

Article

# Bidirectional Tracking Method for Construction Workers in Dealing with Identity Errors

Yongyue Liu , Yaowu Wang and Zhenzong Zhou \* 

School of Civil Engineering, Harbin Institute of Technology, Harbin 150090, China; liuyongyue@hit.edu.cn (Y.L.); ywwang@hit.edu.cn (Y.W.)

\* Correspondence: zhenzongzhou@hit.edu.cn

**Abstract:** Online multi-object tracking (MOT) techniques are instrumental in monitoring workers' positions and identities in construction settings. Traditional approaches, which employ deep neural networks (DNNs) for detection followed by body similarity matching, often overlook the significance of clear head features and stable head motions. This study presents a novel bidirectional tracking method that integrates intra-frame processing, which combines head and body analysis to minimize false positives and inter-frame matching to control ID assignment. By leveraging head information for enhanced body tracking, the method generates smoother trajectories with reduced ID errors. The proposed method achieved a state-of-the-art (SOTA) performance, with a multiple-object tracking accuracy (MOTA) of 95.191%, higher-order tracking accuracy (HOTA) of 78.884% and an identity switch (IDSW) count of 0, making it a strong baseline for future research.

**Keywords:** multi-object tracking; worker tracking; head-integrated; intra-frame processing; inter-frame matching; Kalman filter

**MSC:** 68U10



**Citation:** Liu, Y.; Wang, Y.; Zhou, Z. Bidirectional Tracking Method for Construction Workers in Dealing with Identity Errors. *Mathematics* **2024**, *12*, 1245. <https://doi.org/10.3390/math12081245>

Academic Editors: Faheim Sufi and Jonathan Blackledge

Received: 2 March 2024

Revised: 2 April 2024

Accepted: 18 April 2024

Published: 19 April 2024

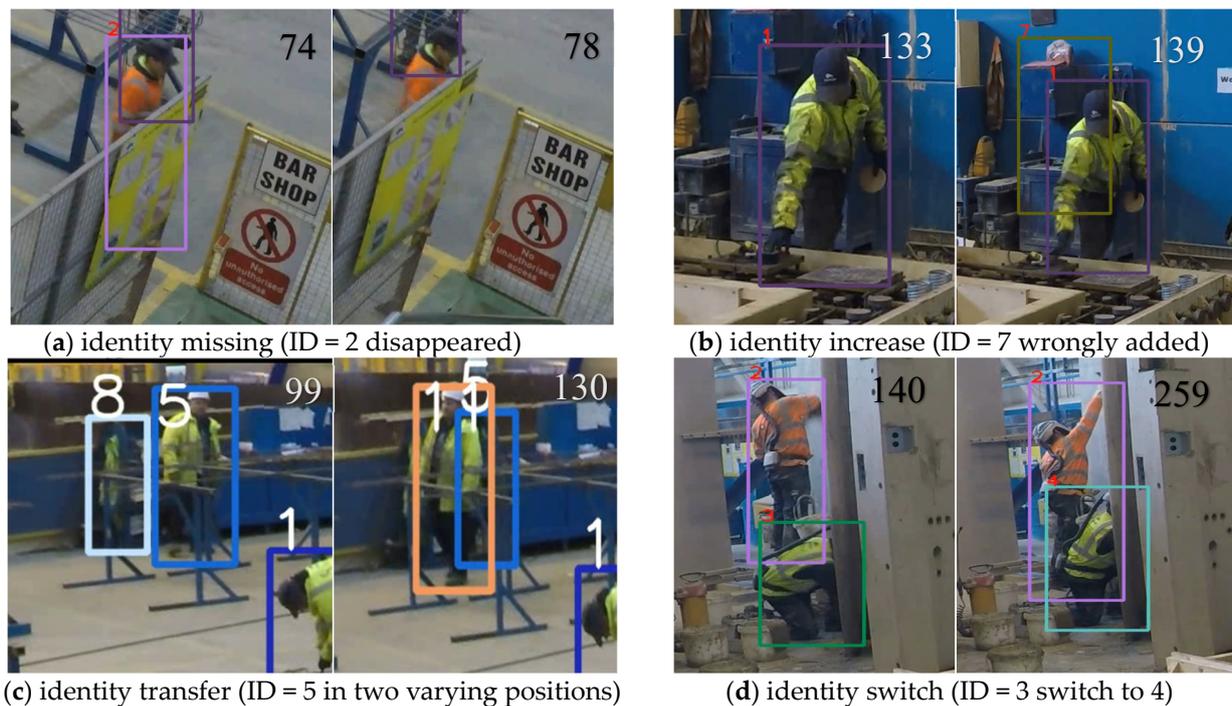


**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The efficacy of worker tracking in construction safety management, both on-site [1] and off-site [2], has been widely acknowledged. The real-time monitoring of workers' locations and identities enables managers to monitor working conditions closely, facilitating prompt interventions to prevent potential accidents [3], such as ensuring helmet usage, maintaining safe distances and avoiding collisions with vehicles [4]. By analyzing tracking data, valuable insights can be derived into workers' productivity, intensity, movement patterns and other critical aspects, ultimately optimizing construction processes and labor allocation, enhancing efficiency, minimizing resource waste and reducing labor-intensive tasks for workers [5].

Nonetheless, complex construction scenarios pose challenges in feature extraction and identity matching, leading to identity-tracking errors. Four types of errors are illustrated in Figure 1: identity missing, identity increase, identity switch (IDSW) and identity transfer (IDTR) [6]. Identity missing occurs when false negatives (FNs) in deep learning models result in missed identifications, as demonstrated by worker ID = 2 in Figure 1a, who was occluded by "Bar shop" in frame #78. Identity increase is due to false positives (FPs), as exemplified by worker ID = 7 in Figure 1b, where an incorrect bounding box detection led to an additional ID. IDSW refers to a single person being assigned different IDs consecutively, as in Figure 1c, where worker ID = 5 had different IDs in frames #99 and #130. Conversely, IDTR occurs when two individuals are mistakenly assigned the same ID, as shown in Figure 1d, where worker ID = 3 switched to ID = 4 between frames #140 and #259. Both IDTR and IDSW stem from issues with association algorithms, which can be interpreted as "accepting false ID" and "rejecting truth ID".



**Figure 1.** Types of identity errors in tracking workers.

Current MOT methods have some intrinsic limitations:

(1) Relying on detection DNNs [7] limits the number of tracked individuals to the detected count per frame. Missed detections lead to a lost person's re-identification (ReID) [8] information and, subsequently, motion estimation for the corresponding sequence, resulting in tracking ID loss.

(2) Current motion estimation techniques are confined to the Kalman filter (KF) [9], which may not converge mathematically. In cases of divergence, the KF's estimates become meaningless and can cause incorrect motion predictions.

(3) Worker tracking, unlike pedestrian tracking, encounters more occlusion challenges in complex scenarios [10], posing higher ID error risks. While pedestrian tracking algorithms typically overlook ID-related metrics, worker tracking, which is crucial for safety management, necessitates their consideration, particularly in MOT challenge-like contexts [11,12].

(4) Head information, which is vital for worker identification, has often been overlooked [2]. Its use is pivotal due to workers' upright posture and less obstructed heads, which can enhance whole-body tracking by leveraging bidirectional matching algorithms. Head tracking, as it aids in body detection and tracking, exploits this relationship to improve robustness.

The main innovation of this article is to provide a simple bidirectional method for tracking workers under complex construction scenarios. This bidirectional interaction improves the accuracy and robustness of body tracking while also addressing issues such as occlusion and environmental changes. The proposed method can significantly improve the safety and efficiency of construction sites by enabling the tracking of worker positions and movements with fewer ID errors.

The contributions of this research are as follows:

(1) Adopting head-tracking-to-body-tracking models in the construction field for the first time and using head motion state to correct whole-body movement.

(2) In the case where KF cannot be abandoned for movement estimation, modifying the body speed into head speed and making it comply with the basic "near-big, far-small" perspective principle.

(3) Changing the unidirectional method to bidirectional and letting tracking outputs correct current frame detections.

(4) Demonstrating the effectiveness of this method from a mathematical perspective.

## 2. Related Works

Vision-based MOT methods aim to identify target IDs (humans, vehicles, etc.) across successive frames. MOT usually consists of three subtasks [13]: object detection, feature extraction and data association.

- Object detection (observation model): using a DNN such as Faster R-CNN [14], YOLO [15], CenterNet [16], or Transformer [17] to obtain the region of interest (ROI) [18], also called the bounding box.
- Feature extraction (appearance model): employing a person re-identification (ReID) [8] network to extract a unidimensional vector from the ROI.
- Data association (motion estimation and linear assignment): matching the IDs [7] between the targets detected by the DNN at the current frame and estimated using the KF at the former frame and the similarity cost matrix with Hungarian [19] or Jonker–Volgenant [7] algorithms.

According to the literature [2], instance segmentation has been posited to enhance detection performance, as exemplified by Xiao et al.'s [2] adaptation of the Mask R-CNN [20] for worker tracking. However, their study relies on datasets without annotations for heavily occluded individuals (occlusion rate above 70%), which precludes the possibility of improving the data association process, as previously noted by [10].

There are two basic paradigms for MOT [21]: tracking by detection (TBD) and joint detection and tracking (JDT).

- TBD treats object detection as a separate detector, while feature extraction and data association are considered trackers [8]. TBD offers the advantage of flexibility in replacing modules with better DNNs or association methods. However, the detector and the tracker cannot enhance each other's performance. If the detector produces missing or false bounding boxes, this will result in the tracker's failure to track or accurately identify the target.
- JDT integrates the detector and tracker into one unified network that can be trained end-to-end, such as Siamese [22] or Transformer networks [23]. JDT relies exclusively on appearance features, but the training of DNNs often demands better GPUs and takes significant time. For instance, TransTrack [24] demands 16.1 GB of GPU memory for inference, making it incompatible with the NVIDIA Tesla T4 (15 GB) on Google Colaboratory [25]. To provide a simple and training-free tracking method, TBD is the better choice.

Matching algorithms in TBD use various cues, including motion information [9], appearance features [26], velocity direction, confidence and height state [27], to calculate the different similarity distances (the cosine distance, the squared Mahalanobis distance, the intersection over union (IoU) distance, etc.) for the cost matrix in the linear assignment problem.

However, unlike pedestrian tracking on roads, the motion of workers always features long stays or frequent crossings [28], which may lead to the continuous growth of the covariance matrix in KF. In practical scenarios, a large covariance matrix can lead to the acceptance of false IDs. Conversely, a small covariance matrix can result in the rejection of the true ID. Hence, it is essential to carefully consider the covariance matrix size to ensure accurate identification. Convergence failure of the covariance matrix implies a mismatch between the selected motion model of the KF and actual motion behavior, but it does not seem to offer an alternative solution for the linear estimator [9].

To provide a comprehensive overview of data association techniques in the MOT domain, we present the technical aspects in Table 1. Contemporary methods focus on retraining detection DNNs for improved input accuracy, while others refine Kalman fil-

ter state vectors. Given our emphasis on data association, object detection and feature extraction details are not elaborated upon in this section.

The SOTA tracking-by-detection methods, as depicted in Table 1, are unidirectional, relying solely on the outputs of detection DNNs. Consequently, their tracking capacity is inherently limited to the number of detected objects, and the tracking ID count never exceeds the detection ID count. Our proposed bidirectional tracking approach encompasses two types of interactions, namely, head-to-body and detector–tracker, which enable the mutual correction of tracking and detection results, enhancing the overall performance.

