# Multi-Person Shopping (MPS) for Cashier-Less Store

Yixin Bao*
Information Networking Institute
Carnegie Mellon University
yixinbao@cmu.edu

Xinyue Cao*
Information Networking Institute
Carnegie Mellon University
xinyue.cao@west.cmu.edu

Chenghui Li*
Information Networking Institute
Carnegie Mellon University
leoli@cmu.edu

Mengmeng Zhang*
Information Networking Institute
Carnegie Mellon University
mengmeng.zhang@west.cmu.edu

## ABSTRACT

Backed up by the rising of Artificial Intelligence and Internet-of-Things, autonomous checkout is rapidly advancing and has tremendous opportunities in the near future. Two key components of a cashier-less store would be inventory monitoring and customer-merchandise association. Regarding these two fields, we built an end-to-end multi-modal solution for autonomous checkout. To achieve inventory monitoring, we constructed a sensor fusion framework through weight sensors and knowledge of item arrangement similar to [1]. To achieve customer-merchandise association, we extracted 3D human keypoints from cameras to assign merchandise to corresponding customers. Our solution could solve various complex shopping scenarios including multi-person shopping, products lift and put back, and products misplacement, etc. We won the CPS-IoT week 2020 AutoCheckout competition with a F1 score 84.4%. [1]

**Figure 1: Multi-person tracking in an autonomous retail shop**

## 1 INTRODUCTION

AIM3S is a novel sensor fusion framework proposed in 2019[1] to enable fully autonomous stores. It combines weight sensors, cameras and prior knowledge of item arrangement to predict products that have been taken off the shelf.

Novel as AIM3S is, there are limitations when it's deployed to real-world stores. After testing on a comprehensive set of test cases in the NanoStore in Campbell, California, we found that the AIM3S system can be improved in both inventory monitoring and customer-merchandise association. Inventory monitoring can be problematic in some special scenarios such as different products are picked up at the same time, or products misplaced to a wrong place after picking up, etc. This is because the data from the weight sensor has a dominant contribution over other components and there is no status tracker in the system. As for customer-merchandise association, the current system fails to associate merchandises with the right customer when there are multiple people shopping, especially when they stand close or even cross arms, which then messes up with everyone's bills.

In this paper, we presented an end-to-end auto-checkout system containing both inventory monitoring and customer-merchandise association. We based our inventory monitoring component on AIM3S[1]. In addition to that, we have implemented multiple features to support more complex shopping scenarios, such as multi-products lifting, products misplacement, real-time dynamic planogram, etc. For customer-merchandise association, we proposed a multi-modal solution based on the prior arrangement knowledge and spatial human pose estimation. When an item is taken off the shelf, the weight sensor is triggered to provide the product spacial location. Meanwhile, we run human pose estimation from multiple cameras to extract 3D human spatial information. Customer-merchandise association is then calculated by the aforementioned information.

As an output for each shopping scenario, our auto-checkout system will generate separate receipts for all targets in the store. Each receipt contains the purchased products and their quantities.

*All authors contributed equally to this research.

[1]Our code is open sourced at: https://github.com/AutoCheckout-CMU/AutoCheckout. As far as we know, we're the first one to open source an end-to-end solution for autonomous store.

## 2 SYSTEM OVERVIEW

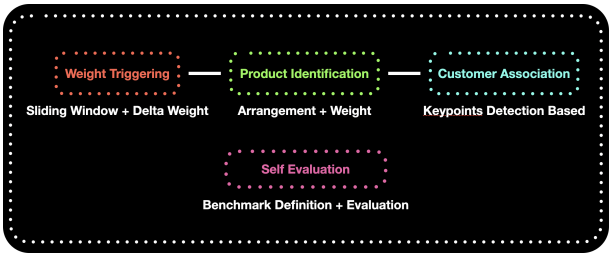


**Figure 2: System Architecture**

Our solution is an end-to-end system that takes in recorded data for shopping events and outputs generated receipts for each customer. The system consists of three major functioning modules: Weight Triggering, Product Identification and Customer Association. And besides the three modules, we have created an Evaluation module that runs our algorithms on a dataset provided by AiFi and evaluates the results with the ground truth we manually labeled.

These modules work cohesively to streamline the process of receipt generation. It all starts with a weight trigger. Whenever a customer interacts with the shelves, the weight sensor would detect a weight change, and this change would be recorded in the database. Having the recorded data, the Weight Triggering module analyzes the sequence of weight changes during a certain time frame and sifts through the data to separate valid weight trigger events from noises. Then the Product Identification module tries to calculate confidence score for each product in the database based on their positions and weight. The Customer Association module takes in the product that has the highest score, associates the product with the exact customer who took it by calculating the distance using detected human body key points.

