A Scale-Invariant Trajectory Simplification Method for Efficient Data Collection in Videos

Yang Liu
Magic Leap
yaliu@magicleap.com

Luiz G. Hafemann
Ubisoft La Forge
luiz@hafemann.ca

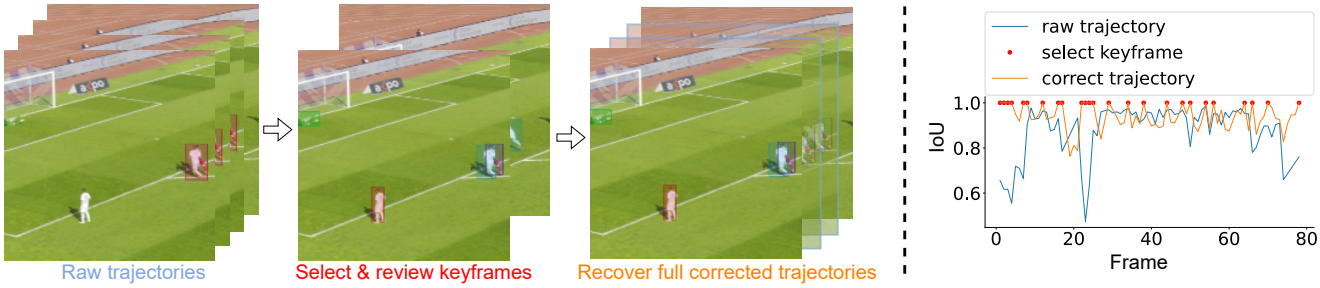


Figure 1. Illustration of our method, where the left part showcases the correction framework and the right part depicts the IoU score between the ground truth trajectory and trajectory in each step of the framework (represented by the corresponding color). More precisely, our approach can automatically choose crucial keyframes (red) for manual inspection and correction from the initial tracking trajectories (blue), which helps to minimize the annotation expenses. After that, the precise trajectories (orange) are reconstructed via interpolation.

Abstract

Training data is a critical requirement for machine learning tasks, and labeled training data can be expensive to acquire, often requiring manual or semi-automated data collection pipelines. For tracking applications, the data collection involves drawing bounding boxes around the classes of interest on each frame, and associate detections of the same "instance" over frames. In a semi-automated data collection pipeline, this can be achieved by running a baseline detection and tracking algorithm, and relying on manual correction to add/remove/change bounding boxes on each frame, as well as resolving errors in the associations over frames (track switches). In this paper, we propose a data correction pipeline to generate ground-truth data more efficiently in this semi-automated scenario. Our method simplifies the trajectories from the tracking systems and let the annotator verify and correct the objects in the sampled keyframes. Once the objects in the keyframes are corrected, the bounding boxes in the other frames are obtained by interpolation. Our method achieves substantial reduction in the number of frames requiring manual correction. In the MOT dataset, it reduces the number of frames by 30x while maintaining a HOTA score of 89.61% . Moreover, it reduces the number of frames by a factor of 10x while achieving a HOTA score of 79.24% in the SoccerNet dataset, and 85.79% in the DanceTrack dataset. The project code and data are publicly released at github/foreverYoungGitHub.

1. Introduction

Object detection and tracking are core problems for video sport analytics [3, 7, 17, 23, 28], as well as other computer vision applications, such as surveillance systems [22, 25, 26] and autonomous vehicles [12, 15, 19]. However, these tasks often require large datasets, and manual annotation in each frame is a time-consuming and expensive effort.

A practical semi-automated approach for data collection involves running off-the-shelf trackers, or an existing tracker for a specific application [4, 10, 30, 31], and rely on a manual cleanup process. This process would correct for mistakes made by the tracking system, by considering errors in the detections (e.g. adding missing bounding boxes, adjusting them or deleting spurious detections), and errors in the association over time (e.g. track switches).

Using tracking results as a prior for the annotation is often more efficient than starting from scratch, but verifying and cleaning the data for each frame still requires a significant effort. In this paper we explore ways to speed up the annotating process by subsampling the tracking data, selecting keyframes to be corrected, such that after correcting only the keyframes, the whole sequence can be obtained by interpolation.

A naive approach is to sample frames uniformly, but this approach is far from optimal, especially with complex tra-

jectories, where some frames are more important than others (e.g. player changing direction). In this case, we can improve over uniform sampling by finding the most important keyframes to compress the trajectory, that is, treating it as a trajectory simplification problem. Several methods for trajectory simplification have been proposed to compress trajectory data in point-based applications [5,6,9,20], such as GPS data. These methods aim to reduce the number of points in a given trajectory, keeping a subset of the points that preserve important information about it. However, these methods were designed for point-tracking, and they are not optimized for bounding boxes (where the scale of an object over time is as important as its position). Another key difference for the problem at hand is that the input trajectories are noisy, and we are interested in preserving the quality of the trajectories *after correction*. For this reason, the proposed method also takes the tracking confidence into consideration when selecting the keyframes.

