is-that-A-CART: A Preliminary Framework for Automated Shopping Cart Retrieval in Retail Environments with YOLOv8

Wentao Jiang
University of Rochester
500 Joseph C. Wilson Blvd., NY 14627
wjiang20@u.rochester.edu

Alvin Jiang
University of Rochester
500 Joseph C. Wilson Blvd., NY 14627
yjiang54@u.rochester.edu

Arthur Masiukiewicz
University of Rochester
500 Joseph C. Wilson Blvd., NY 14627
amasiuki@u.rochester.edu

Abstract

Upon concluding their shopping experience, patrons exiting markets may leave store-owned shopping carts at various locations within the environment without heeding the potential consequences. For instance, carts scattered around retail environments may lead to traffic blockage, and carts on parking spaces may prevent vehicles from parking there. Currently, retail stores such as Target® and Wegmans® hire human resources to retrieve shopping carts and gather them at a specific location for recycling. However, our goal is to automate this process by incorporating computer vision technologies. For our study, we train and fine-tune the ***YOLOv8*** *(You Only Look Once, version 8) network, a state-of-the-art computer vision model, to perform bounding box detection. In addition, we propose a framework with machine integration to streamline shopping trolley retrieval.*

1. Introduction

Managing shopping cart locations presents a considerable logistical challenge in modern retail environments. Customers often leave carts scattered throughout the store, parking areas, and store vicinity, resulting in traffic disruptions and inefficient parking space usage. Furthermore, unattended objects in public may raise security concerns for general public safekeeping [13], necessitating extraneous attention in the surveillance process. Traditionally, stores like Target and Wegmans employ human labor to streamline this process. However, we would like to propose a framework that automates the retrieval of shopping trolleys involving computer vision and robotics technologies. Toward this end, we aim to train a model and achieve high performance on shopping cart segmentation from images. In addition, we would also like to present a holistic review of the modern landscape of robotics and how machine vision technologies can be involved during shopping trolley retrieval.

The remainder of this paper is organized as follows: Section 2 discusses related works in the field, Section 3 describes the methods employed in our study, Section 4 presents our experimental analysis and results, Section 5 reviews the current landscape of robotics in the context of object retrieval, Section 6 presents a preliminary framework for cart retrieval, and Section 7 discusses these findings in the context of existing literature and potential future approach.

2. Related Work

Melegrito et al. [13] presented their findings using the YOLOv3 network for detecting abandoned shopping carts in parking areas. This approach achieved both high training and validation accuracy and precision in identification. This study highlighted the YOLO model's potential application in retail management and operational efficiency. Despite the similarity to this article, our study implements shopping trolley detection in various environments, highlighting its uniqueness in its field of study.

Jadhav and Momin [9] contributed to an identification task of abandoned objects in public through the lens of video surveillance systems. The scholars utilized a dual-background, hybrid model efficient and reliable for stationary object detection. After detection, they extracted object attributes such as height, width, size, color, and time. Their study spearheaded potential future direction for our research

to classify whether detected shopping carts have been unattended.

Albeit not directly related to object detection and retrieval, Wagner et al.'s article [16] provided meaningful insights on the implications of in-store shopping cart movements on consumer shopping behavior. Through disguised observational studies, their aggregate data revealed that shopping trolley placement and movement impacted the duration of a consumer's visit, paths taken within the store, and interaction with products.

3. Methods

3.1. Dataset

We obtained Kornilov's "shopping trolley" dataset [10] from Roboflow to train the YOLOv8 network and to test the model's performance.

3.1.1 Data Volume

Kornilov's dataset [10] comprises a total of 6903 images collected from diverse retail environments to ensure a comprehensive representation of various scenarios involving shopping cart placement. The dataset's split ratio is as follows:

Dataset Split	Number of Samples	Percentage of Total
Training Set	5,996	87%
Validation Set	461	7%
Test Set	446	6%

Table 1. Dataset Split

While the train-test-split ratio deviates from the conventional allocations, which have a smaller training ratio, this variance is justifiable given the substantial volume of [10]. Larger datasets can effectively support different split ratios without compromising the validation and performance of the models, as proposed by Rácz et al. [15].

3.1.2 Data Measurements

Color Channel	Mean	Stdev
Red	0.587	0.199
Green	0.558	0.203
Blue	0.534	0.206

Table 2. RGB Description of the Original Dataset (normalized)

For comparison, the default *ImageNet* normalization parameters are mean = [0.485, 0.456, 0.406] and std = [0.229, 0.224, 0.225].

