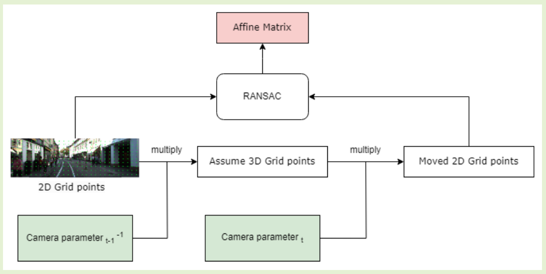
Camera Motion Compensation Based On Camera Parameter

***Abstract*—In this paper, we propose a method to calculate the camera motion through camera parameters and thereby correct the Kalman filter through the camera motion compensation mechanism, this method improves the overall performance of object tracking. Our method uses the change in camera parameters from frame to frame to calculate the displacement caused by camera movement on the target image and thereby corrects the prediction of the Kalman filter. Replacing the sparse optical flow algorithm with this method can significantly reduce the amount of calculation required for the tracking process and reduce the method's dependence on the keypoints of image. In addition, in order to reduce the ID-switch caused by compensation error, we were inspired by papers related to target tracking. In the target association stage, the prediction frame and detection frame are appropriately expanded according to the position between the targets to reduce Tracking is lost due to small errors, which enables our tracking model to achieve faster and more stable performance. We used the KITTI data set to verify our method and compared it with other MOT models. The experimental results show that our method can get better scores in pedestrian recognition performance, and in the execution speed experiment, it is better than the original one. The execution speed of the method can be doubled.**



***Index Terms*—camera motion compensation, data association, kalman filter, camera parameter, multi-object tracking(MOT).**

I. Introduction

MULTIPLE object tracking is a widely used technique with applications ranging from self-driving cars to sports science. Consequently, many people are actively involved in research related to this technology. However, multiple object tracking remains a challenging task, with many issues still requiring improvement, such as occlusion, similar appearances, and interactions between targets.

Nowaday, most multiple object tracking systems follow the paradigm of Tracking-By-Detection [1]. Firstly, a detector is used to identify the locations of targets in the current frame. Then, dynamic system models like the Kalman filter are employed to estimate the predicted bounding boxes based on the target positions from the previous frame. Next, the predicted boxes are matched with the detected boxes based on their positions and appearance features, and each matched pair is assigned a tracking ID. This ID links the same target across frames, forming trajectories representing the tracking of that target.

For predicting the next frame's position of targets, there are generally two main approaches. One is using deep learning models for position prediction [9], [10], and the other is using the Kalman filter [2], [6], [8]. The advantage of the Kalman filter lies in its low computational cost and the absence of the need for pre-training. The Kalman filter constructs a Gaussian distribution and its covariance matrix by computing the relationships between variables in the state vector. Then, based on this distribution, it can calculate the most probable state given the current frame's state vector. After prediction, the covariance matrix is updated based on the predicted results, facilitating subsequent frame position predictions.

Through the Kalman filter, we can effectively predict the motion state of targets. However, when the camera is moving, the movement of the camera causes corresponding shifts in the targets' positions in the image. The Kalman filter cannot account for this shift using historical information, and this shift becomes an external factor outside the system that interferes with prediction, leading to prediction inaccuracies. To address this issue, methods like BoT-SORT [3], [4], [5] utilize camera motion compensation.

For example, BoT-SORT predicts the image feature points of the next frame using sparse optical flow from the detected image feature points of the current frame. By matching the feature points between two frames, they obtain the deviation in the image caused by camera movement. This deviation is then used as an affine matrix to correct the state vector and covariance matrix of the Kalman filter. While this method effectively corrects problems caused by camera movement, there is still room for improvement. Predicting the next frame's feature points using sparse optical flow requires significant computational resources, and this prediction relies on the presence of feature points in the current frame, which may affect prediction accuracy when the image lacks feature points.