In this paper, we introduce a scale-invariant trajectory simplification method for bounding box tracking, that shows potential to significantly reduce annotation times, while aiming to keep the tracking quality as close as possible to the quality obtained if all frames are corrected. Our method is shown in Fig 1: given existing (noisy) trajectory data, it selects keyframes to be corrected. In order to find the optimal simplified trajectory, the keyframes are selected both from high-quality observation and outliers, such that the result minimizes the error metric for the recovered trajectories. We introduce a scale-invariant error metric to guide the trajectory simplification, that penalizes scale changes of the objects in the image. After the keyframes are manually corrected, the full trajectory is recovered by linear interpolation of the keyframes. We perform a thorough evaluation on the MOT20 [8], SoccerNet [7] and DanceTrack [27] datasets. We consider two sets of experiments: (i) using tracking data from state-of-the-art detection ([11]) and tracking ([4], [30]) methods, and (ii) with synthetically corrupted ground truth tracks, that simulate common errors in trackers (e.g. bounding box jitter and track switches) and let us analyze the impact in performance as we vary the amount of noise in the tracking data. Our method is able to generate high-quality trajectory data even in scenarios where only $1/30$ of the frames are corrected in the MOT20 dataset, $1/5$ of the frames in the SoccerNet dataset and $1/10$ of the frames in the DanceTrack dataset, outperforming existing trajectory simplification methods.

The key contributions of our work to the object tracking community are as follows:

- We introduce a scale-invariant trajectory simplification method to speed up semi-automated data collection for object tracking in videos.
- We validate our proposed method on the MOT20, SoccerNet and DanceTrack datasets, that show improved performance compared to other trajectory simplification methods.

2. Related Work

Video and Interactive Annotation. In recent years, the demand for video annotation tools has increased due to their vital role in training data for visual tasks. Interactive recurrent annotation framework introduced by Le *et al*. [16], and semi-automatic annotation method proposed by Ince *et al*. [14] have gained popularity. However, these methods still require checking all frames during annotation, leading to high annotation costs.

Existing video annotation tools, such as CVAT [1] and VATIC [29], offer a practical solution by using linear interpolation to generate bounding boxes and points for most frames, while partially annotating the key frames. However, these tools have limitations as they cannot integrate with existing semi-automatic annotation methods: the annotation process becomes time-consuming and costly by given automatically generated tracking trajectories, as the tools typically mark every frame as a keyframe, and annotators need to review and remove unnecessary annotations.

Trajectory Simplification. Trajectory simplification presents one approach to automatically selecting useful keyframes in tracking trajectories. These methods have been utilized to compress trajectory data from a wide range of systems, including motion capture, touch screens, GPS, and IMUs. The goal of trajectory simplification is to reduce storage and computational resources, which is achieved by taking a sequence of size N and obtaining a subsequence with M points ($M \ll N$) that generates the minimum spatial distance error.

Dynamic programming (DP) [2] could be considered as the first algorithm for trajectory simplification, capable of guaranteeing to find the minimum error with O(N^3) time complexity. DP was later improved in [9], with an approximate simplification method with error bound guarantee. This method minimizes the perpendicular Euclidean distance (PED), which is the shortest distance between points and their anchor segments. An extension of DP called TD-TR [20] exploits the temporal dimension of trajectories and employs a new distance measure, called Synchronous Euclidean Distance (SED), that replaces the perpendicular distance used in DP when finding the split point with the maximum distance. SED considers the time information and uses it as the ratio to find the synchronized location of the points based on the linear interpolation. It calculates the distance between the actual point locations and their synchronized locations on the anchor segment.

A different approach involves constructing a Directed Acyclic Graph(DAG) and optimizing the trajectory by minimizing the integral of error metrics ϵ. Optimization-based

approaches [5, 6] compute the error metric for each corresponding timestamp and minimize the global integral error to improve performance. These methods use two integral errors, commonly known as integral square PED (ISPED) and integral square (ISSED).

It is worth noting that, unlike bounding box trajectories in vision problems, GPS or IMU trajectories are directly captured from the sensors and normally do not contain confidence scores. Therefore, current trajectory simplification methods do not rely on confidence information and cannot filter out low-quality observations. Additionally, these error functions present some issues when applied to bounding boxes, as they treat the two points that define a bounding box separately. This paper addresses these two issues.