Overall, these statistics suggest that Kornilov's dataset [10] contains a diverse set of images with no extreme bias towards any particular color. The high standard deviation values of red, green, and blue confirm the presence of diverse image content and/or varying lighting conditions.

3.1.3 Data Annotation

The ground truth values have been pre-labeled by the author of the dataset [10]. Specifically, these values are bounding boxes that highlight shopping carts within the images in the dataset. Each bounding box is defined by the centerX, centerY, width, and height that specify the rectangular area covering a shopping cart.

3.1.4 Image Augmentation

The preprocessing pipeline integrates a suite of advanced image enhancement and augmentation methods to adapt and scale for the inclusion of additional techniques as needed. To further enhance the model's ability to generalize across different scenarios, we employ a mix of geometric and photometric transformations to enhance the model's ability to generalize across varied real-world conditions. Additionally, we have configured our system to save these transformed images, which facilitates a thorough inspection of the preprocessing effects but also aids in fine-tuning the model by providing a diverse array of training examples.

- **Resizing and Cropping**: Images were resized to 640×640 pixels to ensure consistency in input dimensions across the dataset, and to fit within the standard used for YOLOv8. Post resizing, a dynamic cropping strategy was implemented for test set, where images were scaled between 80% and 100% of their initial size, maintaining an aspect ratio from $3/4$ to $4/3$. This helps the model in detecting shopping carts from varying distances and partial views.
- **Random Vertical Flip**: Each image in the test set had a 50% probability of undergoing a vertical flip. This transformation enhances the dataset's variance by simulating different possible orientations of shopping carts, essential for robust detection.
- **Color Jittering**: To further augment the dataset and account for varying lighting conditions that might affect the appearance of shopping carts, we applied color jittering. This transformation randomly adjusts the brightness, contrast, saturation, and hue of the images, thereby simulating different times of day and lighting conditions found in typical retail environments.

3.1.5 Data Preparation

In processing the annotations, we prioritized the largest object in each image as the primary label, based on its spatial dimensions. This approach was adopted to enhance accuracy and reflect real-world scenarios where the most prominent object is often of the highest relevance.

Figures 1 and 2 show a sample image from our dataset after applying the preprocessing steps outlined above.

Figure 1. Example of a preprocessed image showing resized and cropped dimensions in the Train Set.

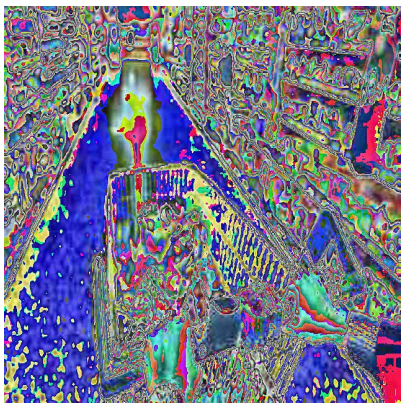

Figure 2. Example of a preprocessed image showing resized and cropped dimensions in the Test Set.

3.2. YOLOv8 Model

3.2.1 Model Analysis

The transfer-learning type YOLOv8 model employs several advanced techniques common in convolutional neural networks (CNNs), specifically tailored for object detection. This model is ideal for real-time applications due to its ability to process images rapidly while maintaining high accuracy. Below is a detailed analysis of its components and how they contribute to the task of shopping cart detection, referencing the official *PyTorch* documentation [3]:

- **Convolutional Layers (Conv2d)**: These layers are the primary feature extractors in the network. By applying varying numbers of filters, they capture both low-level features such as edges and textures, and high-level features like the shapes of shopping carts. The varying kernel sizes and strides enable the model to detect carts at different scales and orientations.
- **Batch Normalization (BatchNorm2d)**: This technique is used to stabilize and speed up the deep network training. It normalizes the activation of a previous layer at each batch, maintaining the mean output close to 0 and the output standard deviation close to 1, which improves gradient flow, allows higher learning rates and reduces the sensitivity to initial weights. Normalizing layer outputs ensures that the network behaves consistently across different training batches, which is vital when adapting the model to diverse lighting and background conditions in parking lots or store entries
- **SiLU Activation**: Sigmoid Linear Unit, defined as $f(x) = x \cdot \sigma(x)$. It offers a balance between ReLU and sigmoid functions. The SiLU activation function helps the model make more effective use of the information it processes by allowing gradients to flow better during backpropagation. This property is particularly beneficial in complex scenes where shopping carts might be partially obscured or overlap with other objects.
- **Skip Connections and Bottlenecks**
  - **Skip Connections** ensure that the model retains important information from early in the network, aiding in the accurate reconstruction of the detected objects in the final output layers. This feature helps preserve the detailed features of carts across the network, i.e., mitigate the vanishing gradient problem.
  - **Bottleneck Layers** effectively reduce the computational burden by minimizing the number of parameters, allowing the model to operate efficiently without sacrificing performance—key for deploying the model in systems with real-time constraints.
- **SPPF (Spatial Pyramid Pooling - Fast)**: An adaptation of the Spatial Pyramid Pooling layer. This layer ensures that the network can handle input images of varying sizes, making it robust to different camera setups and operational fields without requiring reconfiguration of the network for each new camera or viewpoint.
- **Upsampling and Concatenation**
  - **Upsampling** is used to increase the resolution of the feature maps from deep in the network, which enhances the model's ability to locate and delineate shopping carts precisely within the image.
  - **Concatenation** merges feature maps from different layers, allowing the model to use both detailed texture