In this paper, we replace sparse optical flow with camera parameters. These parameters can be extracted during the camera calibration process. By computing the difference in external parameters between consecutive frames, we obtain the camera's movement state. Then, we utilize the camera pinhole model formula to transform the transformation information of 3D space into 2D space image offsets, thus calculating the image shift caused by camera movement. In our experiments, our computational approach, compared to sparse optical flow, increased processing speed from 9 frames per second to 20 frames per second. Moreover, camera parameters, unlike sparse optical flow, do not rely on image feature points and thus more accurately reflect the actual camera movement.

For matching dynamic targets, we also reference the practice of expanding bounding boxes during the matching stage[7]. We designed a new bounding box expansion strategy, allowing the model to maintain a more reasonable margin of error when matching dynamic targets.

II. Related Work

A. BoT-SORT

Although predicting the movement trajectory of targets using the Kalman Filter can yield good tracking results with a stationary camera, significant errors arise when the camera is moving. The reason for this is that the Kalman Filter only considers the positional changes of the target in the image coordinates when predicting its motion. When the camera moves, the target's coordinates are affected not only by its own movement but also by the camera's movement, causing the entire image to shift and leading to prediction errors.

To solve this, the BoT-SORT proposed by Aharon et al. [3] improves the Kalman Filter by incorporating a Camera Motion Compensation (CMC) mechanism when predicting the target's trajectory. This mechanism accounts for the displacement caused by camera movement. First, Shi-Tomasi corner detection is applied to the previous frame to extract key points from the image. Then, using sparse optical flow, the key points in the previous frame are matched to the current frame. RANSAC is then used to compute an affine transformation matrix based on the changes in these key points between the two frames, representing the displacement caused by camera movement.

Simultaneously, they proposed an improved Kalman Filter that corrects its state vector and covariance matrix using the affine transformation matrix before prediction. Since the translation part of the affine matrix only affects the center point of the bounding box, only the first two elements in the state vector representing the bounding box position are modified. The compensation method applying the affine transformation. This adjustment ensures that the prediction accounts for the displacement caused by camera movement. In this article, we are mainly based on this method and improve its computational disadvantages.

B. SMILEtrack

Based on the foundations of ByteTrack and BoT-SORT, Wang et al. [5] propose a MOT framework based on the SDE pattern (as shown in the figure below). In comparison to other models, they primarily address two issues. First, traditional methods like DeepSORT use simple CNNs to extract appearance features, which they argue fail to clearly distinguish appearance features among different objects. To extract more discriminative appearance features, they introduce a new module called SLM. Second, in the association part, they argue that using only IOU can lead to problems such as id-switch when targets are close. To address this issue, they propose a new association model called SMC, based on the previously mentioned ByteTrack and incorporating their SLM module.

In the overall context, they adopted the architecture of BoT-SORT, but they incorporated their SLM module during the first and second matches to enhance the matching of appearance features. The architecture of SLM is depicted in the following figure. By inputting the images of detected boxes from the previous and current frames into SLM, they are processed through their proposed ISA feature extractor, which shares parameters between the two images. After extracting features from the input images, they use a fully connected layer to integrate these features. To learn robust appearance features that can distinguish between various objects, they apply cosine similarity distance to compute the similarity between the two images. The similarity score between the same objects should be as high as possible; otherwise, the similarity score between different objects should be close to zero.

The architecture of the ISA feature extractor is as follows. Its purpose is to minimize model computation and parameter size while maintaining the excellent performance of the Transformer in feature enhancement. Inspired by the Vision Transformer (VIT), they constructed the ISA feature extractor using techniques such as image patching and attention mechanism. ISA first utilizes ResNet-18 to extract features from images within the detection boxes. Then, they divide the feature map into several smaller feature maps and embed one-dimensional positions into each small feature map. Finally, these small feature maps are inputted into the attention layer to extract attention features.

