Supplementary Material
RetailAction: Dataset for Multi-View Spatio-Temporal Localization of Human-Object Interactions in Retail

Davide Mazzini Alberto Raimondi Bruno Abbate Daniel Fischetti David M Woollard
STANDARD.AI
{davide, alberto, bruno, dan, david.woollard}@standard.ai

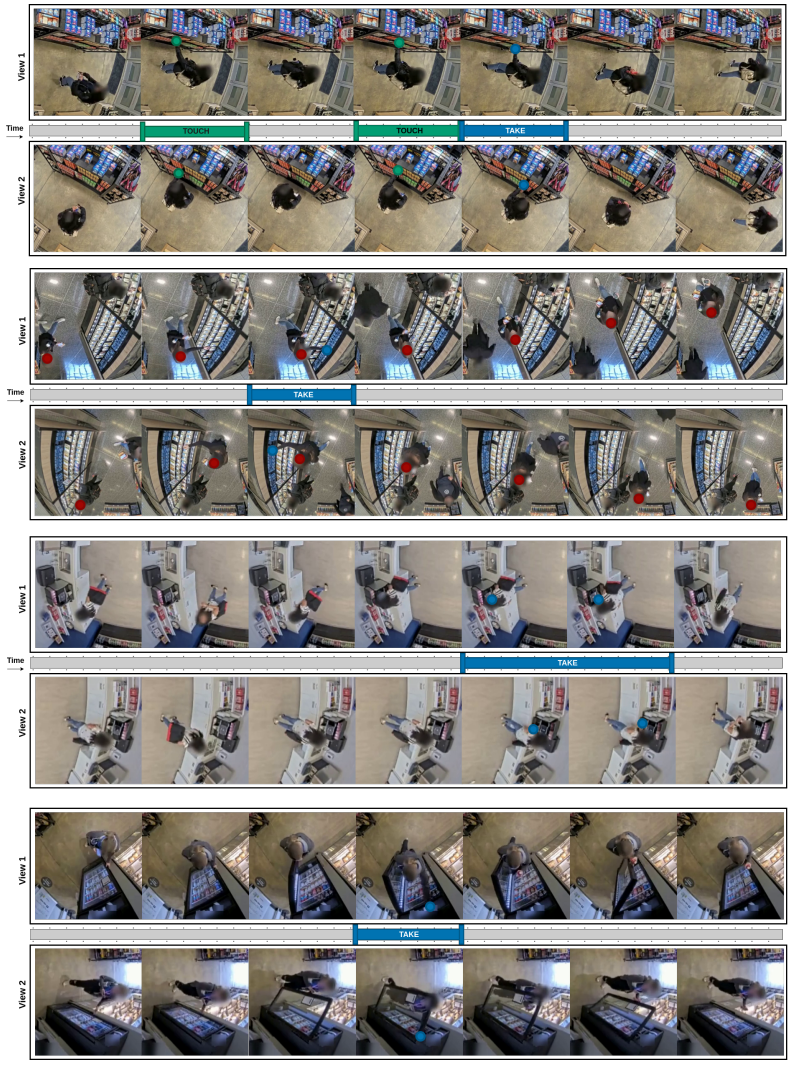


Figure 1. An annotated sample from the RetailAction dataset, consisting of two synchronized video streams from different cameras. The videos capture real people interacting in actual retail stores. The subject performs two actions: taking an item and placing another on a shelf. Each action is annotated with a categorical label (take, put, touch), a temporal range, and spatial coordinates in both views.

1. Dataset Examples

Figure 1 presents additional examples of the RetailAction dataset. These examples illustrate cases with multiple people, multiple actions, or interactions with different types of shelves.

2. Additional Dataset Statistics

In this section, we provide additional statistics about the dataset. In particular, Figures 2 and 3 show the distribution of action duration and segment duration in frames. These graphs can be compared with those representing the duration in seconds of segments and actions (Figure 3 in the paper) to have an idea of the FPS of the generated videos. As mentioned and as shown in Figure 3, the maximum duration of videos in frames is 32 frames (sampled using a frame scoring algorithm). This results in a variable FPS, ranging from 10 FPS to lower values, which can be inferred by considering the segment duration in seconds. Regarding the action distribution, it is important to consider that segments with very short durations in seconds but close to the maximum length of 32 frames will have a higher FPS. As a result, short actions will still be represented by a higher number of frames, which explains the tail in Figure 2.

3. Data Collection Implementation Details

In this section, we provide additional details about the data collection process. As in the paper, the goal is not to present the pipeline at a level of detail that would allow full reproducibility—such a description would be too complex and not sufficiently novel to be covered adequately in these supplemental materials. Rather, our aim is to clarify the role of each component and highlight the complexity involved in the development of the pipeline. Understanding the data collection process can also help readers better interpret the collected data.

