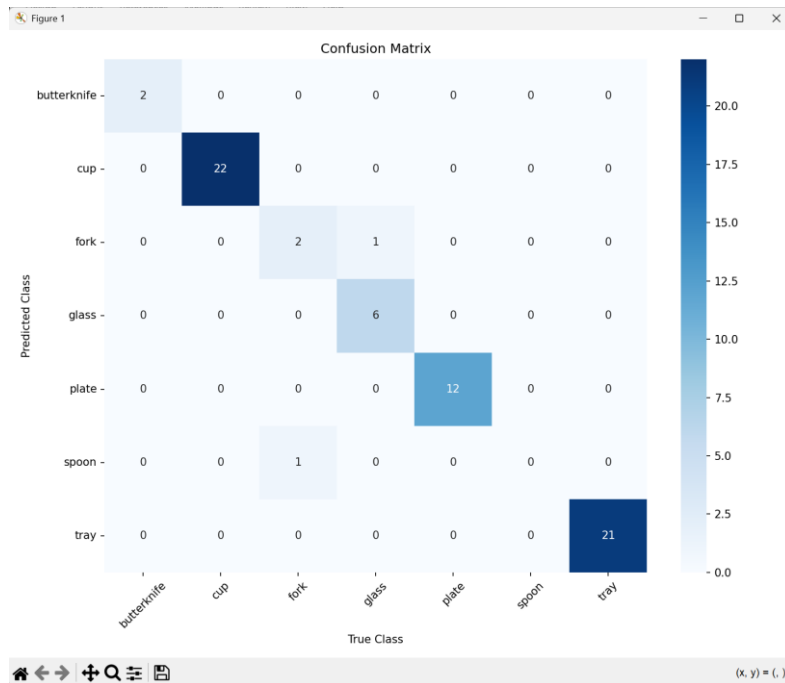


Figure 57 Confusion matrix of tilt 50°

The model detected 2 butterknife,22 cups,2 fork,6 glasses,12 plates,21 trays and mis identified 1 spoon as fork

5.5.2.5 Tilt 60°

Table 40 Comparative table for tilt 60°

Frame no	Precision	Recall	F1	IoU
1	0.9286	0.5200	0.6667	0.8629
2	1.000	0.6000	0.7500	0.8943
3	1.000	0.7931	0.8846	0.8914
4	0.8889	0.6957	0.7805	0.8180
5	1.000	0.5652	0.7222	0.8391

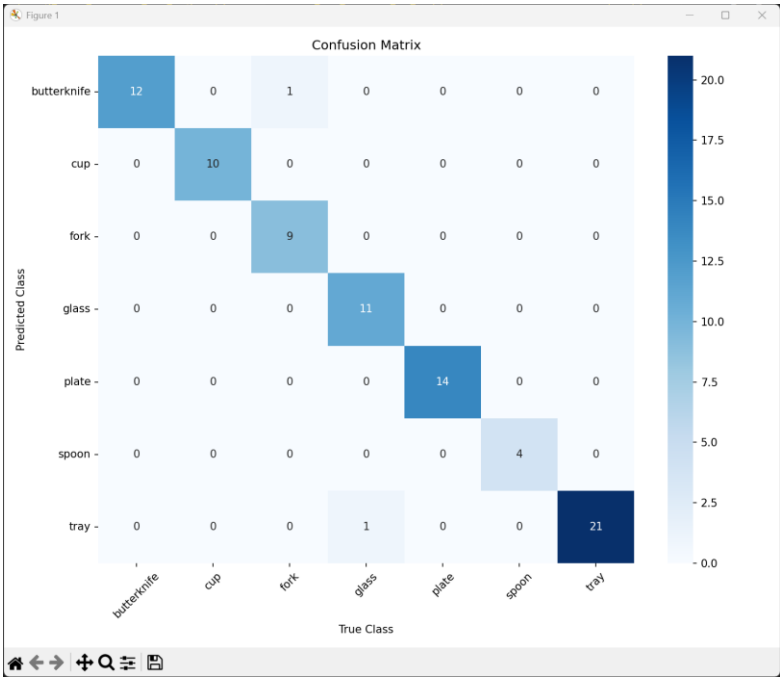


Figure 58 Confusion matrix of tilt 60°

The model classified 12 butter knives, 10 cups, 9 forks, 11 glasses, 14 plates, 4 spoons and 21 trays accurately, while 1 glass and 1 fork were detected as a tray and butter knife, respectively.

5.5.2.6 Box and Whiskers Plot

The box and whisker plot created from the above tables is given in Figure 59.

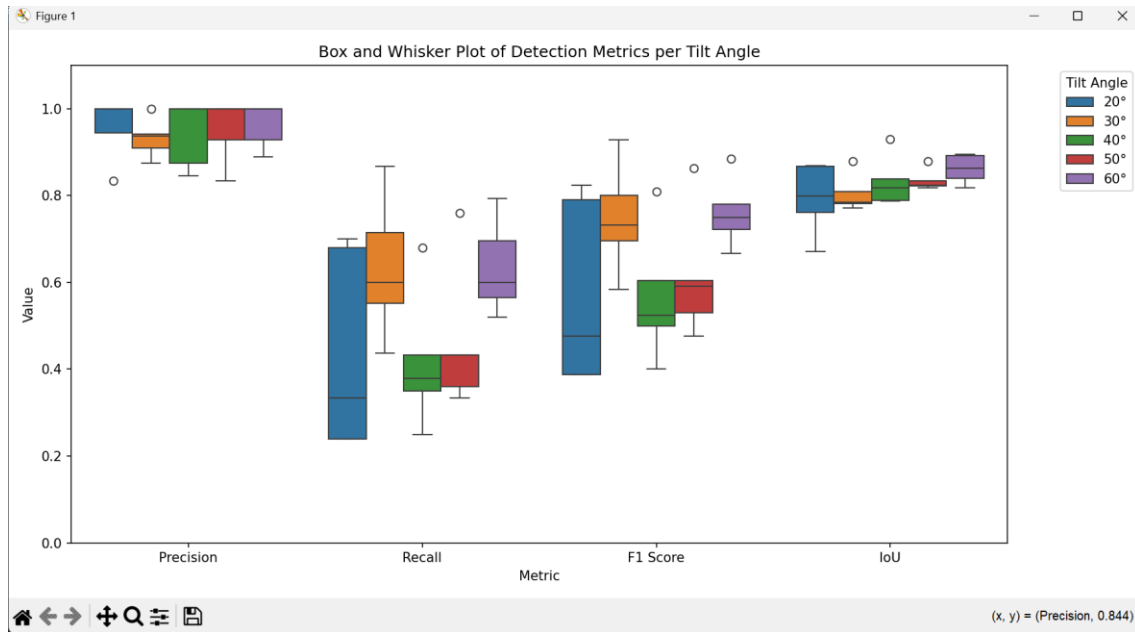


Figure 59 Box and whiskers plot for different tilt angles

From the above figure, tilt 60° has the most consistent high value and the least variations, and hence it is the best model. However, it should be noted that at a 60° tilt, the objects are detected very close to the robot due to the restricted field of view. As a result, the robot sometimes slightly moves over the objects, as it has to detect the object for 5 frames as this detection is made, the robot moves forward, and by the time the 5th frame detection is made and the robot stops, the robot mostly runs over the objects a bit. This can be viewed in the Table 33 as at the stop position, the distance between the robot and the objects is only between 0 to 10 cm.

Hence the second-best tilt angle, which is 30°, will be most suitable as this has consistently high median, is compact, and leaves enough space between the robot and objects and hence no colliding with the object. It also has better

precision and recall while 20°,40° and 50° compromise recall for precision 30°
find the balance between the two.

6 CONCLUSION AND FUTURE WORK

In this thesis, a robust, real-time object detection system is developed for the detection of items in the categories of plates, cups, glasses, trays, spoons, forks and butter knives. To achieve this, various custom datasets for training and testing the models were created. The 2-stage anomaly detection system was later developed, and its performance was evaluated through rigorous testing. The system was then integrated with a robot and tested again for its adaptability and practicality.

The final system is working comparatively better than a YOLOv11-XL model as seen in Tables 10-14 discussed above. The core of the system is based on the YOLOv11-XL model integrated with classifiers using SAM for object isolation. Among the different model combinations tested, the YOLOv11-XL + Submask_ConvNeXt + SAM_ViT model consistently achieved the highest F1-scores across different environments (sunny, gloomy, shade and corridor), demonstrating its robustness.

The integration of the machine vision system to a robot showed that it is integrable to robotic systems, showing its scalability and the vast ways that the system could be implemented. Further, it was noted that increasing the detection logic from 2 frames to 5 frames significantly improved the accuracy. Experiments with different heights showed that the system works in all conditions but is more efficient in moderate heights and distances. The tilt angles' impact on the performance was also studied, and it showed that a 60-degree tilt performed the best, but had the potential of the robot colliding with the objects. Hence, the second-best tilt angle, which was 30 degrees, is the best option. Altogether, the research increases the detection accuracy but also demonstrates how different environments and conditions affect the system and how the system works in all different conditions. The system is adaptable, accurate and capable of functioning in diverse conditions.

The system could be further improved in many areas to increase its robustness and extend it to other areas:

1. Testing how leftover foods or liquids in the objects and utensils affect the system performance and improving the model to work in such conditions.
2. Testing how the speed of the robot affects the detections and to find the balance between the robot speed and object detections.
3. Adding more object classes like napkins, bottles, etc. to increase the detection spectrum.
4. Location identification and integrating an automatic email or alert generation mechanism to notify the staff of the location and objects present wherever the objects are detected.
5. Storing the model detections and use them to auto-train the model at set intervals of time.

References

1. Lana Cook. How AI is transforming hotels—according to hospitality leaders [Internet]. Hospitality Net. 2025 [cited 2025 Aug 15]. Available from: <https://www.hospitalitynet.org/opinion/4127498.html>
2. Gov.Uk. Unattended and Suspicious items [Internet]. 2020 [cited 2025 Aug 15]. Available from: <https://www.gov.uk/government/publications/crowded-places-guidance/unattended-and-suspicious-items>
3. Frye WD. Twelve common housekeeping errors found in many hotels [Internet]. Hospitality Lawyer. 2019 [cited 2025 Jul 20]. Available from: <https://hospitalitylawyer.com/wp-content/uploads/2019/01/12-common-housekeeping-errors.pdf>
4. Augustine. SOP – room service / IRD – tray, trolley, cart removal [Internet]. Setupmyhotel. 2023 [cited 2025 Jul 20]. Available from: <https://setupmyhotel.com/hotel-sop-standard-operating-procedures/food-and-beverage-service-sop/sop-room-service-ird-tray-trolley-cart-removal/>
5. Heimanson & Wolf LLP. Common hazards and risks at hotels, motels, and resorts in Los Angeles [Internet]. Heimanson & Wolf, LLP. [cited 2025 Jul 20]. Available from: <https://www.hwllplaw.com/common-injuries-at-los-angeles-hotels-motels-resorts/#:~:text=Slips%2C%20trips%2C%20and%20falls:,in%20walkways%2C%20and%20uneven%20surfaces.>
6. Health and Safety Executive. Slips and trips in health and social care [Internet]. Health and safety executive. [cited 2025 Jul 20]. Available from: <https://www.hse.gov.uk/healthservices/slips/what-you-need-to-do.htm?>
7. M CK, Basthikodi M. Machine learning approaches for abandoned luggage detection. In: 2023 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER) [Internet].

- Mangalore, India: IEEE; 2023 [cited 2025 Aug 13]. p. 8–12. Available from: <https://doi.org/10.1109/DISCOVER58830.2023.10316663>
8. Nebili B, Louali R, Sakhi S, Laidoudi A, Youcefi A, Khellal A, Boulahmar M. Experimental evaluation of monocular vision-based obstacle detection methods. In: 2024 1st International Conference on Electrical, Computer, Telecommunication and Energy Technologies (ECTE-Tech) [Internet]. Oum El Bouaghi, Algeria: IEEE; 2024 [cited 2025 Aug 13]. p. 1–6. Available from: <https://doi.org/10.1109/ECTE-Tech62477.2024.10851172>
 9. Shah V, Sardhara R. A systematic survey of abnormal event detection models from video with application, limitations and challenges. In: 2024 International Conference on Intelligent Algorithms for Computational Intelligence Systems (IACIS) [Internet]. Hassan, India: IEEE; 2024 [cited 2025 Aug 13]. p. 1–7. Available from: <https://doi.org/10.1109/IACIS61494.2024.10721820>
 10. Fang Q, Su Q. Weakly supervised anomaly detection by utilizing incomplete anomaly information. In: 2024 7th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI) [Internet]. Guangzhou, China: IEEE; 2024 [cited 2025 Aug 13]. p. 01–5. Available from: <https://doi.org/10.1109/ACAI63924.2024.10899497>
 11. R N, S PU, Mohan M, S N, Mohan A, Samuel J. Voice controlled moving robot for smart surveillance. In: 2024 5th International Conference on Circuits, Control, Communication and Computing (I4C) [Internet]. Bangalore, India: IEEE; 2024 [cited 2025 Aug 13]. p. 354–9. Available from: <https://doi.org/10.1109/I4C62240.2024.10748473>
 12. Trisnawan IKN, Jati AN, Istiqomah N, Wasisto I. Detection of Gas Leaks Using the MQ-2 Gas Sensor on the Autonomous Mobile Sensor. In: 2019 International Conference on Computer, Control, Informatics and its Applications: Emerging Trends in Big Data and Artificial Intelligence, IC3INA 2019. Institute of Electrical and Electronics Engineers Inc.; 2019. p. 177–80.