3. Problem Formulation

We formulate the problem as follows: the inputs are noisy tracking observations $\mathbf{B}_n$, consisting of bounding boxes b_t, scores s_t for a set of frames T_n for a given track id n.

$$\mathbf{B}_n = \{b_t, s_t\}, t \in \mathbf{T}_n \tag{1}$$

The trajectory simplification task consists in finding the set $\mathbf{B'}_n$ that subsamples the frames obtaining $\mathbf{T'}_n$ ($|\mathbf{T'}_n| \ll |\mathbf{T}_n|$), aiming to preserve information on the original sequence. We call the ratio of $|\mathbf{T}_n|/|\mathbf{T'}_n|$ the *compression ratio*. More precisely, the simplified trajectory $\mathbf{B'}_n$ can be interpolated to recover the same number of frames of the original trajectory, obtaining $\mathbf{B''}_n$. We can view the subsampled frames as keyframes for the trajectory. In this paper, we consider that $\mathbf{B''}_n$ is recovered by linear interpolation of the simplified trajectory $\mathbf{B'}_n$, and we compare $\mathbf{B}_n$ and $\mathbf{B''}$ with bounding box metrics (IoU) [32] and tracking metrics (MOTA [21], HOTA [18]).

4. Proposed method

We present a trajectory correction framework that utilizes tracking data as input and requires manual correction only for a small subset of this data to obtain the entire corrected sequence. Figure 2 shows an overview of the framework: The proposed method categorizes observations into high-quality (green) and low-quality (blue) subsets. Trajectories are then simplified by selecting a small set of keyframes from the full set using algorithms described in subsequent subsections. Only the selected keyframes require manual review or correction. Groundtruth trajectories are then interpolated from the cleaned simplified trajectories.

4.1. Initializing the Searching Space

To initialize the search space for tracking trajectories, a set of high-quality bounding boxes and outliers of low-quality bounding boxes are selected. The details of this step are presented in Algorithm 1.

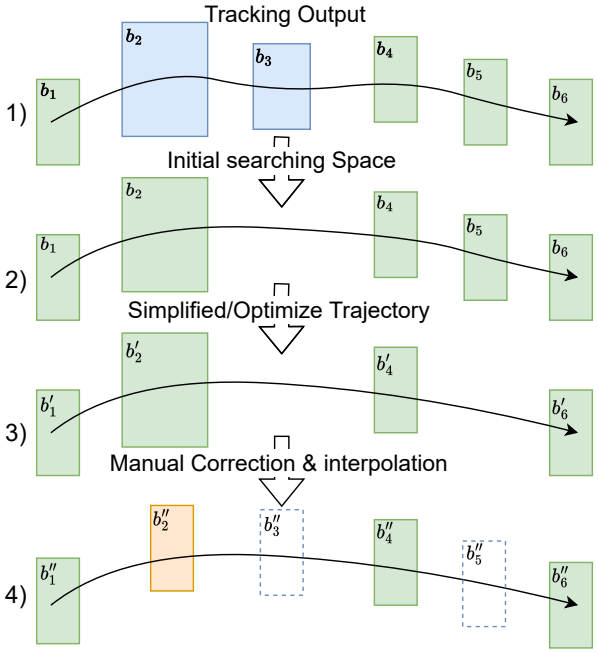


Figure 2. Illustration of the correction framework for one trajectory. Each line shows the trajectory in intermediate steps of the process: 1) the tracking trajectory, which is composed by high-quality observation (green) and the low-quality observation (blue); 2) the initialized searching space (shown in 4.1), which includes the high-quality observation and the outlier of low-quality observations (b_2); 3) the optimized trajectory in searching space by minimizing global error (shown in 4.2); 4) verified (green) and corrected (orange) the simplified trajectory, and recovered the whole trajectory by linear interpolation (dashed blue)

To filter out noisy observations and maintain necessary boxes to recover the trajectory, the high-quality bounding boxes are retained. Based on the assumption that a predicted bounding box b_t with a higher score s_t is closer to the ground truth box b''_t, the confidence scores s_t of predicted bounding boxes are used to identify high-quality bounding boxes.

However, it is also important to include outliers of the low-quality bounding boxes to cover scenarios such as motion blur or irregular appearance which cannot be perfectly interpolated. For selection of the outliers, we draw inspiration from the Douglas-Peucker algorithm [9]. For each anchor segment between high-confidence bounding boxes, we find the box b_t causing the largest error ϵ with respect to a given tolerance threshold ϵ_{th}. If the error ϵ is less than ϵ_{th}, the approximation is accepted, and we only keep the two high-confidence bounding boxes while discarding the remaining boxes within the segment. If the error ϵ is greater than ϵ_{th}, we split the segment into two sub-segments and add b_t into the searching space S.

Scale-Invariant Error Metrics for Bounding Boxes.

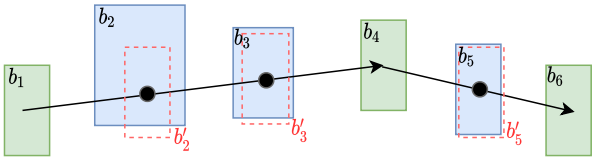


Figure 3. Illustration of Erf_{IoU}. The green boxes are the selected searching space, and it forms two tracking segments b_1b_4 and b_4b_6. The blue boxes are the predicted boxes not in the searching space, while the red boxes are the synchronized boxes for each tracking segment. The error metric for bounding boxes $\text{Erf}_{IoU}(b_t)$ calculates the IoU distance between the predicted bounding box b_t and synchronized box b'_t.

