# PersonGONE: Image Inpainting for Automated Checkout Solution

Vojtěch Bartl Jakub Špaňhel Adam Herout
GRAPH@FIT, Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno, Czech Republic
{ibartl,ispanhel,herout}@fit.vutbr.cz

## Abstract

*In this paper, we present a solution for automatic checkout in a retail store as a part of* AI City Challenge 2022. *We propose a novel approach that uses the "removal" of unwanted objects — in this case, body parts of operating staff, which are localized and further removed from video by an image inpainting method. Afterwards, a neural network detector can detect products with a decreased detection false positive rate. A part of our solution is also automatic detection of ROI (the place where products are shown to the system). We reached* 0.4167 *F1-Score with* 0.3704 *precision and* 0.4762 *recall which placed us at the* 7th *place of* AI City Challenge 2022 *in corresponding* Track 4. *The code is made public and available on GitHub*[1].

## 1. Introduction

*Self-service* is a trend that is extending to more and more aspects of daily life (e.g. airport check-ins, automated teller machines at a bank, *etc*.). Customers are becoming more and more familiar with self-service systems. The triumphant procession of self-service systems seems to be extending to the supermarkets. New automated self-checkout systems enable shoppers to scan, bag, and pay for their purchases without or with minimal help from store personnel. Retailers expect to reduce their costs and gain more flexibility by introducing self-checkout systems. One cashier can now serve multiple customers simultaneously to use staff time efficiently. Shorter checkout queues, a faster checkout process, more privacy, and greater control for the customers are the key arguments being used to convince the retailers to introduce the new self-checkout systems [22].

This year's *AI City Challenge 2022* [27] contains a new Track named *Multi-Class Product Counting & Recognition for Automated Retail Checkout*. This Track aims to automatically detect and report products present in front of a camera view and help during retail store checkout. The goal is to report all products — meaning product name (ID) and the time when the product was present. All products are present in the camera view in some defined area — this

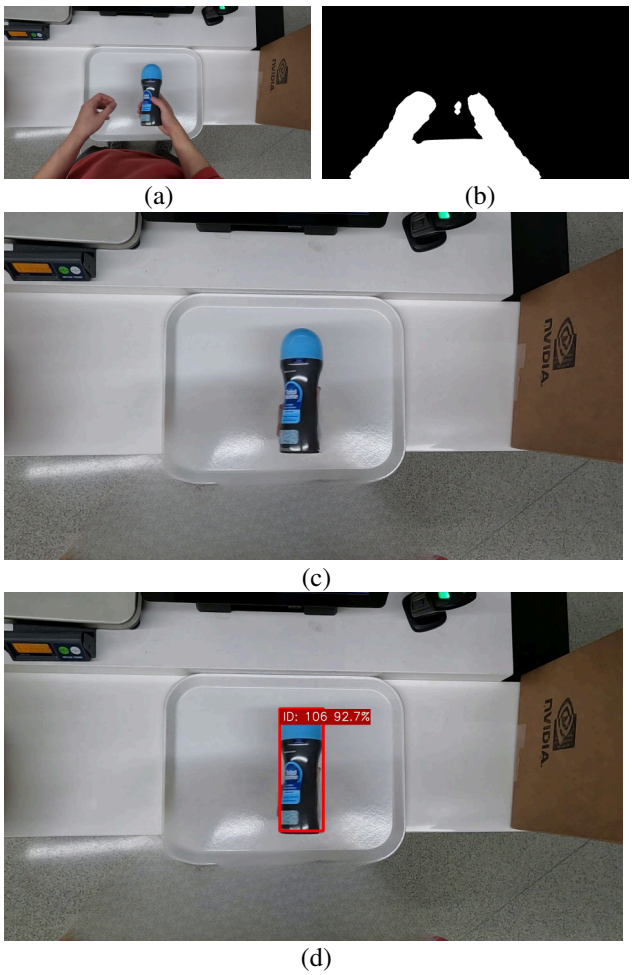


Figure 1. Our proposed solution for product detection. (a): Original video frame. (b): Detected mask of person by CNN. (c): Removed "unwanted" parts by image inpainting algorithm. (d): Resulting detection of product by CNN detector.

[1] https://github.com/BUT-GRAPH-at-FIT/PersonGONE

is a condition typically realizable in real-world scenarios. In this case, the defined area is a white tray which can be seen in Figure 1. The tray position is not defined and must be localized automatically (this process is described in Section 3.2).

