Human Detection and Feature Saving for Resource-Limited Hardware

Jordan Cousineau
Boston University
44 Cummington Mall
jordache@bu.edu

Anthony Southmayd
Boston University
44 Cummington Mall
as0317@bu.edu

Abstract

Deploying computer vision algorithms on low-power microprocessors for real-time robotic applications presents performance challenges. This paper explores the use of the You Only Look Once (YOLO) object detection framework for real-time human detection and the Scale-Invariant Feature Transform (SIFT) algorithm for feature matching in the context of an autonomous human-following robot. To address resource limitations, we apply optimization strategies including Open Neural Network Exchange (ONNX) model conversion [1], grayscale preprocessing, and image cropping. We evaluate the trade-offs between computational efficiency and detection accuracy to improve real-time deployment on constrained hardware.

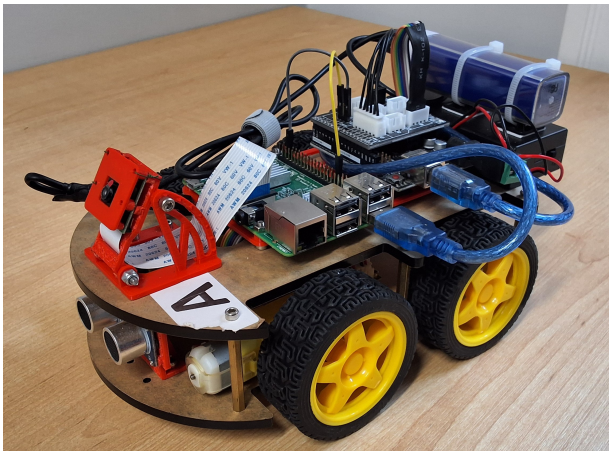

Figure 1. Robot Car Hardware

1. Introduction

Computer vision is a core component of modern robotics, enabling systems to perceive and interact with their environment through visual data. By analyzing camera input, a robotic system can perform tasks such as self-localization, obstacle avoidance, and decision making based on object recognition or scene understanding. However, many state-of-the-art computer vision algorithms are computationally intensive. When deploying these algorithms on embedded microprocessors for real-time performance becomes a significant challenge due to limited processing power and memory.

In this paper, we focus on the computer vision subsystem of an autonomous human-following robot, investigating practical methods for deploying real-time human detection and feature matching on resource-limited hardware. Specifically, we utilize YOLO framework for object detection and the SIFT for feature extraction. To optimize performance, we explore image preprocessing techniques such as grayscale conversion and cropping, as well as model conversion to ONNX format for lightweight inference.

1.1. Related Work

YOLO has been used in a variety of robotic systems. One example of this is an autonomous ATV [4] which has a core task of all terrain navigation and object-tracking. While this paper was an inspiration, and provided some core groundwork in this paper, it has less constrained hardware limitations. The papers use of YOLO is due to the fact that it is fast and accurate which meets our standards. However, there are more ways to increase the speed of YOLO models such as exporting them in ONNX format since this format is designed for cross-platform deployment, and improves the speed of the model [1] as well which makes it well suited for deployment. SIFT is also applied to save important features and descriptors of a person, which is a concept that was taught in CS585.

2. Methodology

We would like to highlight the programs flow, which can be seen in Figure 2. This pipeline demonstrates at what point each process is happening starting with capturing the frame.

2.1. Data Processing

We trained our YOLO models on the 2017 COCO dataset [2] and each image in the dataset has an image resolution of 640×480. ; however, this dataset is used for object detection in general and then the context of a person-following robot we only need to be able to detect whether a human is detected or not, so we made a subset of the original dataset that contained roughly 30k images, where 15% of those images are images that do not contain a human. Additionally, we wanted a grayscale model, so we went through and created a copy of the custom subset and had converted all the images to grayscale using the opencv color converter.

2.2. Model Training

The YOLOv8 nano model was selected due to its lightweight architecture and suitability for real-time inference on resource-constrained devices. Two versions of the model were trained: one using RGB images and the other using grayscale images, to evaluate the trade-off between color information and computational efficiency.

Training was performed using the default YOLOv8 parameters, with the exception of three key settings:

1. Image size was reduced to 320×320 pixels to minimize input data dimensions.
2. Number of epochs was limited to 3 to meet development time constraints.
3. Batch size was set to 2 to accommodate memory limitations during training.