13. Fischer GKJ, Bergau M, Gomez-Rosal DA, Wachaja A, Graeter J, Odenweller M, Piechottka U, Hoflinger F, Gosala N, Wetzel N, Buscher D, Valada A, Burgard W. Evaluation of a Smart Mobile Robotic System for Industrial Plant Inspection and Supervision. *IEEE Sens J* [Internet]. 2024 Jun 15 [cited 2025 Aug 20];24(12):19684–97. Available from: <https://doi.org/10.1109/JSEN.2024.3390622>
14. Lan DT, Yoon S. Anomalous indoor human trajectory detection based on the Transformer encoder and self-organizing map. *IEEE Access* [Internet]. 2023 [cited 2025 Aug 13];11:131848–65. Available from: <https://doi.org/10.1109/ACCESS.2023.3335665>
15. Sujatha E, Janani D. Real-time activity monitoring using deep learning. In: 2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT) [Internet]. Kottayam, India: IEEE; 2024 [cited 2025 Aug 13]. p. 1–6. Available from: <https://doi.org/10.1109/ICITIIT61487.2024.10580124>
16. Prathapareddy SL, Sharma N, V J, Suganthi D, Naveena SS, Kaushal RK. Implementation of video-based human anomalous activity detection using LSTM-RNN network. In: 2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS) [Internet]. Erode, India: IEEE; 2023 [cited 2025 Aug 13]. p. 853–8. Available from: <https://doi.org/10.1109/ICSSAS57918.2023.10331856>
17. Yan J, Yang Y, Naqvi SM. Object detection oriented privacy-preserving frame-level video anomaly detection. In: ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) [Internet]. Seoul, South Korea: IEEE; 2024 [cited 2025 Aug 13]. p. 7640–4. Available from: <https://doi.org/10.1109/ICASSP48485.2024.10447842>
18. Jyothi DN, Reddy GH, Prashanth B, Vardhan NV. Collaborative training of object detection and re-identification in multi-object tracking using YOLOv8 YOLOv8. In: 2024 International Conference on Computing and

- Data Science (ICCDs) [Internet]. Chennai, India: IEEE; 2024 [cited 2025 Aug 13]. p. 1–6. Available from:
<https://doi.org/10.1109/ICCDs60734.2024.10560451>
19. Fabricio F, Oliveira AS de, Teixeira MAS, Günther G. Autonomous mobile pharmacy (AMPharmacy): Identification of hospital objects and their respective positions. In: 2023 Latin American Robotics Symposium (LARS), 2023 Brazilian Symposium on Robotics (SBR), and 2023 Workshop on Robotics in Education (WRE) [Internet]. Salvador, Brazil: IEEE; 2023 [cited 2025 Aug 13]. p. 532–6. Available from:
<https://doi.org/10.1109/LARS/SBR/WRE59448.2023.10333001>
 20. Kumar D G, Varma D R, M VS, Ram G C. Real-time object detection in high-resolution videos using pan-sharpening and CNNs*. In: 2024 International Conference on Emerging Research in Computational Science (ICERCS) [Internet]. Coimbatore, India: IEEE; 2024 [cited 2025 Aug 13]. p. 1–5. Available from:
<https://doi.org/10.1109/ICERCS63125.2024.10895463>
 21. Gao Y, Lin B. Research and design of a small object detection and tracking system based on YOLOv8 and ByteTrack algorithms. In: 2024 IEEE 4th International Conference on Data Science and Computer Application (ICDSCA) [Internet]. Dalian, China: IEEE; 2024 [cited 2025 Aug 13]. p. 109–13. Available from:
<https://doi.org/10.1109/ICDSCA63855.2024.10859594>
 22. Xu Y, Hu H, Zhu X, Nan Y, Wang K, Liu Z, Lian S. RAOD: A benchmark for road abandoned object detection from video surveillance. IEEE Access [Internet]. 2024 [cited 2025 Aug 13];12:123985–94. Available from: <https://doi.org/10.1109/ACCESS.2024.3407955>
 23. Sidyakin S V., Vishnyakov B V. Real-time detection of abandoned bags using CNN. In: Beyerer J, Puente León F, editors. Automated Visual Inspection and Machine Vision II [Internet]. Munich, Germany: SPIE; 2017

- [cited 2025 Aug 13]. p. 103340J. Available from:
<https://doi.org/10.1117/12.2270078>
24. Smeureanu S, Ionescu RT. Real-time deep learning method for abandoned luggage detection in video. 2018 Jun 15; Available from:
<http://arxiv.org/abs/1803.01160>
 25. Guo C, Zhang L. A model-level fusion-based multi-modal object detection and recognition method. In: 2023 7th Asian Conference on Artificial Intelligence Technology (ACAIT) [Internet]. Jiaxing, China: IEEE; 2023 [cited 2025 Aug 13]. p. 34–8. Available from:
<https://doi.org/10.1109/ACAIT60137.2023.10528389>
 26. Xiao Y, Liu Y, Luan K, Cheng Y, Chen X, Lu H. Deep LiDAR-Radar-Visual Fusion for Object Detection in Urban Environments. Remote Sens (Basel) [Internet]. 2023 Sep 1 [cited 2025 Aug 20];15(18). Available from:
<https://www.doi.org/10.3390/rs15184433>
 27. Viswa AV, Dharma Pravardhana V, Thangavel SK, Jeyakumar G. Fall detection for elderly people using LiDAR sensor. In: 2024 3rd International Conference on Artificial Intelligence For Internet of Things (AlIoT) [Internet]. Vellore, India: IEEE; 2024 [cited 2025 Aug 13]. p. 1–6. Available from: <https://doi.org/10.1109/AlIoT58432.2024.10574642>
 28. Wang H, Wang F, Chang CW, Lai F. Real-time fall detection with ground height awareness using LiDAR and a camera of a mobile device. Biomed Signal Process Control [Internet]. 2025 Dec [cited 2025 Aug 20];110:108292. Available from:
<https://doi.org/10.1016/j.bspc.2025.108292>
 29. Madsen K, Li Z, Lauze F, Nasrollahi K. Person Fall Detection Using Weakly Supervised Methods. In: 2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW) [Internet]. Waikoloa, HI, USA: IEEE; 2024 [cited 2025 Aug 13]. p. 154–62. Available from: <https://doi.org/10.1109/WACVW60836.2024.00023>

30. M P V, Shekar M, Pragathi B SL, Ngadiran R, Ravindran S. Fall detection system for monitoring elderly people using YOLOv7-pose detection model. In: 2023 International Conference on Computer, Electronics & Electrical Engineering & their Applications (IC2E3) [Internet]. Srinagar Garhwal, India: IEEE; 2023 [cited 2025 Aug 13]. p. 1–6. Available from: <https://doi.org/10.1109/IC2E357697.2023.10262506>
31. Shabadi L, M C, L M, S G P, Deekshith RK, Narayan L. Real-time elderly fall detection as ambient intelligence in old age homes. In: 2025 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE) [Internet]. Bangalore, India: IEEE; 2025 [cited 2025 Aug 13]. p. 1–6. Available from: <https://doi.org/10.1109/IITCEE64140.2025.10915420>
32. Ultralytics. Ultralytics YOLO11 [Internet]. Ultralytics. [cited 2025 Aug 15]. Available from: <https://docs.ultralytics.com/models/yolo11/>
33. Karyna Naminas. Object detection: Key metrics for computer vision performance [Internet]. label your data. 2025 [cited 2025 Aug 5]. Available from: <https://labelyourdata.com/articles/object-detection-metrics>

Appendices

Appendix A Ethical approval letter



8 May 2025

Dear Ms Robin ,

Reference: CURES/25022/2025

Project ID: 28455

Title: Indoor Anomaly Detection in Smart City Environments

Thank you for your application to the Cranfield University Research Ethics System (CURES).

We are pleased to inform you your CURES application, reference CURES/25022/2025 has been reviewed. You may now proceed with the research activities you have sought approval for.

If you have any queries, please contact CURES Support.

We wish you every success with your project.

Regards,

CURES Team

Appendix B SYSTEM CODES AND DIAGRAMS

B.1 YOLOv11-XL + Submask_ConvNeXt + SAM_ViT

```
from ultralytics import YOLO
import torch
import torch.nn as nn
import torch.nn.functional as F
from PIL import Image, ImageDraw, ImageFont
import numpy as np
import cv2
from torchvision import transforms
from segment_anything import sam_model_registry, SamPredictor
import timm
from torchvision.models import convnext_tiny
from collections import Counter
import os
import glob
from sklearn.metrics import precision_score, recall_score, f1_score
import shutil

# === Create output folder and subfolders ===
output_folder = r"C:\Users\natas\Desktop\IRP\IRP Indoor Anomaly
Detection.v3i.yolov11\yolo_submaskconvnext_samvit_height110_distance240_1
"

tp_folder = os.path.join(output_folder, "TP")
fp_folder = os.path.join(output_folder, "FP")
fn_folder = os.path.join(output_folder, "FN")
if os.path.exists(output_folder):
    shutil.rmtree(output_folder)
os.makedirs(tp_folder, exist_ok=True)
os.makedirs(os.path.join(output_folder, "FP"), exist_ok=True)
os.makedirs(os.path.join(output_folder, "FN"), exist_ok=True)

# === Device ===
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
if device.type == "cuda":
    print("GPU:", torch.cuda.get_device_name(0))

# === Class names ===
# YOLO trained on 7 classes (without background)
class_names = ['butterknife', 'cup', 'fork', 'glass', 'plate', 'spoon',
'tray']
```

```

# === Load models ===
yolo_model = YOLO("runs1/detect/train3/weights/best.pt")
sam = sam_model_registry["vit_b"](checkpoint="sam_vit_b.pth").to(device)
predictor = SamPredictor(sam)
eff_model = timm.create_model("convnext_tiny", pretrained=False,
num_classes=len(class_names))
#eff_model = convnext_tiny(weights=None)
#eff_model.classifier[2] = nn.Linear(eff_model.classifier[2].in_features,
len(class_names))
#eff_model.load_state_dict(torch.load("convnext_tiny_v4_20250624_173816.p
th", map_location=device))
#eff_model.load_state_dict(torch.load("convnext_tiny_20250730_142412.pth"
, map_location=device))
eff_model.load_state_dict(torch.load("convnext_tiny_20250730_181851.pth",
map_location=device))

eff_model.eval().to(device)

vit_model =
timm.create_model("vit_base_patch16_224.augreg2_in21k_ft_in1k",
pretrained=False, num_classes=len(class_names))
#vit_model.load_state_dict(torch.load("vit_b16_v3_vit_20250624_172007.pth
", map_location=device))
#vit_model.load_state_dict(torch.load("vit_b16_with_curves_1_20250730_121
503.pth", map_location=device))
vit_model.load_state_dict(torch.load("vit_b16_sam_cropped_20250730_183634
.pth", map_location=device))

vit_model.eval().to(device)

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

def compute_iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])
    xB = min(boxA[2], boxB[2])
    yB = min(boxA[3], boxB[3])
    interArea = max(0, xB - xA) * max(0, yB - yA)
    boxAArea = max(0, boxA[2] - boxA[0]) * max(0, boxA[3] - boxA[1])

```

```

boxBArea = max(0, boxB[2] - boxB[0]) * max(0, boxB[3] - boxB[1])
if boxAArea + boxBArea - interArea == 0:
    return 0
iou = interArea / (boxAArea + boxBArea - interArea)
return iou

def split_mask(mask):
    num_labels, labels = cv2.connectedComponents(mask.astype(np.uint8))
    masks = []
    for i in range(1, num_labels):
        component = (labels == i).astype(np.uint8)
        if np.sum(component) > 50:
            masks.append(component)
    return masks

def fusion_decision(yolo_label, yolo_conf, conv_label, conv_conf,
vit_label, vit_conf):
    # Normalize labels to lowercase
    yolo_label = yolo_label.lower()
    conv_label = conv_label.lower()
    vit_label = vit_label.lower()

    # Step 1: Check if all three models agree
    if yolo_label == conv_label == vit_label:
        highest_conf = max(yolo_conf, conv_conf, vit_conf)
        return yolo_label, highest_conf

    # Step 2: Check for any two-model agreement
    if yolo_label == conv_label:
        highest_conf = max(yolo_conf, conv_conf)
        return yolo_label, highest_conf
    if yolo_label == vit_label:
        highest_conf = max(yolo_conf, vit_conf)
        return yolo_label, highest_conf
    if conv_label == vit_label:
        highest_conf = max(conv_conf, vit_conf)
        return conv_label, highest_conf

    # Step 3: If all disagree, apply weighted scoring
    model_weights = {
        'yolo': 3,
        'convnext': 2,
        'vit': 1
    }

    scores = {
        'yolo': yolo_conf * model_weights['yolo'],

```

```

        'convnext': conv_conf * model_weights['convnext'],
        'vit': vit_conf * model_weights['vit']
    }

    # Select model with highest weighted score
    best_model = max(scores, key=scores.get)

    if best_model == 'yolo':
        return yolo_label, yolo_conf
    elif best_model == 'convnext':
        return conv_label, conv_conf
    else:
        return vit_label, vit_conf

def run_full_detection(image_path):
    original_image = Image.open(image_path).convert("RGB")
    np_image = np.array(original_image)
    predictor.set_image(np_image)
    results = yolo_model.predict(source=image_path, save=False, conf=0.1)
    yolo_boxes = [tuple(map(int, box.xyxy[0])) for result in results for
box in result.bboxes]
    print(f"YOLO boxes: {len(yolo_boxes)}")

    detections = []

    for result in results:
        for box in result.bboxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            yolo_conf = float(box.conf[0]) * 100
            yolo_label_id = int(box.cls[0])
            yolo_label = yolo_model.names[yolo_label_id]

            if yolo_conf < 20:
                continue

            input_box = np.array([x1, y1, x2, y2])
            masks, scores, _ = predictor.predict(box=input_box,
multimask_output=True)
            full_mask = masks[0].astype(bool)

            # --- ViT receives full mask ---
            vit_masked_np = np_image.copy()
            vit_masked_np[~full_mask] = 0
            vit_crop = vit_masked_np[y1:y2, x1:x2]
            vit_input_tensor =
transform(Image.fromarray(vit_crop)).unsqueeze(0).to(device)