information and higher-level shape information to improve detection accuracy.

- **ModuleList**: This component normally works as a PyTorch container. It provides the architectural flexibility needed to customize the model for specific tasks like shopping cart detection and allows for the easy addition or modification of layers as new challenges or data types are encountered in different retail environments.

3.2.2 Task-specific Deployment

We decided to deploy pre-trained YOLOv8 models of the "nano" variety, meaning models containing roughly 3M parameters. We examined two versions of YOLOv8n, one which was trained initially trained on the COCO (Common Objects in Context) and one on the OpenImagesV7 dataset. In addition, we examined the performance of training YOLOv8n from scratch on our dataset without prior pre-training. Aside from the obvious difference between training from scratch and training a model with previously pre-trained weights, it is important to discuss both datasets used to distinguish between possible future performance differences between both pre-training .

The COCO 2017 dataset is a large-scale object detection, segmentation, and captioning dataset, with over 140k images with 5 objects per image. The bounding boxes are labeled via 91 different classes derived from complex everyday scenes, with shopping trolley, or any version of 'cart' not present as a label [1]. The annotation for COCO was completed manually via Amazon Mechanical Turk, where human workers manually segmented each object instance, and multiple annotators were instructed to confirm the presence of categories and accuracy of instance segmentation [12].

The OpenImagsV7 (OIV7) is an even larger scale dataset; composed of 9 million images, with an average of 8.3 objects per image. The dataset contains bounding boxes dispersed among 600 various categories, with 'cart' being present as one of the 600 categories [2] It is important to note that we are focusing on the detection of shopping carts (trolleys) specifically. However, we expect some shopping carts to be present in the pre-training set with the cart label due to the diverse meaning of the word, and hypothesize that some presence of shopping carts during pre-training will improve this final models' performance. OIV7 was annotated using manual annotators and extreme clicking, where annotators define the most extreme points of an object to form a bounding box - potentially resulting in less precise annotations than COCO - however the larger magnitude of the dataset will most likely outweigh this loss in precision [11].

Leveraging pre-trained weights allows us to benefit from the model's pre-established feature extraction abilities, significantly reducing the need for extensive training from scratch and ensuring faster convergence. This is due to potentially more effective starting weights for our task, present after pre-training. When pre-training, weights are optimized for specific features seen in the datasets used for pre-training. This is an optimized starting point for future feature activations, as they are more relevant from the beginning when training, resulting in faster convergence. When training from scratch weights are randomly initialized, usually from a specified distribution, thus more adjustments to weights are required during back propagation to generalize to the desired dataset. As such, pre-trained models usually generalize better after final training [6].

We trained the YOLOv8 model with the training set of Kornilov's dataset [10] and the following parameters:

- **Batch Size:** The training was conducted with a batch size of 25 and a total of 5996 images in the training set. This batch size ensures that enough data is processed per update to capture significant patterns without straining the computational resources.
- **Epochs:** We trained the model for 100 epochs, allowing multiple iterations over the entire dataset to refine the model weights. This number of epochs was selected to balance between achieving thorough learning and preventing overfitting. This approach ensures the model performs well in real-world conditions.

All training processes were executed on a system equipped with NVIDIA® GeForce® RTX 3090 Ti Graphical Processing Unit, featuring 24 GB of VRAM.

4. Experiment

To examine the performance of our various model checkpoints, we used the 446 image test set found in Kornilov's dataset [10]. To assess the models' performance during testing, we needed to find a method appropriate to examine how well the inferred class instances matched the ground truth of each image in the dataset, when there multiple ground truth bounding boxes, and thus multiple model predictions throughout the image.

