Enhancing Retail Checkout through Video Inpainting, YOLOv8 Detection, and DeepSort Tracking

Arpita Vats
Santa Clara University
Santa Clara, CA, USA
avats@scu.edu

David C. Anastasiu
Santa Clara University
Santa Clara, CA, USA
danastasiu@scu.edu

Abstract

The retail industry has witnessed a remarkable upswing in the utilization of cutting-edge artificial intelligence and computer vision techniques. Among the prominent challenges in this domain is the development of an automated checkout system that can address the multifaceted issues that arise in real-world checkout scenarios, including object occlusion, motion blur, and similarity in scanned items. In this paper, we propose a sophisticated deep learning-based framework that can effectively recognize, localize, track, and count products as they traverse in front of a camera. Our approach, which we call RetailCounter, is founded on a detect-then-track paradigm, wherein we apply tracking on the bounding box of the detected objects. Furthermore, we have incorporated an automatic identification of the detection region of interest (ROI) and efficient removal of unwanted objects from the ROI. The performance of our proposed framework is competitive, as evidenced by our F1 score of 0.8177 and the fourth-place ranking that we achieved in track 4 of the 2023 AI City Challenge.

1. Introduction

In recent times, there has been a notable upswing in the interest surrounding the integration of artificial intelligence (AI) and computer vision (CV) methodologies in the retail industry, specifically in the realm of automatic checkout. The self-service trend has gained significant momentum in various facets of daily life. Track 4 of the 7th AI City Challenge [19], Multi-Class Product Counting and Recognition for Automated Retail Checkout, centers on devising an automated and accurate checkout system for retail stores, which poses substantial hurdles due to real-world factors like object occlusion, motion, item similarity during scanning, and the introduction of new seasonal product stock-keeping units (SKUs). Teams were furnished with a training data set comprising both genuine and synthetic images, totaling 116,500 item scans along with associated segmentation masks. The testing data set includes numerous video clips, with each clip featuring one or more complete scanning actions, performed by customers in a natural manner. Multiple customers participated, each with slightly varied scanning patterns, thereby adding to the complexity of the experiment. A shopping tray was provided for the placement of scanned items in the test scenario, while the camera was positioned overhead and aimed straight down at the checkout counter. In this work, we introduce an innovative framework, named RetailCounter[1], that incorporates video inpainting alongside detection, tracking, and selection modules, for accurately reporting the count of unique items observed in the video.

2. Related Works

The triumph of Amazon Go [28] has sparked a notable interest in self-checkouts at grocery stores, owing to its capacity to eliminate conventional checkout counters and substantially reduce checkout time. This accomplishment is realized through the application of computer vision and sensor fusion to ascertain purchased products and automatically charge customers via a mobile application upon exit, thus minimizing long queues. Panasonic has designed a walk-through self-checkout system that utilizes radio frequency identification (RFID) tags for object detection, a technology that has been previously used for security applications. The system's unique attribute lies in its cost-effectiveness, making it suitable for deployment in grocery stores. The research community has extensively explored the visual object recognition and classification realm, specifically in grocery stores, with a focus on product detection on shelves. Nonetheless, some studies, such as Multiple View Stereo (MVS) [14], have employed Circular Histogram of Gradients (CHoG), a feature descriptor, to extract low-level features from a query image and transmit them to a data server for recognition. Beyond tag reading,

[1] https://github.com/davidanastasiu/RetailCounter

the checkout automation capabilities can expand to detect items by analyzing their visual characteristics and overall appearance. Aquilina *et al*. [2] has pioneered a method for streamlining retail store checkout processes through the utilization of SCARA robots that feature a four-axis robotic system with machine vision. The system identifies items, packs them, and automatically generates a total bill after customers place items on the conveyor belt. Conversely, James *et al*. [12] suggests adopting conventional multi-class detectors that rely on convolutional neural networks to detect and recognize items from a single RGB image.

2.1. Object Detection

Detecting moving objects is a fundamental task in computer vision, and traditional methods such as background subtraction and feature extraction using SIFT or HOG descriptors were commonly used in the past. However, these methods suffered from a high error rate due to variations in object appearance and scale, as well as noise and lighting conditions. Convolutional neural networks (CNNs) have emerged as a dominant technology in object detection, outperforming traditional techniques [1].