```

```

with torch.no_grad():
    vit_out = vit_model(vit_input_tensor)
    vit_probs = F.softmax(vit_out, dim=1)
    vit_conf, vit_pred = torch.max(vit_probs, 1)
    vit_label = class_names[vit_pred.item()]
    vit_conf = vit_conf.item() * 100

# --- ConvNeXt receives submasks ---
if yolo_label.lower() == 'tray':
    submasks = [full_mask.astype(np.uint8)]
else:
    submasks = split_mask(full_mask)

best_conv_label = 'background'
best_conv_conf = 0

for submask in submasks:
    conv_masked_np = np_image.copy()
    conv_masked_np[~submask.astype(bool)] = 0
    conv_crop = conv_masked_np[y1:y2, x1:x2]
    conv_input_tensor =
transform(Image.fromarray(conv_crop)).unsqueeze(0).to(device)

    with torch.no_grad():
        conv_out = eff_model(conv_input_tensor)
        conv_probs = F.softmax(conv_out, dim=1)
        conv_conf, conv_pred = torch.max(conv_probs, 1)
        conv_label = class_names[conv_pred.item()]
        conv_conf = conv_conf.item() * 100

    if conv_conf > best_conv_conf:
        best_conv_label = conv_label
        best_conv_conf = conv_conf

    final_label, final_conf = fusion_decision(
        yolo_label, yolo_conf, best_conv_label, best_conv_conf,
vit_label, vit_conf
    )

    detections.append({
        "box": (x1, y1, x2, y2),
        "confidences": {
            f"YOLO: {yolo_label}": yolo_conf,
            f"ConvNeXt: {best_conv_label}": best_conv_conf,

```

```

        f"ViT:      {vit_label}": vit_conf
    },
    "best_conf": final_conf,
    "display_text": f"{final_label} ({final_conf:.1f}%)",
    "label": final_label,
    "conf": final_conf
})

# IoU filtering
final_detections = []
used = [False] * len(detections)
for i in range(len(detections)):
    if used[i]:
        continue
    keep = detections[i]
    for j in range(i + 1, len(detections)):
        if used[j]:
            continue
        iou = compute_iou(keep["box"], detections[j]["box"])
        if iou >= 0.9:
            if detections[j]["best_conf"] > keep["best_conf"]:
                keep = detections[j]
            used[j] = True
    final_detections.append(keep)
    used[i] = True

# Debugging: Verify final boxes
final_boxes = [det["box"] for det in final_detections]
print(f"Final boxes: {len(final_boxes)}")

return final_detections, original_image

def evaluate_ensemble():
    image_dir =
r"C:\Users\natas\Desktop\IRP\gloomy.v1i.yolov11\train\images"
    image_paths = [os.path.join(image_dir, f) for f in
os.listdir(image_dir) if f.endswith('.jpg')]

    TP = 0
    FP = 0
    FN = 0

    for image_path in image_paths:

```

```

# Load image to get size
img = Image.open(image_path)
img_w, img_h = img.size

# Load ground truth
label_path = image_path.replace("images",
"labels").replace(".jpg", ".txt")
with open(label_path, "r") as f:
    lines = f.readlines()
gt_boxes = []
for line in lines:
    parts = line.strip().split()
    class_id = int(parts[0])
    x_center, y_center, w, h = map(float, parts[1:])
    x1 = int((x_center - w/2) * img_w)
    y1 = int((y_center - h/2) * img_h)
    x2 = int((x_center + w/2) * img_w)
    y2 = int((y_center + h/2) * img_h)
    label = yolo_model.names[class_id]
    gt_boxes.append({"box": (x1, y1, x2, y2), "label": label})

# Run detection
detections, original_image = run_full_detection(image_path)

# Filter detections with conf >= 10 and exclude background
detections = [det for det in detections if det["conf"] >= 10 and
det["label"].lower() != 'background']

# Sort by confidence descending
detections.sort(key=lambda x: x["conf"], reverse=True)

matched_gt = [False] * len(gt_boxes)
tp_detections = []
fp_detections = []

for det in detections:
    det_box = det["box"]
    det_label = det["label"]
    best_iou = 0
    best_gt_idx = -1
    for gt_idx, gt in enumerate(gt_boxes):
        if gt["label"] == det_label and not matched_gt[gt_idx]:
            iou = compute_iou(det_box, gt["box"])
            if iou > best_iou:
                best_iou = iou
                best_gt_idx = gt_idx
    if best_iou >= 0.5 and best_gt_idx != -1:

```

```

        TP += 1
        matched_gt[best_gt_idx] = True
        tp_detections.append(det)
    else:
        FP += 1
        fp_detections.append(det)

    # Unmatched ground truth boxes are false negatives
    fn_boxes = [gt for gt_idx, gt in enumerate(gt_boxes) if not
matched_gt[gt_idx]]
    FN += len(fn_boxes)

    # Draw boxes on the image
    draw = ImageDraw.Draw(original_image)
    font = ImageFont.load_default()

    # Draw TP boxes (green)
    for det in tp_detections:
        x1, y1, x2, y2 = det["box"]
        draw.rectangle([x1, y1, x2, y2], outline="green", width=2)
        draw.text((x1, max(y1 - 20, 0)), det["display_text"],
fill="black", font=font)

    # Draw FP boxes (red)
    for det in fp_detections:
        x1, y1, x2, y2 = det["box"]
        draw.rectangle([x1, y1, x2, y2], outline="red", width=2)
        draw.text((x1, max(y1 - 20, 0)), det["display_text"],
fill="black", font=font)

    # Draw FN boxes (blue)
    for gt in fn_boxes:
        x1, y1, x2, y2 = gt["box"]
        draw.rectangle([x1, y1, x2, y2], outline="blue", width=2)
        draw.text((x1, max(y1 - 20, 0)), f"{gt['label']} (FN)",
fill="black", font=font)

    # Save image to appropriate folder based on content
    base_name = os.path.basename(image_path)
    if tp_detections:
        output_path = os.path.join(tp_folder, base_name)
    elif fp_detections:
        output_path = os.path.join(fp_folder, base_name)
    else:
        output_path = os.path.join(fn_folder, base_name)
    original_image.save(output_path)

```

```

# Print metrics
print(f"True Positives (TP): {TP}")
print(f"False Positives (FP): {FP}")
print(f"False Negatives (FN): {FN}")
print(f"Precision: {TP / (TP + FP):.4f}" if TP + FP > 0 else
"Precision: 0.0000")
print(f"Recall: {TP / (TP + FN):.4f}" if TP + FN > 0 else "Recall:
0.0000")
print(f"F1 Score: {2 * (TP / (TP + FP)) * (TP / (TP + FN)) / ((TP /
(TP + FP)) + (TP / (TP + FN))):.4f}" if (TP + FP) > 0 and (TP + FN) > 0
else "F1 Score: 0.0000")

# Run the evaluation
evaluate_ensemble()

```

B.2 Code for robot integration

```

from ultralytics import YOLO
import torch
import torch.nn as nn
import torch.nn.functional as F
from PIL import Image, ImageDraw, ImageFont
import numpy as np
import cv2
from torchvision import transforms
from segment_anything import sam_model_registry, SamPredictor
import timm
from torchvision.models import convnext_tiny
from collections import Counter
import os
import glob
from sklearn.metrics import precision_score, recall_score, f1_score
import shutil
import requests

import asyncio
import websockets
import json
import ssl

# === Create output folder and subfolders ===

# === Device ===
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print("Using device:", device)
if device.type == "cuda":
    print("GPU:", torch.cuda.get_device_name(0))

```

```

# === Class names ===
# YOLO trained on 7 classes (without background)
class_names = ['butterknife', 'cup', 'fork', 'glass', 'plate', 'spoon',
'tray']

def get_next_run_id(base_dir):
    os.makedirs(base_dir, exist_ok=True)
    existing = [int(f.name.replace("original", "")) for f in
os.scandir(base_dir) if f.name.startswith("original") and
f.name.replace("original", "").isdigit()]
    next_id = max(existing) + 1 if existing else 1
    return next_id

run_id = get_next_run_id("original_frames")
original_save_dir = f"original_frames/original{run_id}"
detection_save_dir = f"detection_frames/detection{run_id}"
os.makedirs(original_save_dir, exist_ok=True)
os.makedirs(detection_save_dir, exist_ok=True)

# === Load models ===
yolo_model = YOLO("runs1/detect/train3/weights/best.pt")
sam = sam_model_registry["vit_b"](checkpoint="sam_vit_b.pth").to(device)
predictor = SamPredictor(sam)

eff_model = convnext_tiny(weights=None)
eff_model.classifier[2] = nn.Linear(eff_model.classifier[2].in_features,
len(class_names))
eff_model.load_state_dict(torch.load("convnext_tiny_v4_20250624_173816.pt
h", map_location=device))
eff_model.eval().to(device)

vit_model =
timm.create_model("vit_base_patch16_224.augreg2_in21k_ft_in1k",
pretrained=False, num_classes=len(class_names))
vit_model.load_state_dict(torch.load("vit_b16_v3_vit_20250624_172007.pth"
, map_location=device))
vit_model.eval().to(device)

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

def compute_iou(boxA, boxB):
    xA = max(boxA[0], boxB[0])
    yA = max(boxA[1], boxB[1])

```

```

xB = min(boxA[2], boxB[2])
yB = min(boxA[3], boxB[3])
interArea = max(0, xB - xA) * max(0, yB - yA)
boxAArea = max(0, boxA[2] - boxA[0]) * max(0, boxA[3] - boxA[1])
boxBArea = max(0, boxB[2] - boxB[0]) * max(0, boxB[3] - boxB[1])
if boxAArea + boxBArea - interArea == 0:
    return 0
iou = interArea / (boxAArea + boxBArea - interArea)
return iou

def split_mask(mask):
    num_labels, labels = cv2.connectedComponents(mask.astype(np.uint8))
    masks = []
    for i in range(1, num_labels):
        component = (labels == i).astype(np.uint8)
        if np.sum(component) > 50:
            masks.append(component)
    return masks

def fusion_decision(yolo_label, yolo_conf, conv_label, conv_conf,
vit_label, vit_conf):
    # Normalize labels to lowercase
    yolo_label = yolo_label.lower()
    conv_label = conv_label.lower()
    vit_label = vit_label.lower()

    # Step 1: Check if all three models agree
    if yolo_label == conv_label == vit_label:
        highest_conf = max(yolo_conf, conv_conf, vit_conf)
        return yolo_label, highest_conf

    # Step 2: Check for any two-model agreement
    if yolo_label == conv_label:
        highest_conf = max(yolo_conf, conv_conf)
        return yolo_label, highest_conf
    if yolo_label == vit_label:
        highest_conf = max(yolo_conf, vit_conf)
        return yolo_label, highest_conf
    if conv_label == vit_label:
        highest_conf = max(conv_conf, vit_conf)
        return conv_label, highest_conf

    # Step 3: If all disagree, apply weighted scoring
    model_weights = {
        'yolo': 3,
        'convnext': 2,
        'vit': 1

```

```

}

scores = {
    'yolo': yolo_conf * model_weights['yolo'],
    'convnext': conv_conf * model_weights['convnext'],
    'vit': vit_conf * model_weights['vit']
}

# Select model with highest weighted score
best_model = max(scores, key=scores.get)

if best_model == 'yolo':
    return yolo_label, yolo_conf
elif best_model == 'convnext':
    return conv_label, conv_conf
else:
    return vit_label, vit_conf

def run_full_detection_frame(np_image):
    original_image = Image.fromarray(np_image).convert("RGB")
    predictor.set_image(np_image)
    results = yolo_model.predict(source=np_image, save=False, conf=0.1)
    yolo_boxes = [tuple(map(int, box.xyxy[0])) for result in results for
box in result.bboxes]

    detections = []

    for result in results:
        for box in result.bboxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            yolo_conf = float(box.conf[0]) * 100
            yolo_label_id = int(box.cls[0])
            yolo_label = yolo_model.names[yolo_label_id]

            if yolo_conf < 60:
                continue

            input_box = np.array([x1, y1, x2, y2])
            masks, scores, _ = predictor.predict(box=input_box,
multimask_output=True)
            full_mask = masks[0].astype(bool)

            vit_masked_np = np_image.copy()
            vit_masked_np[~full_mask] = 0

```

```

        vit_crop = vit_masked_np[y1:y2, x1:x2]
        vit_input_tensor =
transform(Image.fromarray(vit_crop)).unsqueeze(0).to(device)

        with torch.no_grad():
            vit_out = vit_model(vit_input_tensor)
            vit_probs = F.softmax(vit_out, dim=1)
            vit_conf, vit_pred = torch.max(vit_probs, 1)
            vit_label = class_names[vit_pred.item()]
            vit_conf = vit_conf.item() * 100

        if yolo_label.lower() == 'tray':
            submasks = [full_mask.astype(np.uint8)]
        else:
            submasks = split_mask(full_mask)

        best_conv_label = 'background'
        best_conv_conf = 0

        for submask in submasks:
            conv_masked_np = np_image.copy()
            conv_masked_np[~submask.astype(bool)] = 0
            conv_crop = conv_masked_np[y1:y2, x1:x2]
            conv_input_tensor =
transform(Image.fromarray(conv_crop)).unsqueeze(0).to(device)

            with torch.no_grad():
                conv_out = eff_model(conv_input_tensor)
                conv_probs = F.softmax(conv_out, dim=1)
                conv_conf, conv_pred = torch.max(conv_probs, 1)
                conv_label = class_names[conv_pred.item()]
                conv_conf = conv_conf.item() * 100

            if conv_conf > best_conv_conf:
                best_conv_label = conv_label
                best_conv_conf = conv_conf

        final_label, final_conf = fusion_decision(
            yolo_label, yolo_conf, best_conv_label, best_conv_conf,
vit_label, vit_conf
        )
        if final_conf < 75:
            continue

        detections.append({
            "box": (x1, y1, x2, y2),
            "display_text": f"{final_label} ({final_conf:.1f}%)",