We followed the guidance given by Di Yuan et al.: in their Accurate Bounding-box Regression with Distance-IOU Loss for Visual Tracking paper [17]. We created a function the operates by calculating the Intersection over Union (IoU) for each predicted box against each ground truth bounding box in a given image. If the IoU for a ground truth box exceeds a threshold of 0.5 (majority crossover), the prediction is considered a true positive. Predictions not matching any ground labels were counted as False Positives, and ground truth labels without any matching predictions were counted as False Negatives. If multiple predictions meet the threshold, the prediction with the highest IoU that exceeded the threshold was selected as the match.

The following equations detail how IoU was calculated in our process. First, corners of each bounding box are calculated, by converting the width, height, and center coordinates, to the top-left and bottom-right corners of each box.

$$x_{\text{min}} = x - \frac{w}{2}, \quad y_{\text{min}} = y - \frac{h}{2}$$

$$x_{\text{max}} = x + \frac{w}{2}, \quad y_{\text{max}} = y + \frac{h}{2}$$

Then, the intersection coordinates are calculated, otherwise known as the coordinates of the overlapping area between two boxes.

$$x_{\text{left}} = \max(x_{1_{\text{min}}}, x_{2_{\text{min}}}), \quad y_{\text{top}} = \max(y_{1_{\text{min}}}, y_{2_{\text{min}}})$$

$$x_{\text{right}} = \min(x_{1_{\text{max}}}, x_{2_{\text{max}}}), \quad y_{\text{bottom}} = \min(y_{1_{\text{max}}}, y_{2_{\text{max}}})$$

Area of Intersection is calculated by computing the area of the intersecting rectangle found above.

$$\text{Area of Intersection} = (x_{\text{right}} - x_{\text{left}}) \times (y_{\text{bottom}} - y_{\text{top}})$$

The area of the two intersecting boxes are then calculated when there is an intersection.

$$\text{Area of Box}_1 = (x_{1_{\text{max}}} - x_{1_{\text{min}}}) \times (y_{1_{\text{max}}} - y_{1_{\text{min}}})$$

$$\text{Area of Box}_2 = (x_{2_{\text{max}}} - x_{2_{\text{min}}}) \times (y_{2_{\text{max}}} - y_{2_{\text{min}}})$$

Area of Union is calculated by adding the area of both boxes and subtracting the area of intersection from the sum.

$$\text{Area of Union} = \text{Area of Box}_1 + \text{Area of Box}_2 - \text{Area of Intersection}$$

Finally, the IoU is calculated by dividing the area of intersection by the area of union.

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

The final output displayed in Figure 3, is the average overall IoU for the dataset for each model, where all IoUs are summed and divided by the amount of images in the dataset.