C. ExpansionIoU

The current Kalman filter used for motion feature tracking is linear. When the target deviates from linear motion patterns, there may be concerns about the predictions made by the Kalman filter. Therefore, ExpansionIoU achieves better robust tracking performance in dynamic scenes by iteratively amplifying the Intersection over Union (IOU) and deep features. During experiments, they found that expanding the bounding box during the association step can effectively improve tracking performance. Hence, they propose a method for expanding bounding boxes. The expansion of the bounding box is controlled by the expansion ratio E. Given an original bounding box with a height h and width w, we can calculate the expansion lengths h^* and w^* using the following formula:

$$\begin{aligned} h^* &= (2E+1)h \\ w^* &= (2E+1)w \end{aligned} \tag{1}$$

However, in this method, controlling the hyperparameters that govern the expansion of the bounding boxes greatly influences tracking performance. Although it's possible to adjust these hyperparameters post-experiment based on results, doing so would render real-time tracking unattainable. To address this issue, they propose a method to iteratively amplify the bounding boxes, thereby resolving the aforementioned problem.

$$E_t = E_{initial} + \lambda t \tag{2}$$

Where $E_{initial}$ is the initial expansion scale, λ represents the step size of the iterative expansion process, and t denotes the iteration count starting from 0. By employing this method, higher ExpansionIoU trajectories and detection pairs are first associated, followed by gradually searching for pairs with lower overlapping regions. This enhances the robustness of the association process.

III. Method

In this section, we will introduce how we enhance the efficiency and stability of multi-target tracking tasks through camera parameters. Building upon the BoT-SORT module, we have refined the method for calculating affine matrices representing camera motion. Before the module compensates for camera movement, we have introduced a module to calculate camera displacement based on parameters from consecutive frames. Additionally, we have incorporated an algorithm specifically targeting bounding boxes during the initial matching stage to reduce the likelihood of losing targets due to significant offsets.

A. Camera motion compensation

In object tracking tasks, Kalman filtering is commonly used to predict the trajectory of targets. Kalman filtering utilizes the target's positions in previous frames to calculate its velocity and predict its position in the next frame. Under normal circumstances, Kalman filtering can provide stable and accurate predictions. However, in dynamic camera scenarios, Kalman filtering fails to estimate the offset caused by camera jitter, resulting in positional deviations in predictions. To correct this kind of offset, the primary approach is to estimate the affine matrix representing the camera's motion trajectory and compensate for it within the Kalman filter. This allows the Kalman filter to consider information about camera motion for prediction.

For instance, BoT-SORT extracts keypoints from images, then utilizes sparse optical flow for feature tracking to find the positions of keypoints after camera movement. Subsequently, RANSAC is used to calculate the affine matrix representing camera displacement from keypoints before and after movement, enabling compensation. However, such methods rely to some extent on visual information in the scene. We believe that considering camera parameters would be a better choice for calculating the camera motion trajectory.

Camera parameters are commonly used in camera calibration tasks. Camera parameters are divided into two parts: internal parameters and external parameters. Internal parameters include the camera's focal length and principal point center information, while external parameters include rotation and translation information. Through camera parameters, points in three-dimensional space can be transformed into two-dimensional image coordinates. During this transformation process, the world coordinate system is transformed into a camera coordinate system with the camera lens center as the origin through the camera's external parameters.

It is known that external parameters can represent the position of the camera in the world coordinate system. Therefore, by observing the changes in external parameters between consecutive frames, we can obtain the camera's position changes in three-dimensional space. Subsequently, through internal parameters, we can project the changes in three-dimensional space onto two-dimensional images.

Thus, we propose a method to calculate the image offset caused by camera movement. In the process, we first obtain coordinate points (orange points) from the previous frame's image. Using the camera parameters from the previous frame, we project these coordinate points onto an assumed three-dimensional coordinate point (green points). Then, using this assumed three-dimensional coordinate point as the target position in the world coordinate system, we project it back onto the two-dimensional image plane using the camera parameters from the current frame. This process yields the image points' deviation caused by the different positions of the camera due to its movement, allowing us to calculate the offset caused by camera movement.

