Object Detection and Robot Control

Shantanu Shirsath[1] Aakash Deshpande[2]

[1] University of Siegen [2] Department of Electrical Engineering and Computer Science

Objective

The **main objective** of this project is to enable a **fischertechnik** robot to **detect and chase a specific object**, a Cat in this case. A **pre-trained deep learning model** is **retrained** using a **custom made dataset** and **deployed on a Raspberry Pi** . The robot uses a webcam and the model to detect and track a cat in real-time. **PID controllers compute motion commands** based on the cat's position and size, enabling responsive and **continuous pursuit**.

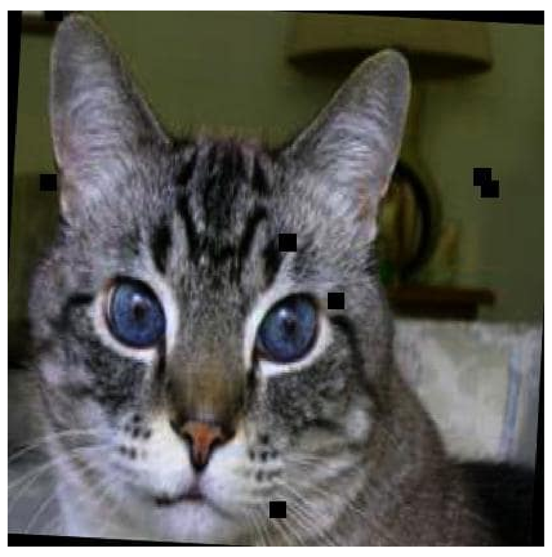
Dataset Generation

Creating a well balanced dataset is crucial in any deep learning project to get good results. Our dataset is a mixture of the following images:

Source	Category	Share (%)	Image Count
COCO [1]	Non-cat	50%	8794
COCO	Cat	25%	4298
Roboflow [2]	Cat	14%	2433
Custom	Cat	11%	2063
Total		100%	17588