$$\text{Overall IoU} = \frac{1}{N} \sum_{i=1}^{N} \text{IoU}_i$$

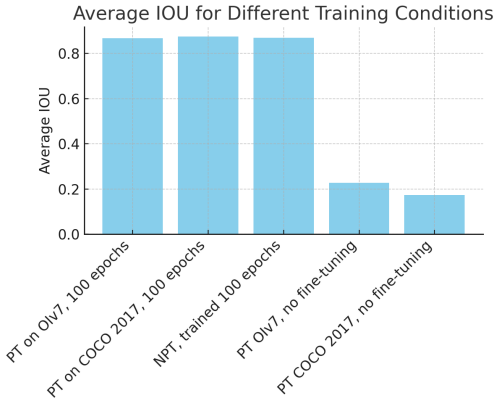

Figure 3. Average IoU across different training conditions

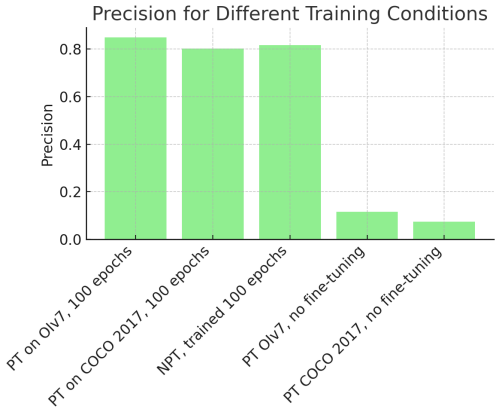

Figure 4. Precision across different training conditions

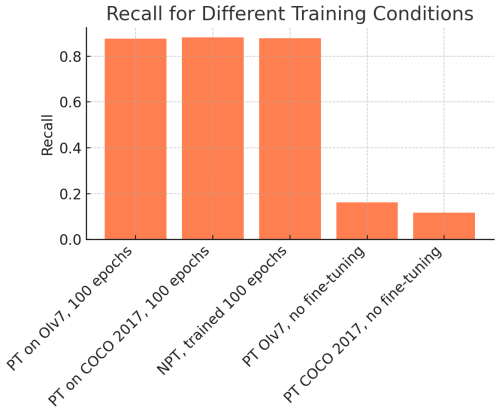

Figure 5. Recall across different training conditions

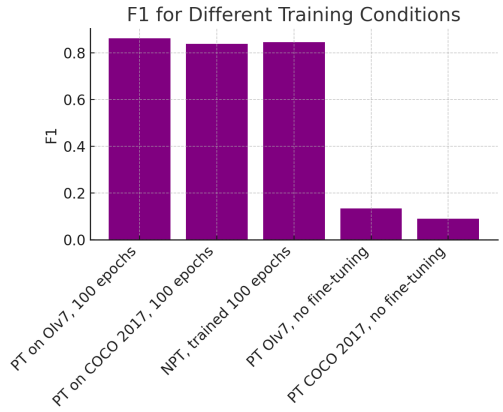

Figure 6. F1 Score across different training conditions

As visible in Figures 3-6, all models trained or fine-tuned on the dataset performed significantly better than the pre-trained models without fine-tuning. The models show similar performance in regards to the above metrics, potentially showing that a larger version of YOLOv8 should have been chosen with more parameters to potentially differentiate between the various model types. Larger models have increased capacity for learning complex features, or may generalize better, at the risk of over fitting [8]. This would allow us to potentially notice a bigger performance difference between models, with potentially for even higher scoring on metrics such as the F1 Score. Ultimately, as visible in Appendix C Table 6, the fine-tuned model that was pre-trained on OpenImages v7 would be our choice for deployment in a robotics application. It had a high amount of true positives comparable to the other fine-tuned model or when training from scratch, while maintaining the lowest amount of false positives between different conditions. This may support our hypothesis of a 'cart' class in pre-training improving post fine-tuning performance. Additionally, this model had the highest IoU and F1 between our models despite only by a few percentage points, further supporting our decision.

5. Robotic Integration: A Review

The modern landscape of robotics offers unique opportunities and challenges. As technology progresses, the integration of computer vision and robotics in pragmatic applications becomes more feasible and valuable. We explore such integration in the context of retail environments, especially focusing on shopping cart management, and propose a framework to simplify this process.

While a popular concept among modern robotics research is self-driving shopping carts [4], realizing this idea requires a complete overhaul of the ongoing trolley system in retail environments, resulting in high financial costs.

Instead, we propose a framework with a surveillance system that adopts YOLOv8 to perform bounding box detection on shopping carts in-store and in the vicinity. A stationary agent would then determine if the cart is unattended; if yes, it would signal an agent equipped with robotic arms to retrieve the cart. Conversely, a single, stationary agent with a mounted camera system could execute all the tasks above.

5.1. Unattended Object Recognition

Identifying whether a shopping cart has been unattended is crucial: the system should restrain from snatching away a cart currently in use. The feasibility of unattended object recognition was demonstrated by Jadhav and Momin [9] using the following methods (the points are directly quoted):

- **Foreground Blob Extraction:** Key features such as the location, height, and width of each blob are extracted, critical for subsequent object identification tasks.
- **Object Classification:** Objects are classified into three categories: stationary, removed, and moving, based on the lengths of their foregrounds.

Long Foreground	Short Foreground	Object Type
1	0	Stationary Object
0	1	Removed Object
1	1	Moving Object

Table 3. Object Classification with Two Foregrounds [9]

- **Object Identification and Responsive Actions:** If a static object is recognized as a person, it remains classified as such. If ambiguity exists, such as a person appearing stationary for long periods, the decision is deferred for manual review. Conversely, if an object is left unattended, the system takes note of this instance. This approach to object classification complements our focus on refining object recognition processes in environments like shopping centers.

While the specific techniques may need to be adjusted in the context of unattended cart identification, Jadhav and Momin's methodology [9] serves as a foundational building block for our task.

5.2. Navigation

Mobile navigation poses another foundational challenge. As robots are deployed in retail settings, they must not only navigate through fixed structures but also adapt to transient changes such as moving people, temporary displays, and varying crowd densities. Our exploration of the agent's navigation is two-fold, both in terms of global and local navigation.

5.2.1 Global Navigation

In global navigation, the agent has access to prior knowledge of the environment [14]. Global navigation refers to the robot's ability to orient itself and plan routes within the larger environment. The most prominent methods developed for global navigation are the Voronoi Graph, Dijkstra Algorithm, and Visibility Graph [14].

5.2.2 Local Navigation

In local navigation, the robot can control its movement and direction autonomously with equipped sensors like ultrasonic, infrared, and camera sensors [14]. Common algorithms to solve the local navigation problem are Fuzzy Logic, Neural Network, Genetic Algorithm, and Simulated Annealing [14].