As described in the main paper, the automated video generation process consists of multiple steps. First, 3D-tracked poses are generated over time for each shopper. Next, time intervals corresponding to potential interactions with items and shelves are identified. Finally, the parameters for selecting and generating the interaction videos are determined, followed by the actual generation and anonymization of the videos.

3.1. 3D Tracked Pose Generation

2D Pose Estimation – For each camera, a dedicated service runs a 2D pose detection model at 10 FPS, detecting all people within the frame and providing their 2D poses with 24 keypoints per frame. Traditional 2D pose estimation models do not perform sufficiently well on 360-degree top-view

Figure 2. Histogram of action durations in frames.

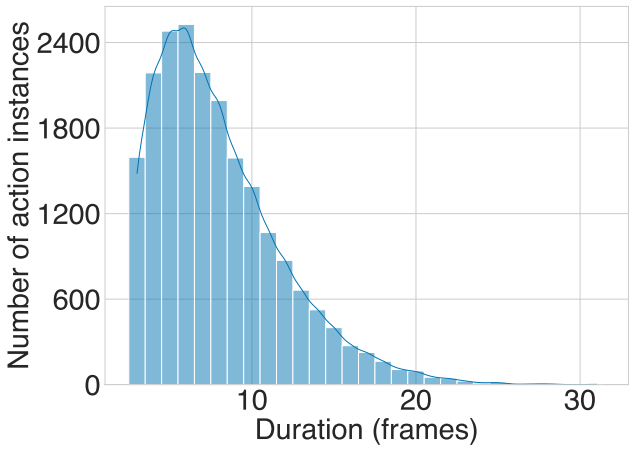
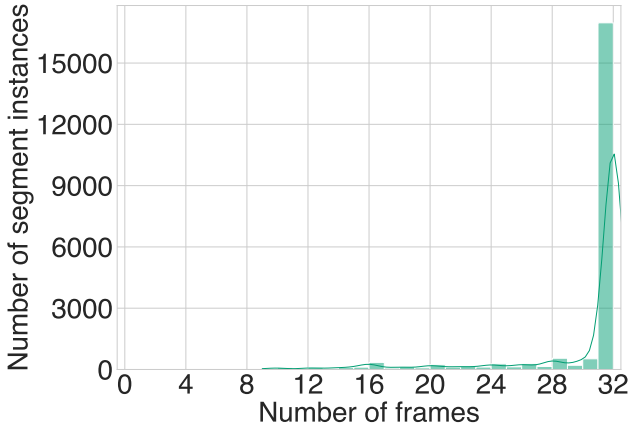


Figure 3. Distribution of segment durations in frames.



cameras. For this reason, we used a version of the PersonLab model[**?**], trained on a proprietary dataset composed of 360-degree views and 2D pose annotations collected from multiple stores. While we are not able to fully report its performance in this paper due to space limitations, the model outperforms many state-of-the-art general-purpose pose detection models, even on stores that were not included in its training set.

Tracking and Triangulation – The 2D poses detected across all cameras are processed by a centralized service that performs multi-view matching and cross-frame tracking of individuals. This service reconstructs the 3D trajectory of each person and generates 3D poses with 24 keypoints tracked at 10 FPS for each shopper in the store. The system operates in three steps: first, 2D poses from each

camera are temporally tracked to produce camera-tracklets. Then, tracklets from different cameras are clustered based on triangulation error to create unified tracks, which are finally triangulated to obtain 3D position estimates.

3.2. Identification of Interaction Time Intervals

For each store, a model of shelf positions and dimensions has been created to provide a spatial reference for tracking interactions. Using this mapping, a kinematic model based solely on the 3D trajectories of individuals and the shelf positions is employed to detect the segments of interest, i.e., time intervals with a high probability of containing real interactions with shelf items (referred to as "interaction segments"). The kinematic model consists of two key steps: Kinematic Interaction Detection and Segment Merging and Splitting.

Kinematic Interaction Detection – This step is carried on by a Graph Convolutional Network [] that operates on 6.4-second windows and predicts, for 0.4-second sub-windows, whether an interaction is occurring, providing a confidence score. Input to the model includes the 3D pose of the individual and the relative distances between the upper body joints (hands, wrists, elbows, shoulders, neck) and the nearest shelf. The model is trained in a supervised manner using labeled data, where labels correspond to time intervals in which shoppers are picking up or placing items on shelves.

Segment Merging and Splitting – A deterministic post-processing step refines the detected interaction segments by recursively merging consecutive 0.4-second windows with positive interaction predictions. Subsequent logic is responsible for merging these small windows to generate what we call interaction segments, aiming to capture complete actions from start to finish, including some context, without splitting them while still keeping distinct actions in separate segments.