The central objective of our method is to generate highly compressed trajectories with low error. This is achieved by minimizing an error metric for the simplified trajectory. Error metrics proposed in the trajectory simplification literature are focused on point-based trajectories such as PED [9, 13] or SED [6, 20], which is not suitable for the bounding boxes trajectories.

In the case of bounding boxes, the Intersection over Union (IoU) score is a commonly used metric for both training and evaluation. IoU is a scale-invariant metric that performs well in pinhole geometry. In this paper, we propose the synchronized IoU distance as the error metric to simplify the tracking trajectories. The synchronized IoU distance is calculated based on the IoU distance between the actual boxes (b_t) and their corresponding synchronized boxes (b'_t) on the anchor segment. An example of the synchronized IoU distance is shown in Figure 3. Here, synchronized IoU distance of box b_5 in between the anchor segment b_4 and b_6 is calculated as the IoU distance between b_5 and the corresponding synchronized box b'_5, which is linearly interpolated based on the time information.

To make the simplified trajectory more robust and reduce manual corrections, we incorporate the confidence score s_t as weights into the synchronized IoU distance. The weighted IoU error metric is defined in Equation 2.

$$\begin{aligned} \text{Erf}_{\text{IoU}}(b_t, b'_t, s_t) &= s_t \times (1 - IoU(b_t, b'_t)) \\ &= s_t \times (1 - \frac{I(b_t, b'_t)}{U(b_t, b'_t)}) \end{aligned} \tag{2}$$

where $I(b_t, b'_t)$ means the intersection area of the actual predict bounding box b_t and synchronized box b'_t, while the $U(b_t, b'_t)$ means the union area of the b_t and b'_t, s_t means the confidence score of the predicted boxes b_t.

4.2. Simplification by minimizing the integral error

To reduce the number of nodes while minimizing the error metric, we utilize a Directed Acyclic Graph (DAG) that describes all potential simplified trajectories within a error

Algorithm 1 Initialize the Search Space

Input
$B = \{\{b_1, s_1\}, ..., \{b_T, s_T\}\}$: boxes with score.
Ω: Confidence threshold.
ϵ_{th}: Error tolerance.
Erf: Error metric.
Output
S: Search spaces.

$S = []$
for i in range(T) **do**
  if $s_i \geq \Omega$ **then** Append index i to S
for $i = 0$ to $|S| - 1$ **do**
  Get subset $\overline{B}$ of B from index $S[i]$ to $S[i+1]$
  indices $L = \text{Search}(\overline{B})$
  Insert indices L to S
return sort(S)

procedure SEARCH($\overline{B}, \epsilon$,Erf)
  if $len(\overline{B}) \leq 2$ **then return** $\{\}$
  $\epsilon, i = \max \text{Erf}(\overline{B})$
  if $\epsilon \leq \epsilon_{th}$ **then return** $\{\}$
  else
    Split $\overline{B}$ to $\overline{B_1}$, $\overline{B_2}$ at index i
    return $\{i\}$ + Search($\overline{B_1}$) + Search($\overline{B_2}$)

tolerance threshold ϵ_{th}. The DAG represents the observation nodes in the search space, with edges connecting any two vertices if the max error between them is less than the threshold ϵ_{th}. In order to minimize the global integral error from the root vertex V_0 to V_T, each vertex V_i stores the integral error from V_0 to V_i. The integral error of the node is obtained by integrating the previous integral error stored in parent vertices and the local errors between the current node and its connected parent node. In each step, the best parent node is selected to minimize the integral error. The DAG is constructed such that the integral error stored in V_T is the minimum of the global integral error. An example of a constructed DAG is shown in Figure 4, where the initial searching space $S = \{\mathbf{B}_0, \mathbf{B}_3, \mathbf{B}_4, \mathbf{B}_5, \mathbf{B}_7, \mathbf{B}_8, \mathbf{B}_9\}$. We begin by checking if the root box $\mathbf{B}_0$ can connect to other boxes. Here, since the max Erf_{IoU} between $\mathbf{B}_3, \mathbf{B}_4, \mathbf{B}_5$ and $\mathbf{B}_0$ is less than ϵ_{th}, we connect them to the root vertex. We do not build the connection between $\mathbf{B}_0$ and $\mathbf{B}_7$ since the max $\text{Erf}_{\text{IoU}}(\mathbf{B}_0 \rightarrow \mathbf{B}_7) > \epsilon_{\text{th}}$. We stop checking the remaining boxes once max $\text{Erf}_{\text{IoU}}(\mathbf{B}_0 \rightarrow \mathbf{B}_8) > \epsilon_{\text{th}}$. We then proceed to check the connections between $\mathbf{B}_7, \mathbf{B}_8, \mathbf{B}_9$ and the current parent vertices $\mathbf{B}_3, \mathbf{B}_4, \mathbf{B}_5$, until we construct the complete DAG.