Fuzzy logic plays a critical role in enhancing the capabilities of local navigation systems in complex and dynamic environments such as retail settings [5]. Unlike traditional binary logic systems that represent truth values in 0-1, fuzzy logic introduces a spectrum of possibilities and enables robots to make decisions in ambiguous scenarios with hidden information [5].

5.3. Robotic Arm Grip

With the unattended object identified and navigation enabled, the mobile agent is ready to retrieve the shopping cart. A core component in object retrieval is determining the gripping method of the agent's arm, which involves the type of grip system, the gripping strength, and the adaptability to various shapes and sizes of the object to be retrieved. Determining an appropriate gripper design is crucial for ensuring efficient object retrieval by the mobile agent. Typically, the design type is a product of the following parameters: size, shape, and material [7]. We determine the following parameters for the handle of a general shopping cart:

Size	31.5 mm to 80 mm
Shape	Circular
Material	Rocks / Electronics
Type	Daily Objects

Table 4. Shopping Cart Handle Parameters

Given these conditions, the most fitting category of gripper system for a shopping cart is Completely constrained-rigid link, with Under-constrained-rigid link as a close second [7]. Recommended structures of the gripper system for these two categories are linear actuator, rotary actuator, pneumatic actuator, cable-driven, and electromagnet [7].

6. Framework

We propose the following framework/pipeline (Figure 7) that integrates the fine-tuned YOLOv8 model, unattended object recognition, navigation, and robotic arm cart retrieval.

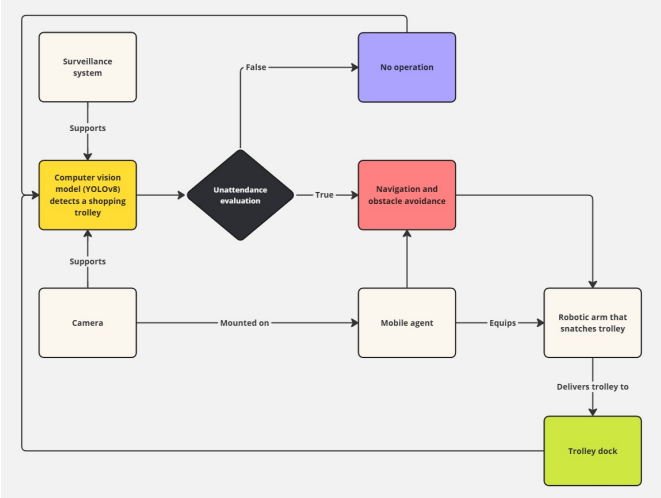

Figure 7. A Comprehensive Framework

This framework describes a system that utilizes a computer vision model (YOLOv8) to detect unattended shopping trolleys in a retail environment.

1. **Surveillance System:** A camera feeds live images to the system.
2. **Computer Vision Detection:** These images are processed by the YOLOv8 model to detect shopping trolleys.
3. **Unattendance Evaluation:** If a trolley is detected as unattended, the system activates further operations. If not, no action is taken.
4. **Navigation and Obstacle Avoidance:** Once an unattended trolley is confirmed, a mobile agent (likely a robot) equipped with navigation and obstacle avoidance capabilities is activated.
5. **Trolley Retrieval:** The mobile agent uses a robotic arm to physically retrieve the trolley.
6. **Trolley Docking:** Finally, the mobile agent delivers the trolley to a designated dock.

This automated process aims to efficiently manage shopping trolley retrieval, reducing labor and improving store operations.

7. Conclusion

We have developed and validated a comprehensive framework that employs the advanced capabilities of the YOLOv8 model for the automated retrieval of shopping carts in retail environments.

Our experimental results, derived from training and testing the robust dataset by Kornilov [10], demonstrate that the fine-tuned YOLOv8 (especially with un-pretrained weights) performs exceptionally well in detecting shopping carts under diverse conditions, achieving high values in IoU, precision, recall, and F1 metrics. This underscores the effectiveness of leveraging deep neural networks for real-world applications.

Moreover, the proposed robotic integration framework, which includes components for unattended object registration, sophisticated navigation systems, and a robust gripping mechanism, paves the way for practical deployment in commercial environments. This system reduces the dependence on human labor by automating manual tasks.

Future work will focus on refining the detection algorithms to handle more complex scenarios, such as varying light conditions and crowded scenes. Additionally, exploring the integration of artificial intelligence (AI)-driven decision-making processes can further enhance the system's autonomy and efficiency.