Products may be occluded or very similar to each other, which may cause problems in detecting proper products and reporting their presence. One goal of the *AI City Challenge 2022* [27] is also to suppress an advantage of external data. Only provided or synthetic data can be used for the training of models (information about these data are available in Section 4.1). Our proposed solution comprises multiple sub-tasks that are done in the following order: person detection, image inpainting, ROI detection, detection of products, tracking of products, and tracks post-processing. The individual steps and their benefits are described below in Section 3.

## 2. Related Work

In this part, a brief overview of related works from every field used during this challenge is provided. Relevant methods and models for automated retail store checkout (Section 2.1), multi-class object and body part detection (Sections 2.2, 2.4), multiple object tracking (Section 2.3) and image inpainting (Section 2.5).

### 2.1. Automated Retail Store Checkout

Self-service checkout systems can be divided into two main categories — centralized or decentralized systems. Centralized systems are located at store exits, often created by self-checkout terminals or tunnel scanners. Decentralized systems use handheld scanners or mobile phones. In both cases, checkout depends on reading RFID [5,6,12,13], EAN or QR code tags [13,33] from retail items.

Automated store checkout might be extended from reading tags to recognizing items during checkout based on visual features of the item and its appearance in general. Aquilina and Saliba [1] presented a method for retail store automated checkout using SCARA robots. Their solution is based on a four-axis robotic system with machine vision, which automatically transfers items placed by a customer on a conveyor to the container, recognizes them, packs them and prepares a total bill.

James *et al*. [16] propose to use conventional multi-class detectors based on convolutional neural networks to detect and recognize items from a single RGB image.

### 2.2. Multi-class Object Detection

Convolutional Neural Networks dominate in object detection of their accuracy compared to older techniques [9–11,25,29,35].

*Single Shot Detectors* (SSD) are one of the detection meta-architectures performing multi-class object detection. Liu *et al*. [25] published a study on a method called SSD which uses a single feed-forward CNN to directly predict classes without a second stage classification operation processing the proposed boxes. The term itself can denote the whole class of such detectors. Typical representatives of this group (aside from the original SSD detector) are also Multibox [35], YOLO-series detectors [3, 17, 29–31] or the Region Proposal Network (RPN) stage of the Faster R-CNN [32], which are used to predict class-independent box proposals.

In recent months, anchor-free detection models have taken the lead in this field. Most of them evolved from anchor-based methods described in the previous paragraph. Chen *et al*. [9] revisits concept of feature pyramids networks (FPN) for one-stage detectors in the *YOLOF* detector. The authors proposed a way to utilize only one-level feature for detection instead of the divide-and-conquer optimization problem solution inside FPN.

Feng *et al*. [10] introduces *TOOD: Task-aligned One-stage Object Detection*. This method combines object localization and classification from attention maps into alignment metrics, which better balances learning task-interactive and task-specific features. Together with the proposed *Task Alignment Learning* for anchor position optimization, this step helps them surpass previous one-stage detectors.

*YOLOX* is an anchor-free evolution of YOLO series detector models created by Ge *et al*. [11]. They also adapt advanced detection techniques such as a decoupled head and the leading label assignment strategy SimOTA. YOLOX outperforms comparable models YOLOv3 [31], YOLOv4 [3] and YOLOv5 [17] by a large margin.

### 2.3. Multiple Object Online Tracking

Simple Online Real-time Tracker (SORT) from Bewley *et al*. [2] is a visual multiple object tracking framework which rely on fundamental data association and state estimation techniques based on Kalman filter [18]. It estimates objects' identities on-the-fly using detections from past and current frames only. It also supports object re-entry in a predefined time window and partial occlusion. The SORT tracker was extended using deep association metric based on image features as DeepSORT tracker [37].

ByteTrack tracker by Zhang *et al*. [41] is a method for multiple objects tracking similar to DeepSORT. ByteTrack does not filter out low score detections (*e.g*. occluded objects, small objects) and associates almost every detection instead of the ones with a score over a threshold. In this case, the tracker is processing detections from the YOLOX detection model; similarity features for detection to track association are extracted from the FastReID model [14]. They outperform many available online trackers.

### 2.4. Detection of Body Parts

A helpful step in automated retail store checkout solutions might be detecting human body parts — especially hands. The known position of hands can be used as additional information when a human is holding individual items during checkout.

The 3D hand pose and shape estimation from a single RGB image has many real-world applications, such as robotics, augmented reality or gesture recognition. The goal is to localize a human hand's semantic keypoints (*i.e*. joints) in the 3D space. It is an essential technique for human behaviour understanding and human-computer interaction. Various deep learning methods have been used. They can be divided into two main categories: joint estimation via keypoint detection and object detection with instance segmentation.