Recent advances in object detection have led to the development of various frameworks, including one-stage and two-stage detectors as well as anchor-free methods. These frameworks rely on a data-driven approach that enables machines to automatically learn feature expressions, thereby eliminating the need for feature extraction. Two-stage detection architectures divide the detection process into the region proposal and classification stages, with models such as R-CNN [11], Fast R-CNN [10], and Faster R-CNN [22] being popular examples. In contrast, one-stage detectors use a single feed-forward fully convolutional network that directly provides bounding boxes and object classification. The most widely used models in this category are SSD [18] and YOLO [21], including variants such as YOLOv4 [4], YOLOv5 [33], Scaled-YOLOv4 [25], YOLOR [27], YOLOX [9], and YOLOv8 [13].

Recently, anchor-free detection models have become more prominent in the field of object detection. These models have evolved from the anchor-based methods discussed earlier. For instance, in the YOLOF detector, Chen *et al*. [5] revisited the concept of feature pyramid networks (FPN) used in one-stage detectors. They proposed a technique to use a single-level feature for detection instead of using FPN's divide-and-conquer optimization approach.

Another example is the Task-aligned One-stage Object Detection (TOOD) method introduced by Feng *et al*. [8]. This approach combines object localization and classification from attention maps into alignment metrics, which helps to balance learning task-specific and task-interactive features. They also proposed the Task Alignment Learning approach for anchor position optimization, which helped them surpass previous one-stage detectors in performance.

In addition, Ge *et al*. [9] created YOLOX, which is an anchor-free evolution of the YOLO series of detector models. They incorporated advanced detection techniques, such as a decoupled head and the leading label assignment strategy, SimOTA [31]. YOLOX significantly outperforms comparable models. YOLOv7 [26] surpasses all known object detectors in both speed (5 to 160 fps) and accuracy (56.8% mAP) among all previously known real-time object detectors capable of an inference speed of 30 or higher fps on an NVIDIA V100 GPU. Furthermore, the newly released YOLOv8 also seems to outperform all the previous versions of YOLO by a significant margin.

To train these models, large datasets such as those from the MS-COCO Detection Challenge [17], ImageNet Large Scale Visual Recognition Challenge [24], and PASCAL VOC Challenge [7] are commonly used, as they cover a vast number of object categories that facilitate good feature learning. After training, the models are often fine-tuned on smaller datasets for specific tasks. However, most object detection methods face a trade-off between accuracy and performance, the challenge being that of improving both simultaneously.

2.2. Object Tracking

Object tracking is a fundamental task in computer vision that involves locating and following objects over time in videos or image sequences. This task is essential for many real-world applications, such as surveillance, autonomous driving, robotics, and human-computer interaction. Object tracking faces many challenges, such as variations in object appearance, occlusion, camera motion, and changes in illumination and scale. To address these challenges, researchers have developed numerous object-tracking algorithms that employ various techniques, such as feature-based methods, deep learning, and probabilistic models. In recent years, object tracking has witnessed significant progress due to advances in machine learning and computer vision, leading to the development of more robust and accurate tracking algorithms.

Alex Bewley *et al*. [3] purposed a practical method named Simple Online Real-Time Tracker (SORT) for multiple object tracking that prioritizes efficient and real-time object association. The research highlights the importance of detection quality in determining tracking performance, where the use of different detectors can lead to an improvement of up to 18.9% in performance. Despite utilizing basic techniques like the Kalman filter and the Hungarian algorithm, the proposed method achieves an accuracy comparable to the latest online trackers. Additionally, the simplicity of this approach allows for a high update rate of 260 Hz, which is more than 20 times faster than other advanced trackers. The DeepSort [29] tracker is an extension of the