To obtain the simplified trajectory, we follow the unique path from the last vertex B_T to the first vertex B_0 in reverse

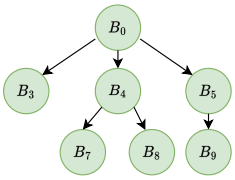


Figure 4. Illustration of constructed DAG, where the green nodes denote the boxes that are present in the search space S.

order. For instance, the final simplified trajectories in Fig 4 is $\{\mathbf{B}_0, \mathbf{B}_5, \mathbf{B}_9\}$. The algorithm for minimizing the error metric is provided in Algorithm 2. Notably, this approach is similar to previous work such as [5, 6]. By utilizing the DAG and minimizing the integral error, our approach can effectively reduce the number of nodes in the search space while maintaining a high level of accuracy.

General Integration (GI) Function. The integral error in each layer integrates local errors between the current node and parents and the integral error in the previous path stored at parents node. To form a general integral function, we extend previous integral functions, such as ISPED [6] and ISSED [5], by using the n-norm to combine errors, which is defined in the Equation 3.

$$\begin{aligned} GI_{0\to 1} &= |[\epsilon_0, \epsilon_1]|_n \\ GI_{0\to c} &= |[GI_{0\to p}, \epsilon_{p+1}, ..., \epsilon_c]|_n \\ &= |[\epsilon_0, ..., \epsilon_c]|_n \end{aligned} \tag{3}$$

where $GI_{0\to 1}$ is the integral error from the root vertex (index 0) to one of its child vertices (index 1), while ϵ_0 is the error at index 0. $GI_{0\to c}$ denotes the integral error from the root vertex (index 0) to the child vertex (index c), and $GI_{0\to p}$ represents the previous integral error from the root to parent vertex (index p) stored in the parent node.

The equation states that the n-norm of the error from the root vertex V_0 to V_c is equal to the n-norm of the previous integral error and the set of errors between the parent and child nodes. This means that the GI between the parent and leaf nodes is equal to the global integral function. By minimizing the GI in each layer, we can minimize the global integral error.

The value of n in the n-norm balances the mean and max of the error samples. When $n = 1$, it represents the sum of all the error metrics in each timestamp. When $n = 2$, it represents the integral square of the error metrics in each subseries. When n is infinite, the integral error represents the maximum error in each subseries.

5. Experimental Protocol

We conducted experiments to evaluate the proposed trajectory correction algorithm on both real tracking data and synthetic data with various levels of noise. In both cases, we utilize a simulated correction pipeline, by matching the predicted bounding boxes on the selected keyframes with the ground truth data. Specifically, this experimental protocol assumes that any frame selected for manual review is perfectly corrected. After correcting the keyframes, we interpolate the trajectory and report metrics that compare the interpolated corrected trajectory with the ground truth. It should be noted that we only used linear interpolation to ensure compatibility with existing annotation toolboxes.

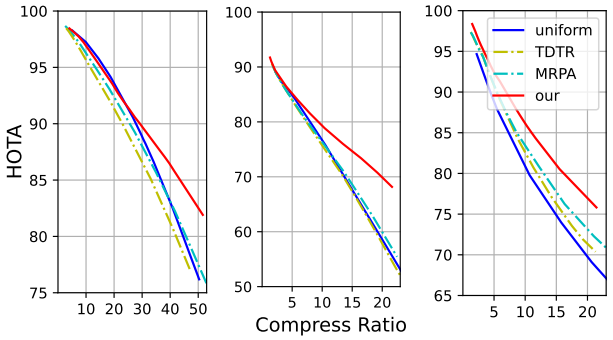
For these experiments, unless explicitly stated, the confidence threshold Ω was selected based on the top 10 percentile of each trajectory. The error metric used was the IoU distance, while the n value was set to 1 in the general integral function. As there are no similar approaches for this task, we compared our method with uniform sampling and the SOTA DP-based [20] and DAG-based [6] point trajectory simplification methods. For the latter, we simplified the bounding box trajectories using the top-left and bottom-right points of the box.