In conclusion, this study not only contributes to the fields of computer vision and robotics but also presents a viable solution to a common logistical problem faced by retailers globally, demonstrating the transformative potential of AI in industry.

Acknowledgements

Our heartfelt thanks go out to Dr. Chenliang Xu, Susan Liang, Luchuan Song, and Pinxin Liu for their guidance on this paper. Susan Liang suggested we look into the challenging yet meaningful frontiers of robotics, which significantly spearheaded the direction of our research.

References

[1] Coco dataset. Ultralytics Documentation, 2024. Available online: https://docs.ultralytics.com/datasets/detect/coco/. 4

[2] Open images v7 dataset. Ultralytics Documentation, 2024. Available online: https://docs.ultralytics.com/datasets/detect/open-images-v7/#sample-data-and-annotations. 4

[3] PyTorch documentation. https://pytorch.org/docs/stable/index.html, 2024. Accessed: 2024-05-07. 3

[4] P. Devaki, Aakarsh Satish, Shivam Dixit, Snehal N. Urs, and Tejasvi Kashyap. Self-driving shopping cart to assist the visually impaired. In *Computational Vision and Bio-Inspired Computing*, pages 659–674, Singapore, 2023. Springer Nature Singapore. 6

[5] Y. Dote. Introduction to fuzzy logic. In *Proceedings of IECON '95 - 21st Annual Conference on IEEE Industrial Electronics*, pages 50–56 vol.1, 1995. 7

[6] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 201–208, Chia Laguna Resort, Sardinia, Italy, 2010. PMLR. 4

[7] Jaime Hernandez, Md Samiul Haque Sunny, Javier Sanjuan, Ivan Rulik, Md Ishrak Islam Zarif, Sheikh Iqbal Ahamed, Helal Uddin Ahmed, and Mohammad H Rahman. Current designs of robotic arm grippers: A comprehensive systematic review. *Robotics*, 12(1), 2023. 7

[8] Xia Hu, Lingyang Chu, Jian Pei, Weiqing Liu, and Jiang Bian. Model complexity of deep learning: A survey, 2021. 6

[9] Lakhan H. Jadhav and Bashirahamad F. Momin. Detection and identification of unattended/removed objects in video surveillance. In *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, pages 1770–1773, 2016. 1, 6

[10] Kirill Kornilov. shopping trolley dataset. https://universe.roboflow.com/kirill-kornilov-kn3yx/shopping-trolley-kn5tj, 2023. visited on 2024-05-06. 2, 4, 8, 9

[11] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. *CoRR*, abs/1811.00982, 2018. 4

[12] 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 *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 4

[13] Mark P. Melegrito, Alvin Sarraga Alon, Sammy V. Militante, Yolanda D. Austria, Myriam J. Polinar, and Maria Concepcion A. Mirabueno. Abandoned-cart-vision: Abandoned cart detection using a deep object detection approach in a shopping parking space. In *2021 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, pages 1–5, 2021. 1

[14] Dr. Anish Pandey. Mobile robot navigation and obstacle avoidance techniques: A review. *International Robotics Automation Journal*, 2:1–12, 2017. 7

[15] Anita Rácz, Dávid Bajusz, and Károly Héberger. Effect of dataset size and train/test split ratios in qsar/qspr multiclass classification. *Molecules*, 26(4), 2021. 2

[16] Udo Wagner, Claus Ebster, Ulrike Eske, and Wolfgang Weitzl. The influence of shopping carts on customer behavior in grocery stores. *Marketing: ZFP – Journal of Research and Management*, 36(3):165–175, 2014. 2

[17] Di Yuan, Xiu Shu, Nana Fan, Xiaojun Chang, Qiao Liu, and Zhenyu He. Accurate bounding-box regression with distance-iou loss for visual tracking. *Journal of Visual Communication and Image Representation*, 83:103428, 2022. 4

A. Appendix A

This appendix includes an illustration of our model's bounding box detection, where green boxes are the predictions and red boxes are the ground truth.

Figure 8. Bounding Box Detection

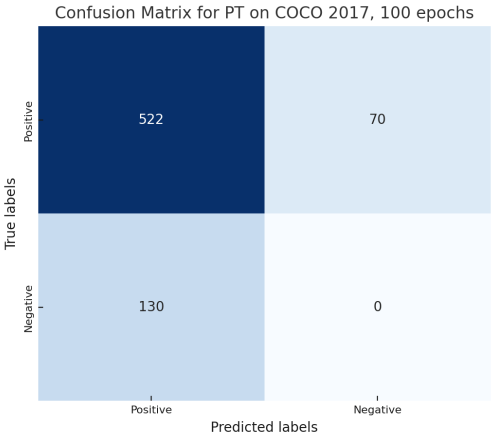

Figure 10. PT on COCO 2017, 100 epochs

B. Appendix B

This appendix includes confusion matrices of all our pre-trained and non-pretrained models, with and without dataset training [10], based on true positives, false positives, and false negatives.