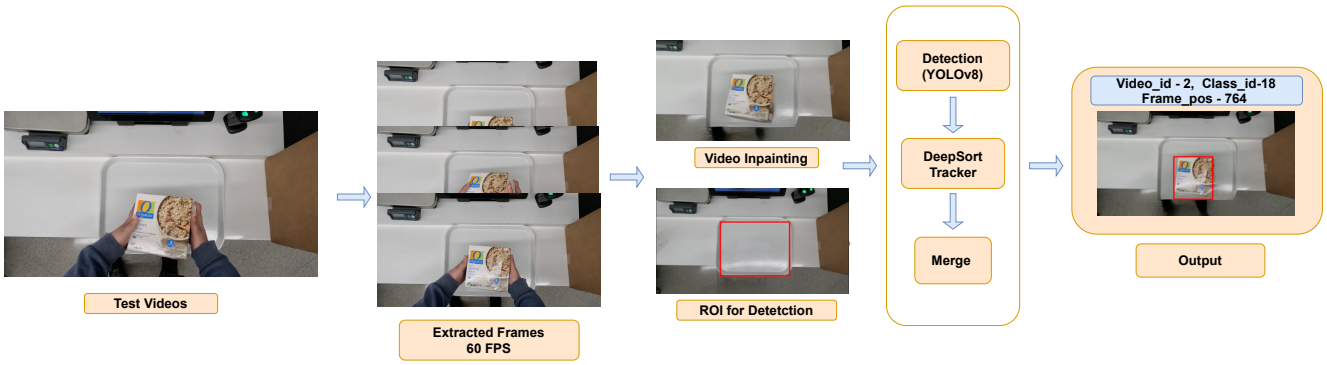

Figure 1. Framework for retail item detection and counting.

SORT tracker that incorporates a deep association metric based on image features.

Yifu Zhang *et al*. [32] proposed an advanced object-tracking algorithm that uses a deep neural network similar to DeepSort to achieve state-of-the-art tracking accuracy. This algorithm is designed to address the challenges posed by real-world object tracking scenarios, including occlusions, scale changes, and motion blur. ByteTracker has demonstrated remarkable performance on multiple benchmarks, outperforming other popular object trackers in terms of both accuracy and speed. By leveraging the power of deep learning, ByteTracker has the potential to advance the field of object tracking and outperforms many tracking algorithms.

2.3. Video Inpainting

Video inpainting is the process of filling in missing or corrupted parts of a video sequence with plausible content. This technique has many practical applications, such as restoring old or damaged films, removing unwanted objects or occlusions from surveillance videos, and even generating realistic content in special effects for film and video games. Video inpainting is a challenging task as it involves reconstructing spatial and temporal information simultaneously. To achieve this, various approaches have been proposed, including patch-based and optimization-based methods, as well as deep learning-based techniques that leverage the power of CNNs to learn spatial and temporal context from the surrounding frames. Despite recent advances in video inpainting, it remains an active area of research with many open challenges, including handling complex scenes, preserving temporal consistency, and dealing with large missing regions.

Zhang *et al*. [30] introduced a novel flow completion network that completes corrupted flows by exploiting relevant flow features within a local temporal window. With the completed flows, they propagated content across video frames and use the flow-guided transformer to synthesize the remaining corrupted regions. By separating transformers along temporal and spatial dimensions, they integrated the locally relevant completed flows to instruct spatial attention only. To further enhance efficiency, they introduced a window partition strategy for both spatial and temporal transformers. Additionally, they designed a flow-reweighting module to precisely control the impact of completed flows on each spatial transformer. Their approach also incorporates a dual-perspective spatial Multi-Head Self-Attention (MHSA) that integrates global tokens with window-based attention.

3. Methodology

Figure 1 depicts our framework. The proposed method is a multi-step approach that will be discussed in detail in the following sections. In our proposed system, frames extracted from test set A are utilized as input. The frames undergo preprocessing in the first step, which results in cropped and masked frames. The second step involves passing the processed frames into the detection network, which generates location bounding boxes. The frames that have location information are then fed into DeepSort and the classification network, producing tracks with category scores. The final step involves using a merge algorithm to fine-tune the object tracks and select the item output frame for each track.

3.1. Data Generation and Model Training

Our object detection model was developed using synthetic images generated from 3D-scanned object models and their corresponding segmentation masks. Owing to limitations on external dataset usage, we produced a background that resembled the tray color in the test video and incorporated Gaussian noise. To enrich the training dataset, each image featured up to three objects from distinct classes, as depicted in Figure 2. Moreover, we explored techniques for enlarging and enhancing the resolution of objects superimposed on the background image,