```

```

        "label": final_label,
        "conf": final_conf
    })

# IoU filtering
final_detections = []
used = [False] * len(detections)
for i in range(len(detections)):
    if used[i]:
        continue
    keep = detections[i]
    for j in range(i + 1, len(detections)):
        if used[j]:
            continue
        iou = compute_iou(keep["box"], detections[j]["box"])
        if iou >= 0.9:
            if detections[j]["conf"] > keep["conf"]:
                keep = detections[j]
            used[j] = True
    final_detections.append(keep)
    used[i] = True

return final_detections, original_image

# WebSocket connection config
WEBSOCKET_URI = "wss://localhost:8443/ws" # Or use
"ws://localhost:8443/ws" if not using SSL

# Classes to trigger detection
trigger_classes = ['butterknife', 'cup', 'fork', 'glass', 'plate',
'spoon', 'tray']
class_counter = {cls: 0 for cls in trigger_classes}
DETECTION_THRESHOLD = 5

async def send_stop_signal():
    ssl_context = ssl.create_default_context()
    ssl_context.check_hostname = False
    ssl_context.verify_mode = ssl.CERT_NONE # For dev only

    try:
        async with websockets.connect(WEBSOCKET_URI, ssl=ssl_context) as
websocket:
            await websocket.send("ping") # Optional keep-alive
            msg = {
                "name": "cancel_goal",

```

```

        "args": ""
    }
    compact_msg = json.dumps(msg, separators=(',', ':')) # compact JSON, no spaces
    await websocket.send(compact_msg)
    print("[WS] Sent stop signal:", msg)
except Exception as e:
    print("[WS ERROR]", e)

# Add this to the very bottom of your script to run live webcam detection
def live_camera_demo():
    cap = cv2.VideoCapture(0)
    # Video writer setup
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    fps = 2

    #fps = int(cap.get(cv2.CAP_PROP_FPS) or 10) # fallback to 20 if FPS
not detected
    width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    original_video_path = f"original_frames/original{run_id}.avi"
    detection_video_path = f"detection_frames/detection{run_id}.avi"
    original_writer = cv2.VideoWriter(original_video_path, fourcc, fps,
(width, height))
    detection_writer = cv2.VideoWriter(detection_video_path, fourcc, fps,
(width, height))

    if not cap.isOpened():
        print("Error: Cannot open webcam.")
        return

    print("Press 'q' to quit. Press 'c' to continue after pause.")

    paused = False
    frame_index = 0

    while True:
        if not paused:
            ret, frame = cap.read()

            if not ret:
                break

            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            detections, pil_image = run_full_detection_frame(frame_rgb)

```

```

        # Track most common class in this frame
        labels_in_frame = [det["label"].lower() for det in detections
if det["label"].lower() in trigger_classes]

        # Update counts
        for cls in trigger_classes:
            if cls in labels_in_frame:
                class_counter[cls] += 1
            else:
                class_counter[cls] = 0

        # Check for threshold
        for cls, count in class_counter.items():
            if count >= DETECTION_THRESHOLD:
                print(f"[ALERT] Detected {cls} in {count} consecutive
frames.")

                # Send POST to FastAPI server
                try:
                    headers = {"Content-Type": "application/json"}
                    payload = {"name": "cancel_goal", "args": ""}
                    response =
requests.post("https://localhost:8443/", json=payload, headers={"Content-
Type": "application/json"}, verify=False)

                    print("[POST] Server responded with:",
response.status_code)
                except Exception as e:
                    print("[POST ERROR]", e)
                    # Also send via WebSocket
                    asyncio.run(send_stop_signal())
                    # Pause video and wait for manual resume
                    paused = True
                    print(f"[PAUSE] Paused after detecting {cls} 5 times.
Press 'c' to continue.")
                    # Reset the counter so it doesn't re-trigger
instantly on resum
                    class_counter[cls] = 0
                    break

        # Annotate and display
        draw = ImageDraw.Draw(pil_image)
        font = ImageFont.load_default()
        for det in detections:
            x1, y1, x2, y2 = det["box"]
            draw.rectangle([x1, y1, x2, y2], outline="green",
width=2)

```

```

        draw.text((x1, max(y1 - 20, 0)), det["display_text"],
fill="black", font=font)

        annotated_frame = cv2.cvtColor(np.array(pil_image),
cv2.COLOR_RGB2BGR)
        # Save original frame
        original_path = os.path.join(original_save_dir,
f"frame_{frame_index:04d}.jpg")
        cv2.imwrite(original_path, frame)
        # Save annotated detection frame
        detection_path = os.path.join(detection_save_dir,
f"frame_{frame_index:04d}.jpg")
        cv2.imwrite(detection_path, annotated_frame)
        frame_index += 1
        if annotated_frame.shape[1] != width or
annotated_frame.shape[0] != height:
            annotated_frame = cv2.resize(annotated_frame, (width,
height))

        original_writer.write(frame)
        detection_writer.write(annotated_frame)

        cv2.imshow("Live Detection", annotated_frame)

    else:
        key = cv2.waitKey(0)
        if key == ord('c'):
            print("[RESUME] Continuing video...")
            paused = False
        elif key == ord('q'):
            break
        continue # skip the rest if paused

    # Handle quit normally
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    cap.release()
    original_writer.release()
    detection_writer.release()
    cv2.destroyAllWindows()
if __name__ == "__main__":
    live_camera_demo()