## 3 METHODOLOGY

### 3.1 Event Detection

We identify a purchase event by detecting weight change within a time frame. The weight sensor will generate a series of time-based weight signals, and an event will be identified by comparing the moving mean and variance of weight among time frames to thresholds.

The first step in MPS pipeline is to detect when events take place and whether there is a put-back or pick-up event. The processing of each change of the inventory starts with a weight change trigger. Because in common use cases, customers are not trying to fool the system. The weight difference on the load sensors is generally enough to detect an event.

In the MPS framework, we compute the mean and variance of the weight values over sliding windows and use the variance threshold to filter out noise data points so that the remained is the event candidates. To become a valid event candidates, the duration when the variance is larger than the variance threshold should be longer than the valid interval threshold. So in Figure 3, there are two event candidates with their own peak values. Apart from this, we check whether all valid event candidates should be selected as final events by comparing the change of moving averages with the threshold. From the drop of the curve in Figure 4 where the delta mean weight is larger than given threshold, one valid event is triggered successfully. At the same time, the final triggered event with start time, end time, delta weight, as well as the position including gondola, shelf and plates where the event happens are returned to MPS system to identify products and locate the frame in videos to do the association between products and customers.

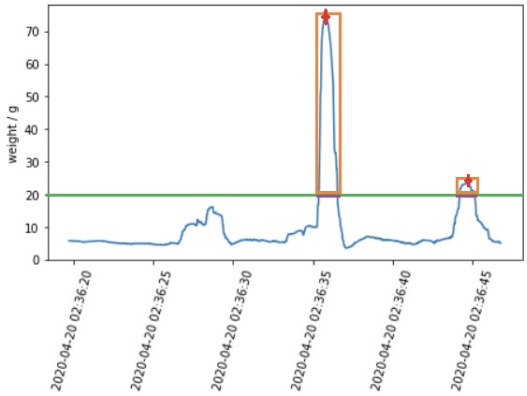


**Figure 3: Aggregated moving variance of weight (blue), variance threshold (green), valid intervals (purple), two event candidates (orange) and peak values (red)**

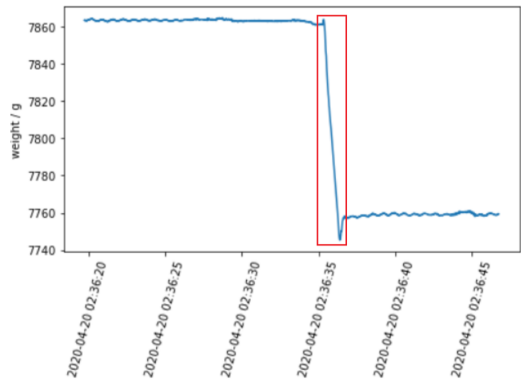


**Figure 4: Aggregated moving mean of weight (blue) during the first half minute, the event detected by MPS (red)**

## 3.2 Product Identification

Having a weight trigger event, the Product Identification module uses the existing data to predict the most possible product that's being picked up or put back during this event. We calculate a confidence score for each product and select the one with the highest score. To calculate the overall confidence score, two properties of the product are taken into consideration: arrangement and weight.

*3.2.1* ***Arrangement-based Product Identification.*** Prior knowledge of merchandise placement and shelf location will be provided in the autonomous shops. Hence getting the merchandise location is relatively trivial.

It would be logical to calculate the arrangement score for a product. Because intuitively the product closest to the plate where the most drastic weight change happened is probably the product being interacted with.

In our design, the weight trigger event is per shelf. It does not make sense to mark an event for each plate because plates on the same shelf are closely placed next to each others and an interaction with one plate will most certainly cause weight changes on other plates too.

When we have an event, we calculate the arrangement score for each product that's on the the shelf. We first assign the a score to each plate on the shelf based on how much weight change they each sensed. For example, if the shelf has 4 plates and their detected weight changes are [-50g, -40g, -10g, 0g], the corresponding scores would be [0.5, 0.4, 0.1, 0.0].