**Table 1.** Strengths and limitations of SOTA methods.

No.	SOTA Methods	Year	Information Types	Advantages (A) & Shortcomings (S)
1	SORT [19]	2016	O: Faster R-CNN; A: None; M: KF + IoU + Hungarian.	A: presented as the baseline; KF state is $x_t = [x_c, y_c, s, r, v_x, v_y, v_s]^T$ , $s$ = aspect ratio and $r$ = area. S: highly dependent on detection performance and many IDSWs; no occlusion-solving considerations.
2	DeepSORT [8]	2017	O: Faster R-CNN; A: ReID (128-d); M: KF + Cosine distance + IoU + Hungarian.	A: presented as the baseline integrate appearance information. KF state is $x_t = [x_c, y_c, \gamma, h, v_x, v_y, v_\gamma, v_h]^T$ , $\gamma$ = aspect ratio and $h$ = height. S: detection performance dependency; occlusion-related IDSWs reduced but still frequent; constant-velocity model.
3	ByteTrack [7]	2022	O: re-trained YOLOX-x on 1400 videos; A: ReID (1024-d); M: KF + Cosine distance + IoU + Hungarian + low scores re-match.	A: best performance in MOT20 and with already published codes. KF state is $x_t = [x_c, y_c, a, h, v_x, v_y, v_a, v_h]^T$ ; $a$ = aspect ratio and $h$ = height. S: highly dependent on detection performance and many IDSWs; no occlusion considerations; constant-velocity model.
4	OC_SORT [9]	2022	O: baseline detections in MOTChallenge; A: None; M: KF + IoU + Hungarian + re-update of KF + motion direction difference.	A: first to explain the KF predict errors accumulation in detail; motion direction difference is added in the association cost matrix. KF state is $x_t = [x_c, y_c, a, s, v_x, v_y, v_a]^T$ , $a$ = area and $s$ = aspect ratio. S: no real online method for KF update; needs future frame; constant-velocity assumption during occlusion, cannot remain effective during long-term occlusions; detection performance dependency; constant-velocity model.
5	Deep OC_SORT [26]	2023	O: YOLOX; A: ReID (SBS50, 287MB) + Camera Motion Compensation + Dynamic Appearance; M: KF + IoU + Hungarian + re-update of KF + motion direction difference.	A: Apply Camera Motion Compensation to correct the KF state for better locations of the bounding box; apply detection confidence to modify ReID output vectors. KF state is $x_t = [x_c, y_c, a, s, v_x, v_y, v_a]^T$ ; $a$ = area and $s$ = aspect ratio. S: the same as the OC_SORT; constant-velocity model.
6	BoTSORT [29]	2022	O: Faster R-CNN; A: ReID + Camera Motion Compensation; M: KF + cosine distance + IoU + Hungarian.	A: modify KF state to $x_t = [x_c, y_c, s, a, v_x, v_y, v_s]^T$ , $s$ = area and $a$ = aspect ratio; apply Camera Motion Compensation to reduce errors of moving cameras; apply new cost matrix with weights of appearance cost and motion cost. S: the same as the OC_SORT; constant-velocity model; slow when working with sparse optical flow.
7	Strong_SORT [30]	2022	O: YOLOX-x; A: ReID (BoT) + Camera Motion Compensation; M: NSA-KF + cosine distance + IoU + Hungarian.	A: apply a new cost matrix with weights of appearance cost and motion cost; KF state is $x_t = [x_c, y_c, a, h, v_x, v_y, v_a, v_h]^T$ ; $a$ = aspect ratio and $h$ = height. S: MOTA is slightly lower, mainly due to the high detection score threshold leading to many missing detections; working speed is not high.
8	TransTrack [24]	2020	O: re-trained transformer; A: None; M: None.	A: Self-Attention Mechanism and Query-Key pipeline. S: hard to train; no motion information utilization; JDT not better than TBD in performance.
9	UniTrack [31]	2021	O: ResNet-50; A: ImageNet-supervised appearance model; M: KF + cosine distance + IoU + Hungarian.	A: can support different tracking tasks and leverage many existing general appearance models. KF state is $x_t = [x_c, y_c, a, h, v_x, v_y, v_a, v_h]^T$ ; $a$ = aspect ratio and $h$ = height. S: not better in terms of metrics performance.

Table 1. Cont.

No.	SOTA Methods	Year	Information Types	Advantages (A) & Shortcomings (S)
10	FairMOT [32]	2020	O: encoder–decoder; A: encoder–decoder; M: KF + cosine distance + IoU + Hungarian.	A: one encoder–decoder network to obtain observation and appearance at the same time, with no need for an independent ReID model; S: need training for about 30 h on two RTX 2080 Ti GPUs; still a SORT-related method.
11	TransMOT [33]	2021	O: spatial–temporal graph Transformer; A: None; M: KF + cosine distance + IoU + Hungarian.	A: A cascaded association structure to handle low confidence detection and long-term occlusion. S: relatively large computing resources and data; no public codes.
12	Hybrid-SORT [27]	2023	O: YOLOX-x; A: ReID + Camera Motion Compensation; M: KF + cosine distance + IoU + Hungarian + weak cues.	A: apply a new cost matrix with weights of appearance cost, motion cost, four corners’ velocity direction, and height-modulated IoU. KF state is $x_t = [x_c, y_c, s, c, r, v_x, v_y, v_s, v_c]^T$ ; $r$ = aspect ratio, $s$ = area and $c$ = confidence score. S: detection performance dependency.
13	Xiao et al. [2]	2023	O: Mask R-CNN; A: ReID (128-d); M: KF + cosine distance + IoU + Hungarian.	A: baseline in worker tracking; S: needs retraining on a new dataset; does not address severe occlusions; no public codes.

Observation DNN: O; appearance DNN: A; motion estimation model and assignment algorithm: M.

### 3. Motion Estimation with KF

#### 3.1. Basic KF Formula

Motion data, including position and velocity, can be effectively extracted using a linear Kalman filter (KF) [10]. The KF estimates the state of a dynamic system from noisy measurements, employing a state equation and an observation equation, as depicted in Equations (1) and (2).

In Equation (1), the prior state vector estimate (denoted by  $\bar{x}_t$ ) is predicted based on the posterior distribution from the previous frame (denoted by  $\hat{x}_{t-1}$ ). The state vector  $x_t = [x_c, y_c, a, h, v_x, v_y, v_a, v_h]^T$  is  $(8 \times 1)$  in frame  $t$  and consists of the bounding box center, with dimensions, width, height,  $a = w/h$  and velocity. The transition matrix  $F_t$  governs the system dynamics, while  $Q_t$  ( $8 \times 8$ ) represents process noise, following a normal distribution.

$$\bar{x}_t = F_t \hat{x}_{t-1} + \omega_t, \quad \omega_t \sim N(0, Q_t) \tag{1}$$

Equation (2) accounts for imperfect estimation, where the measurement  $z_{t-1} = [x_c, y_c, a, h]^T$  (a  $4 \times 1$  vector in frame  $t - 1$ ) and the projected posterior distribution are subject to observation noise (denoted by  $R_t$ , a  $4 \times 4$  matrix). The observation matrix  $H$  facilitates dimensional transformation between state and measurement vectors.

$$z_{t-1} = H_t \bar{x}_{t-1} + v_t, \quad v_t \sim N(0, R_t) \tag{2}$$

The KF’s prediction for the covariance matrix  $P_t$  is shown in Equation (3). Since elements in the vector can vary widely, a covariance matrix is used to quantify the variation between elements.  $P_t$  ( $8 \times 8$ ) is updated regardless of the presence of a tracking ID, with a “dummy update” [9] occurring during tracking, which can lead to error accumulation and non-convergence.

$$\bar{P}_t = F_t \hat{P}_{t-1} F_t^T + Q_t \tag{3}$$

The Kalman gain, denoted in Equation (4), determines the influence of the current observation on state estimation. A higher value of  $K_t$  indicates greater confidence in the

detection, while a lower value suggests a greater confidence in the prediction. Elements of  $K_t$  (an  $8 \times 4$  matrix) are confined to the range  $[0, 1]$  during normal operation.

$$K_t = \frac{\bar{P}_t H_t^T}{(H_t \bar{P}_t H_t^T + R_t)} = \hat{P}_t H_t^T R_t^{-1} \tag{4}$$

Finally, Equations (5) and (6) describe the posterior update of the state vector and covariance matrix, respectively, incorporating the prior estimate and current observation data.

$$\hat{x}_t = \bar{x}_t + K_t(z_t - H_t \bar{x}_t) = F_t \hat{x}_{t-1} + K_t(z_t - H_t F_t \hat{x}_{t-1}) \tag{5}$$

$$\hat{P}_t = (I - K_t H_t) \bar{P}_t \tag{6}$$

### 3.2. KF Divergence Proof

This section elucidates the mechanism behind the non-convergence of the covariance matrix in the KF. When the employed motion estimation model in the KF mismatches the actual dynamics, iterative updates lead to divergence. Given the practical challenge of acquiring a precise motion model for humans, the KF often encounters the following issue: If the KF divergence  $\rightarrow$  elements in the covariance matrix  $P_t$  is larger  $\rightarrow$  elements in the inverse covariance matrix  $P_t^{-1}$  is smaller  $\rightarrow$  Mahalanobis distance of two different persons' IDs is smaller than the threshold  $\rightarrow$  leading to "accepting false ID" errors

Consequently, addressing the KF divergence problem is crucial for reducing such identity errors. From the perspective of Bayesian probabilities, (1) can be expressed as  $p(\bar{x}_t) = N(\mu, \bar{P}_t)$ , and if  $\bar{x}_t$  is given, (2) can be expressed as  $p(z_t|\bar{x}_t) = N(H_t \mu, R)$ ; then,  $p(\hat{x}_t|z_t) = p(z_t|\bar{x}_t) \times p(\bar{x}_t)$ , so the corresponding equation of the covariance matrix is as described as follows (7):

$$\hat{P}_t^{-1} = H_t^T R^{-1} H_t + \bar{P}_t^{-1} \tag{7}$$

Assuming a simple constant-velocity model, the ground truth motion yields (8):

$$\begin{aligned} x_t^* &= x_{t-1}^* + v^* = x_0^* + tv^* \\ z_t^* &= x_t^* + \varepsilon_t = x_0^* + tv^* + \varepsilon_t \end{aligned} \tag{8}$$

Assuming the estimation model is not the same as the ground truth yields (9):

$$\begin{aligned} x_t &= x_{t-1} + v = x_0 + tv, \quad v \neq v^* \& x_0 = x_0^* \\ z_t &= x_t + \varepsilon_t = x_0 + tv + \varepsilon_t \end{aligned} \tag{9}$$

For one element  $p_t$  in the covariance matrix, assuming there is no relevant prior knowledge of noise and initial value at  $t = 0$ , let  $r_0 = \sigma^2$ ,  $p_0 = \infty$ . For the constant-velocity model,  $f_t = h_t = 1$ , the following is obtained (10):

$$\begin{aligned} p_t^{-1} &= [f_t p_{t-1} f_t^T]^{-1} + [h_t r h_t^T]^{-1} = [p_{t-1}^{-1}] + [\sigma^{-2}] \\ &= ([p_{t-2}^{-1}] + [\sigma^{-2}]) + [\sigma^{-2}] \\ &\dots \\ &= [p_0^{-1}] + t \times [\sigma^{-2}] \\ &= \frac{1}{\infty} + \frac{t}{\sigma^2} = \frac{t}{\sigma^2} \end{aligned} \tag{10}$$

(4) can be rewritten in one element of the vector, which yields (11):

$$k_t = \hat{p}_t h_t^T r_t^{-1} = \frac{\sigma^2}{t} \frac{1}{\sigma^2} = \frac{1}{t} \tag{11}$$

Substituting (2) into (5) as one element of the posterior state vector yields (12):

$$\begin{aligned}
 \hat{x}_t &= f_t \hat{x}_{t-1} + k_t(z_t - h_t f_t \hat{x}_{t-1}) = (\hat{x}_{t-1} + v) + \frac{1}{t}(z_t - \hat{x}_{t-1} - v) \\
 &= \frac{t-1}{t}(\hat{x}_{t-1} + v) + \frac{1}{t}z_t \\
 &= \left( \underbrace{\frac{t-2}{t}}_{\rightarrow 0}(\hat{x}_{t-2} + v) + \frac{1}{t}z_{t-1} \right) + \frac{1}{t}z_t \\
 &\dots \\
 &= \frac{1}{t}(z_1 + z_2 + z_3 + \dots + z_t) \\
 &= \frac{1}{t}(x_0 + v + \varepsilon_1) + \frac{1}{t}(x_0 + 2v + \varepsilon_2) + \dots + \frac{1}{t}(x_0 + tv + \varepsilon_t) \\
 &= x_0 + \frac{1}{t} \frac{(1+t)*t}{2}(v + \varepsilon) = x_0 + \frac{(t+1)}{2}v + \frac{1}{t} \sum_{i=1}^t \varepsilon_i
 \end{aligned} \tag{12}$$

Under the constant-velocity assumption, the discrepancy between the ground truth and posterior estimates (13) exhibits an increasing trend with time. This observation implies that the error grows over time, leading to the divergence of the KF under constant perturbations. Consequently, when the employed motion estimation model deviates from the true dynamics, KF performance is significantly compromised, as it is prone to divergence issues. Given the complexity of worker movements, current research efforts, as exemplified by [7], have not been able to establish accurate motion models, resulting in the persistent challenge of mitigating KF divergence still needing a reasonable mathematical solution. Therefore, it is necessary to find ways to bypass the divergence problem.

$$\begin{aligned}
 \Delta &= x_t^* - \hat{x}_t = x_0^* + tv^* - x_0 - \frac{(1+t)}{2}v - \frac{1}{t} \sum_{i=1}^t \varepsilon_i = \frac{t-1}{2}(v^* - v) - \frac{1}{t} \sum_{i=1}^t \varepsilon_i \\
 E(\Delta) &= \frac{t-1}{2}(v^* - v) \\
 D(\Delta) &= \left( \frac{t-1}{2}(v^* - v) \right)^2 + \frac{\sigma^2}{t}
 \end{aligned} \tag{13}$$

### 3.3. KF Processing Example

In this section, we demonstrate that despite the inherent limitations of the KF's motion estimation, achieving convergence can still be accomplished through continuous and accurate detection measurements.

The state equation initialization requires understanding the motion model of the object, such as the uniformly moving, two-dimensional object in (14). On occasions where there was no fixed frame interval time in videos,  $\Delta t$  was assigned the value 1.0, and could be adjusted through velocity.

$$\begin{aligned}
 x_t &= x_{t-1} + v_{t-1}^x \times \Delta t \\
 y_t &= y_{t-1} + v_{t-1}^y \times \Delta t \\
 v_{t-1}^x &= v_{t-1}^x \\
 v_{t-1}^y &= v_{t-1}^y
 \end{aligned} \tag{14}$$

Similar to (14), Equation (1), rewritten into the matrix form, can be presented as (15). At this point, our estimation motion model will be fixed.

$$\vec{x}_t = \begin{pmatrix} x_t \\ y_t \\ a_t \\ h_t \\ v_t^x \\ v_t^y \\ v_t^a \\ v_t^h \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ a_{t-1} \\ h_{t-1} \\ v_{t-1}^x \\ v_{t-1}^y \\ v_{t-1}^a \\ v_{t-1}^h \end{pmatrix} = F_{8 \times 8} \times \vec{x}_{t-1} \quad (15)$$

The initialization of the observation Equation (16):

$$\vec{z}_{t-1} = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ a_{t-1} \\ h_{t-1} \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ a_{t-1} \\ h_{t-1} \\ v_{t-1}^x \\ v_{t-1}^y \\ v_{t-1}^a \\ v_{t-1}^h \end{pmatrix} = H_{4 \times 8} \times \vec{x}_{t-1} \quad (16)$$

The initial noise covariance is empirical and exhibits a weight–height dependency. Vanilla DeepSORT assigns a position noise of 1/20 and a velocity noise of 1/160, assuming higher positional accuracy compared to velocity. Both the Q and R matrices are diagonal, with their values being determined by the height of each detection. It is commonly observed that larger bounding boxes imply higher noise levels, but concurrently, they indicate closer proximity to the camera, leading to less noisy images. This aspect has not been previously addressed in the literature. Consequently, we have revised the noise equation to achieve a more uniform and simplified treatment, as detailed in Equations (17) and (18).

