Coursework Technical Report

1 Introduction

This report presents the methodologies and outcomes of the project designed to address the challenges in the COMP52715 Deep Learning for Computer Vision and Robotics course. The primary task involved developing a model capable of detecting virtual objects within a 3D environment, and then interacting with these objects. The project trains a binary classifier and an object detector using a dataset of 3D point-clouds with features derived from a Kinect system. The another task is to discuss the performance and efficacy of the model and assessing their potential for real-world applications in robotics. The technical report will go from describing the technical details of using the model, to the analysis of the experimental data and discussion of the results, and finally to the conceptual design of the self-service robotics system.

2 Methods

This section describes the development of an advanced object detection model to enhance the precision and efficiency of our detection system, focusing on architectural choices, training procedures, and optimisations necessary to achieve high performance in real-world scenarios.

2.1 Optimisation of PacMan Helper

The research enhanced the efficacy of the `PacMan_Helper.py` code, particularly the points to image function which projects the 3D point cloud onto the 2D image plane and is executed repetitively within the detection-navigation sequence. Utilising Numba's just-in-time compilation significantly improved the speed of the `apply_blending` function compared to the original Python code, allowing for parallel processing and vectorised operations.

2.2 Simple CNN Binary Model

A Convolutional Neural Network (CNN) architecture 1 was implemented for binary classification of image patches. The model consists of convolutional layers followed by ReLU activation and max-pooling, fully connected layers with ReLU activation and dropout, and a final output layer with sigmoid activation. It was trained using the Binary Cross-Entropy loss function and Adam optimiser. Data augmentation techniques were applied to improve generalisation.

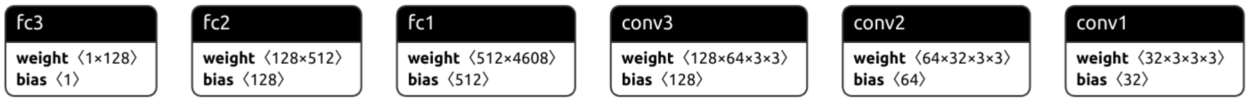


Figure 1: The trained CNN architecture (from right to left)

2.3 Advanced Object Detection Model

The YOLOv8 framework was used to improve detection accuracy and efficiency. Additional training data was generated by simulating different camera positions and orientations within the 3D point cloud environment. Transfer learning with pre-trained weights on the COCO dataset was employed. Hyper-parameters were tuned using a YAML configuration file, and data augmentation techniques were applied.

The training process was monitored using various metrics, including box loss, classification loss, precision, recall, and mean Average Precision (mAP) at different IoU thresholds. Early stopping was implemented to prevent overfitting.

2.4 Navigation and Collection

The trained YOLOv8 model was integrated into the detection-navigation loop. At each iteration, the current view of the point cloud was projected onto the image plane, and the YOLOv8 model was employed to detect the target objects and their bounding boxes. If a target object was detected, the camera moved towards its predicted 3D location. If no target was detected for consecutive iterations, the camera position was adjusted to explore different regions. The process continued until all targets were captured or a maximum number of iterations was reached.

3 Results

This section presents the findings from the deployment of optimised PacMan Helper, binary classifier, and advanced object detection models.

3.1 Optimised PacMan Helper

The optimisation efforts on the `PacMan_Helper.py` code yielded significant performance improvements, as evidenced by the profiling results (Table 1). The optimised `PacMan_Helper_Accelerated.py` code resulted in an approximate 16x speedup for the `points_to_image` function, significantly reducing the overall execution time while maintaining the same output quality (Figure 2).

Table 1: Performance comparison of PacMan Helper

	PacMan_Helper.py		PacMan_Helper_Accelerated.py	
	# Calls	Time (s)	# Calls	Time (s)
Total	8678086	21.696	1617044	1.371
`points_to_image`	1	20.234	1	0.231

3.2 Simple CNN Binary Model

The CNN binary model achieved high accuracy, precision, recall, and F1-score on the training and validation sets (Figure 3). On the test set, the model achieved an accuracy of 0.9817, precision of 1.0000, recall of 0.8261, and F1-score of 0.9048 (Table 2). The lower recall value indicates room for improvement in detecting all positive samples.