Then for each product on the shelf, we create an array of their possibility of being on reach plate, say, if product A is on both the 1st and the 2nd plate, the existence probability array would be [1, 1, 0, 0]

We then do an elementwise multiplication of the plate scores array and the product existence probability array:

$$[0.5, 0.4, 0.1, 0.0] * [1, 1, 0, 0] = [0.5, 0.4, 0.0, 0.0]$$

And we calculate the arrangement score for this product by adding up the elements in the resulting array, thus getting an arrangement score of 0.9 for this product.

*3.2.2* ***Weight-based Product Identification.*** It's also logical to calculate a weight score for each product because intuitively you would think the product which has the most similar weight to the detected weight change should be the one that's being taken or put back.

In our approach, we estimate how close the event weight change is to each product's weight. The shorter the distance, the higher score.

Because of manufacturing deviations, different packages of the same product might weigh a bit differently. To account for this, we model each product's weight as a normal distribution instead of an absolute value. It's also necessary to take the weight sensing noise into consideration. The detected weight change is not always accurate and we modeled it as a normal distribution too.

We then calculate the probability of the two normal distribution overlapping for each product, thus getting a weight score for each product.

Now that we have both arrangement score and weight score for each product, we calculate the overall score like this:

$$TotalScore = arrangementScore * C + weightScore * (1 - C)$$

C: the contribution of the arrangementScore takes up in the totalScore

We then select the product with the highest total score.

*3.2.3* ***Separate handling for putback events.*** For an event that has a delta weight above zero, it means it detected added weight. These events are generally putback events where customers are trying to put the product back to the shelf, and we have a different way of handling the event.

It doesn't make sense to calculate the arrangement score anymore, because customers can place the product on a wrong plate. It also doesn't make sense to predict the product being put back from the whole product database, because the customer can only put back what they already have in hands.

Since our system keeps track of what products the customer has already taken off the shelf, we only need to make a prediction out of their shopping cart. We simply calculate the distance from the product weight to the detected weight change for each product in the shopping cart, and then find the one with the closest weight.

## 3.3 Association

*3.3.1* ***Human Keypoints Estimation.*** We use OpenPose[2] to estimate the 2D spatial information of human keypoints. With 12 cameras in the shop at different locations, we first get the in intrinsic and extrinsic parameters by camera calibration. In stead of conducting 3D reconstruction, we keep the detected human 2D points and project the merchandise into camera coordinate.[2]

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = M \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix}$$

*3.3.2* ***Customer-merchandise association.*** After getting the multi-person identity and spatial location information, we would then associate those identities with different merchandises. For each merchandise, we will calculate $n$ scores representing the association score w.r.t each customer currently in the store.

[2]Please note that in the CPS-IoT 2020 AutoCheckout competition, we used the human keypoints detection results provided by the competition.

We have designed two alternatives to calculate the association score:

$$Score_i = \sum_{k}^{body\_parts} Pr(k)D(k,m) \quad (1)$$

The $Pr(k)$ is the confidence level of OpenPose keypoints detection. The $D(k,m)$ is the $L2$ distance between the corresponding body part and the merchandise.

$$Score_i = \min_{k}\{I_k D(k,m)\} \quad (2)$$

$I$ is an indicator function for confidence score, only body parts which are greater than a threshold will be valid. Intuitively, this approach measures the distance from a merchandise to the closest possible body part.
After getting all the scores, we use *argmax* to assign the merchandise to the customer with the highest association score.

## 4 EXPERIMENT SETUP

A store similar to AiFi's NanoStore [3] is set up for experiment. The store has a set of 48'' shelves, filled with different products. Each shelf is composed by twelve 4" plates, with weight sensor under each plate.

### 4.1 Data

We conducted our experiments on the following data:

- A video feed from 12 cameras inside the store.
- 3D position of all humans inside the store, represented by each person's head, hands, etc..
- Weight Sensors data from all sensors on the shelves.
- A trigger that someone entered/exited the store.
- Layout of the sensors and cameras.
- Layout of the products in the store.
- Detailed information of the products, including product weight, price, pictures from different perspective.

### 4.2 Benchmark Test Cases

The CPS-IoT week 2020 AutoCheckout competition has provided a variety of test cases. Those test cases cover normal shopping behaviors: single person shopping, multiple person shopping, put back selected products, etc. There are also dedicated test cases whose aim is to mislead the system, which includes picking up a product very slowly, or try to take a product which is closer to another customer. We have selected a portion of those tests to make up a benchmark. Table 1 shows the list of test cases that are included in our benchmark.