Let a bounding box in frame  $t = 1$  be

$$z_{t-1} = [x_c, y_c, a, h]^T = \text{np.array}([1062.161303, 316.998036, 0.405503, 273.269825])$$

Then, the height is 273.269825, so the  $(273.269825 \times 1/20)^2 = 186.690993$ .

$$Q_{8 \times 8} = \text{np.square} \left( \begin{pmatrix} \text{weight\_position} \times h_t \\ \text{weight\_position} \times h_t \\ 1e-2 \\ \text{weight\_position} \times h_t \\ \text{weight\_velocity} \times h_t \\ \text{weight\_velocity} \times h_t \\ 1e-5 \\ \text{weight\_velocity} \times h_t \end{pmatrix} \right) \times I_{8 \times 8} \rightarrow \begin{bmatrix} 186.69 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 186.69 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0.0001 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 186.69 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 2.92 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 2.92 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 2.92 \end{bmatrix} \quad (17)$$

$$R_{4 \times 4} = \text{np.square} \left( \begin{pmatrix} \text{weight\_position} \times h_t \\ \text{weight\_position} \times h_t \\ 1e-2 \\ \text{weight\_position} \times h_t \end{pmatrix} \right) \times I_{4 \times 4} \rightarrow \begin{bmatrix} 186.69 & 0 & 0 & 0 \\ 0 & 186.69 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 186.69 \end{bmatrix} \quad (18)$$

Once initialized, the  $F$  and  $H$  will not be changed in any frame during the calculations (independent of  $t$ ). However,  $Q$  and  $R$  will be changed in each frame for the height difference of the bounding box. The noise can be adjusted according to the actual scenarios to achieve better effects.

For the covariance matrix  $P$ , we follow the traditions presented in (19). The covariance matrix  $P$  is also weight–height dependent but with 2-times magnification for position and 10-times magnification for velocity.

$$P_{8 \times 8} = np.square \left( \begin{pmatrix} 2 \times \text{weight\_position} \times h_t \\ 2 \times \text{weight\_position} \times h_t \\ 1e-2 \\ 2 \times \text{weight\_position} \times h_t \\ 10 \times \text{weight\_velocity} \times h_t \\ 10 \times \text{weight\_velocity} \times h_t \\ 1e-5 \\ 10 \times \text{weight\_velocity} \times h_t \end{pmatrix} \right) \times I_{8 \times 8} \rightarrow P_{8 \times 8}^{t=1} = \begin{bmatrix} 746.76 & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 746.76 & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0.0001 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 746.76 & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 291.70 & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 291.70 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 291.70 \end{bmatrix} \quad (19)$$

When the person’s detection is missing for 100 frames without an update of state, the same state vector [1062.161303, 316.998036, 0.405503, 273.269825] remains unchanged, while the elements in the covariance matrix can be magnified by more than 5000 times in (20) (3,894,274.937286/746.763973 = 5214.867).

$$\text{only\_predict} : P_{8 \times 8}^{t=100} = \begin{bmatrix} 3894274.93 & 0. & 0. & 0. & 43609.84 & 0. & 0. & 0. \\ 0. & 3894274.93 & 0. & 0. & 0. & 43609.84 & 0. & 0. \\ 0. & 0. & 0.010134 & 0. & 0. & 0. & 0.000001 & 0. \\ 0. & 0. & 0. & 3894274.93 & 0. & 0. & 0. & 43609.84 \\ 43609.84 & 0. & 0. & 0. & 583.40 & 0. & 0. & 0. \\ 0. & 43609.84 & 0. & 0. & 0. & 583.40 & 0. & 0. \\ 0. & 0. & 0.000001 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 43609.84 & 0. & 0. & 0. & 583.40 \end{bmatrix} \quad (20)$$

Once a subject’s detection is consistently stable and stationary at a specific position [1062.161303, 316.998036, 0.405503, 273.269825], the covariance matrix can be updated in each frame. The reduction in matrix elements, as observed in (21), is approximately 50% (363.719862/746.763973 = 0.48706), indicating that convergence is not reliant on motion and can occur even when the subject is completely stationary. The KF gain also exhibits a converging pattern in (22). Consequently, the non-convergence issue lies in the prediction stage, where an inadequate update occurs due to missing detections.

Each worker has two KFs, one for the head and the other for the body; likewise, different workers maintain different KFs. However, following the above calculations, with the presence of continuous detection values, even if the motion model is not good enough, the convergent  $P$  can also be achieved.

$$\text{predict} + \text{update} : P_{8 \times 8}^{t=100} = \begin{bmatrix} 363.71 & 0. & 0. & 0. & 40.06 & 0. & 0. & 0. \\ 0. & 363.71 & 0. & 0. & 0. & 40.06 & 0. & 0. \\ 0. & 0. & 0.001052 & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 363.71 & 0. & 0. & 0. & 40.06 \\ 40.06 & 0. & 0. & 0. & 29.39 & 0. & 0. & 0. \\ 0. & 40.06 & 0. & 0. & 0. & 29.39 & 0. & 0. \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 40.06 & 0. & 0. & 0. & 29.39 \end{bmatrix} \quad (21)$$

$$K_{8 \times 8}^{t=2} = \begin{bmatrix} 0.86 & 0. & 0. & 0. \\ 0. & 0.86 & 0. & 0. \\ 0. & 0. & 0.019 & 0. \\ 0. & 0. & 0. & 0.86 \\ 0.21 & 0. & 0. & 0. \\ 0. & 0.21 & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0.21 \end{bmatrix} \rightarrow K_{8 \times 8}^{t=100} = \begin{bmatrix} 0.66 & 0. & 0. & 0. \\ 0. & 0.66 & 0. & 0. \\ 0. & 0. & 0.095 & 0. \\ 0. & 0. & 0. & 0.66 \\ 0.073 & 0. & 0. & 0. \\ 0. & 0.073 & 0. & 0. \\ 0. & 0. & 0. & 0. \\ 0. & 0. & 0. & 0.073 \end{bmatrix} \quad (22)$$

Therefore, it is possible to maintain the correct motion if there is a method to retain detections. The following proposed method can utilize head cues to ensure the approximate accuracy of the whole-body detections even with heavy occlusions.

### 4. Methods

The bidirectional method can make the detector and tracker interact with each other, instead of only using the outputs of the previous step without changes.

As depicted in Figure 2, the proposed method consists of the detector, imputers, refiners and tracker; among them, the first three items are detailed and discussed in [10] while the tracker is our concern in this paper. Whereas the unidirectional method only contains the body\_tracker, our tracker contains four parts:

- Head\_tracker for tracking heads (in Section 4.1).
- Body\_tracker for tracking bodies (in Section 4.1).
- Intra-frame processing to delete false positives of heads and bodies (in Section 4.2).
- Inter-frame matching to find the pairing relationship between heads and bodies (in Section 4.3).

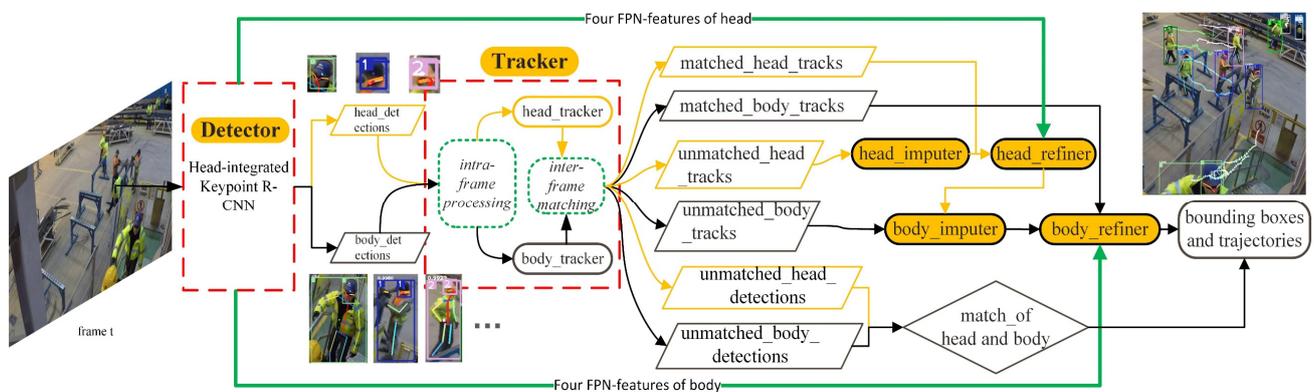


Figure 2. Overview of the proposed tracking method.

#### 4.1. Head Tracker and Body Tracker

The head tracker and body tracker are responsible for associating the head or body with detections, respectively.

The implementation of the head\_tracker is basically the same as the body\_tracker, which contains two steps of matching (shown in Algorithm 1), although the body\_tracker will adopt the velocity of a head other than its own:

- (1) The first match: the cascade matching with the appearance vector’s cosine distance and the bounding box center’s Euclidean distance.
- (2) The second match: the bounding box’s IoU distance matching for the remaining unmatched detections and tracks.

As shown in Figure 3, there are seven tracks of frame  $t - 1$  and five detections of frame  $t$ ; after the first match, (1-B), (2-A) and (4-C) were matched; the remaining (3-D) and (6-E) were successfully obtained in the second match. The tracking ID and detection ID are represented by the rows and columns in the cosine distance cost matrix. To match the tracking ID with the current frame, the Hungarian algorithm (Algorithm 1) is used. If the value is below the cosine similarity threshold of 0.2 and meets the tracking ID criteria, it is considered a successful match.

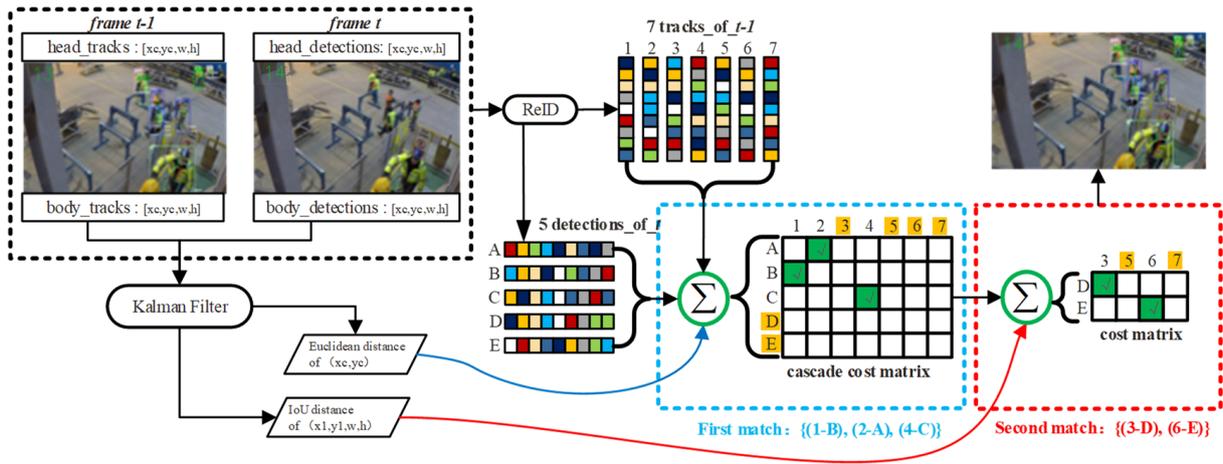


Figure 3. Framework of the head\_tracker or body\_tracker.

We modify the trackers in four ways:

(1) The Mahalanobis distance is modified to the Euclidean distance (23) in KF, to avoid the influence of covariance matrix divergence.  $x_t = [x_c, y_c, a, h, v_x, v_y, v_a, v_h]^T$  is the predicted vector of frame  $t$  over the state distribution (8-d) from frame  $t - 1$ , and it  $y_t = [x_c, y_c, a, h, 0, 0, 0, 0]^T$  refers to the detector’s measurements in frame  $t$ .

$$\begin{aligned} \text{Mahalanobis distance} &= \sqrt{(x_t - y_t)^T P^{-1} (x_t - y_t)} \\ \text{Euclidean distance} &= \sqrt{(x_t - y_t)^T (x_t - y_t)} \end{aligned} \tag{23}$$

(2) The head trajectory can be used as a reference to improve body accuracy. Because the head has limited motion and a fixed size, its use helps reduce errors in direction detection, ensuring that the tracker and imputer are not misled by incorrect detection results.

(3) The cost matrix uses a variable threshold ( $2 \times$  head width), which reduces the number of for-loops to two. The vanilla DeepSORT cost matrix utilizes a fixed threshold of 9.4877, but variable thresholds maintain the visual principle of “near-big, far-small”. Therefore, a smaller threshold is crucial when a worker is far from the camera, has a small head size, and shows minimal movement.