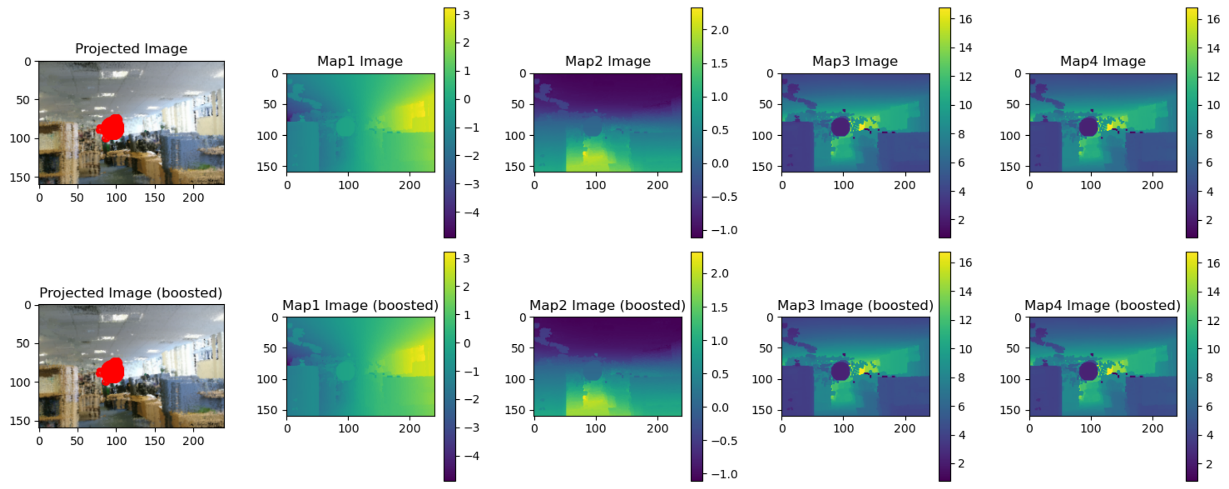


Figure 2: Output comparison of PacMan Helper.

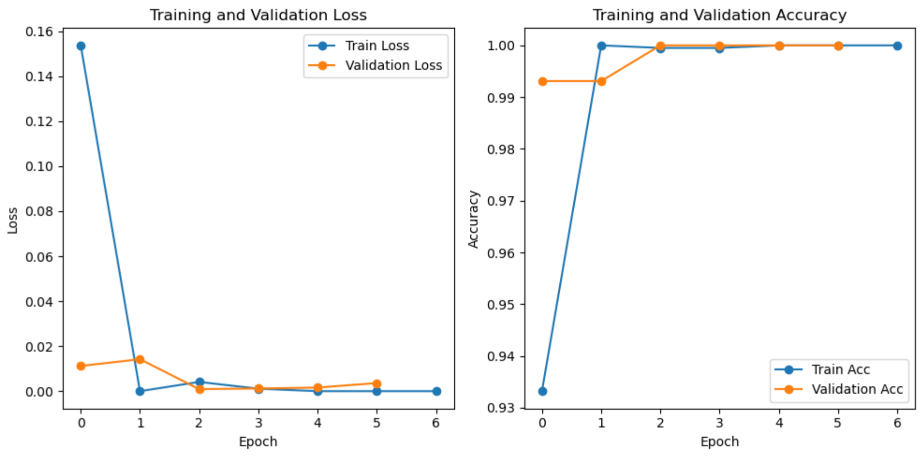


Figure 3: Training and Validation Loss and Accuracy Plots

3.3 *Advanced Object Detection Model*

The YOLOv8 model demonstrated promising results in detecting target objects. The training plots (Figure 4) showed decreasing box and classification losses, and high precision and recall values on both training and validation sets. The model achieved excellent mAP values at different IoU thresholds.

On the test set, the model achieved an F1-score of 0.95 at a confidence threshold of 0.477 and a precision of 0.981 at a recall of 0.5 (Figure 5).