[3] https://www.aifi.io/loop-case-study

| Test cases | Number of people | Number of pickup / putback events | Note |
|---|---|---|---|
| baseline-1 | 1 | 3 | |
| baseline-2 | 1 | 3 | |
| baseline-3 | 1 | 5 | Pickup 1 item and putback to original place |
| baseline-4 | 1 | 5 | Pickup 2 items and putback 1 to original place |
| baseline-5 | 1 | 5 | Same weight items with different colors |
| baseline-6 | 1 | 6 | One 1L bottle water vs two 0.5L bottle water |
| baseline-7 | 2 | 4 | Two people have overlapped hands |
| baseline-8 | 2 | 6 | Two people are very close |
| baseline-10 | 2 | 6 | Two people pickup same products at same time |
| baseline-11 | 3 | 11 | |
| baseline-12 | 3 | 33 | Very complicated, putback to wrong places, pass items between customers |
| baseline-13 | 1 | 2 | Pick up two very close but different product |
| baseline-14 | 1 | 4 | |
| baseline-16 | 1 | 6 | Misplaced items |
| baseline-20 | 1 | 3 | Pick up three very close but different product, then putback one. |
| baseline-22 | 1 | 2 | Pickup 1 item with a super slow speed. |
| baseline-23 | 1 | 6 | Putback 1 item and re-pickup it. |
| baseline-25 | 1 | 4 | |

**Table 1: Benchmark test cases**

### 4.3 Ground Truth

For each test case in the benchmark, we have manully labeled the corresponding ground truth. Below a sample ground truth is provided. The ground truth covers when and

where a shopping event happens, what product is picked or putback, and which customer is involved.


```
{
 "dataset": "BASELINE-2",
 "events": [
  {
   "eventID": 1,
   "observation": {
    "position": {
     "gondola": 3,
     "plates": [
      4
     ],
     "shelf": 2
    },
    "products": [
     {
      "barcodeType": "UPC",
      "id": "012000286209",
      "name": "Pure Leaf Unsweetened Black Tea",
      "price": 1,
      "thumbnail": "https://bit.ly/3eXbCzt",
      "weight": 592.6699829101562
     }
    ],
    "time": "00:10 + 2020-04-20_07-36-24",
    "target_id": "13933261984050196906"
   },
   "putback": 0
  }
 ]
}
```


## 4.4 Metrics

Based on the ground truth, we have defined three metrics to evaluate the overall performance of our system: precision/recall, F1-score, and association accuracy.

*Precision/Recall.* In our experiment, the true positive samples are defined only when both the product ID and customer association are correct identified. False positive samples are defined as wrong items on the final receipt, either product ID or customer association. False negative samples are defined as all items existing at the ground truth but are not on the receipts.

*F1-score.* The F-1 score is calculated based on precision and recall. The AutoCheckout competition is competing on the overall F-1 scores.

*Association Accuracy.* We also have an association accuracy score for association module evaluation. Unlike previous two metrics, this accuracy is to help us refine our solution. It defines the customer-association accuracy for all products on the receipt.

# 5 EXPERIMENT RESULTS

In this section, we will show the overall benchmark result for our MPS model, as well as the benchmark result for two of our most important optimizations for MPS model: use dynamic planogram to handle put-back events, and use event splitting to handle picking-up nearby products.

## 5.1 Overall performance

Figure 5 represents our overall benchmark results. Compared to the reported 86% accurary from self-checkout stations[3], our MPS model has improved the accuracy to 87.5%. Actually the self-checkout stations only need to handle single person shopping, whereas our score represents much difficult cases when different customers shopping at the same time.

It seems that there is still a gap between our accuracy score and the 93.2% accuracy of AIM3S's model[1]. This is because AIM3S makes many assumptions to simplify the environment. It's assumed that for each pick-up or put-back event, those events are correctly detected by the weight sensor. In addition, AIM3S only supports one-person shopping. What's more, AIM3S doesn't release its recall score, whereas our MPS model achieves a quite good score of 92.6%.