Based on the above method, we incorporated a camera motion calculation module before camera movement compensation, replacing the original sparse optical flow calculation. The structure of the module is as follows:

After extracting camera parameters from the camera, which record rotation and translation information in three-dimensional data, we cannot directly correct the two-dimensional image. Therefore, we employ a camera coordinate projection method to transform it. Firstly, we establish a grid coordinate matrix for the previous frame's image. Then, using the formula for camera coordinate projection, we transform the coordinate points in the grid coordinate matrix into assumed three-dimensional points based on the camera parameters of the previous frame. The formula is as follows Equation (3):

$$\begin{bmatrix} x_a & y_a & z_a \end{bmatrix}^T = (P_{ex}^{t-1})^{-1} \cdot (P_{in}^{t-1})^{-1} \cdot \begin{bmatrix} x & y & 1 \end{bmatrix}^T \tag{3}$$

x_a, y_a, and z_a represent the transformed assumed three-dimensional points. $(P_{ex}^{t-1})^{-1}$ and $(P_{in}^{t-1})^{-1}$ denote the inverse matrices of the extrinsic and intrinsic matrices of the camera at time t-1, respectively. x and y are the coordinate position points in the grid coordinate matrix established from the current frame's image. In this formula, we reverse the camera projection formula to reconstruct the two-dimensional coordinates on the image to three-dimensional space. Next, we project the assumed three-dimensional points back to the two-dimensional space using the camera parameters of the current frame. The formula is as follows Equation (4):

$$z_c \cdot \begin{bmatrix} x_c & y_c & 1 \end{bmatrix}^T = P_{ex}^t \cdot P_{in}^t \cdot \begin{bmatrix} x_a & y_a & z_a \end{bmatrix}^T \tag{4}$$

x_c and y_c represent the two-dimensional coordinate points of the pixel after camera movement. P_{in}^t and P_{ex}^t are the

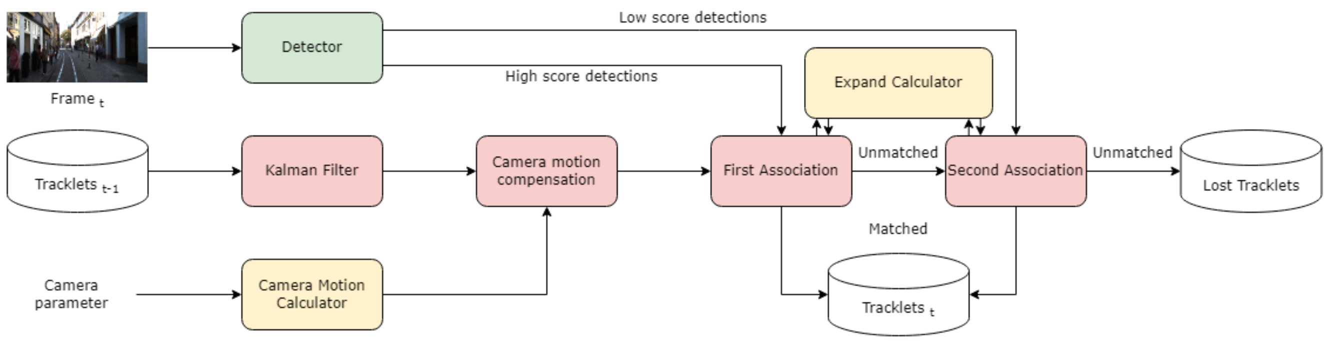


Fig. 1. Model architecture.

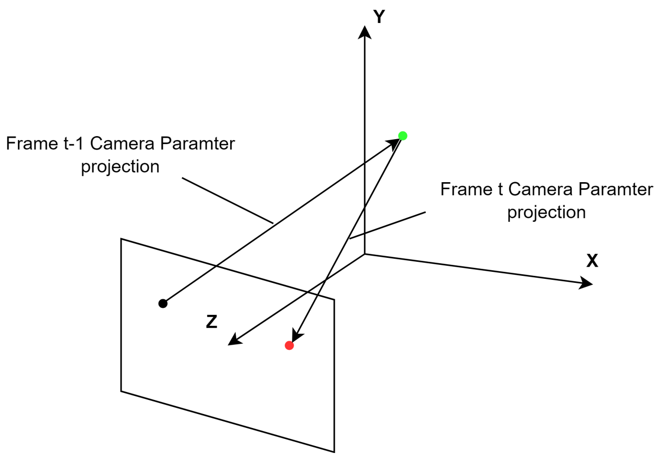

Y
Frame t-1 Camera Paramter projection
Frame t Camera Paramter projection
X
Z

Fig. 2. Coordinate projection diagram.