Additionally, the models were trained exclusively on human images. This decision was made to reduce computational overhead by limiting the detection task to a single object and simplifying the model's classification to improve inference speed. This class-specific training was also aligned with the primary objective of the system: to detect and follow a human in real time.

The average metrics for the YOLO models can be seen in Table 1. While these metrics provide valuable insight into the model's performance, we believe they are not the primary focus of this project. We acknowledge that these values could be further improved through additional fine-tuning of parameters.

2.3. SIFT

SIFT is an algorithm that detects local features in images by extracting keypoints and comparing them using Euclidean distance [3]. While the original method was designed for matching features across static images, we apply SIFT to each frame of a live video feed to track features in real time. In our implementation, we use a distance threshold of 0.8 and require a minimum of 20 matched features. When these

Metric	RGB (AVG)	Grayscale (AVG)
Box Loss	1.63	1.75
Cls Loss	1.87	2.06
mAP@50	0.39	0.36
mAP@50-95	0.21	0.22
Precision	0.55	0.54
Recall	0.38	0.34
Val Box Loss	1.61	1.70

Table 1. Average YOLOv8n Training Metrics (RGB vs Grayscale Models)

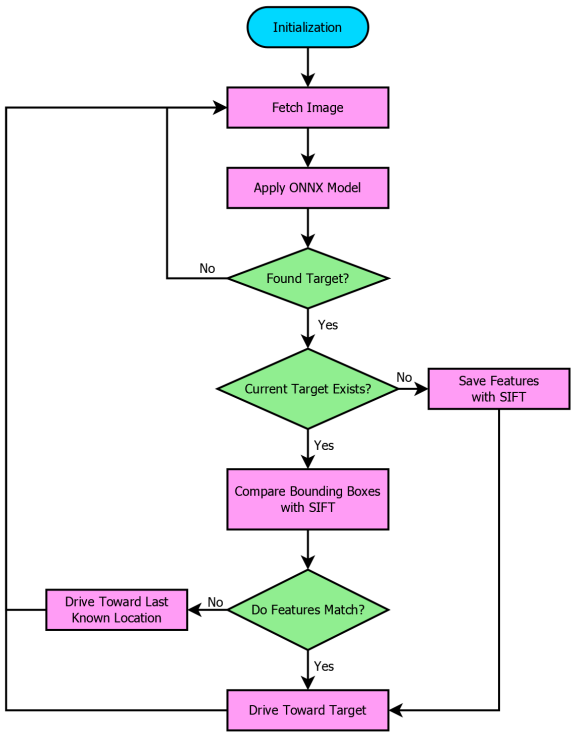

Figure 2. Program Flow

conditions are met, we save the new keypoints to enable tracking under dynamic conditions such as turning, lighting changes, and background shifts. Additionally, the robot will enter a search mode if it is unable to find the previously matched person, which means it will reset the saved features and descriptors and will save the information of the next seen person. To evaluate the effectiveness of SIFT we tested its performance when applied to entire frames versus a cropped version of each frame.

2.4. Experiments

2.4.1. Evaluation of the YOLO Models

The models were deployed and tested in a Python script running on a laptop. We measured performance using

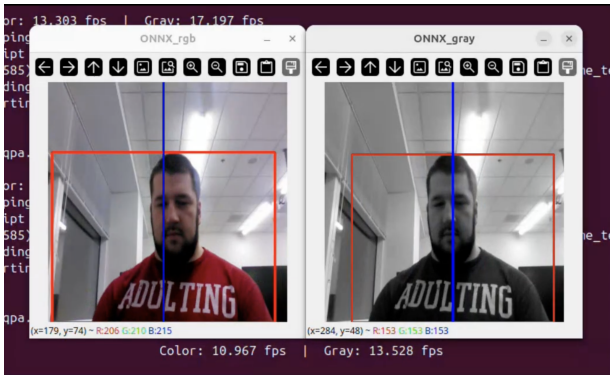

Figure 3. Live Model detecting human (Red Border) with SIFT Detection (Blue Line) showcasing FPS

frames per second (FPS), calculated by timing the execution of both human detection and SIFT processing using timestamps captured before and after each iteration:

$$\frac{1}{endTime - startTime} \tag{1}$$

We recorded FPS values over a 10-second window during live detection and computed the average to assess the performance of each configuration.

The results of this comparison revealed that the grayscale model performed approximately 2.4% better than the RGB model. However, this improvement was not considered significant enough to make noticeable improvements on the microprocessor.