(4) Additional constraints to add a new body ID. The unmatched detections of vanilla DeepSORT were directly added as a new ID, which might have resulted in inflation. The confidence level must be approximately 0.8 or no less than 0.95 during the range intervals near the edge. This effectively filters out most false positives.

In this study, we utilized a Market-1501-based [34] 512-d vector for appearance information, which was better than the original 128-d vector of ReID (see Appendix A) in DeepSORT because an extensive feature dimension is appropriate for small objects such as heads.

---

**Algorithm 1:** Two Stages of the Matching Algorithm for the Trackers

---

**Input:**  $M \leftarrow$  number of workers;  $A \leftarrow$  number of frames;

Two detections set:

$$D = \{d_i^j | 1 \leq i \leq M; 1 \leq j \leq A\}; \quad D_{remain} = \{d_{i,remain}^j | 1 \leq i \leq M_{remain}; 1 \leq j \leq A\};$$

**Output:** Two tracks set:

$$T_{matched} = \{t_i^j | 1 \leq i \leq N_{matched}; 1 \leq j \leq A\}; \quad T_{un\_matched} = \{t_i^j | 1 \leq i \leq N_{un\_matched}; 1 \leq j \leq A\};$$

1: **for** frame  $j$  **in**  $A$  **do:**

2:  $D^j = \{d_1^j, d_2^j, d_3^j, \dots, d_{M^j}^j\}$  /\*observations from the detector and ReID at  $j^*$ /\*

$x^j = \{x_1^j, x_2^j, x_3^j, \dots, x_{N^j}^j\}$  /\*posterior state from KF by  $j - 1$  \*/

---

**Algorithm 1:** *Cont.*


---

```

3. /*The First match: assignment to list of matches, unmatched_tracks, unmatched_detections:*/
    $C^j \leftarrow C_{\text{cosine}}(D^j, x^j) + C_{\text{Euclidean}}(D^j, x^j)$  /*cost matrix at j*/
   Use the linear Hungarian algorithm to solve  $C^j$ .
   if ( $c_{\text{cosine}} > 0.2$  and  $c_{\text{Euclidean}} > 2 \times \text{head\_width}$ ):
        $T_{\text{matched}} \leftarrow t_i^j$  that matched with  $d_i^j$ 
        $T_{\text{un\_matched}} \leftarrow t_i^j$  that is not matched with  $d_i^j$ 
        $D_{\text{remain}} \leftarrow d_i^j$  that is not matched with  $t_i^j$ 
4. /*The second match: for the remaining detecting-ID in  $D_{\text{remain}}$ */
    $C^j \leftarrow C_{\text{cosine}}(D_{\text{remain}}^j, x_{\text{un\_matched}}^j) + C_{1-\text{IoU}}(D_{\text{remain}}^j, x_{\text{un\_matched}}^j)$ 
   Use the linear Hungarian algorithm to solve  $C^j$ .
   if ( $c_{1-\text{IoU}} > 0.7$ ) or (match_times = 0 and  $0.1 \leq c_{\text{cosine}} \leq 0.2$ ):
        $T_{\text{matched}} \leftarrow t_i^j$  that matched with  $d_{i,\text{remain}}^j$ 
        $T_{\text{un\_matched}} \leftarrow t_i^j$  that is not matched with  $d_{i,\text{remain}}^j$ 
        $D_{\text{remain}} \leftarrow d_{i,\text{remain}}^j$  that is not matched with  $t_i^j$ 

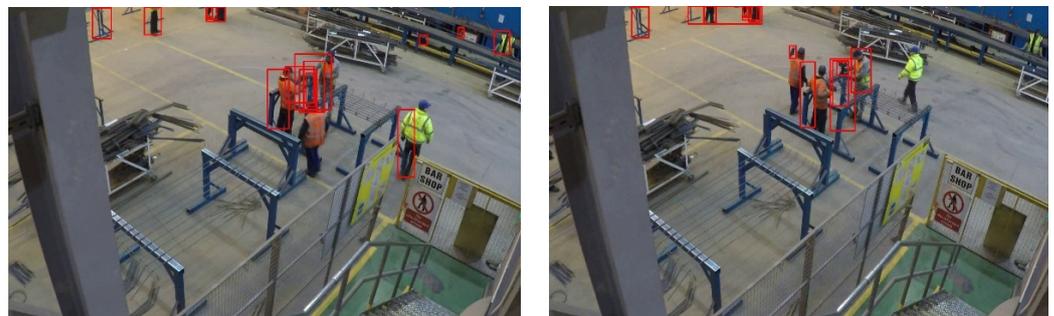
```

---

#### 4.2. Intra-Frame Processing

Intra-frame processing is used to reduce false positives and keep true positive detections, which can be treated as the pre-processing stage of the tracker's inputs, as shown in Figure 2. Intra-frame processing contains the delete process and store process.

First, we need to filter out false positives of detected heads. Compared to height, human heads have smaller differences in size, so heads closer to the camera usually have clearer visual features and greater confidence scores than those that are farther away. Therefore, following the perspective principle of “near-big, far-small” and the detection confidence score, it is possible to filter out most of the false positive detected heads, as shown in Figure 4.



**Figure 4.** Examples of deleted false positives during intra-frame processing.

The head closest to the camera has the highest priority, and any heads with a confidence score of no less than 99% will be considered as true positives directly. After sorting by center vertical coordinates, we determine whether to delete the corresponding item based on the difference in height values and confidence scores between adjacent head boxes, as shown in Algorithm 2.

Second, we need to filter out false positives of detected bodies. Body detections are prone to instability when encountering occlusions or blurring. IoU and keypoints will be taken as conditions to delete unstable bodies in Algorithm 3; first, calculate the IoU between each pair of bounding boxes and if the value is greater than the given threshold ( $\text{iou\_threshold} = 0.6$ ), the lower confidence box is deleted; second, check the number of keypoints to ensure at least two body joints are in the box.

**Algorithm 2:** Filter out False Positives of Detected Heads

---

**Input:**  $M \leftarrow$  number of detected heads;  $N \leftarrow$  number of remain heads;  $j =$  frame;  
 $D_{head} = \{d_i^j | 1 \leq i \leq M; 1 \leq j \leq A\}$ ;  $d_i^j = [x_c, y_c, w, h, confidence, class\_id = 1]_{i=1}^M$

**Output:**  $D_{head}^{del} = \{d_i^j | 1 \leq i \leq N; 1 \leq j \leq A\}$ ;  $D_{head}^{remain} = \{d_i^j | 1 \leq i \leq N; 1 \leq j \leq A\}$ ;

1:  $D_{head} \leftarrow \text{sort}\{d_i^j, y_c\}$ ; /\*sorted by head center  $y_c$ \*/

2: **while**  $i + 1 < M - 1$  **do**:  
 $\Delta_i \leftarrow d_i^j - d_{i+1}^j$ ;  
**if**  $\Delta_i < -5$ : /\*two heads are very close, maybe FP in here\*/  
**if**  $d_i^j[4] \geq 99\%$ : /\* confidence  $\geq 0.99$ \*/  
 $D_{head}^{del} \leftarrow d_i^j$  /\*delete  $i$ \*/  
**elif**  $d_{i+1}^j[4] \geq d_i^j[4]$ :  
 $D_{head}^{del} \leftarrow d_{i+1}^j$  /\*delete  $i + 1$ \*/  
**else**:  
 $D_{head}^{del} \leftarrow d_i^j \text{ and } d_{i+1}^j$  /\*delete  $i$  and  $i + 1$ \*/  
 $i += 1$

3:  $D_{head}^{remain} \leftarrow D_{head} - D_{head}^{del}$

---

**Algorithm 3:** Filter out False Positives of Detected Bodies

---

**Input:**  $M \leftarrow$  number of detected bodies;  $N \leftarrow$  number of remain bodies;  $j =$  frame;  
 $D_{body} = \{d_i^j | 1 \leq i \leq M; 1 \leq j \leq A\}$ ;  $d_i^j = [x_c, y_c, w, h, confidence, class\_id = 0]_{i=1}^M$

**Output:**  $D_{body}^{del} = \{d_i^j | 1 \leq i \leq N; 1 \leq j \leq A\}$ ;  $D_{body}^{remain} = \{d_i^j | 1 \leq i \leq N; 1 \leq j \leq A\}$ ;

1: **for**  $i, k$  **in**  $M$  **do**:  
 $IoU_i \leftarrow \text{Compute\_IoU}(d_i^j, d_{i+1}^j)$ ;  
**if**  $IoU_i \geq 60\%$ : /\*two bodies are very close, maybe FPs in here\*/  
 $D_{body}^{del} \leftarrow \min(d_i^j[4], d_{i+1}^j[4])$  /\*lower confidence deleted\*/  
 $i += 1$   
 $k += 1$

2: **for**  $i$  **in**  $\text{length}(\text{keypoints})$  **do**:  
/\*if the body has no more than two effective keypoints in total\*/  
**if**  $\text{torch.sum}(\text{one\_key}[:, -1] \geq 0.05) < 2$ :  
 $D_{body}^{del} \leftarrow d_i^j$  /\*lower confidence deleted\*/  
 $i += 1$

3:  $D_{body}^{remain} \leftarrow D_{body} - D_{body}^{del}$

---

After these filtering steps, the remaining heads and bodies are passed to the store process. To handle potential crowd scenarios where multiple heads may occupy a single body location, we employed Python dictionaries (`head_sequence_set` and `body_sequence_set`) to store data. This approach allows for persistent tracking even when individuals enter or exit the field of view. Additionally, this method mitigates indexing issues that may arise due to variable head–body pairings, such as the example provided:

$$\text{body\_head\_matched\_ids} = [(3, 0), (0, 1), (2, 2), (\text{None}, 3), (4, \text{None}), (1, 4)]$$

By deferring matching to the inter-frame tracking stage, our method can utilize appearance features in addition to intra-frame location data, thus avoiding the potential identity confusion caused by false positives, as shown in Figure 4.

In summary, the intra-frame processing stage employs a combination of confidence scores, size principles and keypoint analysis to refine head and body detections, reducing false positives and preparing the data for more accurate inter-frame tracking using the RseNet-50-based [10] head-integrated keypoint R-CNN model.

### 4.3. Inter-Frame Matching

Since head detection can aid in body detection [10], head tracking can likewise assist in body tracking, as described in this section. Because a less occluded head has an inclusion relationship with the entire body, the corresponding tracking can be achieved through bidirectional matching algorithms.

Inter-frame matching refers to associating the heads and bodies in consecutive frames. Unlike the traditional matching procedure of “head to head” or “body to body” in vanilla DeepSORT, we have expanded the scope of association to “head to body”.

Because the head and body do not belong to the same recognition category ( $class\_id = 0$  for body;  $class\_id = 1$  for head), it is meaningless to directly use the appearance features of the head to match a body but this is very useful for filtering newly added tracking IDs, as shown in Algorithm 4. This process can provide additional checking conditions for new IDs. In the case of mixed tracking, the correspondence between the body and head does not need to be calculated in every frame but only during the track ID initialization frame.

The head feature is the critical factor in identifying a worker in practical construction scenarios, which may lead to dilemmas in the annotation of tracking IDs in manual labeling. Due to the size of the head being relatively small and fixed, we used the Euclidean distance of the bounding box center ( $3 \times width$ ) and a higher confidence threshold (95%) to restrict the newly added head’s tracking ID in Algorithm 3. Based on the assumption that the head tracking is more accurate than the body tracking, the newly added body should not be around those already-existing heads by the IoU threshold of 0.8.

Whenever there is a newly added ID, we calculate the elements  $z$  of the cost matrix as (24) and use Algorithm 4 to find the corresponding head or body ID.

$$z = \frac{1 - \text{modified\_IoU}}{\text{confidence}} \quad (24)$$

$$\text{modified\_IoU} = \frac{\text{body} \cap \text{head}}{\min(\text{body}, \text{head})} \quad (25)$$

where the modified\_IoU in (25) is no longer focused on the union area of the body and head, but on the smaller area. This type of calculation tends to retain a high-overlap “head to body” pair, and the low-confidence pair will lose priority to an extent.