Table 1. Breakdown of Dataset Sources for Cat Detection

COCO Cat



Roboflow Cat

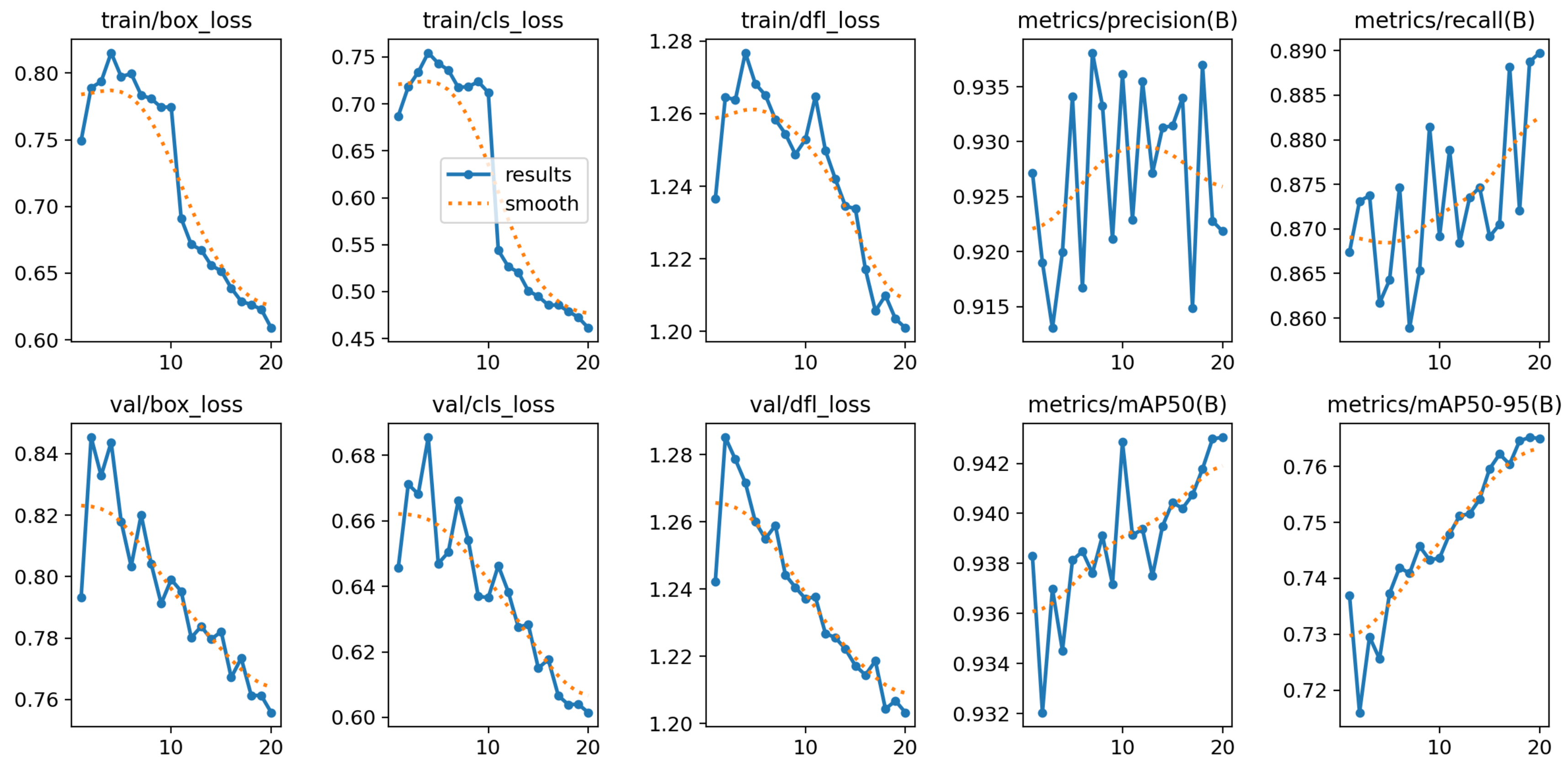
Custom Cat

Sample cat images from datasets

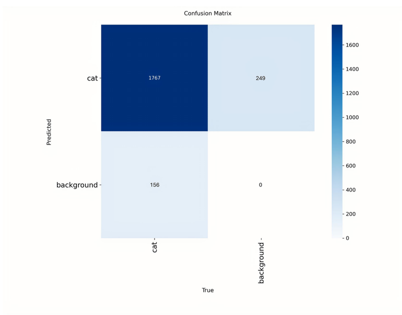
A **video was recorded** using a webcam and **frames were extracted**. Each frame was manually **annotated** with bounding boxes and **converted to YOLO format** using YOLO model [3] itself as the base.

Selecting, training and Integration of the model onto a Raspberry Pi-5

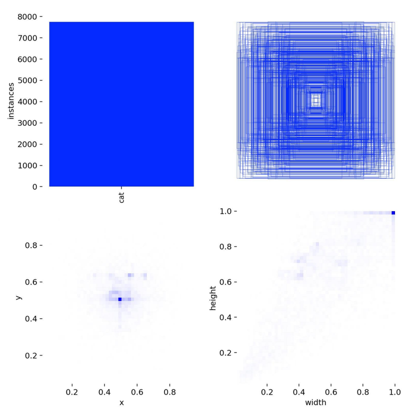
To focus on cat detection, we just **tweaked the detecting head** of the lightweight **YOLOv8n model**. Even on a small custom dataset, **quick training** and **good accuracy** were achieved by freezing the pretrained layers to leverage general visual features..