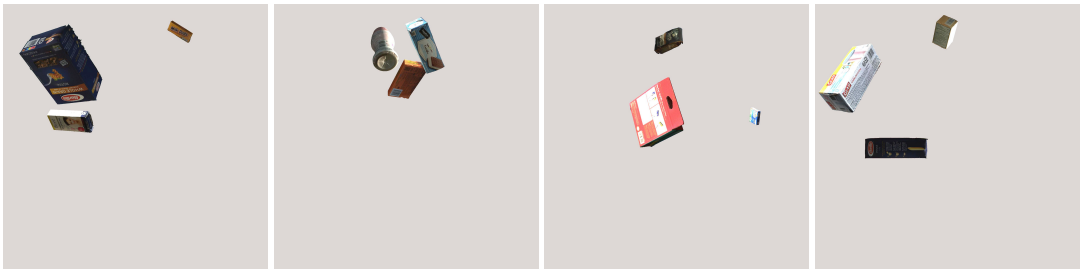

Figure 2. Generating our training dataset.

considering the original images' subpar quality. While the Super-Resolution Convolutional Neural Network [6] model did not yield improvements in image resolution, we successfully employed the Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network (SRGAN) [15] model to obtain high-resolution images, resulting in superior training image quality. In total, we generated 130,000 training images and 20,000 validation images. To train our model, we fine-tuned the pretrained weights of YOLOv8, which is currently acknowledged as the state-of-the-art model for object detection.

Figure 3. Real-time tracking of the region of interest.

3.2. Outlier Object Removal

The dataset used for our training was comprised of synthetic images of individual products that were embedded into ordinary images, as detailed in Section 3.1. During training, these products were isolated in the frame and placed in "free space" without any other objects in close proximity. However, during inference, the model sometimes detected the worker's hands or body as false positives, even when no product was present in the scene. To tackle this issue, we employed computer vision techniques to first detect human body parts, particularly hands, in the given frame and generate a mask to represent their location. This involved using keypoint detection or instance segmentation methods to estimate the position of the hands' semantic key points or identify the hands as objects, respectively. Subsequently, we applied Flow-Guided Video Inpainting (EFGVI) [16], which utilizes three trainable modules – flow completion, feature propagation, and content hallucination – to jointly optimize the inpainting process. Our proposed method yielded superior results both qualitatively and quantitatively, outperforming state-of-the-art techniques and demonstrating promising effectiveness, as evidenced in Figure 4. By utilizing video inpainting, we were able to significantly reduce the false positive detection rate.

3.3. Region of Interest Detection

Since the primary goal of the challenge was to identify objects solely above the "white tray", we decided to dynamically identify the bounding box of the tray as our ROI and limit detection and tracking within that region. This improved both detection accuracy and overall efficiency across our processing pipeline. The detection process was initiated by extracting the background image of each video using a Gaussian Mixture Model [23]. Foreground images were then separated by using background subtraction methods [20] for each frame of the video while incorporating the previous frames. Given that the video camera did not actively move in our scenario, we extracted ROI coordinates only at key frames, once every second. However, to avoid outlier ROI detections, at each key frame, we computed the ROI for the current frame along with the previous and following n frames, and selected the ROI with the median bounding box area within that list.

To extract the ROI in a given frame, the background image was converted to grayscale, and the Scharr operator, which is an enhanced version of the Sobel operator, was employed for edge detection. The Scharr operator was executed in both the x and y directions and then combined. The largest rectangle in the frames was used as the ROI (white tray), as depicted in Figure 3. We also tried using the flood fill algorithm on a differenced image to identify pixels with similar values to a given seed. In this case, the seed was located at the image center, but it could be set arbitrarily. In this way, all pixels that were connected until edges were detected, and they were labeled as the "tray". However, this method yielded worse overall results.

3.4. Detection and Tracking

After completing all the necessary preprocessing steps, such as video inpainting and ROI detection, we utilized the YOLOv8 detector for object detection, as depicted in Figure 5. This state-of-the-art detector takes the ROI-enhanced