---

**Algorithm 4:** Matching of “Head to Body” across Frames

---

**Input:**

$M_T \leftarrow$  number of tracked heads;  $N_T \leftarrow$  number of tracked bodies;  $j = \text{frame}$ ;

$T_{\text{body}} = \left\{ t_{i,\text{body}}^j \mid 1 \leq i \leq N_T; 1 \leq j \leq A \right\}$ ;  $t_{i,\text{body}}^j = [x_c, y_c, w, h, \text{confidence}, \text{class\_id} = 0]_{i=1}^{N_T}$

$T_{\text{head}} = \left\{ t_{i,\text{head}}^j \mid 1 \leq i \leq M_T; 1 \leq j \leq A \right\}$ ;  $t_{i,\text{head}}^j = [x_c, y_c, w, h, \text{confidence}, \text{class\_id} = 1]_{i=1}^{M_T}$

$L_{\text{body,head}} = \left\{ \left( t_{i,\text{body}}^j, t_{a,\text{head}}^j \right) \mid 1 \leq i \leq N_T; 1 \leq j \leq A; 1 \leq a \leq N_T \right\}$ ;

**Output:**  $L_{\text{body,head}}$

1: **if**  $\exists ID_{\text{head}} \notin T_{\text{head}}$ :

**for**  $k$  **in**  $L_{\text{body,head}}$  **do**:

        /\*find the closest head, and calculate the Euclidean distance of the center\*/

**if** Euclidean  $\left| ID_{\text{head}} - t_{a,\text{head}}^j \right| > 3 \times w$  **and**  $ID_{\text{head}}[4] > 0.95$ :

$T_{\text{head}} \leftarrow ID_{\text{head}}$

2: **if**  $\exists ID_{\text{body}} \notin T_{\text{body}}$ :

**for**  $k$  **in**  $L_{\text{body,head}}$  **do**:

        /\*find the closest body, and calculate the IoU\*/

**if**  $\text{IoU} \left( ID_{\text{body}}, t_{a,\text{head}}^j \right) > 0.8$  **and**  $ID_{a,\text{head}}^j \neq \text{None}$ :

$T_{\text{body}} \leftarrow ID_{\text{body}}$

3: **if**  $T_{\text{head}} \leftarrow ID_{\text{head}}$  **or**  $T_{\text{body}} \leftarrow ID_{\text{body}}$  : /\*matching of new added head–body pairs\*/

$\forall C^j \in \text{cost\_matrix}, C^j \leftarrow \frac{1 - \text{modified\_IoU}}{\text{confidence}}$  when  $C^j \leq 1.0$ ;

$C^j \leftarrow 100000$  when  $C^j > 1.0$ ;

    Use the linear Hungarian algorithm to solve  $C^j$

    # row\_indices, col\_indices = linear\_assignment(cost\_matrix)

$L_{\text{body,head}} \leftarrow (ID_{\text{body}}, ID_{\text{head}})$

---

By adding head tracking, the weakness of relying solely on body tracking can be overcome. Figure 5 (the frame #37 in video-3) shows the effective processing of inter-frame matching from left to right images: the left image is the detector’s outputs, while the middle image is the tracker’s outputs and the right image is the final results. Head and body IDs

can obtain the pair-wise relationship and stable trajectories successfully. The newly added head ID = 17 in the middle image (confidence = 92.2% in the left image) was deleted as a false positive association and did not affect the final tracking outputs.



Figure 5. Process images of the inter-frame matching.

In vanilla DeepSORT, the matching algorithm was only restricted to two frames ( $t - 1$  and  $t$ ) as the Markov property in the KF in (2), whereas former frames ( $1, 2 \dots t - 2$ ) were forgotten. Then, unmatched head or body detections could be directly and easily treated as a new track ID for the next frame; consequently, if the ReID feature vector of the current frame is not good enough due to occlusions or blurring, better feature vectors from previous frames will be replaced and cannot be recovered. Using the head as a benchmark can maintain correspondence with past frames during the bad modification of body features in the current frame; the “head to body” matching reduces possible ID errors effectively.

Note: to clearly understand how head tracking aids in body tracking, please refer to Appendix B for the calculation details.

#### 4.4. Evaluation Metrics

CLEAR MOT [35] proposed the IDF1, MOTA and MOTP metrics in (26)–(28).

$$IDF1 = \frac{2 \times IDTP}{2 \times IDTP + IDFP + IDFN} \tag{26}$$

$$MOTA = 1 - \frac{\sum_i (IDFP_i + IDFN_i + IDSW_i)}{\sum_i GT_i} \in (-\infty, 1] \tag{27}$$

$$MOTP = \frac{\sum_{i,t} d_{i,t}}{\sum_i c_i} \in (0, 1) \tag{28}$$

where True Positive ID (IDTP) is the number of correctly assigned IDs throughout the entire video; False Positive ID (IDFP) is the number of incorrect IDs; False Negative ID (IDFN) is the number of missed IDs; IDSW is the identity switches of one object; GT means the number of manually labeled ground truth boxes;  $i$  is one frame in video set;  $t$  is a worker identity;  $d$  refers to the distance between the prediction box and ground truth box,  $d$  may be assigned as the IoU distance.

IDTP does not have the same value as TP. The former emphasizes maintaining ID consistency, whereas TP primarily focuses on bounding box IoU. IDTP can only be obtained after the tracking association calculation is complete, whereas TP can be obtained during the earlier detection stage. The same is true for IDFP and FP, and IDFN and FN.

IDF1 gauges the proportion of correctly identified detections to the mean count of ground truth and computed detections. It offers a fair assessment of all trackers based on their identification precision and recall through their harmonic mean [36].

MOTA measures the combined errors of IDFNs, IDFPs and IDSW. However, IDFPs have a greater influence than IDFNs and IDSW. For instance, if the GT tracking target has

an ID error, it results in an increment of IDFP + 1. On the other hand, if the GT does not have a corresponding detection, it is considered missing, resulting in IDFN + 1. Changes in IDFPs will lead to an increase in IDFNs but not vice versa. It is important to note that MOTA does not include a measure of localization error.

To compensate for this, MOTP measures a tracker's ability to accurately estimate object positions, regardless of its proficiency in recognizing object configurations and maintaining consistent trajectories. It primarily evaluates positional accuracy, which is more relevant to the detector's performance than the overall performance of the tracker. As a result, MOTChallenge removes MOTP as an evaluation metric.

HOTA [6] balances the effect of three accuracies: detection, association and localization. Equations (29) and (30) decompose HOTA into separate DetA and AssA scores.

$$HOTA_{\alpha} = \sqrt{\frac{\sum_{c \in \{TP\}} A(c)}{|TP| + |FN| + |FP|}} = \sqrt{DetA_{\alpha} \cdot AssA_{\alpha}} \quad (29)$$

$$HOTA = \int_0^1 HOTA_{\alpha} d\alpha \approx \frac{1}{19} \sum_{\alpha \in \{0.05, 0.1, \dots, 0.95\}} HOTA_{\alpha} \quad (30)$$

where  $\alpha$  is the localization threshold of IoU, with 19 different distinct values (0.05 to 0.95 in 0.05 intervals) for integral calculation;  $c$  is one worker ID. More details of the HOTA formula can be found in [6].

HOTA identified two errors in ID distinction for tracking: ID switches (IDSWs) and ID transfers (IDTRs). IDSW is designed for continuous trajectories, while IDTR is designed for intermittent trajectories.

IDSW calculates the frequency at which a tracked trajectory switches its matched GT identity. It is worth mentioning that this definition is applicable only while the target is within the field of view. IDSW occurs when, in consecutive frames, the same worker has varying IDs, but it is restricted to TP and not FN. If a trajectory abruptly vanishes, it will not be counted as IDSW.

IDTR happens when one worker exits the frame and another worker enters the subsequent frame, but with the former's ID. IDTR is common in scenarios involving frequent entry and exit. When worker A leaves frame- $i$ , worker B enters frame- $i + 1$ . If worker B is mistakenly recognized as worker A in frame- $i + 1$ , worker A's ID will be transferred to worker B, but IDSW will not be added since worker A was interrupted. Therefore, IDTR is also considered an undesirable behavior during tracking, which MOTA could not consider.

Fortunately, HOTA can effectively resolve IDTR errors [6]. The IDSW in the CLEAR MOT evaluation only measures short-term tracking and cannot accurately assess long-term tracking. HOTA can evaluate the global long-term tracking by comparing FN and FP in the matched trajectories.

## 5. Results and Discussion

The proposed tracking method achieved an IDF1 of 97.609%, MOTA of 95.191% and HOTA of 78.884% on the testing dataset of nine videos [2] with a total of 41 workers. Additionally, the method achieved an IDSW of 0, even in scenarios with heavy occlusions. This is the first report of the IDSW value since previous research on tracking workers.

The model is implemented using Python 3.8.8 and PyTorch 1.8.0. The method is based on the GitHub code [37], containing KF and Hungarian algorithms. The computer has one NVIDIA RTX 3080 GPU, an Intel(R) Core(TM) i9-10900K CPU @ 3.70 GHz and one 32 GB RAM with Win10. The testing dataset of nine videos is the same as [2,38], with a resolution of  $1920 \times 1080$  and 30 FPS; more details can be found in [2].

The ground truth data with frame-ID, track-ID and bounding box in a four-tuple ( $x$ ,  $y$ , width, height) were manually annotated as follows:  $\langle frame \rangle$ ,  $\langle id \rangle$ ,  $\langle bb\_x \rangle$ ,  $\langle bb\_y \rangle$ ,  $\langle bb\_width \rangle$ ,  $\langle bb\_height \rangle$ ,  $\langle conf = 1 \rangle$ ,  $\langle x = -1 \rangle$ ,  $\langle y = -1 \rangle$ ,  $\langle z = -1 \rangle$ . This was performed with TrackEval [39].

### 5.1. Quantitative Results

Table 2 presents a comprehensive analysis of the performance evaluation outcomes for the nine videos across seven metrics: MOTA, IDF1, HOTA, AssA, AssRe, AssPr and LocA. It is important to note that the aggregated value in the final row does not result from a simple average of individual video scores due to variations in total frames; instead, the cumulative frames of all videos are considered.

**Table 2.** Performance of our tracking method on the testing dataset.

	Metrics	Video-1	Video-2	Video-3	Video-4	Video-5	Video-6	Video-7	Video-8	Video-9	Combined
1	MOTA↑ (%)	100	94.08	93.943	93.582	94.833	98.859	95.44	96.843	93.972	95.191
2	IDF1↑ (%)	100	97.04	96.966	96.824	97.468	99.431	97.73	98.422	97.007	97.609
3	HOTA↑ (%)	93.023	77.601	81.446	75.7	74.801	76.911	78.273	81.799	77.446	78.884
4	AssA↑ (%)	93.023	79.784	86.066	78.962	81.28	85.451	80.484	83.341	78.149	83.296
5	AssRe↑ (%)	94.298	84.026	89.916	84.525	87.219	89.145	85.352	86.762	84.28	87.83
6	AssPr↑ (%)	94.298	84.026	90.052	83.247	84.142	88.895	84.634	86.689	83.016	86.976
7	LocA↑ (%)	92.568	83.922	86.915	82.642	82.643	81.798	84.326	85.815	85.614	84.725
8	IDSW↓	0	0	0	0	0	0	0	0	0	0
9	Frag↓	0	4	7	21	22	4	6	10	12	86
10	Hz↑	9.05	6.01	5.69	5.96	7.11	6.69	6.62	5.33	6.82	6.58
11	MT	1	3	9	5	5	4	4	6	4	41

(‘↑’ means higher is better, ‘↓’ means lower is better).

The method consistently demonstrates stability and reliability across diverse scenarios and conditions, as shown in Figure 6. The challenge for tracking algorithms, particularly in video 3, with the highest number of workers, is exemplified by HOTA reaching 81.446%, which is marginally higher than the combined score of 78.884%, thereby highlighting the effectiveness of the crowd-tracking method.



**Figure 6.** Testing videos' tracking outputs at their final frames.

### 5.2. Comparison of Other SOTA Methods

Pedestrian tracking on roads presents a relatively straightforward task from a programming standpoint, despite some inherent complexities and uncertainties in traffic

environments. In contrast, construction sites pose a greater challenge due to their dynamic nature. The simplicity of pedestrians' movement patterns further contributes to the relative ease of tracking on roads. Table 3 highlights eight state-of-the-art pedestrian tracking methods employed in recent years, particularly in the context of worker tracking. However, these methods exhibit considerable disparities in application domains, tracked objects and technical challenges. Consequently, selecting the most suitable tracking techniques for practical applications is crucial, as it ensures alignment with the diverse needs and requirements of different scenarios.

**Table 3.** Performance of classical pedestrian tracking method on the testing dataset.

	Metrics	DeepSORT	ByteTrack	Deep OC_SORT	BoTSORT	OC_SORT	Strong_SORT	TransTrack	UniTrack
1	MOTA↑ (%)	69.914	62.515	56.429	60.699	66.002	56.698	39.345	27.441
2	IDF1↑ (%)	79.771	81.829	74.471	77.365	80.633	77.065	70.459	61.887
3	HOTA↑ (%)	68.418	66.915	62.425	63.208	66.596	63.893	57.607	49.201
4	AssA↑ (%)	74.585	77.995	71.394	73.429	76.166	74.7	72.685	63.686
5	AssRe↑ (%)	78.896	82.717	75.462	76.777	80.316	78.679	76.712	68.208
6	AssPr↑ (%)	84.488	83.637	83.043	85.385	86.221	84.348	82.495	74.389
7	LocA↑ (%)	85.353	80.694	82.01	82.833	82.657	82.058	80.222	69.791
8	IDSW↓	21	5	51	8	10	12	7	15
9	Frag↓	101	107	178	180	168	140	161	268
10	Hz↑	1.77	9.09	8.33	7.14	8.47	7.69	5.13	4.30

(‘↑’ means higher is better, ‘↓’ means lower is better).

The codes of ByteTrack [7], Deep OC\_SORT [26], BoTSORT [29], OC\_SORT [9] and Strong\_SORT [30] are from [40]. These five methods were retrained on 26.6 K images of CrowdHuman, MOT17, Cityperson and ETHZ datasets [7]. TransTrack [24] and UniTrack [31] are in the authors' original codes, and DeepSORT uses an implementation from [41]. These three methodologies rely on vanilla YOLOX-x (yolox\_x.pth, 756 MB).

As displayed in Table 3, YOLOX-based DeepSORT achieved the highest HOTA and MOTA scores, at 68.418% and 69.914%, respectively, but exhibited the lowest FPS (1.77) and the worst IDSW score (21). In contrast, ByteTrack demonstrated the best performance, with an IDF1 score of 81.829%, the fastest FPS (9.09) and the lowest IDSW score (5). Despite these achievements, none of these pedestrian tracking methods surpasses our proposed bidirectional tracking method.

