Adaptive RoI with pretrained models for Automated Retail Checkout

Anudeep Dhonde Prabhudev Guntur Vinitha Palani
{anudeep.dhonde, prabhudev.guntur, vinitha.palani}@centific.com,
Centific Global Solutions Inc.

Abstract

In this paper, we present a solution for automatic checkout in a retail store as a part of AI City Challenge 2023 Track 4. We propose a methodology which involves usage of pretrained Yolov5 models to detect person and media pipe models to detect hands of the person. This information is utilized to compute the Region of Interest (RoI) which is adaptive in nature. Afterwards, a custom trained object detection model is used detect products in the frame. We then use a tracker to track the products across video frames to avoid duplicated counting. The method is evaluated on the AI City challenge 2023 – Track 4 and gets the F1 score 0.6571 on the test A set, which places us on 6th place on the public leader board. The code is made public and available on GitHub.[1]

1. Introduction

Self-checkout is becoming more prevalent in retail stores in the recent times. The cause of this due to rapid advances in AI to identify products accurately. Customers have become accustomed to such systems. There are multiple benefits for both retailers and customers in using self-checkout systems. Retailers can save labor, and thereby reduce operating costs; Customers can do faster checkout thereby saving time and enhancing customer experience.

This year's AI city challenge 2023 contains a track named Multiclass Product Counting and Recognition for Automated Retail Checkout. This objective of this track is to build a system to automatically recognize products present in a camera view during retail checkout. The intended solution has to report all the products – meaning product name (ID) and the frame number at which it was present in the Regional of Interest (RoI) which is the white tray under the product as shown in Fig. 1. The tray position is not defined and must be localized automatically.

This data set contains a total of 116,500 synthetic images which consists of products across 116 classes. Both product images and segmentation masks are shared as part of the dataset. Few sample images and segmentation masks are shown in Fig. 1. Our proposed solution comprises of multiple sub-tasks such as RoI Localization, Object Detection, Object Tracking and tracks post processing. The individual tasks are described below in Section. 3

[1]https://github.com/centific-aicoe/AICity-Prod-Counting-2023

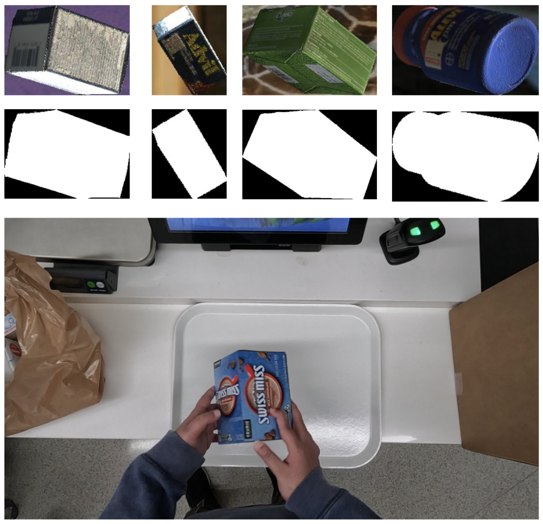
Figure 1. Top two rows in the figure show the images and segmentation masks. Bottom row shows the scene from test video where product is over the white tray

2. Related Work

2.1. Object Detection

Object detection refers to the task of localizing and classifying object in an image. Convolutions neural network-based architectures are generally used for object detection. Object detectors mainly contain two variants of neural network architectures two-stage and single-stage. Faster-RCNN [6] and Mask-RCNN [2] stands for the two-stage approach, while RetinaNet [4] and YOLO [5] are repre-

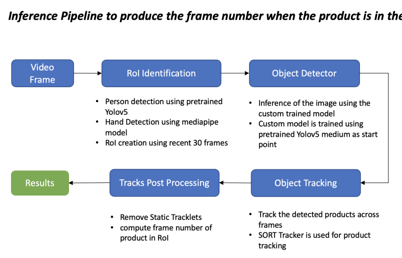
Inference Pipeline to produce the frame number when the product is in the RoI
Video Frame
RoI Identification
Object Detector
Person detection using pretrained Yolov5
Hand Detection using mediapipe model
RoI creation using recent 30 frames
Inference of the image using the custom trained model
Custom model is trained using pretrained Yolov5 medium as start point
Results
Tracks Post Processing
Object Tracking
Remove Static Tracklets
compute frame number of product in RoI
Track the detected products across frames
SORT Tracker is used for product tracking

Figure 2. Our proposed inference pipeline. Every frame is processed by RoI identification module. The frame is then fed to object detector and tracker to detect and track products. Tracklets are then processed to arrive at final results

sentatives for the one-stage approach. Single shot detectors are the most popular family of architectures used for multi class object detection. The famous single shot detectors are YOLO (You only look once). These models detect all the objects in an image with a single pass over the neural network unlike other models where image traverse multiple times. In our proposed method, we use Yolov5 pretrained model and custom trained a model to identify product classes.