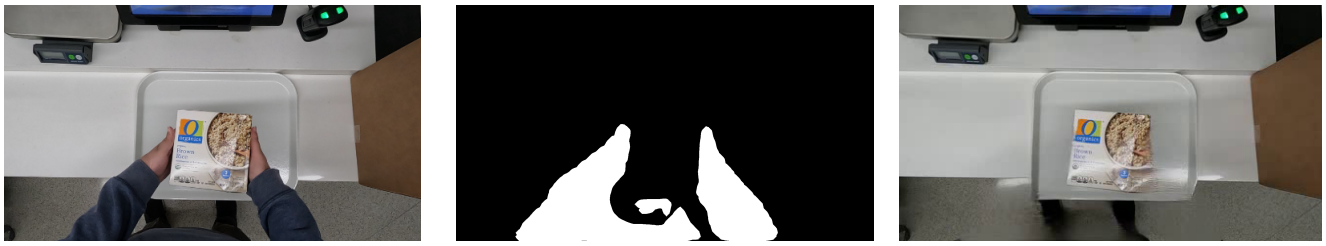

Figure 4. Video Inpainting. The image on the left displays the original image. The image in the center shows the image with a mask applied, while the image on the right displays the video after the mask has been replaced with background data (inpainted).

Figure 5. Object detection within the ROI.

image as input, which is resized to 640×640. To ensure the highest level of precision and quick inference, we implemented the DeepSort tracker. Our model was trained on a dataset of generated objects, with 116 output classes, and achieved an impressive 98.3% mean Average Precision (mAP) on the validation set during training. Notably, our post-processing of tracks takes advantage of both detection confidence and class confidence, in addition to the object position.

Our solution for individual product tracking involves the utilization of two online tracking algorithms, namely SORT and DeepSort. These algorithms are well-known for their effectiveness in object tracking and rely on the bounding rectangles of detections to track the objects of interest. The Kalman filter is employed in both algorithms to make predictions about the future position of each object. This prediction is then merged with the corresponding tracks, ensuring that each object is accurately tracked throughout the entire video sequence. The combination of these algorithms results in an approach that is both accurate and computationally efficient, making it ideal for real-world applications.

SORT and DeepSort are particularly well-suited for scenarios where objects are in close proximity or occluded, and they are designed to handle high-speed tracking. Additionally, the algorithms are able to handle changes in the size, orientation, and appearance of objects, making them robust to variations in lighting and background conditions. By leveraging these tracking algorithms, our solution is able to achieve high levels of accuracy in tracking individual products in a variety of scenarios. Overall, our approach offers a combination of high accuracy and computational efficiency, which is essential for real-world applications where both speed and accuracy are critical. This makes our solution a practical choice for a wide range of applications such as inventory management, object detection in autonomous driving, and surveillance systems.

3.5. Track Merging

Our chosen tracker provides our algorithm with a set of tracklets with different IDs for each. In each tracklet, for each frame, we maintain the bounding box coordinates of the object that was detected, its class assignment, and the confidence of that detection. We then assign a class label to each tracklet as the class with the highest mean confidence within all detections of the tracklet. We then analyze whether some tracklets are extensions of a single track that may have resulted through some occlusion and merge any such tracklets. For any two tracklets with the same assigned class label, we compare the two tracklets in ascending order and merge them if the x and y coordinates of the center points of the last frame in the first tracklet and the first frame in the second are within K pixels of each other.

3.6. Choosing Output Frames

Once tracks have been determined for each object that passes through the ROI, we determine the frame ID that should be reported for each object as the frame with the most centered object bounding box within the track. To find this frame, we first compute the center points of all detection bounding boxes in the track, and then compute the Euclidean distance between those points and the center of their respective key frame ROI bounding boxes. Finally, we report the object detection at the frame with the minimum distance from its respective ROI center.

4. Experimental Evaluation

In the following section, we provide an extensive discussion about the implementation process of our project, including detailed information on the execution environment we used. We also include a thorough analysis of the results we obtained from our experiments. By doing so, we aim to provide a comprehensive understanding of the various technical aspects of our project and offer insights into the effectiveness of our approach. Additionally, we will delve into the challenges we faced during the implementation process and how we overcame them, providing a valuable perspective on the practical applications of our methodology. Through this detailed discussion, we hope to provide a deeper understanding of our project and contribute to the ongoing research in this field.

4.1. Evaluation Environment

Our experiments were carried out on a system running Ubuntu Linux 18.04 and equipped with an 8-core Intel(R) Core(TM) i9-7920X CPU operating at a clock speed of 3.60 GHz, 64 GB RAM, and 2 NVIDIA TITAN Xp 12G GPUs. The system's hardware configuration provided us with sufficient computational power to execute the experiments in a reasonable amount of time.