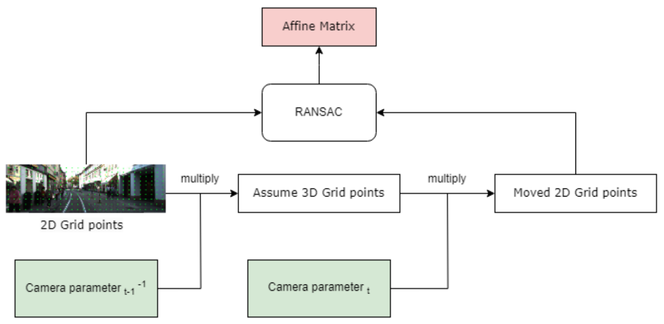


Fig. 3. Camera Motion Calculator.

intrinsic and extrinsic matrices of the camera at time t, respectively. x_a, y_a, and z_a are the assumed three-dimensional points transformed from Equation (3). Through the transformations in Equation (3) and Equation (4), we obtain the position of the previous frame's image coordinates after camera movement in the current frame. Next, following the Random Sample Consensus (RANSAC) algorithm used in BoT-SORT, we calculate the affine matrix of camera movement offset by matching the coordinates before and after transformation. Finally, we compensate for this affine matrix in the Kalman filter.

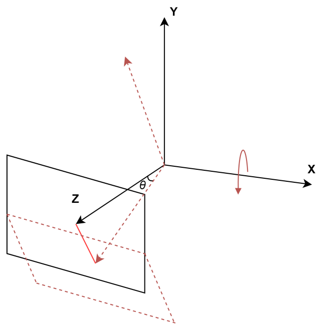


Fig. 4. For rotation with x-axis. It caused a shift in the image along the y-axis. We calculate the displacement of the image using the chord of a sector with an angle of θ_x .

B. Rotation angle compensation

By utilizing the camera parameters from consecutive frames, we can correct errors caused by camera movement more quickly. However, due to the lack of parameters during the reverse process, we can only compute assumed three-dimensional points. This limitation may result in less precise calculations in certain scenarios, thereby reducing tracking stability. Hence, in addition to the camera coordinate projection method, we aim to reduce errors by incorporating additional information.

From the rotation matrix in the camera's extrinsic parameters, we can obtain the rotation angles around the X, Y, and Z axes. By calculating the difference in rotation angles between consecutive frames, we can determine the rotation angles around the X, Y, and Z axes caused by the camera's movement between the two frames. We observed that in frames with errors, the rotation angles around the X, Y, and Z axes more closely match the deviations caused by real camera movement. Therefore, we compare the rotation direction with the displacement direction calculated through camera coordinate projection and correct the displacement direction in error frames.

Additionally, when the camera rotates around the Y axis, the target in the image will shift on the X axis; rotation around the X axis causes a shift on the Y axis, and rotation around the Z axis causes image rotation. Hence, we can calculate the affine matrix of image displacement caused by camera movement by adding the rotation angles to the translation matrix in

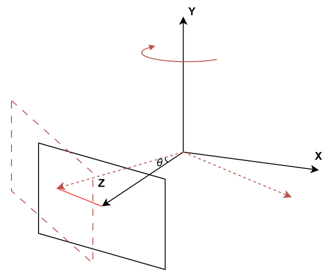


Fig. 5. For rotation with y-axis. It caused a shift in the image along the x-axis. We calculate the displacement of the image using the chord of a sector with an angle of θ_y .

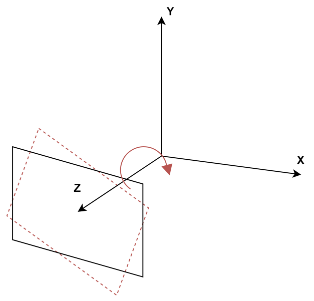


Fig. 6. For rotation with z-axis. caused a rotation in the image. We calculate the 2D rotation matrix using θ_z to determine the displacement it caused in the image.