*Hand-joint detection* is based on keypoint regression networks primarily. This covers also SCNet [23], HRNet [36], or baseline methods of 3D hand pose datasets like FreiHand [44] or InterHand2.6M [26]. These models can be further enhanced by using Unbiased Data Processing (UDP) [15], during the training phase.

Another way to solve this task is employing *object detector with instance segmentation* — mask of the detected object. DetectorRS [28], PointRend [19], YOLACT [4] and HTC [7] are a few samples of such detectors. The knowledge of object masks might be used for image preprocessing to boost the performance of the aforementioned multi-class object detectors.

### 2.5. Image Inpainting

Image inpainting refers to the task of completing missing regions of an image. This fundamental computer vision task has many practical applications, such as object removal and manipulation, image retargetting, image compositing, and 3D photo effects.

Previously used *patch-based methods* (copy-pasting patches from known regions) or *diffusion-based methods* (color filling using partial differential equation) are outdated nowadays. Generative adversarial neural networks (GANs) are taking the lead in recent years, and they are still making great progress in generated image quality and preciseness. GAN models are composed of two main components, *generator*, the part responsible for image synthesis and *discriminator* in the role of referee.

Two-stage networks predict an intermediate representation of an image in the form of edge, gradient, segmentation map, or a smoother image for final output enhancement. In order to augment the adversarial loss and suppress artifacts, many works often train the generator with additional reconstruction objectives such as perceptual, contextual, or $l_1$ loss.

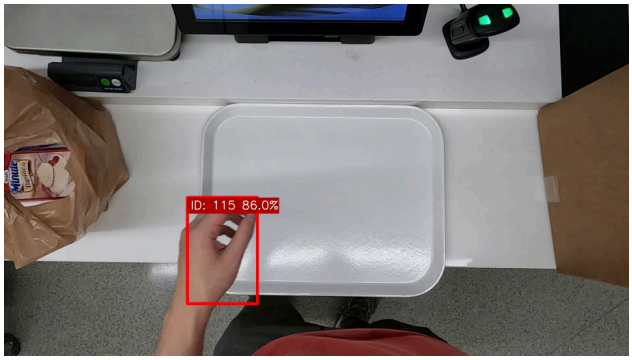


Figure 2. False positive product detection on person body in original frame.

Yu *et al*. [39] aimed at expanding the receptive field of the network by incorporating dilated convolutions to the generator and designing contextual attention to explicitly let the network borrow patch features at a global scale.

To enhance global prediction capacity, Zhao *et al*. [42] propose an encoder-decoder network that leverages style code modulation for global-level structure inpainting.

Suvorov *et al*. [34] propose to use Fourier convolution to acquire a global receptive field and segmentation networks to compute perceptual loss to achieve better performance. Furthermore, feature gating such as gated or partial convolution is proposed to handle invalid features inside the hole.

## 3. Methodology

As mentioned earlier, our proposed method for checkout consists of several steps. Each of the steps is described in more detail in the following sections.

### 3.1. Person Removal

As the data provided consists of synthetic images of individual products, these synthetic images were used for training by inserting them into ordinary images (more details are in Section 4.2). Objects are inserted into the "free space", and therefore, these products were often isolated in frame during training, and there were no other objects near the annotated products. For this reason, it has often been the case during inference that the worker's hands or body are detected as products even in cases where no product occurs in the scene (as can be seen in Figure 2). Therefore, we decided to use an image inpainting method to remove the person, which significantly reduced the false positive detection rate.

The method used to "delete" a person is LaMa [34]. This method requires an image and a related mask as its input. Thus, it is necessary to detect the person's mask as the first step. Instance segmentation methods are used for this purpose — we tried several different methods for instance seg-