Although our approach uses both CPU and GPU resources, it is important to note that we only utilized one GPU for our experiments. The CPU-based processing was implemented using multi-threaded techniques to ensure efficient usage of the available computational resources. However, our method's primary processing unit was the GPU, which was responsible for the bulk of the computation. Our experiments were designed to be scalable, which means that they can be executed on a range of hardware configurations, including systems with fewer GPUs or CPU cores.

4.2. Implementation Details

During inference, we focused on detecting and tracking an object that appears within the ROI in videos. We achieved this by extracting only the pixels in the ROI area, with a small buffer, which we then resized to 640×640 to provide as input to our detector. The ROI was detected at the start of each second, choosing the ROI with the median area within a window of $n = 10$ frames before and after the ROI key frame. Since the video was encoded with 60 frames per second, given our choice of $n = 10$, our method avoided ROI calculations for 65% of the frames.

The images used to train our detection model were the ones we created, as described in Section 3.1. We found the SRGAN model to be most effective at enhancing the training object images that were randomly composed in our training set. We initially fine-tuned a pretrained medium YOLOX model for 80 epochs, which achieved a respectable mean average precision (mAP) of 96.8%. Our final detection model was fine-tuned using the medium YOLOv8 model for 75 epochs and achieved a higher mAP of 98.3%. All other network settings were left unchanged from the original configuration, as described on the YOLOv8 website.

For tracking, we experimented with two different algorithms, SORT and DeepSort. In both cases, we increased the period for which the track could be broken to 30 frames, enabling more robust and reliable tracking capabilities. We experimented with different lengths of frames, but we were able to get the best result with 30 frames. Finally, we set K, the maximum number of pixels between the center of the last frame in the previous track and the center of the first frame in the next track in our merge algorithm to 100.

Table 1. Ablation Study

Detector	ROI	Tracker	F1 score
YOLOX	Mean Frame	SORT	0.590
YOLOX	Windowed ROI Median	SORT	0.651
YOLOX	Mean Frame	Deep SORT	0.681
YOLOX	Windowed ROI Median	Deep SORT	0.701
YOLOv8	Mean Frame	SORT	0.628
YOLOv8	Windowed ROI Median	SORT	0.737
YOLOv8	Mean Frame	Deep SORT	0.768
YOLOv8	**Windowed ROI Median**	**Deep SORT**	**0.817**

4.3. Ablation Study

We investigated the effectiveness of various stages of our framework in achieving desirable outcomes and show the results in Table 1. Specifically, we aimed to identify the individual contributions of each stage in the pipeline. To this end, we used test set A to test our pipeline. We found that the YOLOv8 detector model paired with the DeepSort tracking method yielded the best results, achieving an F1 score of 0.817. This demonstrates the importance of carefully selecting the detector model and tracking method to optimize performance. In addition, we also explored different approaches to ROI detection. We found that taking the ROI of each key frame improved the performance of the pipeline compared to using a single ROI for the entire video. This highlights the importance of adapting the ROI detection approach to the specific conditions of the video surveillance task at hand.

4.4. Results and Discussion

Our model's performance in the competition is a testament to the rigor of our experimentation process and the robustness of our object detection pipeline. As shown in Table 2, we achieved an F1 score of 0.8177, placing us in the $4th$ position out of 10 teams that submitted results to the Track 4 public leaderboard.

Table 2. Top-10 of the AIC 2023 Track 4 Leader Board

Rank	Team Name	F1 score
1	SKKU Automation Lab	0.9792
2	BUPT_MCPRL	0.9787
3	Zebras	0.8254
4	**SCU_Anastasiu**	**0.8177**
5	Fujitsu R&D Center	0.7684
6	Centific	0.6571
7	dtb2023	0.4757
8	Fu	0.4215
9	HCMIU-CVIP	0.3837
10	UTE_AI	0.3441

To obtain this positive performance, we conducted extensive experimentation with different YOLO models, including YOLOX and YOLOv8, with the aim of optimizing object detection performance. After a thorough analysis of the results, we concluded that YOLOv8 outperformed the other models. We also explored the impact of different image sizes on the results, and while we found that variants produced similar and almost identical results, we concluded that the input image size may not be a critical factor in optimizing performance.