Datasets and Evaluation Metric. We evaluate our pro-

Algorithm 2 Minimizing the integral error

Input
$B = \{\{b_1, s_1\}, ..., \{b_T, s_T\}\}$: boxes with score.
$S = \{i_1, ..., i_M\}$: Search space.
ϵ_{th}: Error tolerance.
Erf: Error metric.
GI: Integral function.
Output
R: Result for the simplified indices.

```
//visit status V: unvisited, current leaf, current parent, visited
set V to visited for all the nodes in search space S
set V_0 as the root node with status current parent
set E = [0, inf, ..., inf] to store the integral errors GI_{0→t} for all B
set P for parent node index of all B
while V_end not visit do
    for i_c in unvisited nodes do
        for i_p in current parent nodes do
            Get subset B̄ of B from index i_p to i_c
            ε_B̄ = Erf(B̄)            ▷ ε_B̄ is a list of errors
            ε = max(ε_{B̄_S})
            GI_{0→i_c} = GI(E_{i_p}, ε_B̄)
            if ε < ε_th and GI_{0→i_c} < E_{i_c} then
                set V_{i_c} is current leaf
                set E_{i_c} = GI_{0→i_c} and P_{i_c} = i_p
            else if ε > 2ε_th then break to while
    set V of node in current parents to visited
    set V of node in current child to current parents
return ravel through the unique path from P_end → P_0 for parent node index of all B_S
```



(a) MOT20 (b) SoccerNet (c) DanceTrack

Figure 5. The HOTA score for trajectory correction as we vary the compression rate. The uniform sampling, TDTR [20], MRPA [6], and our proposed method are represented by the blue, yellow, cyan, and red curves, respectively. A higher accuracy score indicates better correction results.

posed method on MOT20 [8], SoccerNet [7] and DanceTrack [27] datasets. The MOT20 dataset [8] consists of multi-person tracking data for pedestrians, with the scale of the bounding box being relatively consistent in the full track. SoccerNet [7] is a tracking dataset consisting of multiple objects, including players, the ball, and referees, in soccer broadcast videos, that may involve camera movement. In this paper, we consider only tracking of the *person* class in the evaluation. The DanceTrack [27] dataset consists of multi-person tracking in dance videos. In DanceTrack, the bounding boxes vary significantly in the nearby frames and the entire sequence. To evaluate the accuracy of the interpolated corrected trajectory, we adopt IoU scores in detection metrics and MOTA [21] and HOTA [18] in tracking metrics.

Input tracking data. For the real tracking data, we apply YoloX [11] pretrained on CrowdHuman [24] and MOT20 [8] datasets to extract bounding boxes in each frame, and we use the combination of the ocsort [4] and byte [30] algorithms to track the bounding boxes without any re-identification features. We match the actual trajectory with the ground truth trajectory by Hungarian assignment in each frame to associate the bounding boxes with the corresponding ground truth data. Each actual trajectory is then simplified based on our proposed method, generating a simplified trajectory that is ready for data correction. To mimic the data correction process, we assign the ground truth bounding box and track IDs back to the keyframes in the simplified trajectory to fix the bounding box jitters and ID switches. In cases where the first or last objects in the ground truth trajectory do not have a corresponding match in the tracking data, we add these missing objects to the corrected trajectory in order to align it with the ground truth trajectory and facilitate meaningful comparison.

Synthetic data. To investigate the algorithm's sensitivity to different levels of noise in the input tracking data, we perform additional experiments using synthetic data.

MOT20 [8]	IoU(mean)	IoU(min)	MOTA	HOTA
Raw Tracking	-	-	93.28%	75.14%
Uniform(x30)	88.58%	60.52%	98.14%	88.59%
TD-TR [20](x30)	86.01%	59.47%	96.89%	82.23%
MRPA [6](x30)	87.44%	62.57%	98.58%	85.12%
ours(x30)	**91.37%**	**70.28%**	**99.36%**	**89.61%**
SoccerNet [7]	IoU(mean)	IoU(min)	MOTA	HOTA
Raw Tracking	-	-	49.22%	49.55%
Uniform(x10)	79.81%	12.55%	84.07%	75.77%
TD-TR [20](x10)	79.04%	14.34%	85.97%	74.18%
MRPA [6](x10)	80.08%	15.09%	87.66%	75.5%
ours(x10)	**82.96%**	**16.2%**	**89.7%**	**79.24%**
DanceTrack [27]	IoU(mean)	IoU(min)	MOTA	HOTA
Raw Tracking	-	-	91.33%	58.7%
Uniform(x10)	83.54%	31.75%	93.78%	79.78%
TD-TR [20](x10)	83.4%	37.58%	94.95%	80.72%
MRPA [6](x10)	84.31%	40.86%	95.09%	80.9%
ours(x10)	**87.54%**	**46.58%**	**98.46%**	**85.79%**

Table 1. The data correction accuracy for real trajectory in MOT20(upper) and DanceTrack(lower) datasets. The x30 and x10 in the scope means the compression rate is 30 times and 10 times respectively. The higher accuracy score means the better result we get.

We corrupt the ground truth trajectories with two common tracking mistakes: bounding box jitter and track switches. We simulate bounding box jitter by perturbing the bounding box positions (center and scale) with Gaussian noise. For track switches, we switch the tracking IDs of bounding boxes with high overlap (IoU > 0.5) with a probability p. A probability of 0 indicates that the input trajectory is noise-free and therefore the process is equivalent to simplifying ground truth trajectory. Following the same procedure used for the actual tracking data, we apply the correction methods to select the keyframes and correct them based on the matched ground truth data.