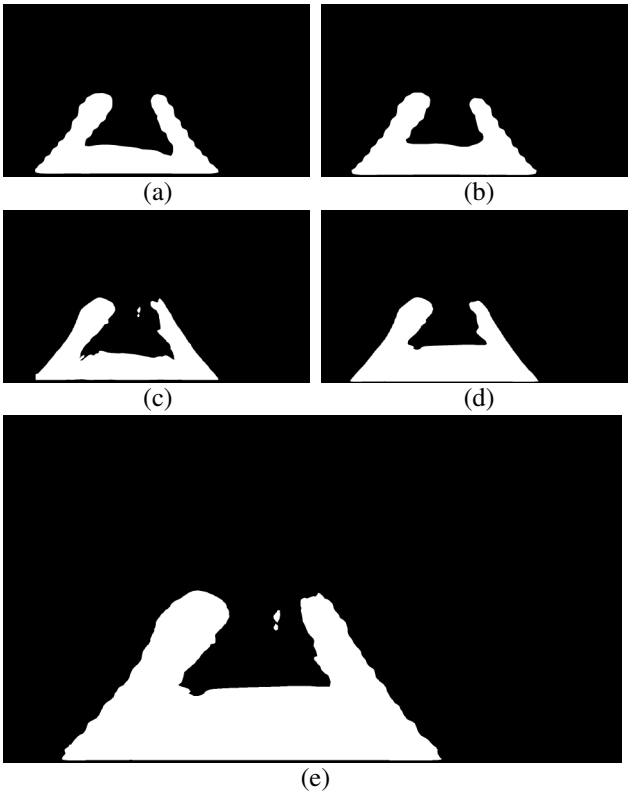

Figure 3. Construction of person mask for further "removal". (a): DetectoRS output. (b): HTC output. (c): PointRend output. (d): YOLACT output. (e): Combination of all the methods.

mentation from *MMDetection* toolbox [8]. In particular, we tested the models *DetectoRS* [28], *HTC* [7], *PointRend* [19], and *YOLACT* [4]. All of these methods suffer from some inaccuracies. We combined the outputs of all the methods — some examples can be seen in Figure 3.

The usage of dilation additionally enhances the detected person masks. It helps to make some higher external borders, and in some failure cases, "ghost objects" do not appear in the frame. The dilation value must be set carefully as low dilation keeps the mentioned "ghost objects". On the other hand, too high a value removes potentially necessary information. The inpainting is processed frame-by-frame, so in future work, video inpainting could be used (*e.g.* [20, 24, 40]). However, these methods have high memory requirements, and for our goal, image inpainting is sufficient. An example of the convenience of the method used can be seen in Figure 4.

### 3.2. ROI Localization

The trained model also had problems that it detect objects outside the required area, and at the same time, the goal of the track is to report objects above the "white tray". An example of unwanted detection outside the area can be

Figure 4. The difference between detection in the original and the inpainted frame.

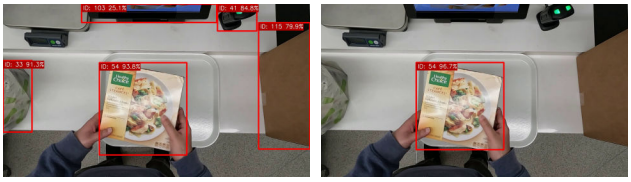

Figure 5. Detection of products without/with the usage of detected ROI.

seen in Figure 5. The localization of ROI (region of interest) is made automatically; the first step in this process is to extract the background image. This image for each scene is extracted by following: Gaussian Mixture Model [45] extracts background the part of each frame of the video sequence (with the usage of previous frames), and the mean value of all these background images is computed as the resulting background model. For the computation, inpainted video frames are used, as it makes the process easier by removing unnecessary objects (person). Examples of resulting background images are depicted in Figure 6 *top row*.

When the mean background image is extracted for the scene, the image is transformed to grayscale, and the Scharr operator (an enhanced variant of the Sobel operator) for edge detection is applied. The Scharr operator is applied in $x$ and $y$ direction and combined together (resulting edge detection is in Figure 6 — *bottom left*). A *flood fill* algorithm is used on this image with detected edges, which searches the same/similar values as a seed (in our case, the seed is in the image center, but can be set arbitrary). In this way, all pixels until edges are connected and marked as the "tray" in our case. Detected ROI can be seen in Figure 6 — *bottom right*. When ROI is detected, it serves for filtering detections outside (with some extension). The resulting bounding box is axis-aligned rectangle of all pixels found by flood fill algorithm.

### 3.3. Product Detection and Tracking

As mentioned earlier in the Introduction (Section 1) the next step after person removal and ROI detection is the detection of the products themselves. For detection, we use a multi-class *YOLOX* [11] detector, which reaches high precision on public datasets evaluation together with fast inference speed. We trained our own model with 116 output

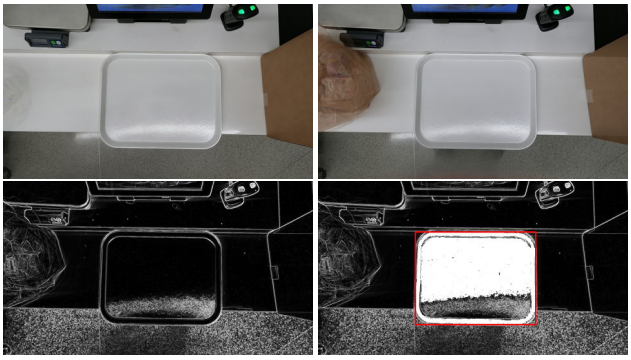

Figure 6. *Top row*: Mean background images for two sample scenes. *Bottom left*: Edges detection by the Scharr operator. *Bottom right*: Detection of the "tray" by the usage of flood fill algorithm; bounding rectangle in red.