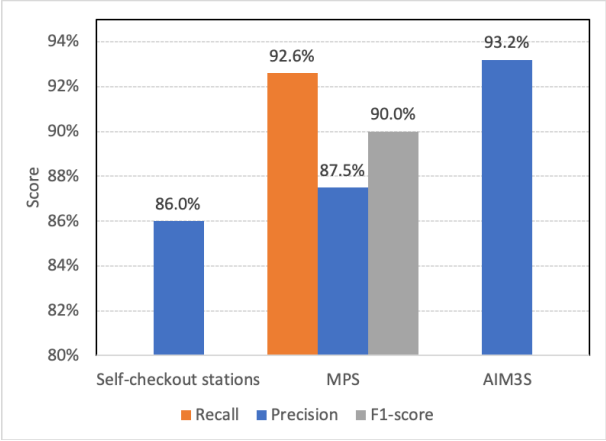


**Figure 5: Compare MPS performance with related works**

## 5.2 Dynamic Planogram Performance

For each test case, a prior knowledge of product arrangement is available, a.k.a the planogram. Our MPS model replies on this planogram to calculate the arrangement score. However, this planogram will get out-dated when one customer picks up one product and puts it back to another plate or shelf. To address this issue, a dynamic planogram is

added to the MPS model. Each time a customer puts back a product to the shelf, the MPS model will update this shelf's planogram to reflect that a new product is added to this place. Figure 6 shows that the F1-score of our MPS model is improved by 6.7% when enabling dynamic planogram.

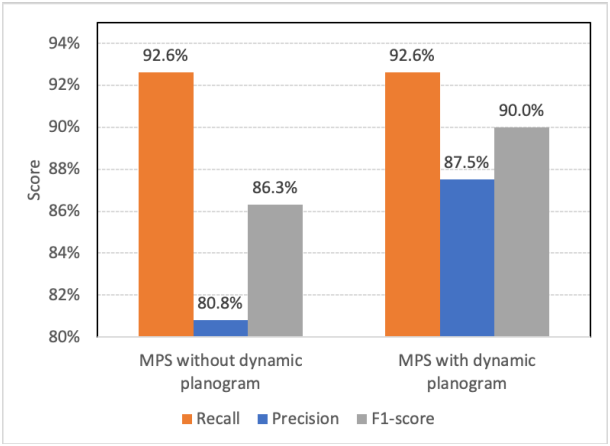


**Figure 6: Performance with / without dynamic planogram**

## 5.3 Event Splitting Performance

The basic MPS model can only triggers one event per shelf, which couldn't handle the case that customers will pick up two nearby products simultaneously. In order to address this issue, the MPS model needs to refer to the planogram and split the per-shelf event into smaller sub-events. We call this feature event splitting. Figure 7 shows that the F1-score of our MPS model is improved by 5.6% when enabling event splitting.

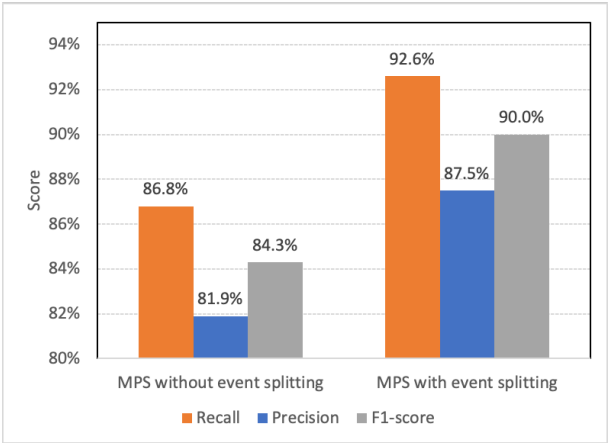


**Figure 7: Performance with / without event splitting**

## 6 CONCLUSION

In this paper, we presented Multi-Person Shopping (MPS) system for cashier-less convenience stores. Utilizing weight difference, weight-based and location-based product identification, human keypoints estimation, and customer-merchandise association monitors products placed on or picked up from shelves without human-in-the-loop with up to 0.992 event detection precision, 0.926 recall for receipt from real world scenarios and 0.875 for the overall precision in real world scenarios. To the best of our knowledge, this is the first fully autonomous system that solves the complex scenarios of multi-person shopping, fusing multiple sensing modalities to identify items and associate with customers, without relying on human-in-loop approaches.

## ACKNOWLEDGEMENT

The authors would like to thank Carnegie Mellon University, AiFi Inc., especially João Diogo Falcão, Pei Zhang, for their help and support.

## REFERENCES

[1] Carlos Ruiz, Joao Falcao, Shijia Pan, Hae Noh, and Pei Zhang. Aim3s: Autonomous inventory monitoring through multi-modal sensing for cashier-less convenience stores. pages 135–144, 11 2019. ISBN 978-1-4503-7005-9. doi: 10.1145/3360322.3360834.

[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[3] Adrian Beck. Self-scan checkouts and retail loss: Understanding the risk and minimising the threat. *Security Journal*, 24:199–215, 2011.