Trial	RGB	Grayscale
1	29.656	31.031
2	31.375	31.14189723
3	21.13589979	24.560131
4	34.74301284	30.89222775
5	28.7724622	31.40157409
Average	29.13665959	29.80552804

Table 2. Results of the FPS Comparison.

2.4.2. Evaluation of SIFT

Similar to our model evaluations, we use frames per second (FPS) as the primary performance metric, use the same FPS calculation, and take the average of the first 10 seconds worth of FPS values. In this experiment, we compare the application of SIFT to the full video frame versus a cropped region from the bounding box of human detection. This comparison is conducted for both the RGB model and grayscale model to assess how input and frame size affect SIFT's real-time performance. In Figure 3 we can see the comparison between the two models, as well as the FPS under their feeds; however, it should be noted that the blue line that bisects the bounding box of the human represents that that is the person with the saved keypoints and descriptors. If another person were replace the current person in frame the blue line would disappear, indicating that this is not the human to be tracked.

The RGB full-frame configuration outperformed the RGB cropped configuration by approximately 47%, while the grayscale cropped configuration performed about 0.24% better than its full-frame counterpart. These RGB results were unexpected; we initially hypothesized that cropping would improve performance by reducing the number of pixels processed. This discrepancy is notable, as the results in Table 2 show significantly higher performance, whereas Table 3 reports values consistently around 20 FPS. Interestingly, this trend does not hold for the grayscale model, where cropping did lead to a slight performance improvement.

Trial	RGB Full	RGB Cropped	Grayscale Full	Grayscale Cropped
1	26.507	20.542	29.115	31.246
2	28.049	19.323	30.350	25.672
3	28.799	18.656	30.685	28.705
4	28.836	18.812	29.680	35.772
5	29.087	18.765	29.923	28.724
Average	28.255	19.220	29.951	30.024

Table 3. FPS comparison between full-frame and cropped SIFT on RGB and Grayscale inputs.

3. Conclusion

3.1. Results

Based on our experimentation, we were able to see which model performed better under each condition. It was surprising to see that the RGB model with SIFT applied to the full frame was performing significantly better than its cropped frame counterpart. However, the fact that the grayscale model with SIFT applied to the cropped frame performing the best of all the tests makes sense and aligns with our original hypothesis, as we assumed it would be processing the least amount of necessary information to get results. Overall, the grayscale model did outperform the RGB model even if only slightly but the SIFT application ended up depending on which model it was being applied to.

We can also confirm that by training the model to only detect humans it was performing better on the mi-

croprocessor, since when a person entered the frame the FPS would decrease, and increase when they left frame. If anything else was being detected in frame it would also decrease the FPS. When running the whole program on the microprocessor we were achieving around 1-2 FPS which granted is slow, still achieved successful results in tracking and following a human.

3.2. Limitations

One limitation of the trained models is a reduction in detection accuracy when a person is only partially visible in the camera frame. In such cases, the model occasionally fails to detect the human altogether. This behavior is likely influenced by training dataset, which predominantly contains images of visible and centrally framed individuals, potentially biasing the model. Additionally, the resizing of input images to 320×320 pixels may introduce distortion or result in the loss of fine details, particularly near the edges of the frame, further reducing the model's ability to detect partially occluded or cropped subjects.

3.3. Future Work

One area for future improvement involves addressing the camera's low-angle perspective 1, which differs significantly from the viewpoint in the training images. Since the camera is mounted closer to the ground and angled slightly upward, the captured frames often show humans from a lower, skewed perspective. We hypothesize that applying a homography transformation to warp the input frames could improve detection accuracy by making the camera view more closely resemble the training dataset's frontal or top-down perspectives. This preprocessing step could help reduce the domain gap between training and inference, potentially leading to more consistent and reliable detections in real-world deployment.

References

[1] Desen Bu, Bei Sun, Xiaoyong Sun, and Runze Guo. Research on yolov8 uav ground target detection based on rk3588. In *2024 2nd International Conference on Computer, Vision and Intelligent Technology (ICCVIT)*, pages 1–5, 2024. 1

[2] 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. *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014. 2

[3] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004. 2

[4] Joseph M. Phillips, Sam Shue, and James M. Conrad. A design architecture for steering an autonomous all-terrain vehicle using object tracking via computer vision and yolov8. In *2024 IEEE 21st International Conference on Smart Communities: Improving Quality of Life using AI, Robotics and IoT (HONET)*, pages 242–247, 2024. 1