6. Result and Discussion

6.1. Tracking trajectory correction

To validate the performance of our algorithm on real tracking data, we experiment with different compression ratios on the three selected datasets. The compression ratio indicates how much we compressed the trajectory (e.g. a ratio of 10 indicate that 1/10th of the frames were selected for correction). We report the results in terms of HOTA scores, which are shown in Figure 5. Our algorithm outperforms existing methods on both datasets. In general, we observe that the higher the compression ratio, the greater the improvement gained from our algorithm.

Figure 5a revealed an intriguing pattern on the MOT20 dataset: at low compression rates, trajectory simplification methods may underperform compared to the straightforward uniform sampling. For instance, when the compression rate is less than 20x, our algorithm performs similarly or slightly worse than uniform sampling. Uniform sampling

works well in this scenario due to the MOT20 trajectories having linear motion within the short time window [27], while trajectory simplification methods tend to prioritize covering outliers caused by ID switches instead of reducing the frame gap, which could decrease detection accuracy (DetA). However, when the time window increases, non-linear motions become more prevalent within the trajectories, which is where our method excels by increasing the compression rate. Conversely, in the DanceTrack dataset (Figure 5c), all trajectory simplification methods outperform uniform sampling with all compression rates by selecting keyframes that capture the trajectories' significant variations resulting from large and non-linear motions.

Another important observation from Figure 5 is that reducing the compression rate is critical to obtaining high-quality ground truth trajectories for complex motions and noisy tracking trajectories. This is due to the fact that, for the same compression rate, the accuracy of the corrected trajectory in SoccerNet is lower than that in the less complex much cleaner MOT20 dataset. For instance, when our method is used to sample trajectories 20 times and correct them, the mean IoU and HOTA scores for the fully recovered trajectories in MOT20 are 93.93% and 93.16%, respectively, while the mean IoU and HOTA scores decrease to 75.48% and 70.76% in SoccerNet. It should be noted that a visual shift in bounding boxes may be observed when the IoU scores are lower than 90%.

Table 1 presents the accuracy scores of the raw tracking trajectories and the corrected trajectories. The simulated correction pipeline is effective in significantly boosting the HOTA scores since all id switches are corrected in the recovered trajectory. For instance, in the DanceTrack dataset, the HOTA score only achieves 58.7% in the raw tracking trajectories due to the low association accuracy (41.91%). However, our proposed correction method is capable of increasing the HOTA score to 85.79%.

The results in Table 1 demonstrate that the classic point-based trajectory simplification methods, TDTR [20] and MRPA [6], perform similarly or slightly worse than uniform sampling in both datasets when the mean IoU metric is considered. This suggests that these methods struggle to identify keyframes for the visual bounding box trajectories. In contrast, our proposed model achieves substantially higher mean IoU scores in all datasets. Moreover, although trajectory simplification methods may slightly underperform compared to uniform sampling for the entire trajectories, they are more effective in covering the outlier cases based on the min IoU metric. It is important to note that our proposed method has a more significant impact on the outlier cases, as compared to uniform sampling. In particular, using our method, the min IoU scores for the corrected trajectories increase by 9.76% and 14.83% for the MOT20 and DanceTrack datasets, respectively, demonstrating its effectiveness in handling challenging tracking scenarios.

HighConf	Outlier	min-ϵ	Dist	IoU_{mean}	IoU_{min}
	✓		SED	83.4%	37.58%
		✓	SED	84.31%	40.86%
	✓		IoU	86.03%	47.18%
		✓	IoU	87.33%	45.91%
✓		✓	IoU	86.81%	39.83%
✓	✓	✓	IoU	87.54%	46.58%
✓	✓	✓	DIoU	87.74%	46.87%

Table 2. Ablation studies on different modules in our proposed method, which were evaluated on the actual track generated from the DanceTrack dataset at an approximate compression rate of 10x.

6.2. Algorithm Modules Analysis

To understand our algorithm better, we carried out controlled experiments to examine how each component affects performance. For all experiments, we use the same settings and real tracking trajectory data, except for specified changes to the settings or component(s). The relative results are shown in Table 2.

Using scale-invariant error metrics, such as IoU, is crucial for accurate visual tracking annotation. Table 2 shows that adopting the IoU as the error metric leads to the most significant improvement, boosting mean IoU scores by 3%. Further improvements in accuracy can be achieved by using the DIoU [32] and CIoU [33] distance metrics, which increase mean IoU scores by 0.2%.

Combining high-quality and outlier boxes for initializing the searching space produces the best results. When optimizing the integral error for only high-quality bounding boxes, useful information is removed, resulting in a min IoU score of 39.83%. However, incorporating all bounding box trajectories can bias the simplified trajectories with noisy low-quality bounding boxes. By combining high-quality and outlier bounding boxes, our proposed method is able to filter noise while retaining necessary trajectory information, achieving a mean IoU score of 87.74% and a min IoU score of 46%.