2.2. Multiple Object Tracking

Simple Online Real-time Tracker (SORT) from Bewley et al [1] is a visual multi object tracking framework which is based on data association and state estimation techniques using Kalman Filter. It estimates object identities on the fly by using detections from past and current frames. It also supports some features like object re-entry in a predefined time window and partial occlusion, number of detections to be available before a tracklet is initialized

3. Proposed Method

Our proposed method consists of four major steps as shown in Fig. 2

1. RoI (Region of Interest) Identification
2. Custom Training for Object Detection
3. Object Tracking
4. Tracks Post Processing

3.1. RoI Identification

RoI identification refers to the process of approximately locating the tray in the frame. Any changes to the camera movement needs to be re-adjust the RoI accordingly.

To address such scenarios, we propose a dynamic RoI methodology, where in we consider the most recent 30 frames (0.5 seconds) to determine the RoI.

Our RoI consists of identifying three white lines, one which is on the left side of the tray, one on the right side of the tray and one just below the tray as shown in Fig. 3

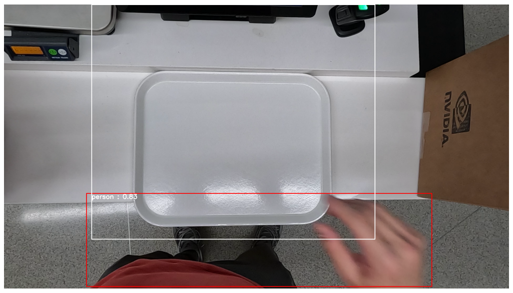

Figure 3. Three white lines define our RoI. The one on the left is called **left_vertical**, one on right is called **right_vertical** and the one which is horizontal is called **lower_horizontal**

To identify these co-ordinates, we use two pretrained models:

- Yolov5 [3] for person detection
- Mediapipe [7] to detect the hands of the person

For every frame in our inference video, we check two criteria as detailed below

1. person is detected with a confidence of more than 0.8 using yolov5 pretrained model as shown in Fig. 3
2. two hands are detected by mediapipe [7]

if both the above criteria are satisfied, then we capture the information of bounding box of the person **(x_person_center, y_person_center, person_box_width)**.

This information is computed for every frame and maintained in a queue of size 30 which holds information from only recent 30 frames with both criteria satisfied.

The RoI lines would be computed using the formula shown below by using values across recent 30 frames.

- **left_vertical_line_coord** = mean(x_person_center) - (mean(person_box_width)/ 2)
- **right_vertical_line_coord** = mean(x_person_center) + (mean(person_box_width)/ 2)
- **lower_horizontal_line_coord** = mean(y_person_center)

As per our methodology, we are using proxy of bounding box of person (with both hands detected) as an approximate estimator of RoI using the formula mentioned above. The RoI can readjust itself when anything changes in the camera view because person would move accordingly. We have tried increasing the size of the queue to a bigger size, but then RoI re-adjustment takes more time

3.2. Custom Training for Object Detection

As part of AI City challenge, we received 116,500 product images and segmentation masks. Some of the sample images are shown in Fig. 1

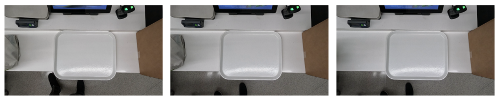

Figure 4. the three background images which we used to create object detection dataset

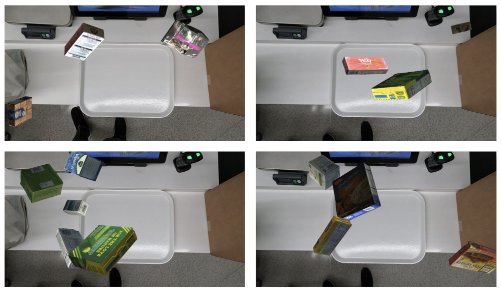

Figure 5. Sample images of the object detection dataset

Instead of directly using segmentation masks, we created a dataset ready to train for object detection. In order to prepare the synthetic dataset for object detection, we need two things

- Background
- Foreground (which contains products)

For background, we randomly selected few frames from the inference video test_1.mp4. We selected 3 background images as shown in Fig. 4