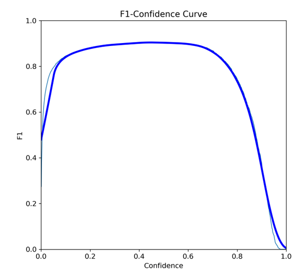
Training Curves



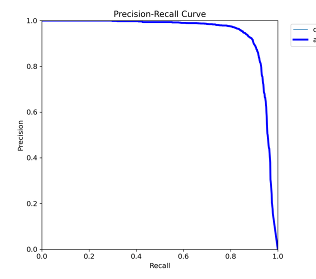
Confusion Matrix



Label Distribution



F1 Score vs Confidence

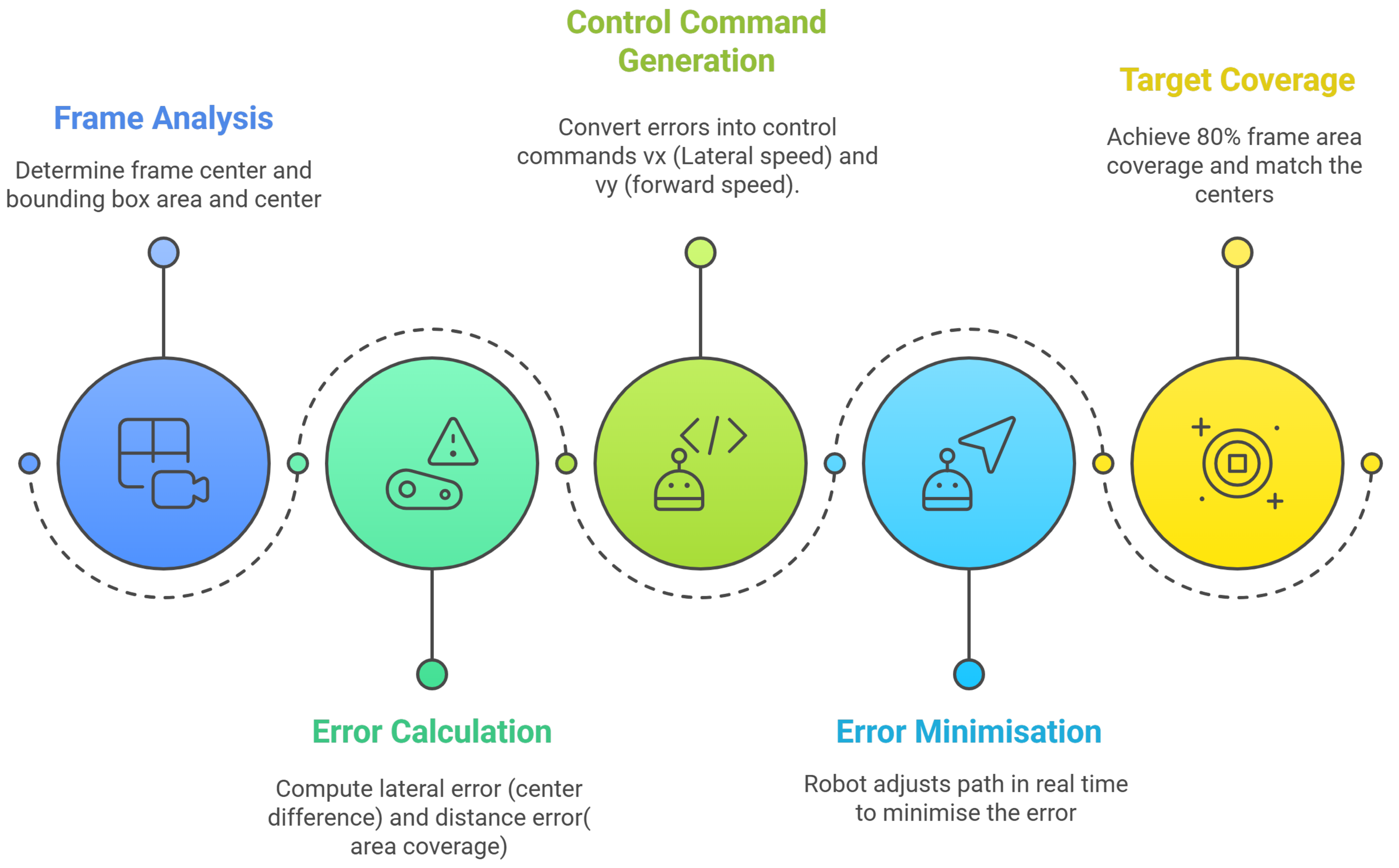


Precision–Recall Curve

We used the **NCNN [4] inference engine** to **deploy** our retrained YOLOv8 model on the **Raspberry Pi**. The model was exported in **ARM CPU-optimized** format. A small Python wrapper is used to load the model and does inference on images. This makes it possible for edge devices without GPUs to identify objects quickly and with minimal power. This setup is ideal for embedded AI applications such as real-time object detection.