In addition to the YOLO models, we also evaluated the performance of the SORT and DeepSort trackers. While both trackers exhibited excellent performance, we found that the latter exhibited better stability and achieved an F1 score of 0.8167. The result is impressive considering the image resolution we used was only 640×640 and might be able to be improved by using higher resolution inputs in the tracking.

5. Conclusion

Our submission to the 2023 AI City Challenge Track 4, Multi-Class Product Counting and Recognition for Automated Retail Checkout, presents a comprehensive framework that is able to accurately detect and count individual items that pass through an automatically detected region of interest. Our approach employs video inpainting to enhance detection results and minimize the occurrence of false positives. By automatically detecting the region of interest and segmenting humans, followed by their removal from the scene, our approach delivers competitive results that are comparable to those of the top 3 teams. We achieved these results by utilizing the YOLOv8 detection network, which runs in real-time, and trackers that solely rely on bounding boxes.

Our method's performance achieved the fourth position on the Public leaderboard, with an F1 score of 0.8177. Future work includes expanding our approach to accommodate more complex scenarios, exploring the use of deep learning to enhance object detection, and incorporating contextual information to further improve the system's accuracy. Furthermore, we aim to investigate the use of advanced tracking methods, such as particle filters and Kalman filters, to achieve even better tracking accuracy in challenging scenarios.

References

[1] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, 2017. 2

[2] Yesenia Aquilina and Michael A. Saliba. An automated supermarket checkout system utilizing a scara robot: preliminary prototype development. *Procedia Manufacturing*, 2019. 2

[3] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2016. 2

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 2

[5] Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature, 2021. 2

[6] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network, 2016. 4

[7] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010. 2

[8] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R. Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection, 2021. 2

[9] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021, 2021. 2

[10] Ross Girshick. Fast r-cnn, 2015. 2

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014. 2

[12] Namitha James and Student. Automated checkout for stores: A computer vision approach. *Revista Gestão Inovação e Tecnologias*, Vol. 11 No. 3 (2021):Vol. 11 No. 3 (2021), 07 2021. 2

[13] Glenn Jocher. Yolov8. Accessed on April 05, 2023. 2

[14] Guoyu Lan, Heng Qi, Keqiu Li, Kai Lin, Wenyu Qu, and Zhiyang Li. A framework of mobile visual search based on the weighted matching of dominant descriptor. In *MM '14: Proceedings of the 22nd ACM international conference on Multimedia*, New York, NY, USA, 2014. Association for Computing Machinery. 1

[15] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017. 4

[16] Zhen Li, Cheng-Ze Lu, Jianhua Qin, Chun-Le Guo, and Ming-Ming Cheng. Towards an end-to-end framework for flow-guided video inpainting, 2022. 4

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. 2

[18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 21–37, Cham, 2016. Springer International Publishing. 2

[19] M. Naphade, S. Wang, D. C. Anastasiu, Z. Tang, M. Chang, Y. Yao, L. Zheng, M. Shaiqur Rahman, A. Venkatachalapathy, A. Sharma, Q. Feng, V. Ablavsky, S. Sclaroff, P. Chakraborty, A. Li, S. Li, and R. Chellappa. The 6th ai city challenge. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2022. 1

[20] OpenCv. Opencv. Accessed on April 05, 2023. 4

[21] Redmon. Yolo9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017. 2

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 2

[23] Douglas A. Reynolds. Gaussian mixture models. In *Encyclopedia of Biometrics*, 2009. 4

[24] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. 2

[25] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4: Scaling cross stage partial network, 2021. 2

[26] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022. 2

[27] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks, 2021. 2

[28] Kirti Wankhede, Bharati Wukkadada, and Vidhya Nadar. Just walk-out technology and its challenges: A case of amazon go. In *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*, pages 254–257, 2018. 1

[29] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017. 2

[30] Kaidong Zhang, Jingjing Fu, and Dong Liu. Flow-guided transformer for video inpainting, 2022. 3

[31] Tianxiao Zhang, Bo Luo, Ajay Sharda, and Guanghui Wang. Dynamic label assignment for object detection by combining predicted ious and anchor ious. *Journal of Imaging*, 8(7), 2022. 2

[32] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box, 2022. 3

[33] Xingkui Zhu, Shuchang Lyu, Xu Wang, and Qi Zhao. Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios, 2021. 2