In order to create synthetic detection data, we iterated on the steps below till all the segmentation dataset is exhausted

1. Randomly select a background image from our 3 backgrounds
2. Randomly select up to 5 products and their segmentation masks (from 116,500 set of images)
3. For each product do the following
    - Randomly generate x,y co-ordinates for the product
    - Paste them on the background at those location (x,y) subject to conditions of minimum 75% visibility in the frame
    - If above step fails, go to the first step a and re-generate co-ordinates
    - Create a text file with details of product location as per yolo format

The above exercise was done till we exhausted all the 116,500 images in the dataset. By running the synthetic data generation for object detection process, we generated 33,349 images. A few sample images are shown in Fig. 5

This object detection dataset with 33,349 images trained using yolov5 [3] medium model for 70 epochs. In inference

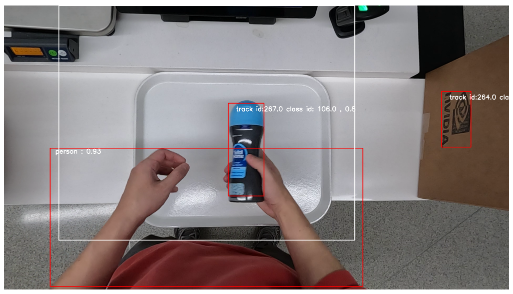

Figure 6. Sample inference image using the trained object detection model

mode, the trained model takes in an input image and detects products in the image.

An example of object detection inference is available in Fig. 6

3.3. Object Tracking

Along with object detection for every frame, we track the products across frames and assign a unique track id. As a part of our solution, we have used SORT tracker with parameters of max_age of 100 frames, min_hits as 3, and iou threshold of 0.3 We have used SORT tracker and modified to suit our use case of handling multiple classes

3.4. Tracks Post Processing

The last step in our pipeline is post processing of the tracks, by this step we have information of all tracks and their respective product class. We have implemented one post processing method to identify static tracklets. This method takes as input all the detection bounding boxes of the respective tracklet and works as below

1. Compute mean of the center (x_center, y_center) of the product bounding boxes across various frames
2. Compute difference between mean and each value delta_x = (x_center – x_mean) and delta_y = (y_center - y_mean)
3. If all the delta_x and delta_y are less than or equal to 10, then the tracklet is static

Post removing the static tracklets, we are left with the remaining tracklets. For each of the remaining tracklet, we compute the frame number at which the product is in the RoI using the methodology below

1. Capture the first frame (f_start) when the product is in the RoI
2. Capture the last frame (f_end) when the product is in the RoI
3. Compute mean of f_start and f_end i.e., (f_start+f_end)/2 which is the frame_id to be used for submission

Rank	Team Name	Score
1	SKKU Automation Lab	0.9792
2	BUPT_MCPRL	0.9787
3	Zebras	0.8254
4	SCU Anastasiu Lab	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

Table 1. Public leaderboard of AI City Challenge 2023 Track 4

The inference pipeline was executed for all the test videos as part of submission and space delimited text file was created in the format as below ⟨video_id⟩ ⟨class_id⟩ ⟨frame_id⟩

4. Results

We tried various threshold of queue size for RoI localization. 0.5 seconds gave us the best results on the AI City challenge submission. Also, we experiment with SORT tracker with few parameters and best result came at max_age of 100 frames, min_hits as 3, and iou threshold of 0.3 Our best result is **0.6571 F1-Score**, so we placed **6th** in the public part of Track 4 challenge. The public leaderboard can be seen in Tab 1

5. Conclusion

We participated in AI City Challenge 2023, Track 4 called MultiClass Product Counting & Recognition for Automated Retail Checkout. We proposed an approach based on using pretrained yolov5 models for person detection and leveraging Mediapipe for hand detection. These two components were used to propose a dynamic RoI identification. Our solution scores at F1 = 0.6571 on the testA of the AI City challenge 2023 Track 4.

The method can further be improved by adding few augmentation techniques in the custom object detection training module. Since we are using reconstructed images of products, we can augment them by adding more lighting, brightness etc., while creation of object detection dataset

6. Acknowledgment

We would like to thank our colleagues Umar Abdullah, Bandi Jagadeeshwar Reddy and others at Centific for valuable discussion and help in reviewing the paper.

References

[1] 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)*, pages 3464–3468. IEEE, 2016. 2

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 1

[3] Glenn Jocher. YOLOv5 by Ultralytics, 5 2020. 2, 3

[4] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 1

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 1

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 1

[7] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, and Matthias Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020. 2