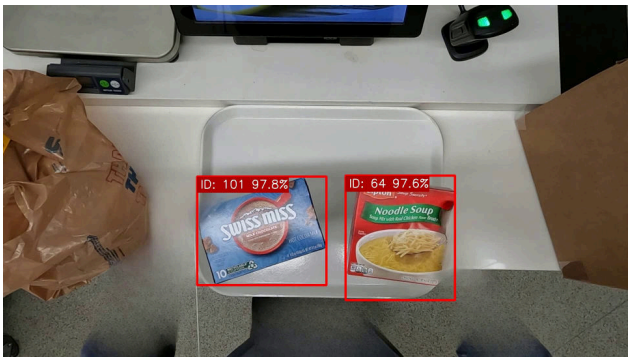


Figure 7. Multi-class detection with our trained *YOLOX* detector.

classes on the provided dataset of generated objects as described in Section 4.2. We reached AP 97.47 % on the validation set during training. Together with object position, the detection confidence and class confidence are available, which are used further in the tracks post-processing. An example of object detection is available in Figure 7.

Together with the detection of products, tracking of individual products is performed. As a part of our solution, we tried two tracking algorithms — *SORT* [2] and *ByteTrack* [41]. Both algorithms work online with bounding rectangles of the detections and use the Kalman filter to predict the future positions and merge these detections/predictions to corresponding tracks.

### 3.4. Tracks Post-processing

The last step in our pipeline is a post-processing of the detected tracks. Each track $t^i$ contains detections $d^i_j$. Each detection $d^i_j$ is composed of a bounding box, class ID, class confidence, and detection confidence as was mention previously. As a track can contain certain inaccurate class values (in one track $t^i$ can be detections $d^i_j$ with different class IDs) it is necessary to set a single class value for the whole track $t^i$.

For this purpose, classes are merged together by their class $id$ as $c^i_{id}$. For each single class $c^i_{id}$ in track $t^i$ is computed its count $|c^i_{id}|$ and mean class confidence value $\overline{c^i_{id_{\text{class\_conf}}}}$ of all detections belonging to the corresponding class $id$. To each class $id$ is computed its weighted confidence value as:

$$c^i_{id_{conf}} = \overline{c^i_{id_{\text{class\_conf}}}} \frac{|c^i_{id}|}{N^i},$$

where $N^i = |d^i_j|$ is the count of all detections in track $t^i$. The resulting class for the whole track $t^i$ is the class with the highest weighted confidence value $c^i_{id_{\text{conf}}}$. In this way, not only count of single class detections, but also class confidences of each detection play a role in decision of final track class — this class is then reported as the final whole track class.

A part of the submission is also the time when the object was localized in front of the camera. The time should be any second (*integer*) when the object was above the ROI (detection described earlier in Section 3.2). As a part of our solution, we have frame numbers for each detection $d^i_j$. The class ID for each track is determined based on the procedure described earlier. Thus, the last necessary step is proper time computation. For each track, $t^i$ is a computed list of time values in seconds as $d^i_{j\text{frame}}/\text{fps}$ and rounded down (*math floor*) to the nearest integer. These time values are accumulated into individual *bins* corresponding to proper values in seconds. The resulting (reported) second is the position of the bin with the highest accumulated value (the most detections in the proper second).

## 4. Experiments

This section describes the datasets used in this work, performed experiments, and the achieved results. Datasets are divided into AIC Challenge Dataset for Track 4 and our generated synthetic dataset for the training of the aforementioned *YOLOX* detector.

### 4.1. AIC Challenge Track 4 Dataset

Dataset for multi-class product counting and recognition for automated retail checkout is provided as part of the *AI City Challenge 2022* (Track 4). This dataset contains 116, 500 synthetic images, generated using a pipeline by Yao *et al*. [38] with masks for training, created from 116 different merchandise item models captured by a 3D scanner. Random background images, which are selected from Microsoft COCO [21], are used to increase the dataset diversity. Sample images, together with masks, for some classes are shown in Figure 8.