```

B.3 Test-wise confusion matrix for different tilts

B.3.1 Tilt 20°

Test 1

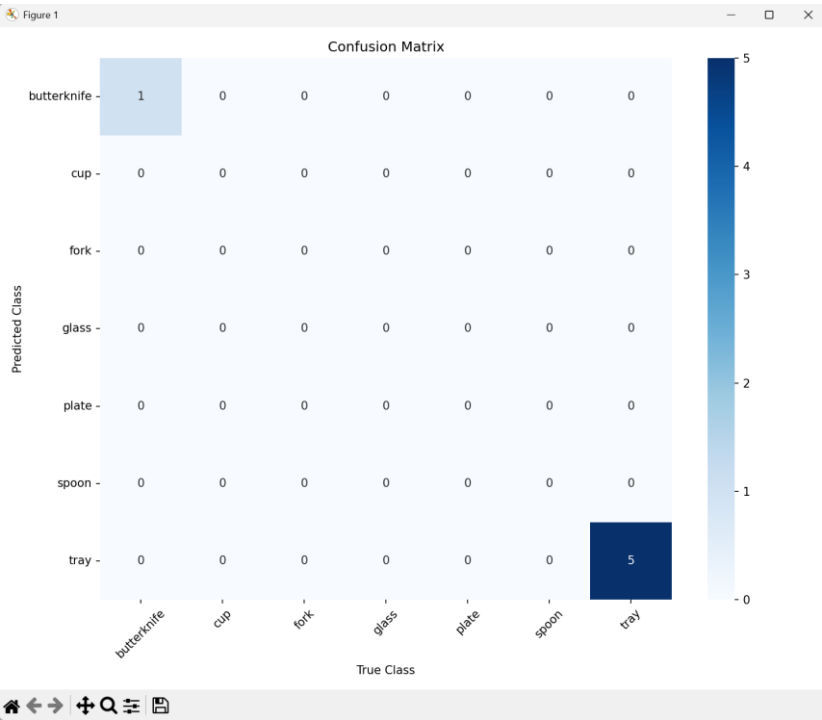


Figure B-1 Confusion matrix tilt 20° , Test 1

Test 2

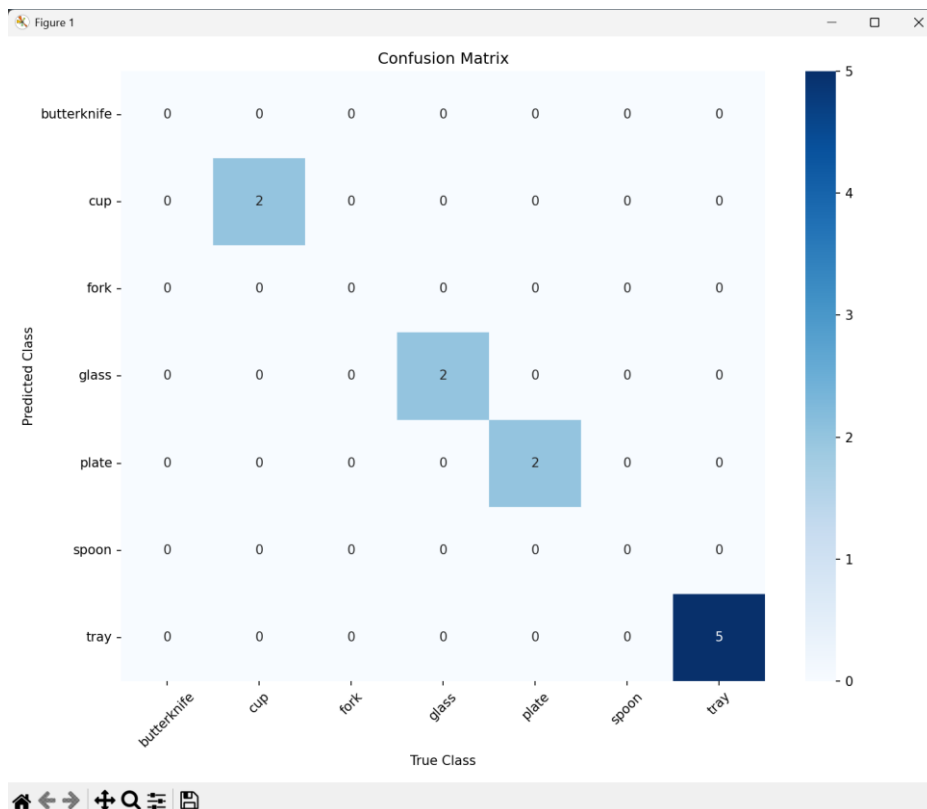


Figure B-2 Confusion matrix tilt 20° , Test 2

Test 3

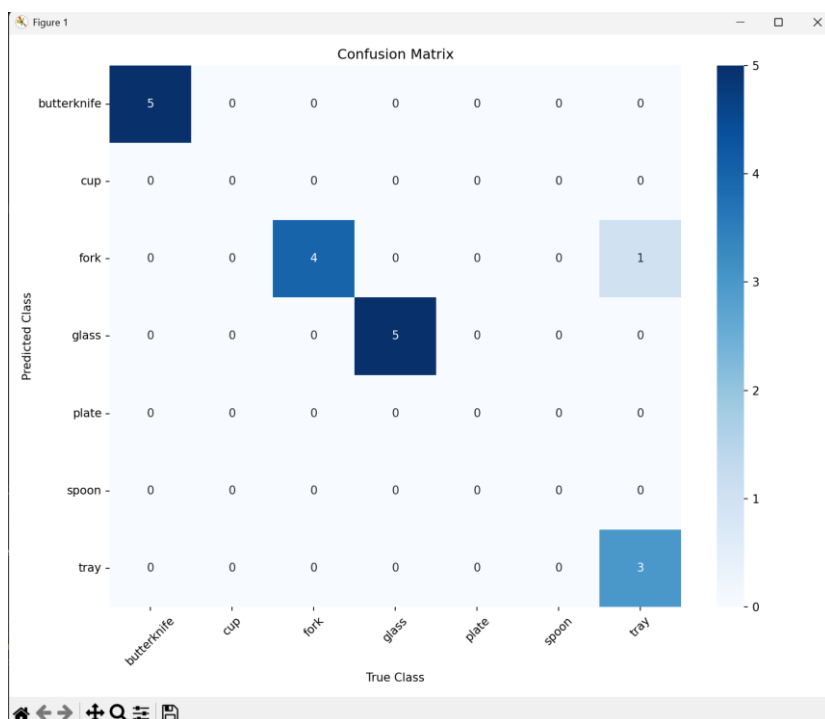


Figure B-3 Confusion matrix tilt 20° , Test 3

Test 4

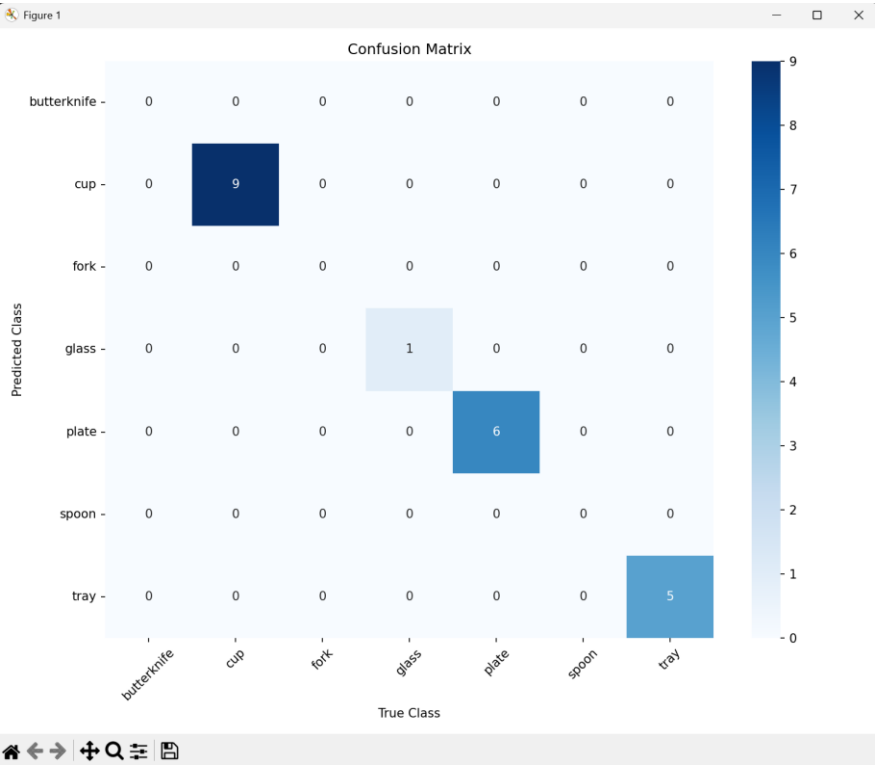


Figure B-4 Confusion matrix tilt 20° , Test 4

Test 5

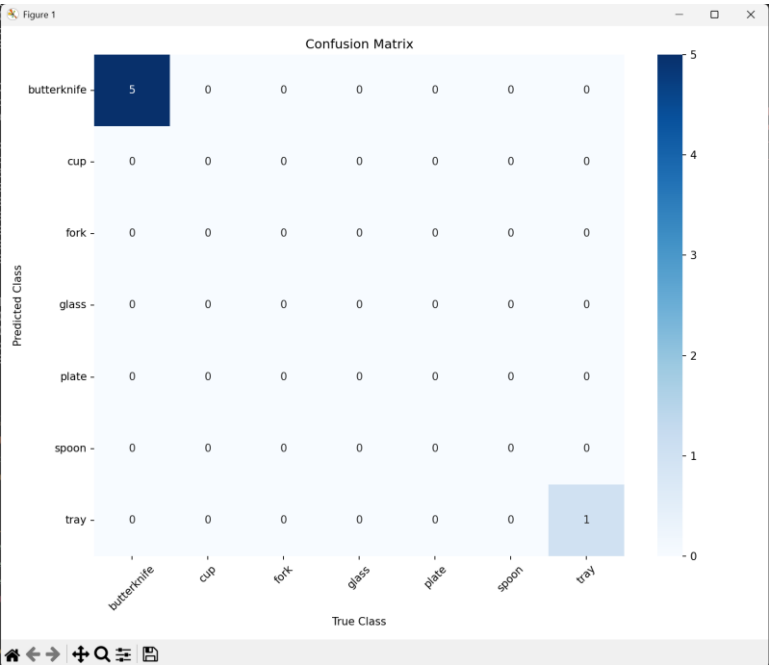


Figure B-5 Confusion matrix tilt 20° , Test 5

B.3.2 Tilt 30°

Test 1

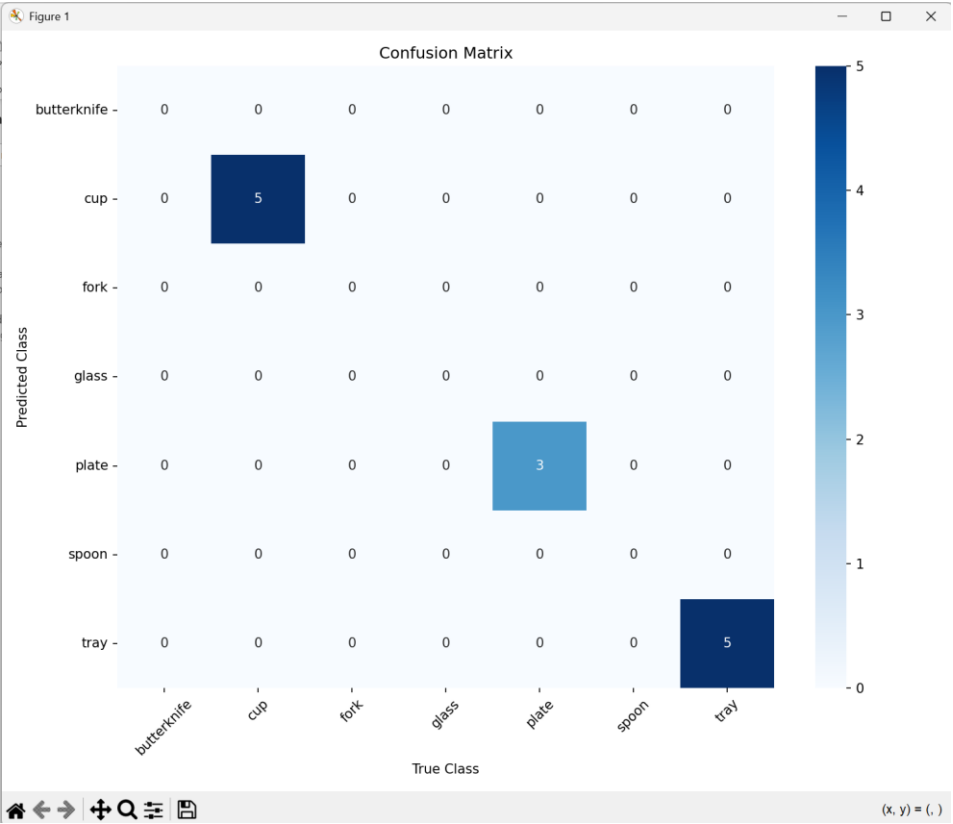


Figure B-6 Confusion matrix tilt 30° , Test 1

Test 2

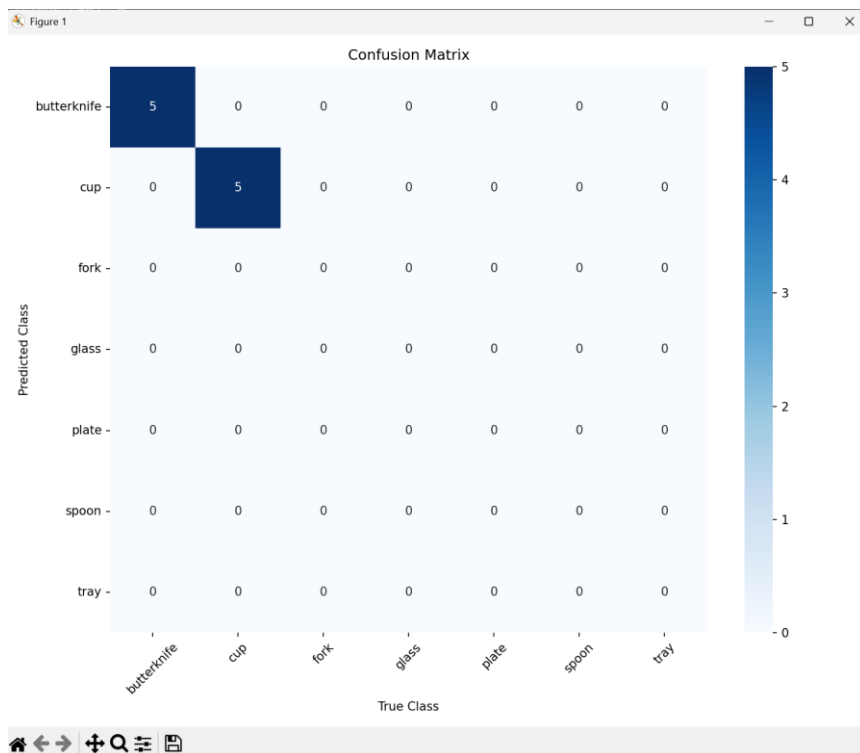


Figure B-7 Confusion matrix tilt 30° , Test 2

Test 3

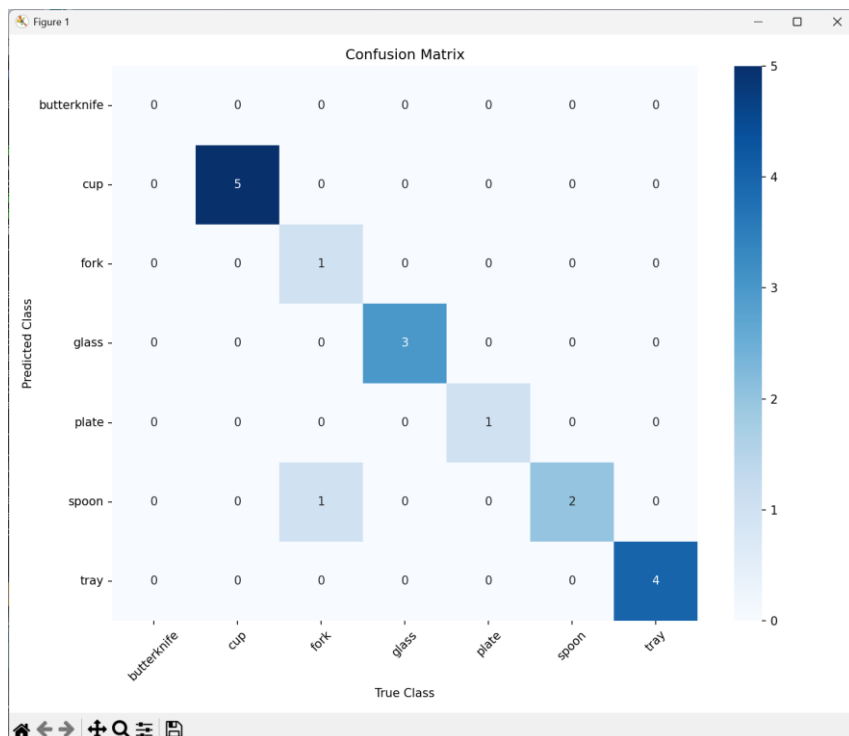


Figure B-8 Confusion matrix tilt 30° , Test 3

Test 4

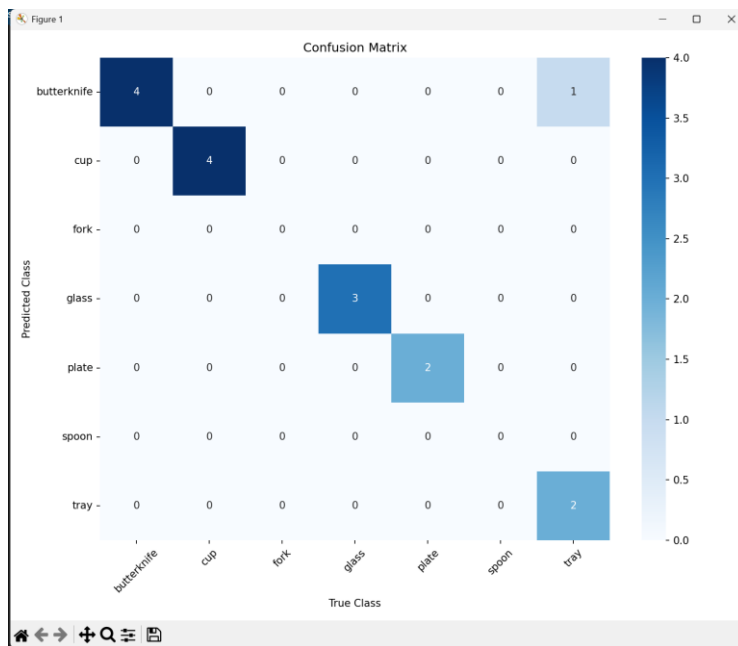


Figure B-9 Confusion matrix tilt 30° , Test 4

Test 5

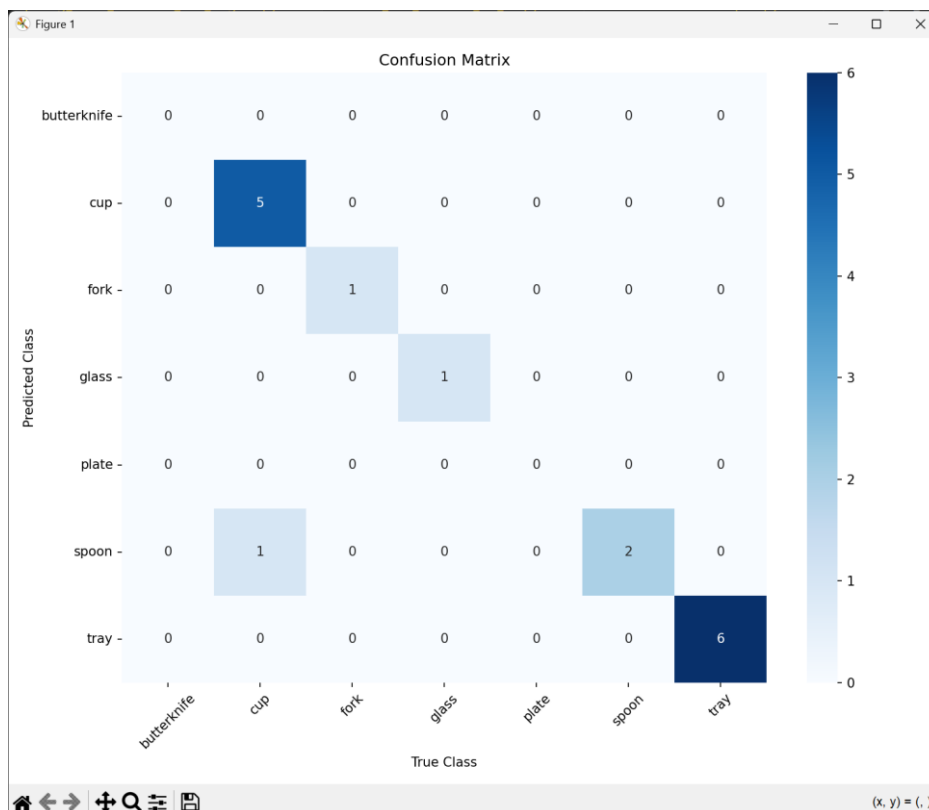


Figure B-10 Confusion matrix tilt 30° , Test 5

B.3.3 Tilt 40°

Test 1

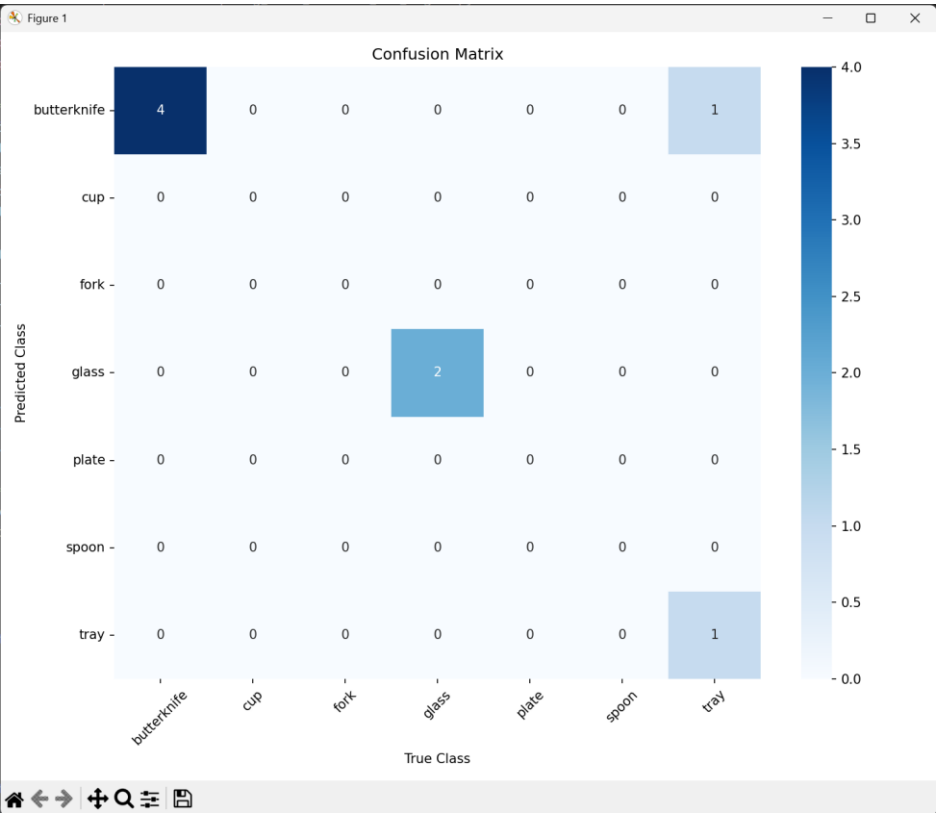


Figure B-11 Confusion matrix tilt 40° , Test 1

Test 2

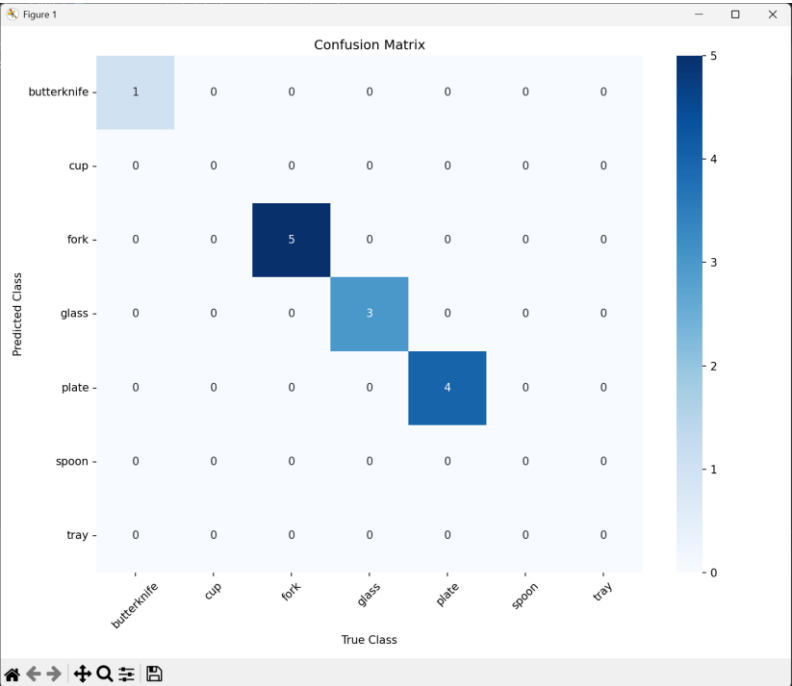


Figure B-12 Confusion matrix tilt 40° , Test 2

Test 3

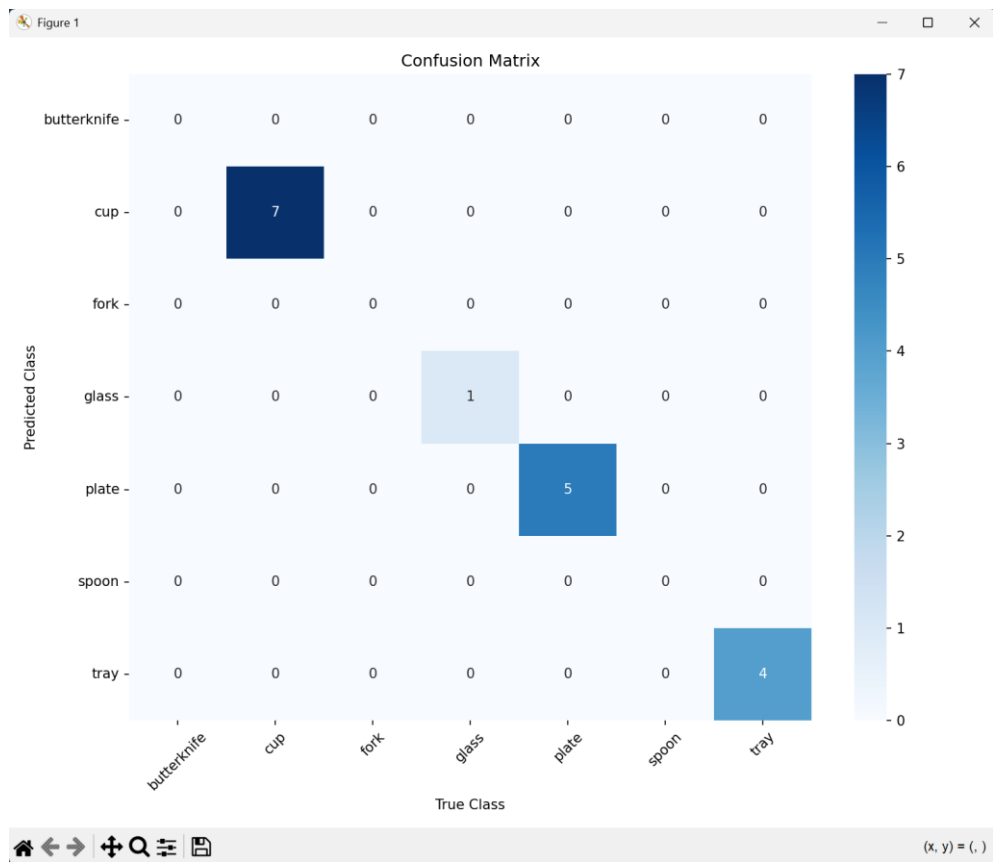


Figure B-13 Confusion matrix tilt 40° , Test 3

Test 4

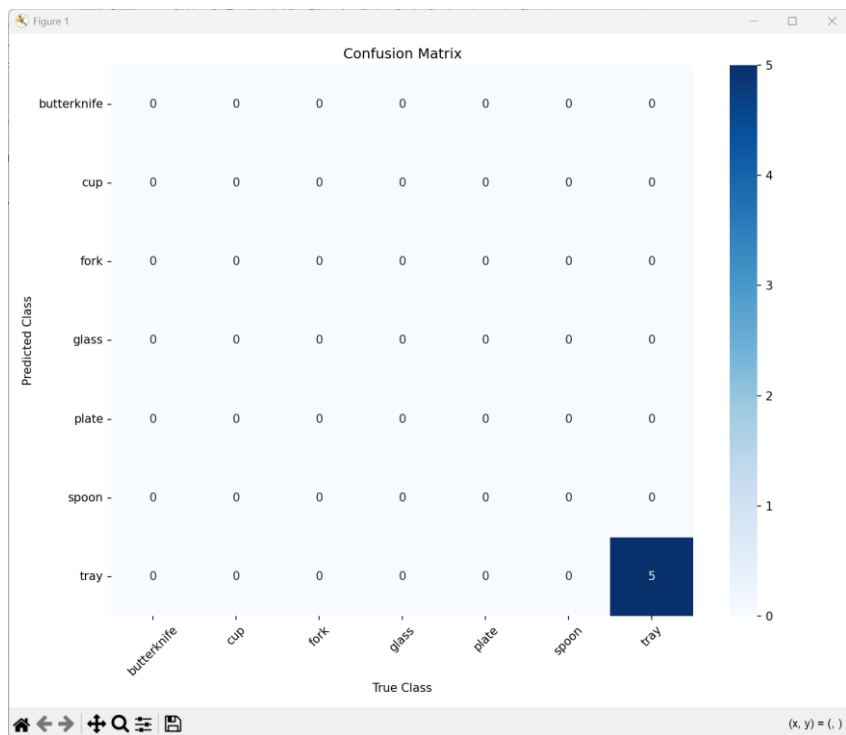


Figure B-14 Confusion matrix tilt 40° , Test 4

Test 5

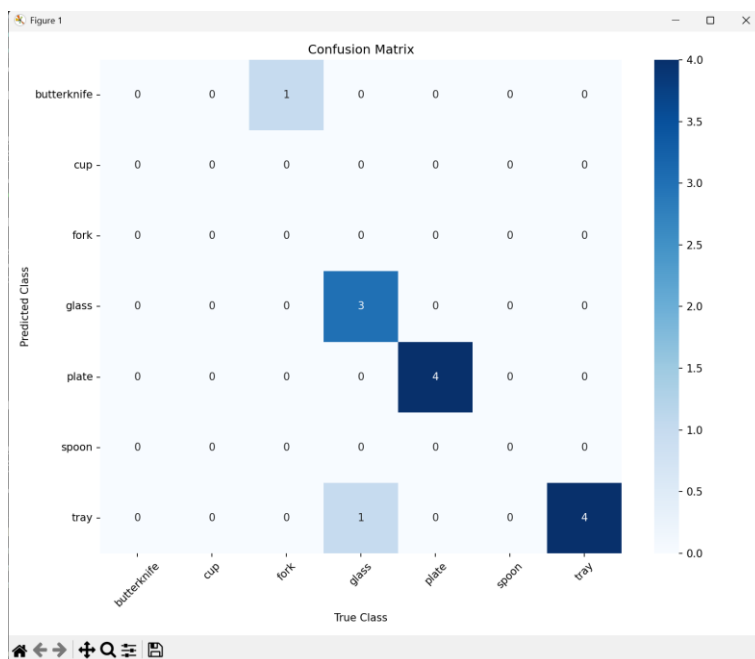


Figure B-15 Confusion matrix tilt 40° , Test 5

B.3.4 Tilt 50°

Test 1

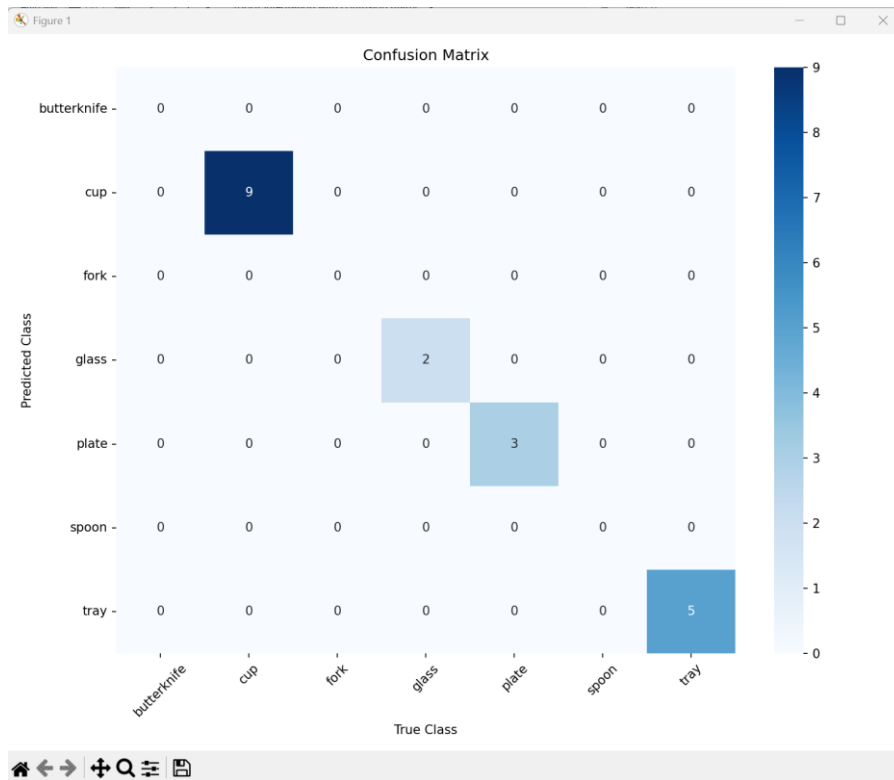


Figure B-16 Confusion matrix tilt 50° , Test 1

Test 2

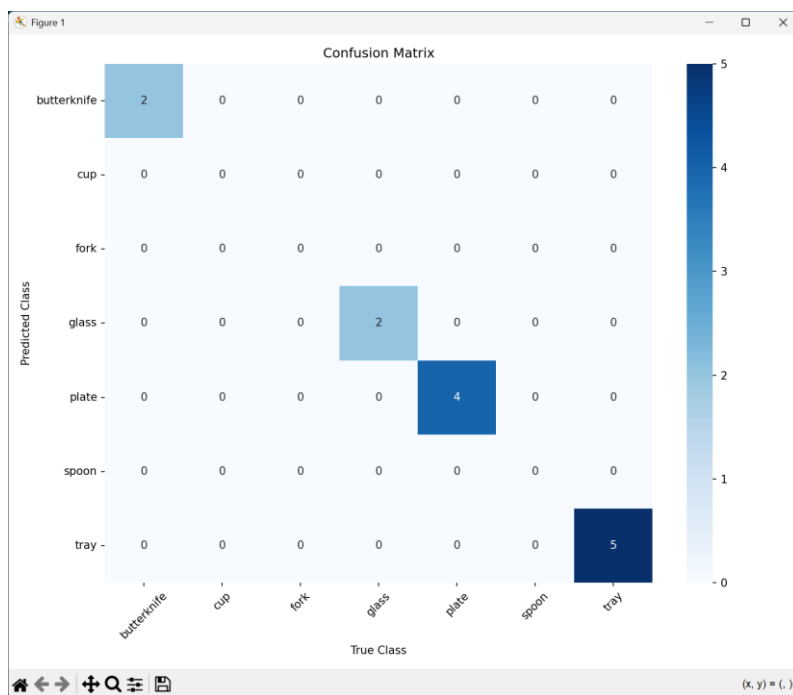


Figure B-17 Confusion matrix tilt 50° , Test 2

Test 3

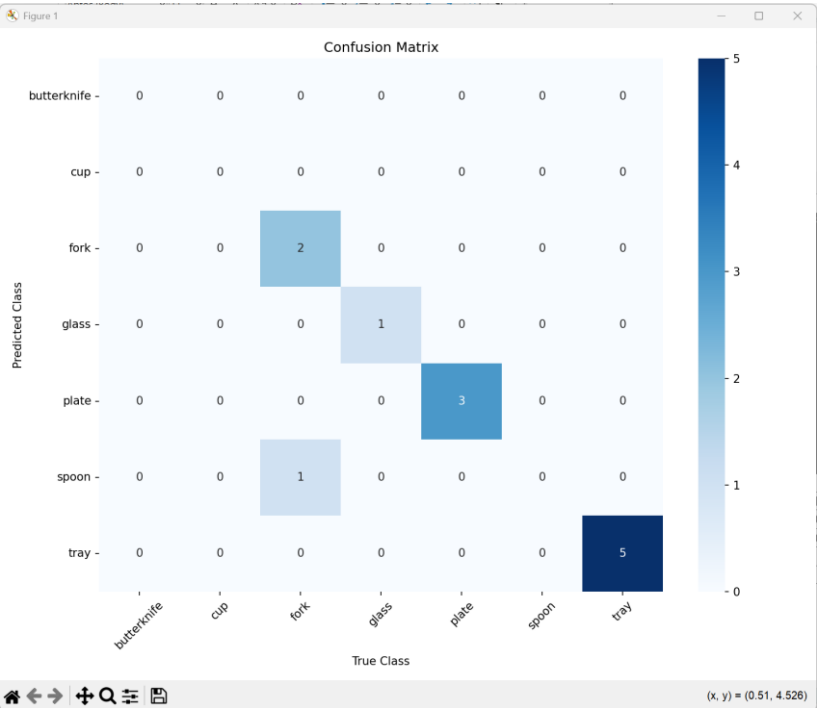


Figure B-18 Confusion matrix tilt 50° , Test 3

Test 4

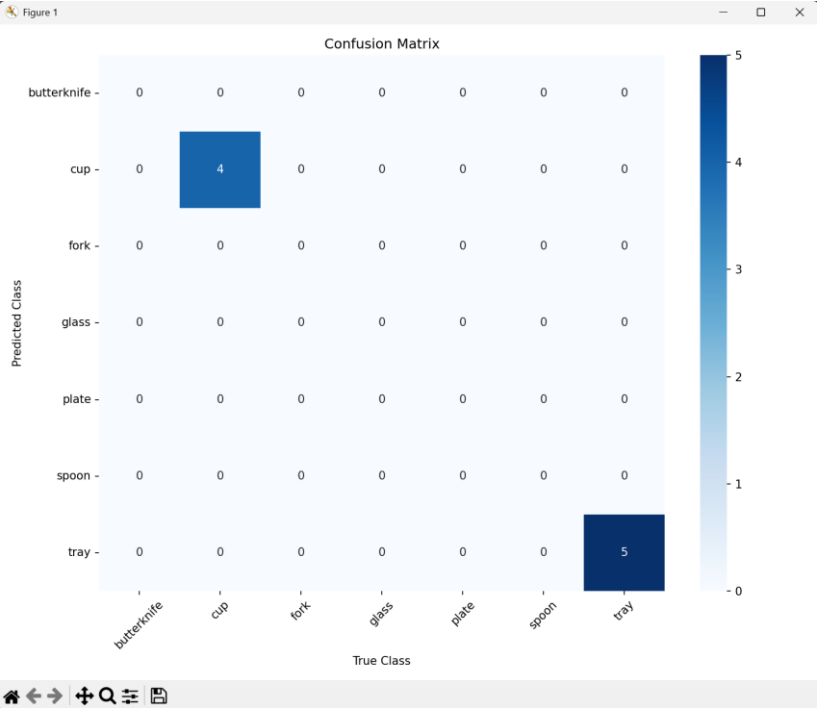


Figure B-19 Confusion matrix tilt 50° , Test 4

Test 5

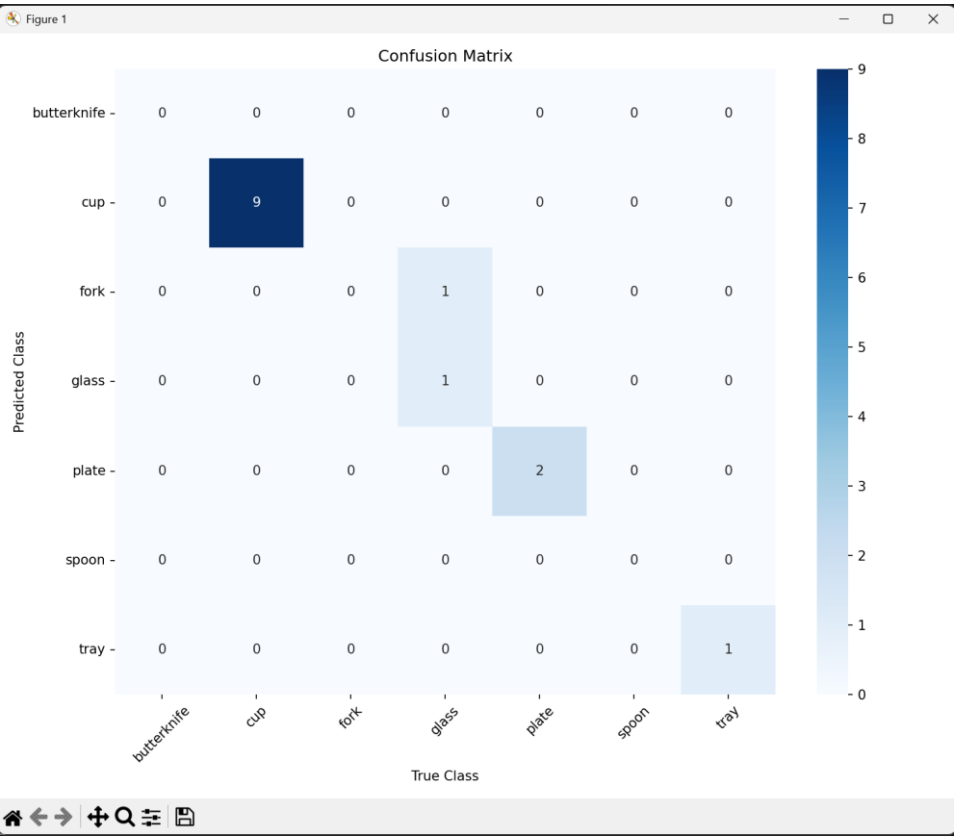


Figure B-20 Confusion matrix tilt 50° , Test 5

B.3.5 Tilt 60°

Test 1

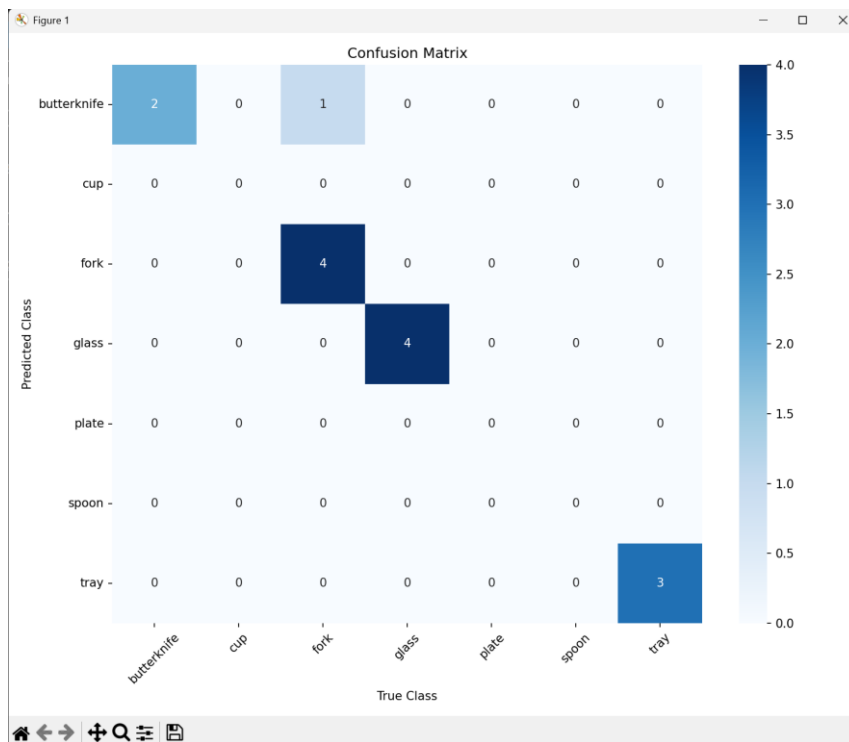


Figure B-21 Confusion matrix tilt 60° , Test 1

Test 2

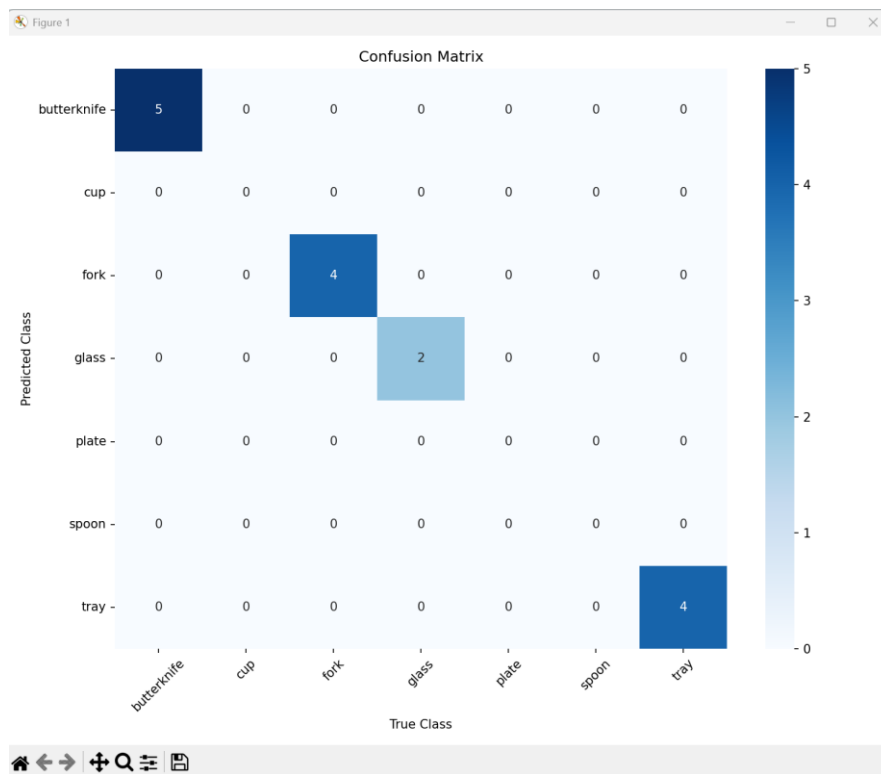


Figure B-22 Confusion matrix tilt 60° , Test 2

Test 3

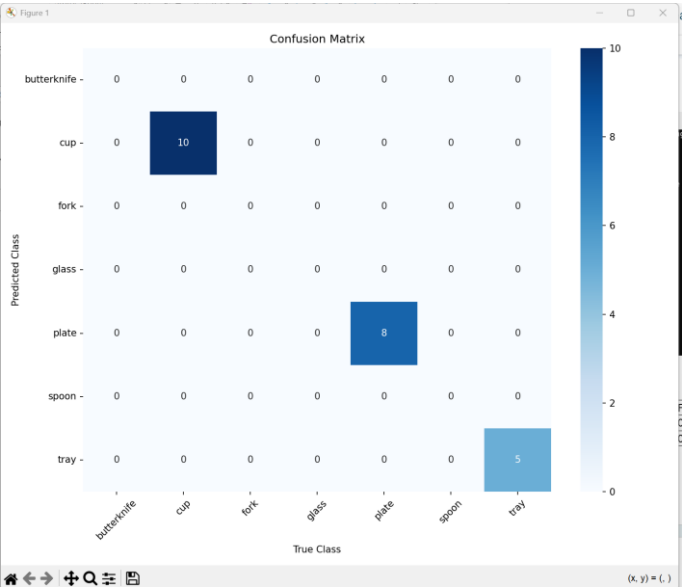


Figure B-23 Confusion matrix tilt 60° , Test 3

Frame 4

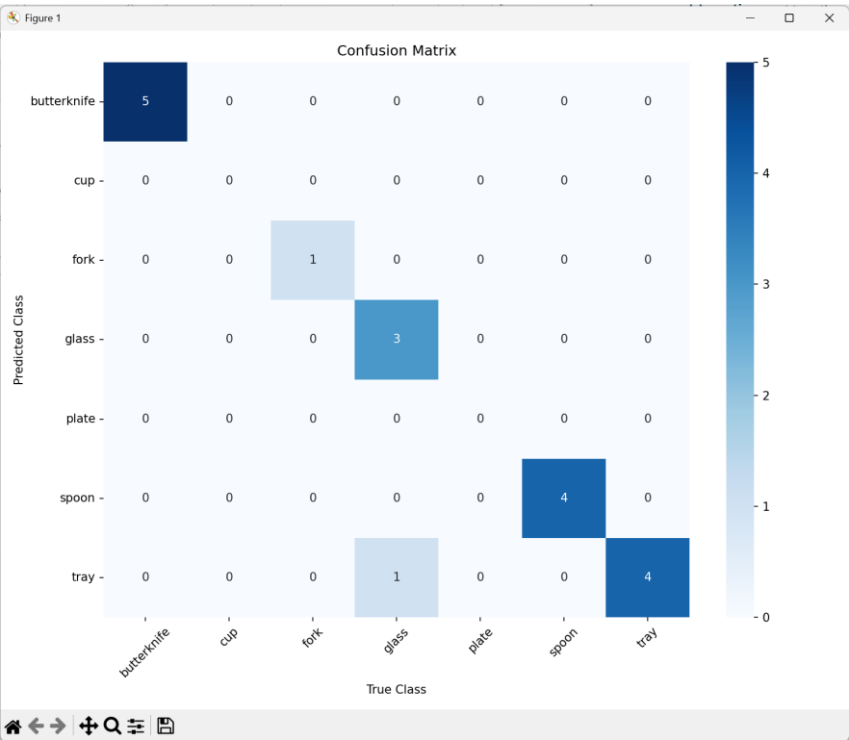