For instance, if two actions occur far apart in time, they should be assigned to different segments. Similarly, if actions take place in sufficiently distant locations or involve different shelves that are far from each other (e.g., one in front of the other), they should remain separate.

To achieve this, an algorithm iterates over all small windows where an interaction is predicted—i.e., above a certain confidence threshold, which is set very high to ensure we meet the required target for our application (99%). The windows are analyzed sequentially, and merging occurs only if the following conditions are met:

- The time gap between the segments is below a predefined threshold (fine-tuned to 1.5s).
- At least one of the following holds:
  - The person's average position across the two segments is within a predefined distance threshold (1m).
  - The person is not oriented towards two different shelves in the two segments.
- The model's interaction confidence scores at the segment boundaries are sufficiently high.

An additional temporal padding (fine-tuned to 0.6s) is applied when merging. All these thresholds were fine-tuned qualitatively by reviewing thousands of events.

3.3. Video Selection

Best Camera Selection – For each interaction segment, the system selects two video streams from the most suitable cameras to ensure optimal visibility of the interaction. A scoring algorithm evaluates each frame based on whether the person is visible, potential occlusions caused by shelves, and the visibility of the upper body joints, particularly the hands. In a second step, a score grouping algorithm aggregates the scores across frames within the interaction segment to finally provide a unique ranking of the cameras per segment and select the two best views.

Spatial Cropping Selection – Once the temporal intervals have been selected and the camera that best captures the action has been identified, a crop must be selected within the frame from the fisheye camera. The goal is to focus only on the user performing the interaction, ensuring that the entire action is fully visible without cutting out relevant parts such as arms or items, while also avoiding excessive context that is unnecessary for action recognition. To achieve this, we use a model to select the crop, which analyzes the 2D poses of the person in the selected camera view, tracking all the pixels occupied by their hands throughout the segment. The 3D left and right hand positions over time, denoted as $\mathbf{h}_t^{(3D)}$, are first projected onto the camera image plane. The cropping box is then determined by computing the minimum and maximum projected coordinates $\mathbf{h}_t^{(2D)}$ across the segment, with an additional padding term p_x, p_y that accounts for the person's distance from the camera in order to always correspond to 0.4 meters:

$$x_{\min} = \min_t h_{t,x}^{(2D)} - p_x, \quad x_{\max} = \max_t h_{t,x}^{(2D)} + p_x,$$
$$y_{\min} = \min_t h_{t,y}^{(2D)} - p_y, \quad y_{\max} = \max_t h_{t,y}^{(2D)} + p_y.$$

This ensures that the cropped region encompasses the entire hand movement while preserving relevant contextual information.
Algorithm 1 shows pseudo-code that describes the spatial cropping selection process, including the extraction of 3D hand positions, projection onto the image plane, and computation of the final crop bounds.

Temporal Frame Scoring – Finally, a frame selection algorithm assigns a temporal importance score to each frame 115 based on kinematic information. This component is crucial because these scores are used to perform subsampling, removing only the less relevant parts of the

video—for example, moments where the person remains still for a long time. This algorithm relies solely on kinematic information, specifically the person's 3D pose. As mentioned in the paper, this scoring algorithm analyzes the velocity and acceleration of the person's hands, prioritizing frames with significant movement while downsampling static frames. In particular, the temporal score for each frame t is computed as:

$$\text{score}(t) = \alpha \left(\frac{a(t)}{A_{\text{max}}} \right) + \beta \left(\frac{v(t)}{V_{\text{max}}} \right) + (1 - \alpha - \beta) \cdot c(t)^2$$

Where $a(t)$ and $v(t)$ are the acceleration and the velocity at time t, A_{max} and V_{max} are the maximum expected acceleration and velocity, $c(t)$ is the confidence score of the kinematic interaction detection at time t, α and β are parameters to give different weights to velocity and acceleration components.

In Algorithm 2 we provide a pseudo-code description of how this algorithm works. Once again, we emphasize that this method has proven to be qualitatively effective in selecting the most important frames.

Anonymization and Storage – A final processing step applies facial blurring to anonymize all individuals appearing in the videos. Face detection is performed using 2D pose predictions. Additionally, all timestamps are anonymized by resetting them to the reference date of 1970, and any references to store names are either removed or blurred if visible in the videos. Once anonymization is complete, videos are stored alongside metadata.