3.4 *Navigation and Collection*

The navigation and collection task successfully captured all target objects in a single iteration, taking 23.1654 seconds (Figure 6). This showcases the robustness and computational efficiency of the approach.

Table 2: Binary CNN model test performance

Metric	Value
Test Loss	0.0211
Accuracy	0.9817
Precision	1.0000
Recall	0.8261
F1-score	0.9048
Elapsed Time	0.0692 seconds

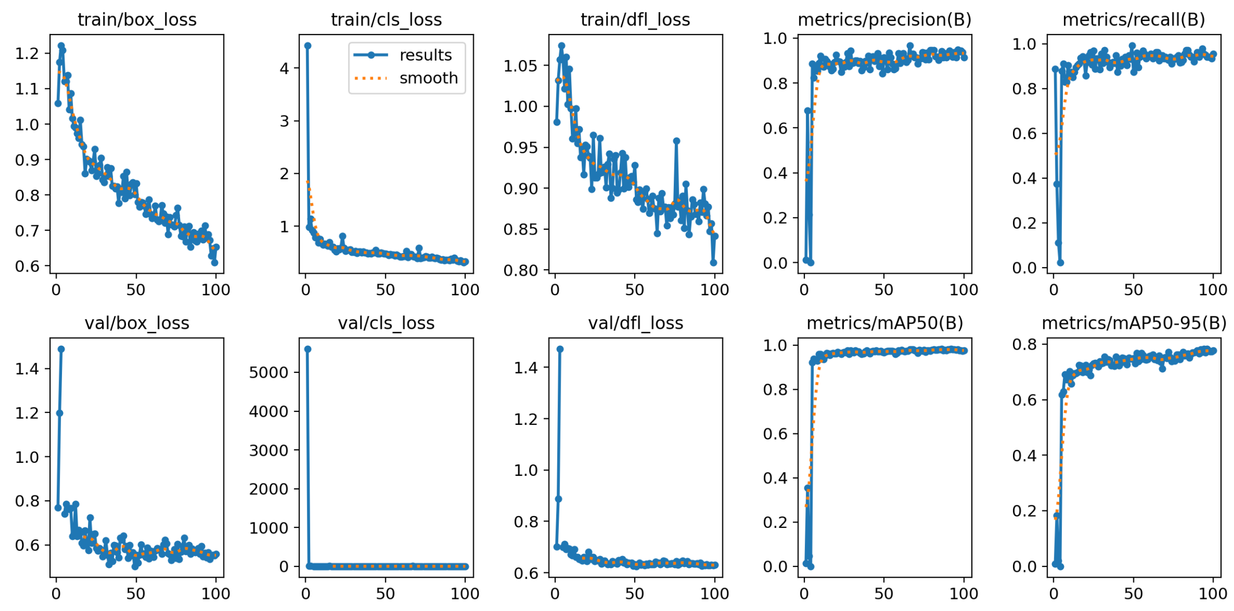


Figure 4: YOLO object detection model training plots

4 Robot Design

A robotic system design based on the Kuka YouBot platform is proposed. It leverages the robot's omnidirectional mobile base and 5 DOF manipulator arm for navigation and object manipulation. The existing sensor suite, including the Microsoft Kinect RGB-D camera, laser range finders, and IMU, is utilised for perception, mapping, and localisation tasks.

The YOLOv8 object detection model is integrated into the system for detecting spherical targets. Navigation algorithms, such as rapidly-exploring random trees, are implemented for collision-free path planning. Control strategies for the manipulator arm are developed for grasping and capturing targets. Sensor fusion techniques, like Extended Kalman Filter, are employed for accurate localisation and odometry.

Optimisation techniques are applied to ensure real-time performance and robustness of the system. Proper sensor calibration and data filtering are implemented to improve the reliability of sensor data.