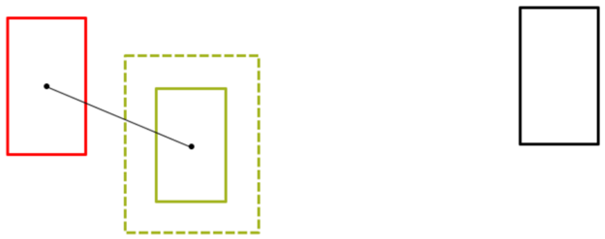


Fig. 7. For two closely spaced predicted boxes, we enlarge the distance between them by half, ensuring that the two predicted boxes do not overlap after expansion.

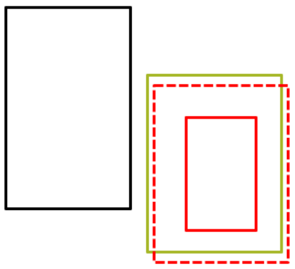


Fig. 8. For the predicted box (solid red box), we will search for the nearest neighboring box (green box) and expand the predicted box according to the expansion magnitude of that detection box, resulting in a dashed red box.

the extrinsic parameters. For the rotation matrix part, we can calculate the rotation around the Z axis using the formula for a two-dimensional rotation matrix. For the translation matrix part, we can calculate the displacement caused by camera rotation by computing the chord of a sector with the rotation angle as the sector angle and the focal length as the radius, as shown in Equation (5):

$$R = \begin{bmatrix} \cos(\theta_z^t - \theta_z^{t-1}) & -\sin(\theta_z^t - \theta_z^{t-1}) \\ \sin(\theta_z^t - \theta_z^{t-1}) & \cos(\theta_z^t - \theta_z^{t-1}) \end{bmatrix} \quad (5)$$

$$\begin{aligned} trans_x &= 2 \cdot f \cdot \sin((\theta_y^t - \theta_y^{t-1})/2) + (T_x^t - T_x^{t-1}) \\ trans_y &= 2 \cdot f \cdot \sin((\theta_x^t - \theta_x^{t-1})/2) + (T_y^t - T_y^{t-1}) \end{aligned} \quad (6)$$

$$Trans = \begin{bmatrix} trans_x & trans_y \end{bmatrix} \quad (7)$$

T is the calculated offset, f represents the focal length of the current frame parameters, R^t and R^{t-1} are the rotation angles between the current frame and the previous frame,T^t and T^{t-1} are the values in the translation matrices of the current and previous frames, respectively. With this, we can obtain a set of affine matrices that follow the same camera coordinate projection method. Finally, we blend the two sets of affine matrices according to Equation (8) and compensate for them in the Kalman filter, completing the compensation for camera motion offset.

$$A = \begin{bmatrix} R_{CMC} \cdot W + R_{RAC} \cdot (1 - W_m) & Trans_{CMC} \cdot W + Trans_{RAC} \cdot (1 - W_m) \end{bmatrix} \quad (8)$$

A represents the blended affine matrix, R_{CMC} and T_{CMC} are respectively the rotation matrix and translation matrix calculated by the camera coordinate projection method, R_{RAC} and T_{RAC} are respectively the rotation matrix and translation matrix calculated by the rotation angle method, and W_m is an adjustable weight parameter.

C. Expand Bounding box

After compensating for camera motion offset, we found that using camera parameters to correct predicted boxes results in more minor fluctuations compared to using sparse optical flow. These fluctuations can introduce instability, particularly when tracking smaller objects.

In the paper "Iterative Scale-Up ExpansionIoU and Deep Features Association for Multi-Object Tracking in Sports," the authors improve IOU matching performance by enlarging the bounding boxes. We believe that without causing ID-switch conditions to affect neighboring boxes, we should expand the boxes as much as possible. Therefore, we employ the following method to enlarge the boxes: first, for each detection box, we search for the nearest other detection boxes and expand them by half of the distance, as illustrated in Figure (7), ensuring that the detection boxes do not interfere with each other after expansion.

Since IoU matching is used during matching, when the detection boxes are expanded, the predicted boxes matched with them should also be expanded by the same extent. We search for the detection box closest to the predicted box and expand the predicted box according to the magnitude of expansion of that detection box, as shown in Figure (8).

IV. EXPERIMENTS

TABLE I
MOT RESULTS USING THE KITTI TRAINING SEQUENCE.