Figure B-24 Confusion matrix tilt 60° , Test 4

Test 5

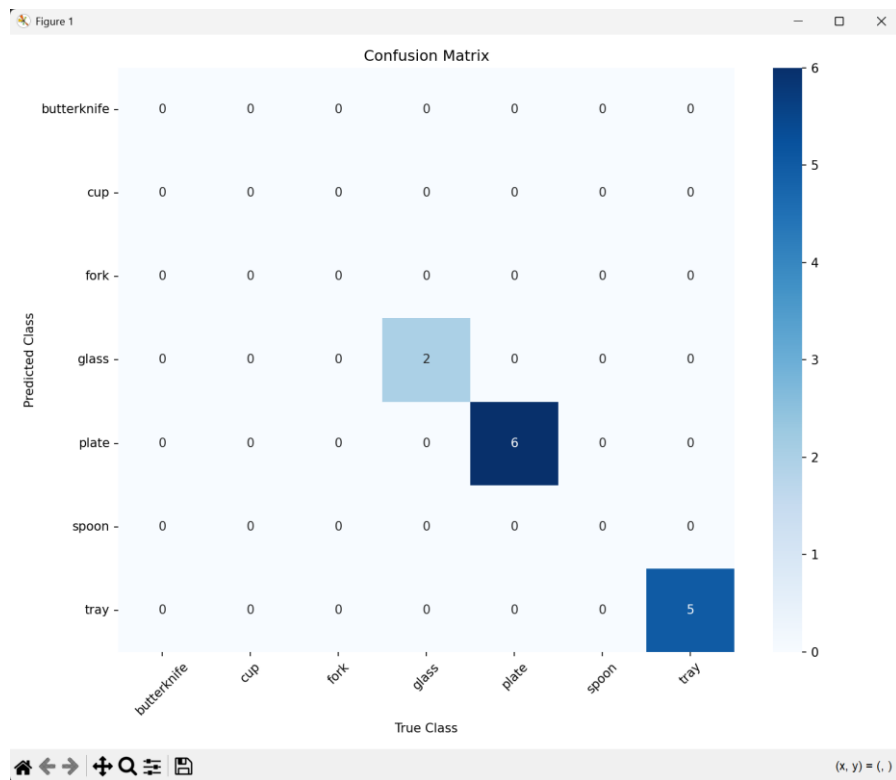


Figure B-25 Confusion matrix tilt 60° , Test 5

Appendix C Gantt chart

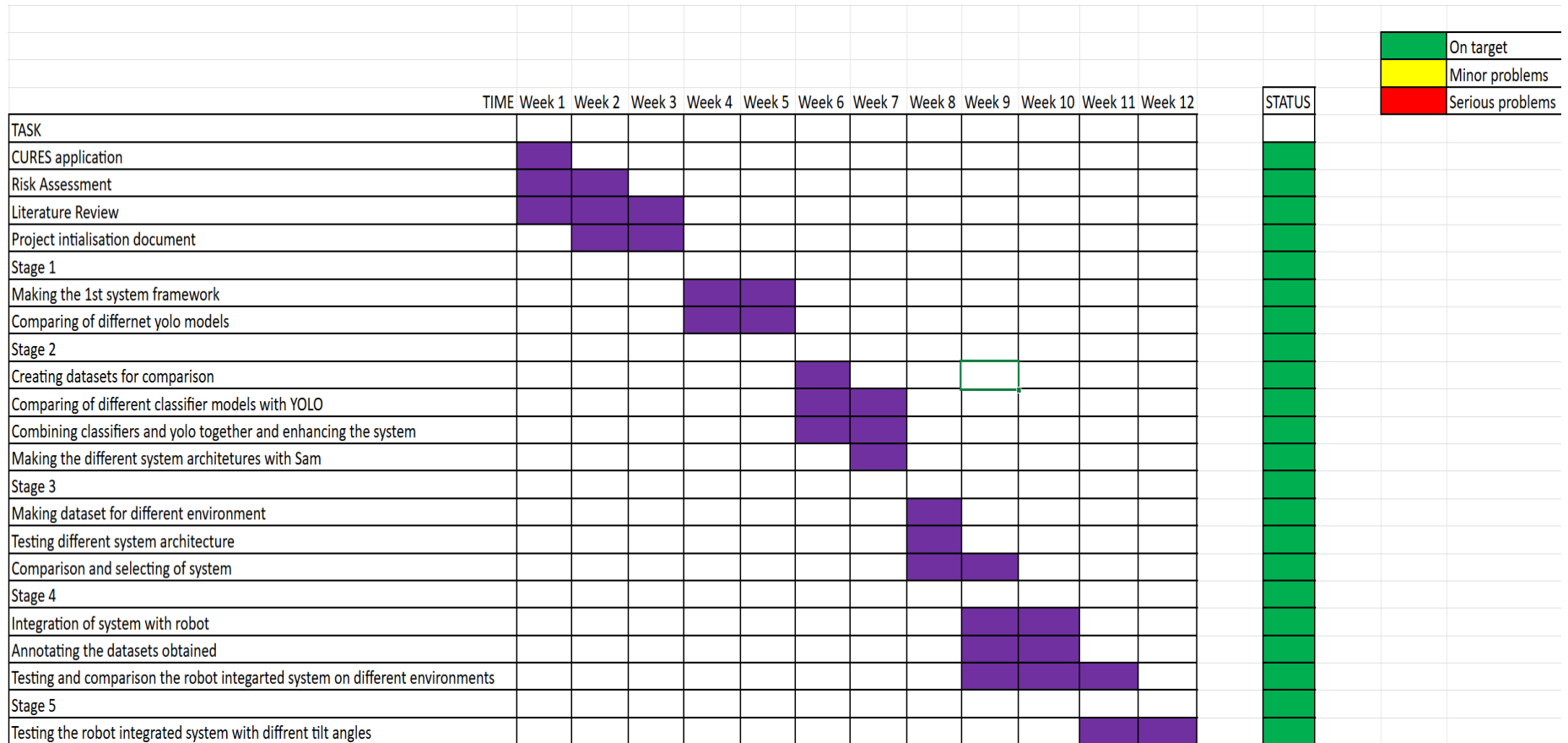


Figure C-1 Gantt chart

Appendix D Risk assessment



Risk Assessment Form

CU-SHE-FORM-3.01

V3.0

Risk Assessment Number:	RA-5685-0525	Title:	Use of SWIFT Robot for Anomaly Detection			
Task/Activity assessed:	Programming and Operation of the SWIFT Mobile Robot within indoor spaces on Campus, including operating in the lab, and also specific corridor spaces.					
Name/job role of people consulted during assessment:	Gilbert Tang, James Fowler, Smart City Consultancy and SWIFT Robotics, Natasha Robin (Robotics MSc IRP)	Date of Assessment:	Tuesday, May 13, 2025	Review Date:	Thursday, May 13, 2027	

Acknowledgements, Sign off and Authorisation				
	Acknowledgement	Name	Signature	Date
Risk Assessor:	By signing this risk assessment I acknowledge my responsibility as the Risk Assessor for conducting this risk assessment in accordance with CU-HAS-PROC-3.01, Risk Assessment Procedure.	James Fowler		Tuesday, May 13, 2025
Checked by: (where required)	By signing this risk assessment I acknowledge my responsibility as the checker for this risk assessment in accordance with CU-HAS-PROC-3.01, Risk Assessment Procedure.			
Authorising Person:	By signing the risk assessment, I acknowledge my responsibility as the Line manager/Supervisor for reviewing and approving this risk assessment and communicating controls and any additional controls to			

	staff/students (as appropriate).			
--	----------------------------------	--	--	--

Significant hazards	Who is affected and how <ul style="list-style-type: none"> Who might be hurt? How bad could it reasonably be? 		What are your existing controls? (Reference all Safe Systems Of Work (SSOW), Standard Operating Procedures (SOP) and Emergency Procedures)	Existing Risk Rating (Consequence x Likelihood = Total)			Are additional controls needed? Y/N (If Yes, RAMP required)
				C	L	TOTAL	
Manual handling	Student, Staff	As the robot weighs around 40kg, if somebody attempted to lift the robot it could cause injury, especially if the robot falls on them.	-Lifting handles are on the top of the robot and should be used when lifting -The robot shall only be lifted by two persons who have completed manual handling training -Safety Footwear to be worn when lifting the robot	3	1	3	No