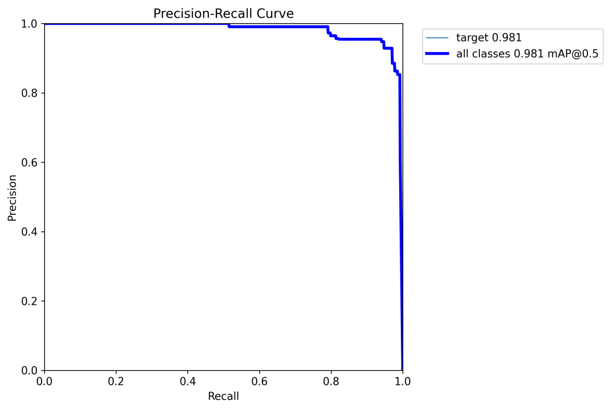
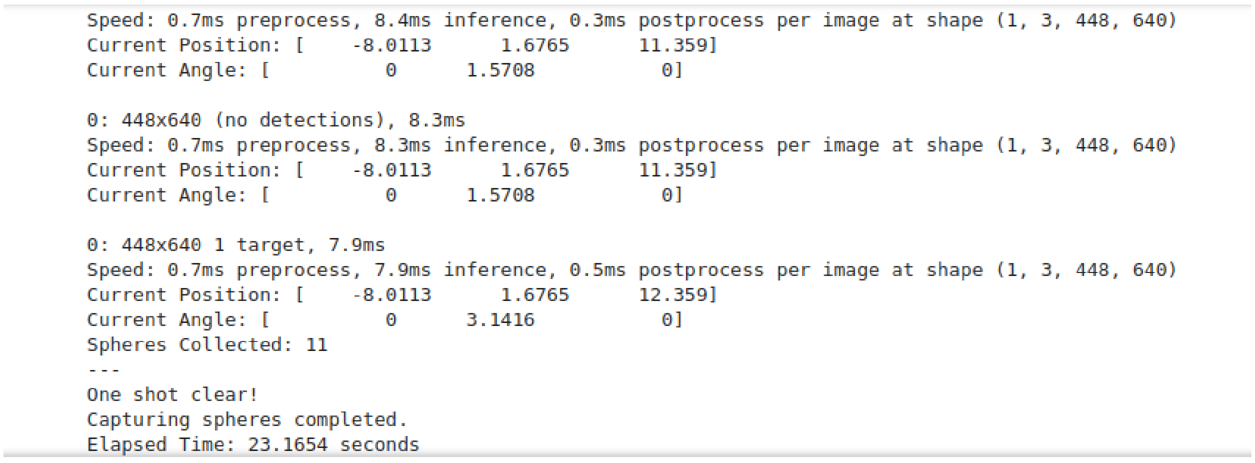


Figure 5: PR curve of testing



```
Speed: 0.7ms preprocess, 8.4ms inference, 0.3ms postprocess per image at shape (1, 3, 448, 640)
Current Position: [    -8.0113       1.6765      11.359]
Current Angle: [          0      1.5708           0]

0: 448x640 (no detections), 8.3ms
Speed: 0.7ms preprocess, 8.3ms inference, 0.3ms postprocess per image at shape (1, 3, 448, 640)
Current Position: [    -8.0113       1.6765      11.359]
Current Angle: [          0      1.5708           0]

0: 448x640 1 target, 7.9ms
Speed: 0.7ms preprocess, 7.9ms inference, 0.5ms postprocess per image at shape (1, 3, 448, 640)
Current Position: [    -8.0113       1.6765      12.359]
Current Angle: [          0      3.1416           0]
Spheres Collected: 11
---
One shot clear!
Capturing spheres completed.
Elapsed Time: 23.1654 seconds
```

Figure 6: The output of the `capture_spheres` function

5 Conclusion

The project successfully developed an advanced system for detecting and capturing spherical targets in a 3D point cloud environment. It leveraged deep learning methods, optimisation techniques, and a well-designed robotic system based on the Kuka YouBot platform. The results demonstrated the effectiveness and efficiency of the approach in terms of object detection accuracy, computational performance, and navigation capabilities.

The proposed robotic system design incorporates the necessary components, algorithms, and control strategies for real-world deployment, considering aspects of efficiency, robustness, and extensibility.

The outcomes of this project highlight the potential of integrating deep learning and robotics to solve complex challenges in computer vision and autonomous systems. The developed solutions and design principles can be further extended and adapted to various applications, such as autonomous navigation, object manipulation, and interactive robotics.