### 5.3. Discussion

Compared with the unidirectional method, the bidirectional method has the following advantages:

(1) Low dependency of detection DNN

The bidirectional approach allows the tracker to modify the results of the detector in reverse with imputers and refiners. Therefore, poor detections (ResNet-50 [10]) in the current frame will be modified into acceptable outputs and saved into head or body sequence sets for the next frame's prediction. Thus, these poor detections will not greatly change the value of the covariance matrix.

(2) Application of head tracking aid in body tracking

Head cues' distinctive attributes, including low variation, visibility and reduced occlusion, contribute to their effectiveness in tracking tasks. In tracking algorithms, the utilization of head appearance features for body tracking marks a novel approach. Worker tracking, far from being a mere metric comparison, demonstrates its practicality in complex situations like the occlusions depicted in Figures 7 and 8. Despite individual differences in head shape, the relatively consistent size of heads compared to other body parts ensures more reliable appearance features compared to limbs.

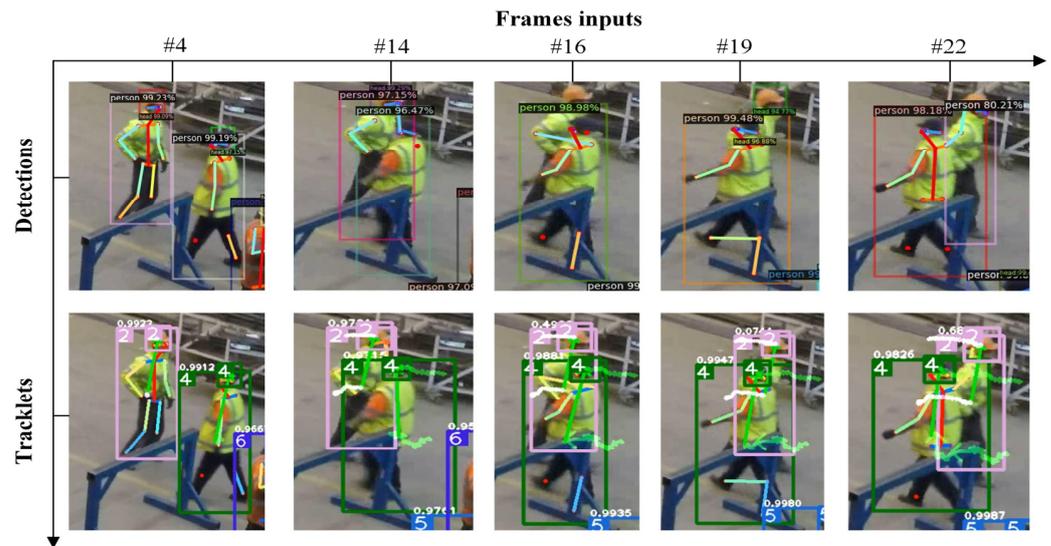


Figure 7. Example of intra-class occlusions in video 3.

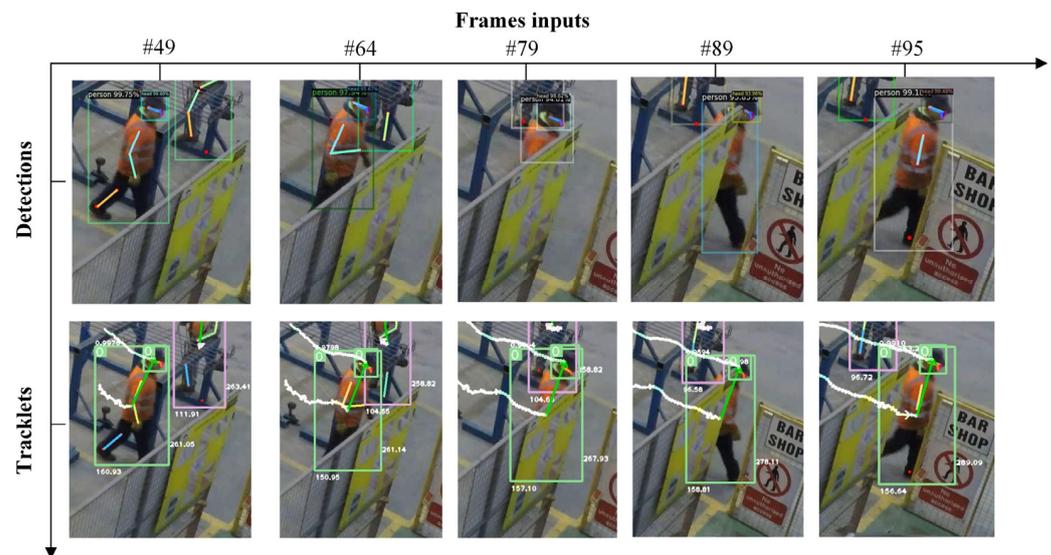


Figure 8. Example of inter-class occlusions in video 7.

In Figure 7, from frames #4 to #14, workers with ID = 2 and ID = 4 of the same categories are moving in the opposite direction. In frame #16, ID = 2 loses the detection of the head and body until frames #19 and #22, respectively. After calculating, ID = 2 maintained continuous tracklets during the overlap by ID = 4, proving its effectiveness in handling intra-class occlusions.

In Figure 8, due to the obstruction of the fence category, only a few appearance features are visible in the worker category ID = 0. Frame #79 demonstrates that the scaled-down bounding box of ID = 0 was repaired to a normal height and width, indicating that inter-class occlusions can be fixed even during bad detections.

(3) Avoidance of the impact of KF divergence issues

The prediction of the whole body’s speed can now be replaced by the speed of the head, instead of solely relying on KF prediction. The head’s speed can serve as a reference to refine body velocity predictions in the KF. If body bounding box coordinates are inaccurate, the correct head coordinates can be employed to correct them, preventing the tracker and imputer from being misled by faulty detections.

For instance, in the top row of Figure 9, the body detector’s output for the left worker’s full-body box in frames 112 to 114 shows an unexpected increase in the x-coordinate (ID 4).

The expected trend is a decrease as the worker moves leftward, but the detection error pushes the x-coordinates to the right (upper row: 667.91  $\rightarrow$  668.17  $\rightarrow$  666.25). Conversely, the head's x-coordinate accurately reflects the leftward movement (lower row: 668.50  $\rightarrow$  653.50  $\rightarrow$  648.00). By utilizing head-based linear interpolation, the x-coordinate of the full-body bounding box can be corrected to the correct value (lower row: 666.65  $\rightarrow$  665.5  $\rightarrow$  660.5), ensuring accurate tracking.



**Figure 9.** Example of body's reverse movement for ID = 4.

#### (4) Focus on metrics performance with ID errors

Addressing identity errors can be efficiently achieved with meticulous handling of new head or body IDs, as outlined in the intra-frame processing and inter-frame matching sections. By not immediately assigning unmatched detections to new identities, these methods can significantly reduce false positives (FPs) and false negatives (FNs), as demonstrated by Algorithms 2–4. These heuristic algorithms streamline the process by avoiding the need for retraining and effectively leveraging head information for identity association. By attentively managing new IDs, they contribute to enhanced tracking system performance, presenting a simple, effective and cost-efficient solution. This approach is particularly suitable for practical applications due to its wide applicability.

The main limitation of the proposed method is the fact that many innovations are heuristic and difficult to fully prove using mathematical formulas. Another limitation is that it is not possible to completely abandon the use of KF to estimate velocity values.

A potential future research direction is to track worker movement data, obtain movement characteristics and make trajectory predictions.

## 6. Conclusions

This article introduced a training-free tracking method for tracking workers. The novel method has a bidirectional interaction mechanism between the detector and tracker, which allows head information to supply more stable and precise navigation for body tracking. We have also analyzed in detail the mathematical reasons for the non-convergence phenomenon, and the proposed method can successfully prevent the problem of non-convergence KF in traditional one-way tracking and significantly reduce identity errors regarding workers through the designed newly added ID-checking algorithms. During testing, our method achieved an HOTA of 78.884%, an MOTA of 95.191% and an IDSW of 0 across nine video datasets, demonstrating that the method remains highly effective even in scenarios with serious occlusions or non-convergence issues.

**Author Contributions:** Conceptualization, Y.L. and Y.W.; methodology, Y.L.; software, Y.L.; validation, Y.L. and Z.Z.; formal analysis, Y.L.; investigation, Y.L.; resources, Y.L.; data curation, Y.L.; writing—original draft preparation, Y.L.; writing—review and editing, Y.L. and Z.Z.; visualization, Y.L.; supervision, Y.W.; project administration, Y.L. and Y.W.; funding acquisition, Y.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

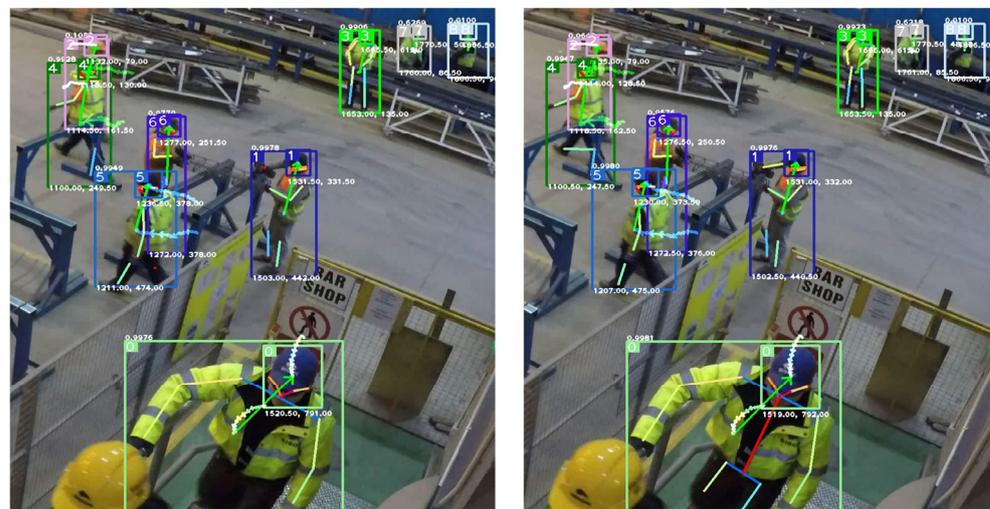
**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Conflicts of Interest:** The authors declare no conflicts of interest.

### Appendix A

Some studies [42,43] suggest that the similar high-visibility apparel (Hi-Vis [38]) worn by workers must lead to indistinguishable appearance features, which need additional retraining on a new dataset of construction scenarios. Hi-Vis is a special type of clothing commonly used to make workers more visible in busy or low-light environments, thereby increasing their safety. Hi-Vis clothing typically uses eye-catching colors and pattern designs. Due to the introduction of distinguishable appearance features in Hi-Vis clothing, the previously trained model may not be able to accurately recognize or process these new features. Therefore, in order for the model to adapt to these new changes and work properly in new construction scenarios, it is necessary to retrain the model. Retraining typically involves using datasets containing new features (in this case, Hi-Vis clothing) to allow the model to learn and adapt to these new appearance features.

However, our calculations demonstrate that this attitude is wrong. The appearance features of workers are distinguishable. For instance, in Figure A1, which captures frames #18–#19 of video 3, featuring nine workers, our experiments revealed that even in scenarios where workers wear helmets and similar Hi-Vis clothing, a standard human detection ReID model (ckpt.t7/43.9 MB [44]) can still differentiate individuals accurately.



**Figure A1.** Frames #18 (left) and #19 (right) of a construction video.

The following can be observed from the two cost matrices equations below:

(1) For smaller headers, the order of magnitude of successfully matched elements is 1~10 times less than other values, e.g., in the first row of the following Cosine\_cost matrix of heads equation:

$$(0.272943 + 0.280426 + 0.268017 + 0.318667 + 0.237262 + 0.289022 + 0.245184 + 0.309502) \div 8 \div 0.026227 = 10.58.$$

(2) For larger bodies, it is 10~100 times less. Larger objects are easier to distinguish than smaller ones, similar to how body elements are approximately 10 times smaller than the head, e.g., in the first row of the following Cosine\_cost matrix of bodies equation,

$$(0.17495 + 0.220428 + 0.207515 + 0.245856 + 0.277771) \div 5 \div 0.005517 = 40.8.$$

When using the default cosine threshold of 0.2, the ReID for pedestrian detection can accurately identify even the smallest heads of distinct workers. Such simple calculations provide evidence that the ReID model for pedestrians can be applied to workers without retraining.

$$\text{Cosine\_cost matrix of bodies} = \begin{bmatrix} 0.005517 & 0.17495 & 0.220428 & 0.207515 & 0.245856 & 0.277771 \\ 0.228444 & 0.269655 & 0.002582 & 0.235785 & 0.305651 & 0.189078 \\ 0.218299 & 0.175027 & 0.221523 & 0.162441 & 0.193348 & 0.256855 \\ 0.272631 & 0.173115 & 0.321291 & 0.181186 & 0.00269 & 0.33439 \\ 0.207658 & 0.174372 & 0.196977 & 0.011115 & 0.182533 & 0.2122 \\ 0.164177 & 0.009771 & 0.234517 & 0.231651 & 0.197819 & 0.286635 \\ 0.246577 & 0.224236 & 0.212872 & 0.196327 & 0.241916 & 0.029123 \\ 0.129127 & 0.131971 & 0.134357 & 0.116552 & 0.138499 & 0.185706 \\ 0.129708 & 0.127821 & 0.15255 & 0.127272 & 0.138029 & 0.196159 \end{bmatrix}$$

$$\text{Cosine\_cost matrix of heads} = \begin{bmatrix} 0.026227 & 0.272943 & 0.280426 & 0.268017 & 0.318667 & 0.237262 & 0.289022 & 0.245184 & 0.309502 \\ 0.266658 & 0.202771 & 0.255982 & 0.014943 & 0.358701 & 0.245981 & 0.343662 & 0.292406 & 0.375493 \\ 0.303788 & 0.271091 & 0.223952 & 0.29911 & 0.271148 & 0.016107 & 0.321069 & 0.189749 & 0.279182 \\ 0.334715 & 0.023847 & 0.310186 & 0.227934 & 0.251481 & 0.233631 & 0.280038 & 0.286123 & 0.336056 \\ 0.314554 & 0.29344 & 0.046261 & 0.27025 & 0.300371 & 0.197424 & 0.355366 & 0.271777 & 0.173631 \\ 0.328805 & 0.260196 & 0.280138 & 0.371353 & 0.250941 & 0.253999 & 0.037608 & 0.292974 & 0.356252 \\ 0.343413 & 0.300676 & 0.344513 & 0.434718 & 0.320221 & 0.268226 & 0.312919 & 0.031341 & 0.190585 \\ 0.32014 & 0.216402 & 0.172925 & 0.28769 & 0.112796 & 0.174532 & 0.293966 & 0.220967 & 0.250534 \\ 0.383001 & 0.340808 & 0.301298 & 0.429668 & 0.447917 & 0.341719 & 0.396959 & 0.19489 & 0.021867 \end{bmatrix}$$

### Appendix B

To clearly show the details of how head tracking aids in body tracking, we choose one output example (shown in Table A1) of our source code, executed in video 7 from frames #1 to #8. Each frame contains five procedures:

detector calculation → Intra-frame processing → head\_tracker → body\_tracker → Inter-frame matching.