Method	MOTA↑	MOTP↑	Recall↑
AB3DMOT	66.98	67.77	72.82
StrongFusionMOT	73.63	-	-
EagerMOT	93.14	73.22	-
PolarMOT	93.48	-	93.66
Ours	**97.65**	**92.39**	**98.86**

A. Datasets

In common multi-object tracking datasets, the primary data typically consist of consecutive images along with the detected bounding box positions and target IDs. However, our method requires datasets to include the camera parameters for each frame, which poses a challenge when using widely adopted benchmarks like MOTChallenge for evaluating our method.

In our experiments, we choose to use the KITTI dataset. This dataset contains images captured by a stereo camera mounted on the roof of a vehicle, recording camera parameters for each frame. It consists of 21 training sequences and 29 test sequences, and each image sequence contains a different number of frames from 150 to 1100. Among them, we use the training sequence for testing and use the bounding box position in the label file it contains as the result of the detector. This approach aimed to eliminate potential experimental errors caused by different tracking models using different detectors and weights, thus focusing the experiment on comparing the tracking performance of various tracking models.

B. Implementation Details

In terms of performance evaluation, we mainly calculate MOTA, IDSW and other scores based on the CLEAR metric to compare with other excellent tracking models to test the effectiveness of the method. In addition, we also compared the processing speed (quantified in frames per second (FPS)) of the two tracking models in the tracking step with a prototype (SMILEtrack) with a small amount of data to compare the operating efficiency of different tracking models. In addition, we also divided our method into two parts: camera motion compensation and bounding box expansion, and tested their respective changes in tracking performance through ablation experiments.

C. Experimental Results

First, we compare it with other MOT models validated using the KITTI dataset. According to the official validation tool provided by KITTI, the test results can be divided into two parts: cars and pedestrians. Since our model is designed to track pedestrians, here we compare the pedestrian part with other models.

In this experiment, our method can reach 97.65 in MOTA, which is 4.17 higher than the second best PolarMOT, and is also higher than other models in MOTP, Recall, etc., as shown in Table I.

TABLE II
COMPARE MOTA WITH PROTOTYPE USING THE KITTI 0013 TRAINING SEQUENCE.

Method	MOTA↑	IDF1↑	IDP↑	IDR↑	IDSW↓
SMILEtrack	91.83	89.04	90.62	87.51	67
Ours	93.21	93.51	95.17	91.90	25

TABLE III
COMPARE PROCESSING SPEED WITH PROTOTYPE USING THE KITTI 0013 TRAINING SEQUENCE.

Method	Execution Time↓	Frame Per Second↑
SMILEtrack	54.47	9.94
Ours	26.32	20.58

Next, in order to verify the overall improvement of our method, we used the 0013 sequence in the KITTI data set to test our method and prototype (SMILEtrack) respectively. Compared to the original method which had 67 ID-switch occurrences, our method only had 25 ID-switch occurrences. We believe this improvement is attributed to expanding the detection frame, providing a more reasonable matching space for moving targets during the matching phase. This enhancement also resulted in better scores across other comprehensive metrics such as MOTA and IDF1, as shown in Table II..

On the other hand, we compared the processing speed of the trackers with the prototype in the same way, comparing the processing speed of different trackers based on the number of frames per second (FPS) they can handle. From the table below, we observe that after replacing sparse optical flow with our method, the processing time decreased from 54.47 to 26.32, and the frames per second increased from 9.94 to 20.58. These data demonstrate that our method achieves a speedup of more than double compared to the original method in terms of processing speed, as shown in Table III.

D. Ablation Study

In this part, we evaluated the performance of two proposed improvements separately: camera motion estimation and expanded box matching to understand how each of these two improvements can help track performance. Firstly, concerning the camera motion estimation module, we temporarily removed the expanded box matching part and compared with not compensate model using 0013 sequence, then compared the MOTA performance of the model with and without compensation for camera motion. With the inclusion of camera parameter calculation for compensation, there was a slight improvement in all evaluation scores, and the occurrence of ID-switch was slightly mitigated, as shown as Table IV.