The automated retail store checkout quality is evaluated on the testing part of the provided dataset. This part is

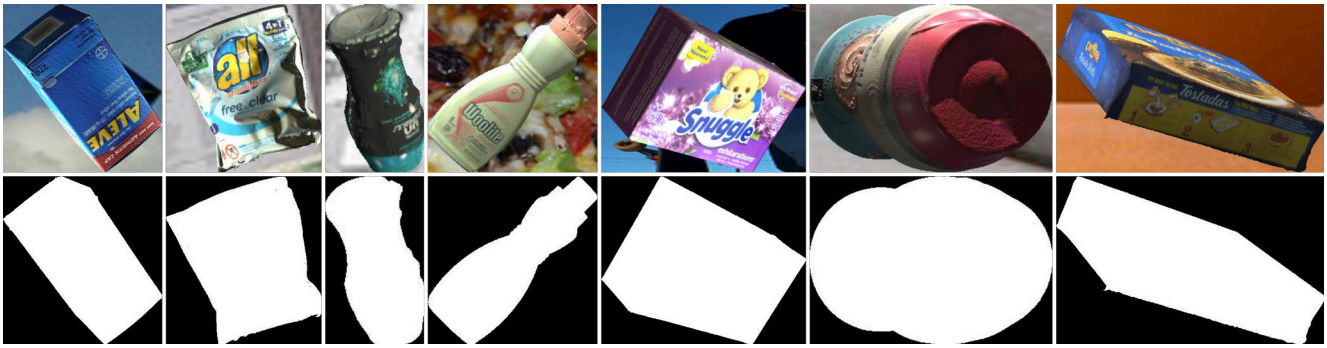

Figure 8. Sample of the synthetic dataset of products (*top row*) with generated object masks (*bottom row*).

formed from 25 recorded test videos. These videos capture different checkout procedures in a simulated environment from the top view. The task is to correctly identify and count items in the region of interest at different times. A sample of these data can be found in Figure 1 (a).

The test set is split between Test sets A and B with a ratio 20 % to 80 % accordingly. Test set A was published for testing and result evaluation over the AIC evaluation server. Test set B is dedicated for further evaluation and the final ranking.

### 4.2. Synthetic Dataset for Detector Training

The assignment of the *AI City Challenge 2022* is quite strict in the usage of external data, and thus we created our own dataset for the training of the detector as mentioned in Section 3.3. The only allowed "extra" data are those from other challenge tracks, and therefore we used these. We extracted 15, 531 frames from Track 1 and Track 3 datasets and inserted synthetic images (Figure 8) into them. The inserted objects are cropped by the available masks; synthetic objects are thus freely placed in each frame (see Figure 9). In total, 100, 000 and 20, 000 images were generated for the training and validation set, respectively. For each (random) frame, 1 to 4 objects were randomly selected from the available synthetic dataset (Section 4.1) and randomly placed into the frame.

### 4.3. Implementation Details

As mentioned in Section 3.1, we need to localize an exact person mask by combination of different instance segmentation methods (*DetectoRS* [28], *HTC* [7], *PointRend* [19], and *YOLACT* [4]). All these methods are implemented in *MMDetection* toolbox [8] and also weights pretrained on *COCO* dataset [21] are available[2].

Once the instance masks are available, we use LaMa [34] for image inpainting — a model pretrained on *Places2* dataset [43] is also available[3]. Before the image inpainting application, the object mask is dilated. We experimented with different values and possible settings and found the best dilation solution with a cross kernel of size 9 and 3 iterations. Expansion of ROI for filtering is set to the value of 0.1 (expansion of width and height by 10 %).

Table 1. Results with different input image size

| Variant | F1-Score | Precision | Recall |
|---|---|---|---|
| 640 | 0.4082 | 0.3571 | 0.4762 |
| 736 | 0.4000 | 0.3448 | 0.4762 |
| 800 | 0.4167 | 0.3704 | 0.4762 |

We trained the *YOLOX* [11] detector on our generated dataset (Section 4.2); as the best we found variant *YOLOX-L* trained for 75 epochs with input image size 640 × 640. All other training setting was equal to original network setting (see website[4]). For tracking, we tried two trackers: *SORT* [2] and *ByteTrack* [41] with the period for which the track can be broken increased to 30 frames.