Controlling the Robot to chase the Cat

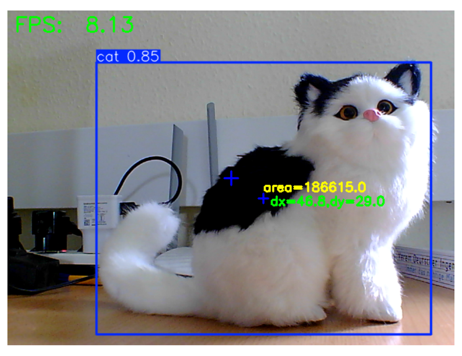
We chose a **Proportional–Integral–Derivative (PID) controller** for its simplicity, stability, and fast response. PID is lightweight, interpretable, and **perfect for real-time control** on low-power hardware like Raspberry Pi.



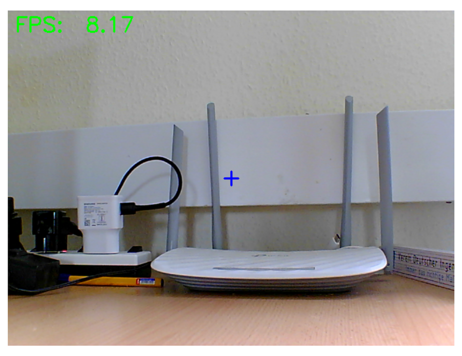
Robot navigation system

Controller	P	I	D	Control Target
Lateral (PD)	0.8	0	0.1	Horizontal alignment
Longitudinal (P)	0.0005	0	0	Object distance

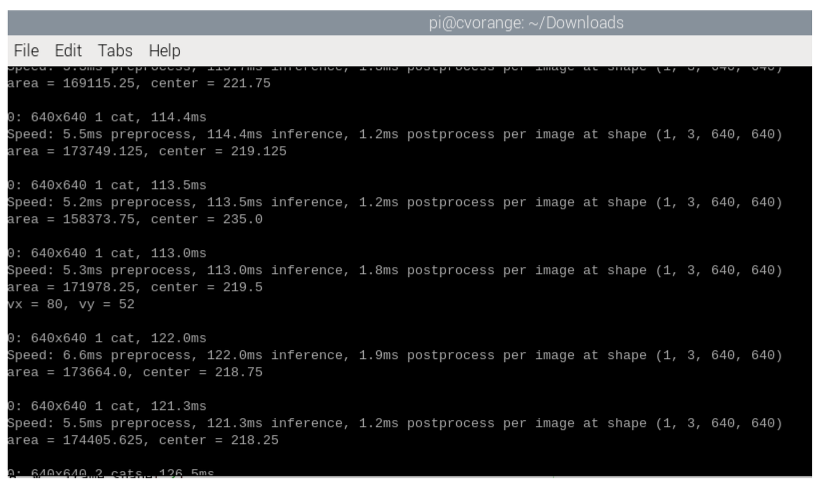
Table 2. Configuration of PID Controllers



Detection of cat



No object detected



Console Display

References

[1] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

[2] Mohamed Traore. Cats dataset. `https://universe.roboflow.com/mohamed-traore-2ekkp/cats-n9b87`, nov 2022. URL `https://universe.roboflow.com/mohamed-traore-2ekkp/cats-n9b87`. visited on 2025-07-10.

[3] Glenn Jocher, Jing Qiu, and Ayush Chaurasia. Ultralytics yolo, 2023. URL `https://ultralytics.com`.

[4] Hui Ni and The ncnn contributors. ncnn, 2017. URL `https://github.com/Tencent/ncnn`.

We would like to thank **Prof. Dr. Michael Möller**, **Jan Philipp Schneider** and **Alexander Auras** and acknowledge their supervision and guidance.