Algorithm 1: Spatial Cropping Selection

Input: Camera ID cam_num, Person Track $person\_track$, Padding Size $bounds\_pad$
Output: Cropping Bounds $(x_{\min}, x_{\max}, y_{\min}, y_{\max})$

```
/* Extract 3D positions of left and right hands over time */
hand_pos_3d ← get_hand_positions(person_track) ;

/* Retrieve camera model */
camera ← get_camera(cam_num) ;

if len(hand_pos_3d) = 0 then
    return None /* No valid hand positions found */

/* Project 3D hand positions onto 2D image plane */
proj ← camera.project_points(hand_pos_3d) ;

/* Compute crop boundaries */
x_min ← min(proj[:, 0]),   x_max ← max(proj[:, 0]) ;
y_min ← min(proj[:, 1]),   y_max ← max(proj[:, 1]) ;

/* Apply padding to maintain a fixed real-world size of bounds_pad (0.4m) */
if bounds_pad > 0 then
    (p_x, p_y) ← get_padding_based_on_distance(camera, hand_poses_array, bounds_pad) ;
    if not any(isnan([p_x, p_y])) then
        x_min ← x_min − p_x,   x_max ← x_max + p_x ;
        y_min ← y_min − p_y,   y_max ← y_max + p_y ;

return (x_min, x_max, y_min, y_max)
```

Algorithm 2: Temporal Frame Scoring Algorithm

Input: $left\_hand\_pos, right\_hand\_pos$: 3D hand positions over time, $times$: corresponding timestamps, $model\_confidences$: list of model confidences for each timestamp

Output: $scores$: list of timestamps and their corresponding scores

$MAX\_SPEED \leftarrow 1.5$ /* m/s */
$MAX\_ACCELERATION \leftarrow 30.0$ /* m/s^2 */
$SPEED\_WEIGHT \leftarrow 0.35$
$ACCELERATION\_WEIGHT \leftarrow 0.6$
$lookback \leftarrow 3$
$default\_speed \leftarrow MAX\_SPEED/5$, $default\_acceleration \leftarrow MAX\_ACCELERATION/10$

$SCORE\_KERNEL \leftarrow array([0.6, 0.7, 0.8, 0.9, 1.0, 0.9, 0.8, 0.7, 0.6])$
$SCORE\_KERNEL \leftarrow SCORE\_KERNEL/sum(SCORE\_KERNEL)$ /* This kernel gives more weight to the central part of the segment, emphasizing the middle frames while reducing the influence of frames at the edges of the segment. */

$prev\_left, prev\_left\_velocity, prev\_right, prev\_right\_velocity \leftarrow []$
$scores \leftarrow []$

foreach $(left\_hand, right\_hand, ts, conf) \in (left\_hand\_pos, right\_hand\_pos, times, model\_confidences)$ **do**
  $speeds, accelerations \leftarrow [], []$
  foreach $(current\_hand, prev\_hands, prev\_velocity) \in [(left\_hand, prev\_left, prev\_left\_velocity), (right\_hand, prev\_right, prev\_right\_velocity)]$ **do**
    $hand\_is\_none \leftarrow (current\_hand = None) \vee (\texttt{isnan}(current\_hand[0]))$
    if *len*$(prev\_hands) = 0 \vee hand\_is\_none$ **then**
      $speed \leftarrow default\_speed$, $acceleration \leftarrow default\_acceleration$
      if *not* $hand\_is\_none$ **then**
        Append $(current\_hand, ts)$ to $prev\_hands$
    else
      $velocity \leftarrow \frac{1}{lookback} \sum_{prev \in prev\_hands[-lookback:]} \frac{(current\_hand - prev[0])}{ts - prev[1]}$
      $speed \leftarrow ||velocity||$
      if *len*$(prev\_velocity) = 0$ **then**
        $acceleration \leftarrow default\_acceleration$
      else
        $acceleration \leftarrow ||\frac{1}{lookback} \sum_{prev \in prev\_velocity[-lookback:]} \frac{(velocity - prev[0])}{ts - prev[1]}||$
      Append $(current\_hand, ts)$ to $prev\_hands$
      Append $(velocity, ts)$ to $prev\_velocity$
    Append $speed$ to $speeds$, Append $acceleration$ to $accelerations$
  $speed \leftarrow \min(MAX\_SPEED, \max(speeds))$
  $acceleration \leftarrow \min(MAX\_ACCELERATION, \max(accelerations))$
  $score \leftarrow ACCELERATION\_WEIGHT \times \frac{acceleration}{MAX\_ACCELERATION} + SPEED\_WEIGHT \times \frac{speed}{MAX\_SPEED}$
  $+(1 - SPEED\_WEIGHT - ACCELERATION\_WEIGHT) \times conf^2$
  Append $(ts, score)$ to $scores$
if *len*$(scores) > 0$ **then**
  $scores\_array \leftarrow [score \text{ for } (\_, score) \in scores]$
  $scores\_array \leftarrow \texttt{convolve}(scores\_array, SCORE\_KERNEL)$
  $scores \leftarrow [(ts, score) \text{ for } (ts, \_), score \text{ in } \texttt{zip}(scores, scores\_array)]$
return $scores$