**Table A1.** The process details of the head tracking aid in body tracking.

No.	Calculation Descriptions					
	Input frame #1					
	<ul style="list-style-type: none"> <li>The output of Detector (head_integrated Keypoint R-CNN) as:</li> </ul>					
	Bounding boxes in (x1, y1, x2, y2, confidence, class_id) = tensor([					
	1102.4229	275.0947	1219.5381	563.0864	0.9966	0 #body ID = 0
	733.7047	1.0617	789.8318	97.3516	0.9864	0 #body ID = 1
1	1312.4270	197.8513	1434.9780	470.1823	0.9817	0 #body ID = 2
	601.3785	0.0000	654.7200	104.9832	0.9561	0 #body ID = 3
	1148.3512	274.9141	1189.9109	322.3585	0.9955	1 #head ID = 0
	1382.5782	198.2840	1422.7518	249.7969	0.9886	1 #head ID = 1 ])

Table A1. Cont.

No.	Calculation Descriptions																																				
Input frame #1																																					
Number of Body IDs = 4, Number of Head IDs = 2.																																					
2	<ul style="list-style-type: none"> <li>Intra-frame processing: delete the false positives: Algorithm 2: <math>\text{delta\_del\_head} = []</math>, Algorithm 3: <math>\text{delta\_del\_body} = [3]</math>. Then the detection = [601.3785, 0.0000, 654.7200, 104.9832, 0.9561, 0] is deleted, and the left bounding box is: <math>\text{boxes\_xyxy} = [</math>  <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1102.422852</td> <td>275.094666</td> <td>1219.538086</td> <td>563.086426</td> <td>0.996561</td> <td>0</td> </tr> <tr> <td>733.704712</td> <td>1.061707</td> <td>789.831787</td> <td>97.351639</td> <td>0.986442</td> <td>0</td> </tr> <tr> <td>1312.427002</td> <td>197.851257</td> <td>1434.978027</td> <td>470.182251</td> <td>0.98167</td> <td>0</td> </tr> <tr> <td>1148.351318</td> <td>274.914093</td> <td>1189.910889</td> <td>322.35849</td> <td>0.995536</td> <td>1</td> </tr> <tr> <td>1382.578247</td> <td>198.283997</td> <td>1422.751831</td> <td>249.796875</td> <td>0.988596</td> <td>1]</td> </tr> </table> </li> </ul>	1102.422852	275.094666	1219.538086	563.086426	0.996561	0	733.704712	1.061707	789.831787	97.351639	0.986442	0	1312.427002	197.851257	1434.978027	470.182251	0.98167	0	1148.351318	274.914093	1189.910889	322.35849	0.995536	1	1382.578247	198.283997	1422.751831	249.796875	0.988596	1]						
1102.422852	275.094666	1219.538086	563.086426	0.996561	0																																
733.704712	1.061707	789.831787	97.351639	0.986442	0																																
1312.427002	197.851257	1434.978027	470.182251	0.98167	0																																
1148.351318	274.914093	1189.910889	322.35849	0.995536	1																																
1382.578247	198.283997	1422.751831	249.796875	0.988596	1]																																
3	<ul style="list-style-type: none"> <li>Head_tracker in Algorithm 1: the first match: <math>\text{matches\_a}</math>, <math>\text{unmatched\_tracks\_a}</math>, <math>\text{unmatched\_detections} = [] [] [0, 1]</math> the second match: <math>\text{matches\_b}</math>, <math>\text{unmatched\_tracks\_b}</math>, <math>\text{unmatched\_detections} = [] [] [0, 1]</math> <math>\text{matches}</math>, <math>\text{unmatched\_tracks}</math>, <math>\text{unmatched\_detections} = [] [] [0, 1]</math> The tracking ID in the head sequence set is: 0, 1.</li> </ul>																																				
4	<ul style="list-style-type: none"> <li>Body_tracker in Algorithm 1: the first match: <math>\text{matches\_a}</math>, <math>\text{unmatched\_tracks\_a}</math>, <math>\text{unmatched\_detections} = [] [] [0, 1, 2]</math> the second match: <math>\text{matches\_b}</math>, <math>\text{unmatched\_tracks\_b}</math>, <math>\text{unmatched\_detections} = [] [] [0, 1, 2]</math> The tracking ID in the body sequence set is: 0, 1, 2.</li> </ul>																																				
5	<ul style="list-style-type: none"> <li>Inter-frame matching in Algorithm 4: <math>\text{body\_id\_list} = [0, 2]</math>; <math>\text{head\_id\_list} = [0, 1]</math> <math>\text{cost\_matrix} = [</math>  <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0.003819</td> <td>100,000.</td> </tr> <tr> <td>100,000.</td> <td>100,000.</td> </tr> <tr> <td>100,000.</td> <td>0.]</td> </tr> </table>  <math>\text{row\_indices} = \text{bodys}</math>, <math>\text{col\_indices} = \text{heads}</math>: [0 2] [0 1] The matched body-ID and head-ID are <math>\text{self.match\_body\_head} = [(0, 0), (2, 1), (1, \text{None})]</math></li> </ul>	0.003819	100,000.	100,000.	100,000.	100,000.	0.]																														
0.003819	100,000.																																				
100,000.	100,000.																																				
100,000.	0.]																																				
Input frame #2																																					
1	Bounding boxes in (x1, y1, x2, y2, confidence, class_id) = tensor ([ <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>1100.2812</td> <td>274.1615</td> <td>1218.8927</td> <td>563.6195</td> <td>0.9977</td> <td>0</td> </tr> <tr> <td>1317.1239</td> <td>199.0990</td> <td>1435.9489</td> <td>468.3447</td> <td>0.9848,</td> <td>0</td> </tr> <tr> <td>733.1741</td> <td>0.5491</td> <td>783.7123</td> <td>96.3040</td> <td>0.9834</td> <td>0</td> </tr> <tr> <td>598.9022</td> <td>0.0000</td> <td>644.2981</td> <td>103.0769</td> <td>0.9701</td> <td>0</td> </tr> <tr> <td>1148.8118</td> <td>275.4363</td> <td>1190.7026</td> <td>322.5742</td> <td>0.9915</td> <td>1</td> </tr> <tr> <td>1383.2902</td> <td>199.0062</td> <td>1423.4681</td> <td>248.5701</td> <td>0.9893</td> <td>1])</td> </tr> </table> Number of Body IDs = 4, Number of Head IDs = 2.	1100.2812	274.1615	1218.8927	563.6195	0.9977	0	1317.1239	199.0990	1435.9489	468.3447	0.9848,	0	733.1741	0.5491	783.7123	96.3040	0.9834	0	598.9022	0.0000	644.2981	103.0769	0.9701	0	1148.8118	275.4363	1190.7026	322.5742	0.9915	1	1383.2902	199.0062	1423.4681	248.5701	0.9893	1])
1100.2812	274.1615	1218.8927	563.6195	0.9977	0																																
1317.1239	199.0990	1435.9489	468.3447	0.9848,	0																																
733.1741	0.5491	783.7123	96.3040	0.9834	0																																
598.9022	0.0000	644.2981	103.0769	0.9701	0																																
1148.8118	275.4363	1190.7026	322.5742	0.9915	1																																
1383.2902	199.0062	1423.4681	248.5701	0.9893	1])																																
2	<ul style="list-style-type: none"> <li>Intra-frame processing: delete the false positives: Algorithm 2: <math>\text{delta\_del\_head} = []</math>, Algorithm 3: <math>\text{delta\_del\_body} = [3]</math>. [598.9022, 0.0000, 644.2981, 103.0769, 0.9701, 0] is deleted.</li> </ul>																																				
3	<ul style="list-style-type: none"> <li>Head_tracker in Algorithm 1: the first match: <math>\text{matches\_a}</math>, <math>\text{unmatched\_tracks\_a}</math>, <math>\text{unmatched\_detections} = [(0, 0), (1, 1)] [] []</math> the second match: <math>\text{matches\_b}</math>, <math>\text{unmatched\_tracks\_b}</math>, <math>\text{unmatched\_detections} = [] [] []</math></li> </ul>																																				

Table A1. Cont.

No.	Calculation Descriptions																																			
	Input frame #2																																			
4	<ul style="list-style-type: none"> <li>Body_tracker in Algorithm 1:</li> </ul> the first match: matches_a, unmatched_tracks_a, unmatched_detections = [(0, 0), (1, 2), (2, 1)] [] [] the second match: matches_b, unmatched_tracks_b, unmatched_detections= [] [] []																																			
5	<ul style="list-style-type: none"> <li>Inter-frame matching in Algorithm 4:</li> </ul> The matched body-ID and head-ID is: self.match_body_head = [(0, 0), (2, 1), (1, None)] body-np_xyxy_final = [ <table border="1" style="margin-left: 20px;"> <tr> <td>1100.495617</td> <td>274.254816</td> <td>1218.957032</td> <td>563.566199</td> <td>0.997711</td> <td>0</td> </tr> <tr> <td>731.006493</td> <td>0.616912</td> <td>786.759376</td> <td>96.442545</td> <td>0.983394</td> <td>1</td> </tr> <tr> <td>1315.511496</td> <td>198.934008</td> <td>1436.811807</td> <td>468.587705</td> <td>0.984758</td> <td>2]</td> </tr> </table>	1100.495617	274.254816	1218.957032	563.566199	0.997711	0	731.006493	0.616912	786.759376	96.442545	0.983394	1	1315.511496	198.934008	1436.811807	468.587705	0.984758	2]																	
1100.495617	274.254816	1218.957032	563.566199	0.997711	0																															
731.006493	0.616912	786.759376	96.442545	0.983394	1																															
1315.511496	198.934008	1436.811807	468.587705	0.984758	2]																															
	⋮ ⋮																																			
	Input the frame #7																																			
1	Bounding boxes in (x1, y1, x2, y2, confidence, class_id) = tensor ([ <table border="1" style="margin-left: 20px;"> <tr> <td>1103.0753</td> <td>278.1667</td> <td>1218.2045</td> <td>573.1316</td> <td>0.9963</td> </tr> <tr> <td>1327.7861</td> <td>199.1935</td> <td>1435.1885</td> <td>466.8471</td> <td>0.9849</td> </tr> <tr> <td>576.0401</td> <td>0.9442</td> <td>636.5553</td> <td>104.2498</td> <td>0.9785</td> </tr> <tr> <td>721.5332</td> <td>2.7390</td> <td>797.3731</td> <td>105.7765</td> <td>0.9664</td> </tr> <tr> <td>646.1354</td> <td>0.9752</td> <td>769.4316</td> <td>103.6911</td> <td>0.8158</td> </tr> <tr> <td>1153.6139</td> <td>279.1069</td> <td>1196.0039</td> <td>327.2707</td> <td>0.9968</td> </tr> <tr> <td>1388.5093</td> <td>198.7742</td> <td>1425.3273</td> <td>243.2225</td> <td>0.9838]</td> </tr> </table> Number of Body IDs = 5, Number of Head IDs = 2.	1103.0753	278.1667	1218.2045	573.1316	0.9963	1327.7861	199.1935	1435.1885	466.8471	0.9849	576.0401	0.9442	636.5553	104.2498	0.9785	721.5332	2.7390	797.3731	105.7765	0.9664	646.1354	0.9752	769.4316	103.6911	0.8158	1153.6139	279.1069	1196.0039	327.2707	0.9968	1388.5093	198.7742	1425.3273	243.2225	0.9838]
1103.0753	278.1667	1218.2045	573.1316	0.9963																																
1327.7861	199.1935	1435.1885	466.8471	0.9849																																
576.0401	0.9442	636.5553	104.2498	0.9785																																
721.5332	2.7390	797.3731	105.7765	0.9664																																
646.1354	0.9752	769.4316	103.6911	0.8158																																
1153.6139	279.1069	1196.0039	327.2707	0.9968																																
1388.5093	198.7742	1425.3273	243.2225	0.9838]																																
2	<ul style="list-style-type: none"> <li>Intra-frame processing: delete the false positives:</li> </ul> Algorithm 2: delta_del_head = [], Algorithm 3: delta_del_body = [4]. [646.1354, 0.9752, 769.4316, 103.6911, 0.8158] is deleted.																																			
3	<ul style="list-style-type: none"> <li>Head_tracker in Algorithm 1:</li> </ul> the first match: matches_a, unmatched_tracks_a, unmatched_detections = [(0, 0), (1, 1)] [] [] the second match: matches_b, unmatched_tracks_b, unmatched_detections = [] [] []																																			
4	<ul style="list-style-type: none"> <li>Body_tracker in Algorithm 1:</li> </ul> the first match: matches_a, unmatched_tracks_a, unmatched_detections = [(0, 0), (1, 2), (2, 1)] [3] [] the second match: matches_b, unmatched_tracks_b, unmatched_detections = [] [3] []																																			
5	<ul style="list-style-type: none"> <li>Inter-frame matching in Algorithm 4:</li> </ul> Find a new added body-ID: initiate_np_xyxy = [[721.533203, 2.738953, 797.373108, 105.776482, 0.966378, 3]] head-candidates_tlwh = [ <table border="1" style="margin-left: 20px;"> <tr> <td>1153.614397</td> <td>278.977509</td> <td>42.391931</td> <td>48.180364</td> </tr> <tr> <td>1388.411052</td> <td>198.606596</td> <td>37.066294</td> <td>44.685888]</td> </tr> </table> body_id_list= [0, 2, 1]; head_id_list = [0, 1, None] #match between one newly added body-ID and two existing head-ID cost_matrix = [[100,000. 100,000.]] #100,000 is bigger than the threshold, then body ID = 3 with head-ID = None self.match_body_head = [(0, 0), (2, 1), (1, None), (3, None)] #the newly added track ID's bounding box is not in the current frame, but will be shown in the next frame. body-np_xyxy_final = [ <table border="1" style="margin-left: 20px;"> <tr> <td>1102.914528</td> <td>278.100212</td> <td>1218.567574</td> <td>572.995262</td> <td>0.996286</td> <td>0</td> </tr> <tr> <td>570.531019</td> <td>0.777104</td> <td>631.442964</td> <td>104.358195</td> <td>0.978527</td> <td>1</td> </tr> <tr> <td>1322.21706</td> <td>198.937752</td> <td>1440.727778</td> <td>466.620414</td> <td>0.984871</td> <td>2]</td> </tr> </table>	1153.614397	278.977509	42.391931	48.180364	1388.411052	198.606596	37.066294	44.685888]	1102.914528	278.100212	1218.567574	572.995262	0.996286	0	570.531019	0.777104	631.442964	104.358195	0.978527	1	1322.21706	198.937752	1440.727778	466.620414	0.984871	2]									
1153.614397	278.977509	42.391931	48.180364																																	
1388.411052	198.606596	37.066294	44.685888]																																	
1102.914528	278.100212	1218.567574	572.995262	0.996286	0																															
570.531019	0.777104	631.442964	104.358195	0.978527	1																															
1322.21706	198.937752	1440.727778	466.620414	0.984871	2]																															