Then, we conducted performance comparisons for the expanded box matching module. We utilized a model without

TABLE IV
CAMERA MOTION COMPENSATION ABLATION TEST USING THE KITTI 0013 TRAINING SEQUENCE.

Method	MOTA↑	IDF1↑	IDP↑	IDR↑	IDSW↓
Without Compensation	88.81	86.38	87.92	84.90	113
Camera parameter calculator	89.91	90.43	92.04	88.88	71

TABLE V
EXPANDED BOX MATCHING ABLATION TEST USING THE KITTI 0013 TRAINING SEQUENCE

Method	MOTA↑	IDF1↑	IDP↑	IDR↑	IDSW↓
Without Expanded	88.81	86.38	87.92	84.90	113
Expand Box	93.00	91.55	93.18	89.98	44

expansion and the 0013 sequence to compare its performance during the matching phase with and without the addition of the expanded box module. Experimental results indicate that through our expanded box strategy, the issue of bounding box loss can be effectively addressed. The MOTA score increased from 88.81 to 93.00, while the occurrence of ID-switch drastically decreased from 113 to 44 times, as shown as Table V.

V. CONCLUSION

In this paper, we propose a method based on the SMILE-Track tracking model architecture to replace the sparse optical flow approach with camera parameter calculation for camera displacement. This replacement aims to reduce the computational complexity required by sparse optical flow, and when combined with our proposed expanded box strategy, allows the model to have a more reasonable search range during the matching phase. Through MOTA comparisons, we evaluate the performance of our model. The experiments demonstrate that our model, when tracking under camera motion, significantly reduces occurrences of tracking loss and achieves tracking speeds of more than double compared to the baseline.

REFERENCES

[1] Zhang, Yifu, Chunyu Wang, Xinggang Wang, Wenjun Zeng and Wenyu Liu. "FairMOT: On the Fairness of Detection and Re-identification in Multiple Object Tracking." International Journal of Computer Vision 129 (2020): 3069 - 3087.

[2] Liu, Zelin, Xinggang Wang, Cheng Wang, Wenyu Liu and Xiang Bai. "SparseTrack: Multi-Object Tracking by Performing Scene Decomposition based on Pseudo-Depth." ArXiv abs/2306.05238 (2023): n. pag.

[3] Aharon, Nir, Roy Orfaig and Ben-Zion Bobrovsky. "BoT-SORT: Robust Associations Multi-Pedestrian Tracking." ArXiv abs/2206.14651 (2022): n. pag.

[4] Maggiolino, Gerard, Adnan Ahmad, Jinkun Cao and Kris Kitani. "Deep OC-Sort: Multi-Pedestrian Tracking by Adaptive Re-Identification." 2023 IEEE International Conference on Image Processing (ICIP) (2023): 3025-3029.

[5] Wang, Yuhan, Jun-Wei Hsieh, Ping-Yang Chen, Ming-Ching Chang, Hung Hin So and Xin Li. "SMILEtrack: SiMIlarity LEarning for Occlusion-Aware Multiple Object Tracking." AAAI Conference on Artificial Intelligence (2022).

[6] Yi, Kefu, Kai Luo, Xiaolei Luo, Jiangui Huang, Hao Wu, Rongdong Hu and Wei Hao. "UCMCTrack: Multi-Object Tracking with Uniform Camera Motion Compensation." AAAI Conference on Artificial Intelligence (2023).

[7] Huang, Hsiang-Wei, Cheng-Yen Yang, Jenq-Neng Hwang and Chung-I Huang. "Iterative Scale-Up ExpansionIoU and Deep Features Association for Multi-Object Tracking in Sports." 2024 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW) (2023): 163-172.

[8] Cao, Jinkun, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang and Kris Kitani. "Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking." 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022): 9686-9696.

[9] Zeng, Fangao, Bin Dong, Tiancai Wang, Cheng Chen, X. Zhang and Yichen Wei. "MOTR: End-to-End Multiple-Object Tracking with TRansformer." ArXiv abs/2105.03247 (2021): n. pag.

[10] Zhou, Xingyi, Tianwei Yin, Vladlen Koltun and Philipp Krähenbühl. "Global Tracking Transformers." 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2022): 8761-8770.