### 4.4. Evaluation

We tried several variants of the *YOLOX* network for object detection (Medium, Large, X-large) in two possible settings (network pretrained on COCO dataset and training from scratch). As a result, we selected variant *Large* trained from scratch. We also experimented with input image size — results can be seen in Tab. 1. All variants seem to be similar and produce very close results, and thus the input image size is probably not so important.

We also experimented with *SORT* and *ByteTrack* trackers. In our experiments, *ByteTrack* seems to be more stable and achieved the same result (0.4167 F1-Score) with lower image resolution (640×640) compared to the *SORT* tracker. However, all results are very close, and it is almost impossible to say which one is better. Both variants are implemented, and users can switch between them due to conditions. Our best result is 0.4167 F1-Score, so we placed *7th*

[2] https://github.com/open-mmlab/mmdetection
[3] https://github.com/saic-mdal/lama
[4] https://github.com/Megvii-BaseDetection/YOLOX

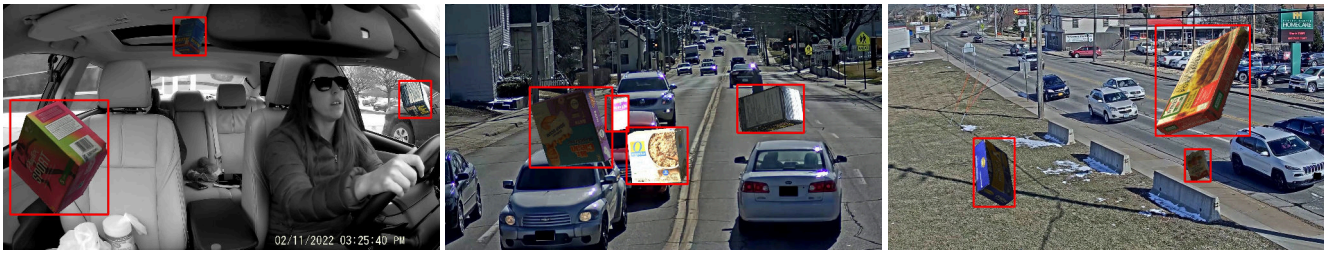

Figure 9. Sample images from our generated dataset for detector training with marked detections.

Table 2. Public leaderboard of *AI City Challenge 2022* Track 4

| Rank | Team Name | Score |
|---|---|---|
| 1 | BUPT-MCPRL2 | 1.0000 |
| 2 | SKKU Automation Lab | 0.4783 |
| 3 | The Nabeelians | 0.4545 |
| 4 | mizzou | 0.4400 |
| 5 | RongRongXue | 0.4314 |
| 6 | Starwar | 0.4231 |
| 7 | GRAPH@FIT | 0.4167 |
| 8 | HCMIU-CVIP | 0.4082 |
| 9 | CyberCore-Track4 | 0.4000 |
| 10 | UTE-AI | 0.4000 |

in the public part of Track 4 challenge. The public leaderboard can be seen in Tab. 2.

### 4.5. Processing Data from Multiple Streams

An automated retail store checkout system should be installed at every checkout spot in the store to improve the Quality of Experience for customers. The solution based on computer vision and machine learning technologies could be computationally heavy and need adequate hardware for every checkout spot. Another possible approach is to use a distributed form of processing using cloud-native applications. In this case, image data are transferred to the cloud, where each part can be computed individually using multiple workers. The result of each sub-tasks is passed to the following processing step. This approach corresponds with NetApp for distributed processing of the *5G Enhanced Robot Autonomy* project. The system proposed in this work is an excellent example of a task for distributed processing, where a single computational cluster located in the store or cloud may process many checkout spots.

## 5. Conclusion

We participated in *AI City Challenge 2022*, Track 4 called *Multi-Class Product Counting & Recognition for Automated Retail Checkout*. We proposed a novel approach based on image inpainting, which significantly improves the detection results and reduces the rate of false positive detections. As a part of our solution, we also automatically detect the region of interest and automatically segment out parts of humans and further "delete" them from the scene. We achieve competitive results to most other teams with YOLOX-L detection network, which can run in real-time and trackers based only on bounding boxes (without deep learning). In the final leaderboard, we placed *7th* with F1-score 0.4167, which placed us in the first half of the participants.

## Acknowledgment

This publication is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101016681.