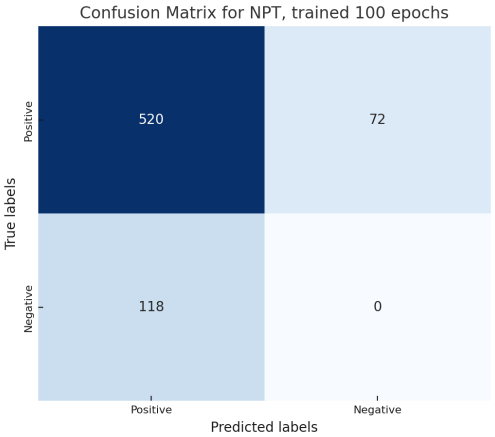

Figure 11. NPT, trained 100 epochs

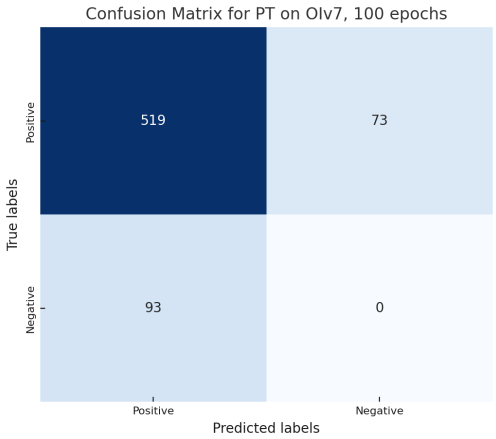

Figure 9. PT on OIv7, 100 epochs

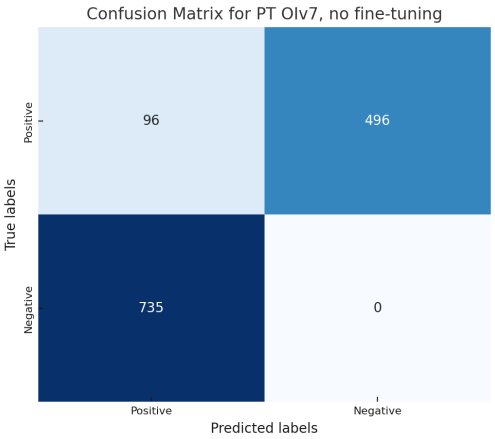

Figure 12. PT OIv7, no fine-tuning

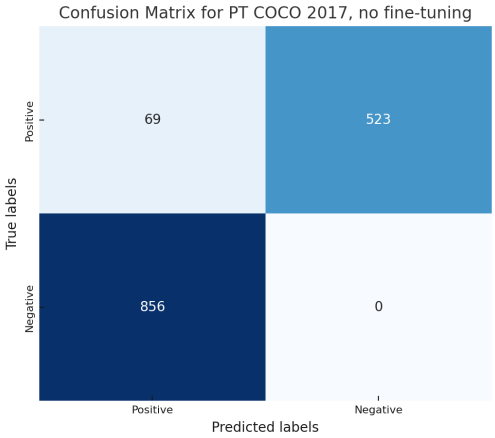

Figure 13. PT COCO 2017, no fine-tuning

C. Appendix C

This appendix includes raw values supporting the metrics graphs above and confusion matrices in appendix A.

Table 5. YOLOv8n: Performance Scores under Various Training Conditions

Model	**Average IoU**	**Precision**	**Recall**	**F1**
Pretrained on OIv7, 100 epochs	0.8671	0.8480	0.8767	0.8621
Pretrained on COCO 2017, 100 epochs	0.8739	0.8006	0.8818	0.8392
Pretrained OIv7, no fine-tuning	0.2268	0.1155	0.1622	0.1349
Pretrained COCO 2017, no fine-tuning	0.1725	0.0746	0.1166	0.0909
Non-pretrained, trained 100 epochs	0.8693	0.815	0.8784	0.8455

Table 6. YOLOv8n: Detection Counts under Various Training Conditions

Model	**True Positives**	**False Positives**	**False Negatives**
Pretrained on OIv7, 100 epochs	519	93	73
Pretrained on COCO 2017, 100 epochs	522	130	70
Pretrained OIv7, no fine-tuning	96	735	496
Pretrained COCO 2017, no fine-tuning	69	856	523
Non-pretrained, trained 100 epochs	520	118	72