Impact	Student, Staff	Robot may crash into somebody or something	- Staff and students that are required to work around the robot during motion are	2	2	4	No

		<p>causing injury and or other damage to property.</p>	<p>required to be aware and maintain their distance from the robot when it is moving.</p> <ul style="list-style-type: none"> - Only authorised people are allowed to work with the robot in the allocated space. <p>Sign board warning people about the mobile robot usage.</p> <ul style="list-style-type: none"> - Emergency stop button on the hands of the operator to press in case of emergency stopping the robot immediately. - People in the working area of the robot should wear closed toe shoes and be attentive and keep a safe distance when the robot is in motion. - The robot will be operated in controlled areas such as the lab. When used in the corridor, signage will be used and 				
--	--	--	---	--	--	--	--

			<p>communication will be made to other users of the space for awareness.</p> <p>- The robot only operates at a slow speed <3mph therefore impact force would be low and time to avoid</p> <p>- The robot has smooth, rounded sides which would further reduce the impact force and potential for any bruising/cutting as a result of impact</p>				
Electricity	Student, Contractors, Staff	As the robot requires a 240v mains supply for charging - there is a risk of electric shock or burns.	<p>-The robots charging cable is correctly inspected for any defects before each use.</p> <p>Only trained members of staff and students will be charging the robot.</p> <p>-PAT Tested charger cable</p>	4	1	4	No

			<p>-The robot is used only by a person who has been authorised to do so and received appropriate training and information</p> <p>-The robot is locked in a secure room when not in use</p>				
Ground/Flooring	Staff, Student, Cleaners	A person may walk into or trip on the robot if it is obscured (robot is approximately knee height), furthermore when it is being charged, the cable could pose a trip hazard	<p>-Charge only in specific area, away from doors and gangways.</p> <p>-Make sure when not in use, the robot is parked in a corner of the lab so that it is not obstructing anything, particularly fire exits.</p> <p>-Use of trip mats to cover charger cable if required</p>	3	2	6	
PERSONAL PROTECTIVE EQUIPMENT (PPE):							
							
☐	☐	☐	☐	☐	☒	☐	☐
							
						☐	☐
							
							☐