Table A1. Cont.

No.	Calculation Descriptions					
	Input the frame #8					
	body-np_xyxy_final = [					
	1104.565028	282.556513	1212.796813	574.978288	0.99675	0
5	563.08197	1.296683	623.345229	103.507251	0.983013	1
	1323.317717	198.276943	1441.398572	466.717254	0.983189	2
	722.836121	2.213905	787.554428	106.905021	0.941132	3] #new added body-ID = 3

## References

- Teizer, J. Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites. *Adv. Eng. Inform.* **2015**, *29*, 225–238. [CrossRef]
- Xiao, B.; Xiao, H.; Wang, J.; Chen, Y. Vision-based method for tracking workers by integrating deep learning instance segmentation in off-site construction. *Autom. Constr.* **2022**, *136*, 104148. [CrossRef]
- Golizadeh, H.; Hon, C.K.H.; Drogemuller, R.; Hosseini, M.R. Digital engineering potential in addressing causes of construction accidents. *Autom. Constr.* **2018**, *95*, 284–295. [CrossRef]
- Freimuth, H.; Koenig, M. Planning and executing construction inspections with unmanned aerial vehicles. *Autom. Constr.* **2018**, *96*, 540–553. [CrossRef]
- Guo, H.; Yu, Y.; Skitmore, M. Visualization technology-based construction safety management: A review. *Autom. Constr.* **2017**, *73*, 135–144. [CrossRef]
- Luiten, J.; Os, A.A.; Dendorfer, P.; Torr, P.; Geiger, A.; Leal-Taixé, L.; Leibe, B. HOTA: A Higher Order Metric for Evaluating Multi-object Tracking. *Int. J. Comput. Vis.* **2021**, *129*, 548–578. [CrossRef] [PubMed]
- Zhang, Y.; Sun, P.; Jiang, Y.; Yu, D.; Weng, F.; Yuan, Z.; Luo, P.; Liu, W.; Wang, X. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. *arXiv* **2022**, arXiv:2110.06864. [CrossRef]
- Wojke, N.; Bewley, A.; Paulus, D. Simple Online and Realtime Tracking with a Deep Association Metric. In Proceedings of the 24th IEEE International Conference on Image Processing (ICIP), Beijing, China, 17–20 September 2017; pp. 3645–3649. [CrossRef]
- Cao, J.; Pang, J.; Weng, X.; Khirodkar, R.; Kitani, K. Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking. *arXiv* **2022**, arXiv:2203.14360. [CrossRef]
- Liu, Y.; Zhou, Z.; Wang, Y.; Sun, C. Head-Integrated Detecting Method for Workers under Complex Construction Scenarios. *Buildings* **2024**, *14*, 859. [CrossRef]
- Dendorfer, P.; Ošep, A.; Milan, A.; Schindler, K.; Cremers, D.; Reid, I.; Roth, S.; Leal-Taixé, L. MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking. *arXiv* **2020**, arXiv:2010.07548. [CrossRef]
- Leal-Taixé, L.; Milan, A.; Reid, I.; Roth, S.; Schindler, K. MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking. *arXiv* **2015**, arXiv:1504.01942. [CrossRef]
- Ciaparrone, G.; Sánchez, F.L.; Tabik, S.; Troiano, L.; Tagliaferri, R.; Herrera, F. Deep Learning in Video Multi-Object Tracking: A Survey. *arXiv* **2019**, arXiv:1907.12740. [CrossRef]
- Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *arXiv* **2016**, arXiv:1506.01497. [CrossRef] [PubMed]
- Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2015**, arXiv:1506.02640. [CrossRef]
- Duan, K.; Bai, S.; Xie, L.; Qi, H.; Huang, Q.; Tian, Q. CenterNet: Keypoint Triplets for Object Detection. *arXiv* **2019**, arXiv:1904.08189. [CrossRef]
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. *arXiv* **2017**, arXiv:1706.03762. [CrossRef]
- Girshick, R. Fast R-CNN. *arXiv* **2015**, arXiv:1504.08083. [CrossRef]
- Bewley, A.; Ge, Z.Y.; Ott, L.; Ramov, F.; Upcroft, B. Simple Online and Realtime Tracking. In Proceedings of the 23rd IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, 25–28 September 2016; pp. 3464–3468. [CrossRef]
- He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. *arXiv* **2018**, arXiv:1703.06870. [CrossRef]
- Bashar, M.; Islam, S.; Hussain, K.K.; Hasan, M.B.; Rahman, A.B.M.A.; Kabir, M.H. Multiple Object Tracking in Recent Times: A Literature Review. *arXiv* **2022**, arXiv:2209.04796. [CrossRef]
- Shuai, B.; Berneshawi, A.; Li, X.; Modolo, D.; Tighe, J. SiamMOT: Siamese Multi-Object Tracking. *arXiv* **2021**, arXiv:2105.11595. [CrossRef]
- Meinhardt, T.; Kirillov, A.; Leal-Taixé, L.; Feichtenhofer, C. TrackFormer: Multi-Object Tracking with Transformers. *arXiv* **2021**, arXiv:2101.02702. [CrossRef]
- Sun, P.; Cao, J.; Jiang, Y.; Zhang, R.; Xie, E.; Yuan, Z.; Wang, C.; Luo, P. TransTrack: Multiple Object Tracking with Transformer. *arXiv* **2020**, arXiv:2012.15460. [CrossRef]
- Google. Google Colaboratory. 2023. Available online: <https://colab.research.google.com/> (accessed on 29 August 2023).

26. Maggiolino, G.; Ahmad, A.; Cao, J.; Kitani, K. Deep OC-SORT: Multi-Pedestrian Tracking by Adaptive Re-Identification. *arXiv* **2023**, arXiv:2302.11813. [[CrossRef](#)]
27. Yang, M.; Han, G.; Yan, B.; Zhang, W.; Qi, J.; Lu, H.; Wang, D. Hybrid-SORT: Weak Cues Matter for Online Multi-Object Tracking. *arXiv* **2023**, arXiv:2308.00783. [[CrossRef](#)]
28. Duan, P.; Zhou, J.; Goh, Y.M. Spatial-temporal analysis of safety risks in trajectories of construction workers based on complex network theory. *Adv. Eng. Inform.* **2023**, *5*, 101990. [[CrossRef](#)]
29. Aharon, N.; Orfaig, R.; Bobrovsky, B.-Z. BoT-SORT: Robust Associations Multi-Pedestrian Tracking. *arXiv* **2022**, arXiv:2206.14651. [[CrossRef](#)]
30. Du, Y.; Zhao, Z.; Song, Y.; Zhao, Y.; Su, F.; Gong, T.; Meng, H. StrongSORT: Make DeepSORT Great Again. *arXiv* **2022**, arXiv:2202.13514. [[CrossRef](#)]
31. Wang, Z.; Zhao, H.; Li, Y.L.; Wang, S.; Torr, P.; Bertinetto, L. Do Different Tracking Tasks Require Different Appearance Models? *arXiv* **2021**, arXiv:2107.02156. [[CrossRef](#)]
32. Zhang, Y.; Wang, C.; Wang, X.; Zeng, W.; Liu, W. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *arXiv* **2020**, arXiv:2004.01888. [[CrossRef](#)]
33. Chu, P.; Wang, J.; You, Q.; Ling, H.; Liu, Z. TransMOT: Spatial-Temporal Graph Transformer for Multiple Object Tracking. *arXiv* **2021**, arXiv:2104.00194. [[CrossRef](#)]
34. Zheng, L.; Shen, L.; Tian, L.; Wang, S.; Bu, J.; Tian, Q. Person Re-identification Meets Image Search. *arXiv* **2015**, arXiv:1502.02171. [[CrossRef](#)]
35. Bernardin, K.; Stiefelhagen, R. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *Eurasip J. Image Video Process.* **2008**, *2008*, 246309. [[CrossRef](#)]
36. Ristani, E.; Solera, F.; Zou, R.S.; Cucchiara, R.; Tomasi, C. Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking. *arXiv* **2016**, arXiv:1609.01775. [[CrossRef](#)]
37. KubaRurak. Detectron2-Deepsort-Repo. 2021. Available online: <https://github.com/KubaRurak/detectron2-deepsort-repo> (accessed on 31 August 2023).
38. Konstantinou, E.; Lasenby, J.; Brilakis, I. Adaptive computer vision-based 2D tracking of workers in complex environments. *Autom. Constr.* **2019**, *103*, 168–184. [[CrossRef](#)]
39. JonathonLuiten. TrackEval. 2021. Available online: <https://github.com/JonathonLuiten/TrackEval> (accessed on 31 August 2023).
40. Mikel-Brostrom. YOLO\_Tracking. 2023. Available online: [https://github.com/mikel-brostrom/yolo\\_tracking#real-time-multi-object-segmentation-and-pose-tracking-using-yolov8--yolo-nas--yolox-with-deepocsort-and-lightmbn](https://github.com/mikel-brostrom/yolo_tracking#real-time-multi-object-segmentation-and-pose-tracking-using-yolov8--yolo-nas--yolox-with-deepocsort-and-lightmbn) (accessed on 31 August 2023).
41. pmj110119. YOLOX\_Deepsort\_Tracker. 2021. Available online: [https://github.com/pmj110119/YOLOX\\_deepsort\\_tracker](https://github.com/pmj110119/YOLOX_deepsort_tracker) (accessed on 31 August 2023).
42. Xiao, B.; Kang, S.-C. Vision-Based Method Integrating Deep Learning Detection for Tracking Multiple Construction Machines. *J. Comput. Civ. Eng.* **2021**, *35*, 04020071. [[CrossRef](#)]
43. Xiao, B.; Lin, Q.; Chen, Y. A vision-based method for automatic tracking of construction machines at nighttime based on deep learning illumination enhancement. *Autom. Constr.* **2021**, *127*, 13. [[CrossRef](#)]
44. Drive, G. Deepsort\_Parameters. 2023. Available online: [https://drive.google.com/drive/folders/1xhG0kRH1EX5B9\\_Iz8gQJb7UNnn\\_riXi6](https://drive.google.com/drive/folders/1xhG0kRH1EX5B9_Iz8gQJb7UNnn_riXi6) (accessed on 31 August 2023).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.