## References

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

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *International Conference on Image Processing (ICIP)*, 2016. 2, 5, 6

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020. 2

[4] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *International Conference on Computer Vision (ICCV)*, 2019. 3, 4, 6

[5] MFM Busu, I Ismail, MF Saaid, and SM Norzeli. Auto-checkout system for retails using radio frequency identification (rfid) technology. In *Control and System Graduate Research Colloquium*, 2011. 2

[6] Gavin Chappell, David Durdan, Greg Gilbert, Lyle Ginsburg, Jeff Smith, and Joseph Tobolski. Auto-id in the box: the value of auto-id technology in retail stores. *Auto-ID Center*, 2003. 2

[7] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid

task cascade for instance segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 4, 6

[8] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 4, 6

[9] Qiang Chen, Yingming Wang, Tong Yang, Xiangyu Zhang, Jian Cheng, and Jian Sun. You only look one-level feature. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[10] Chengjian Feng, Yujie Zhong, Yu Gao, Matthew R Scott, and Weilin Huang. Tood: Task-aligned one-stage object detection. In *International Conference on Computer Vision (ICCV)*, 2021. 2

[11] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 2, 4, 6

[12] Matthias Hauser, Sebastian Günther, Christoph Flath, and Frédéric Thiesse. Leveraging rfid data analytics for the design of an automated checkout system. 2017. 2

[13] Matthias Hauser, Sebastian A Günther, Christoph M Flath, and Frédéric Thiesse. Towards digital transformation in fashion retailing: A design-oriented is research study of automated checkout systems. *Business & Information Systems Engineering*, 2019. 2

[14] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. FastReID: A pytorch toolbox for general instance re-identification. *arXiv preprint arXiv:2006.02631*, 2020. 2

[15] Junjie Huang, Zheng Zhu, Feng Guo, and Guan Huang. The devil is in the details: Delving into unbiased data processing for human pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[16] Namitha James, Nikhitha Theresa Antony, Sara Philo Shaji, Sherin Baby, and Jyotsna Annakutty. Automated checkout for stores: A computer vision approach. *Revisita Geintec-Gestao Inovacao E Tecnologias*, 2021. 2

[17] Glenn Jocher. Yolov5. online. 2

[18] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960. 2

[19] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. *arXiv preprint arXiv:1912.08193*, 2019. 3, 4, 6

[20] 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. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 4

[21] 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. 5, 6

[22] Thorsten Litfin and Gerd Wolfram. *New Automated Checkout Systems*, pages 189–203. Springer Berlin Heidelberg, 2010. 1

[23] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Changhu Wang, and Jiashi Feng. Improving convolutional networks with self-calibrated convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[24] Rui Liu, Hanming Deng, Yangyi Huang, Xiaoyu Shi, Lewei Lu, Wenxiu Sun, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fuseformer: Fusing fine-grained information in transformers for video inpainting. In *International Conference on Computer Vision (ICCV)*, 2021. 4

[25] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[26] Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. Interhand2. 6m: A dataset and baseline for 3d interacting hand pose estimation from a single rgb image. In *European Conference on Computer Vision (ECCV)*, 2020. 3

[27] Milind Naphade, Shuo Wang, David C Anastasiu, Zheng Tang, Ming-Ching Chang, Yue Yao, Liang Zheng, Sharifur Rahman, et al. The 6th ai city challenge. In *Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2022. 1, 2

[28] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution. *arXiv preprint arXiv:2006.02334*, 2020. 3, 4, 6

[29] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *Computing Research Repository (CoRR)*, abs/1506.02640, 2015. 2

[30] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[31] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018. 2

[32] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015. 2

[33] Matthew Ritchie, Tiana Longino, Daniel Garza, and Alp Katranci. Optimized automated checkout process for major food retailers. 2021. 2

[34] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. *arXiv preprint arXiv:2109.07161*, 2021. 3, 6

[35] Christian Szegedy, Scott Reed, Dumitru Erhan, Dragomir Anguelov, and Sergey Ioffe. Scalable, high-quality object detection. *arXiv preprint arXiv:1412.1441*, 2014. 2

[36] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui

Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 3

[37] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *International Conference on Image Processing (ICIP)*, 2017. 2

[38] Yue Yao, Liang Zheng, Xiaodong Yang, Milind Napthade, and Tom Gedeon. Attribute descent: Simulating object-centric datasets on the content level and beyond. *arXiv preprint arXiv:2202.14034*, 2022. 5

[39] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3

[40] Yanhong Zeng, Jianlong Fu, and Hongyang Chao. Learning joint spatial-temporal transformations for video inpainting. In *European Conference on Computer Vision (ECCV)*, 2020. 4

[41] 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. 2021. 2, 5, 6

[42] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. *arXiv preprint arXiv:2103.10428*, 2021. 3

[43] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 6

[44] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[45] Zoran Zivkovic and Ferdinand van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 2006. 4