FOR EVERY ITEM OF PPE REQUIRED, SPECIFY THE TYPE AND OTHER RELEVANT INFORMATION BELOW:

TYPE	OTHER RELEVANT INFORMATION <i>E.G MATERIAL, LEVEL OF PROTECTION, ETC.</i>
Boots	

Emergency Planning Arrangements relating to operations/event

First Aider is on stand-by and made aware of the operations.

Everybody working in the Lab with the robot is given a health & safety lab induction.

Risk Rating Matrix

RISK MATRIX					
Consequence Likelihood	Negligible (1)	Minor (2)	Medium (3)	Major (4)	Severe (5)
Almost Certain (5)	5	10	15	20	25
Likely (4)	4	8	12	16	20
Possible (3)	3	6	9	12	15
Unlikely (2)	2	4	6	8	10
Very Unlikely (1)	1	2	3	4	5

Rating	Interpretation	Authorisation
≤ 6 = Low Risk	Acceptable but ensure that controls are maintained	Line Manager or equivalent
9 -12 = Medium Risk	Adequate but look to improve if reasonably practicable	Line Manager or equivalent
15 – 25 = Unacceptable Risk	STOP activity and make immediate improvements	PVC School/Director of PSU

CONSEQUENCE (considered WITH controls in place)		
5	Severe	<ul style="list-style-type: none"> Fatality (ies) Severe or chronic illnesses or permanent life changing impact
4	Major	<ul style="list-style-type: none"> Injury such as fracture of bones, dislocation, or acute ill health e.g. occupational asthma, occupational dermatitis
3	Medium	<ul style="list-style-type: none"> An injury that requires first aid treatment and subsequent treatment by health care professional No lost time illnesses and no chronic/acute health effects

LIKELIHOOD (considered WITH controls in place)		
5	Almost Certain	<ul style="list-style-type: none"> Will occur/greater than a likelihood of 1 in 1(yr)
4	Likely	<ul style="list-style-type: none"> Known to occur/probably occurs most circumstances/No greater than a likelihood of 1 in every 10
3	Possible	<ul style="list-style-type: none"> Might occur /no greater than a likelihood of 1 in 1000

2	Minor	<ul style="list-style-type: none"> An injury that requires basic first aid treatment such as administering a plaster, individual able to continue at work e.g. minor cuts, bruising, abrasions, strains or sprains
1	Negligible	<ul style="list-style-type: none"> Superficial or no physical injury or health effects

2	Unlikely	<ul style="list-style-type: none"> Not likely/could occur at some time/no greater than a likelihood of 1 in 10,000
1	Very Unlikely	<ul style="list-style-type: none"> May only occur in exceptional circumstances/no greater than a likelihood of 1 in 100,000