6.3. Trajectory Noise Impact Analysis

We now consider the results on the synthetic dataset, in order to quantify the impact on performance of bounding box jitter and track id switches.

Correction of trajectories with noisy detection. The HOTA scores of the corrected trajectories are shown in Fig 6a. Here, we only use uniform sampling as a baseline reference since it is not impacted by the bounding box noise, and therefore only one curve is plotted for uniform sampling.

In Figure 6a, we observed that our proposed algorithm outperforms uniform sampling in general. The HOTA scores generated by our algorithm closely match the simplified ground truth trajectory ($p = 0$) when the noise prob-

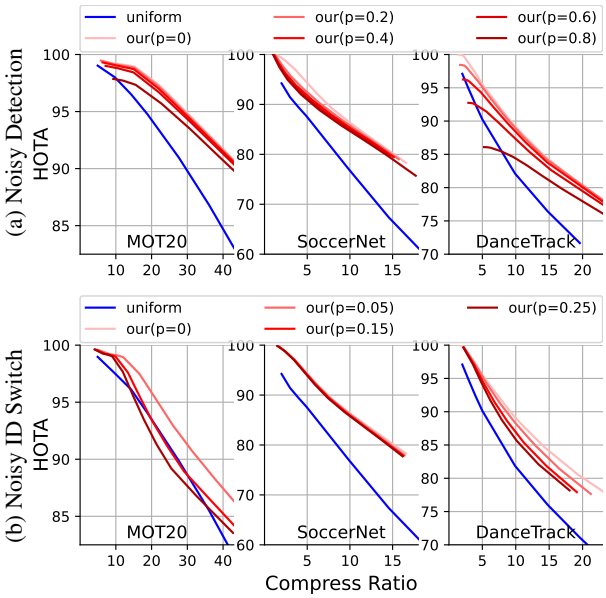


Figure 6. The HOTA score for trajectory correction with noisy bounding box (a) and noisy id switches (b). The blue curves represent the scores for uniform sampling, whereas the red curves correspond to the results obtained from our proposed methods. The label on the curves includes the noisy probability p, with darker lines indicating higher levels of detection noise / id switches in the trajectories.

ability is less than 0.4. This demonstrates that our algorithm effectively filters out the noisy bounding boxes and retains the most informative keyframes. However, the accuracy score notably decreases when the noisy jitter rate surpasses 0.4, especially at low compression rates. This is because nearly every bounding box experiences shifts when the noise probability is extremely high. In such cases, as we minimize the integral error by sampling more data, we inadvertently incorporate more noisy bounding box jitters. Thus, the proposed algorithm is highly sensitive to high levels of bounding box jitter.

Correction of trajectories with track id switches. The HOTA scores of the corrected trajectories are shown in Fig 6b. Our proposed algorithm consistently outperforms uniform sampling in all sampling rates and id switch noise rates. Moreover, the algorithm is able to capture errors that occur when id switches happen, resulting in HOTA scores that are close to the simplified ground truth trajectory when the noise rate is low. However, when the noise probability is high, such as in the case of the MOT20 dataset with a sampling rate of 20x, there is a concave curve in the HOTA scores of our algorithm. That is because there are many id switches that occur frequently back and forth in the adjacent frames with high noise probability. In this case, the uniform sampling directly skips some of the id switches segments, while our algorithm captures these id switches densely as they happen, sacrificing the accuracy of the rest of the trajectory, which supports our assumption in Section 6.1. It should be noted that the id switches were simulated at higher frequencies than what is typically observed in real tracking data, as the objective of this section was to evaluate the algorithm's performance under varied levels of noise.

It is worth noting that although we have added significant amounts of detection and tracking noise to generate the synthetic dataset, the trajectories remain mostly complete without any missing frames. This is in contrast to real data, as illustrated by the example of SoccerNet in Fig 5b), the HOTA score after correcting all frames in the tracking trajectories is only 92.22% due to the incompleteness of the generated trajectories by the pre-trained model. On the other hand, based on the result shown in Fig 6, if we can provide acceptable trajectories for SoccerNet generated by the fine-tuned model, our method can achieve 90.21% HOTA by correcting only 3 frames per second, and 94.75% HOTA by correcting only 5 frames per second. In comparison, the uniform sample method requires correcting half of the frames to achieve a HOTA score of 94.23%.

7. Limitation

Our experimental results have identified certain limitations of the proposed method. It has been observed that if the tracked motion predominantly follows a linear pattern, the proposed method may not demonstrate an improvement over uniform sampling. As observed in MOT20 dataset, where the proposed method only outperforms uniform sampling at compression rates exceeding 20x. Additionally, limited to the existing annotation tools, the use of linear interpolation between keyframes imposes a constraint on the extent to which complex trajectories can be compressed. As observed in the DanceTrack dataset, although the proposed method exhibits significant improvements over other methods, when the compression rate is over 10x, the IoU drops below 90% which leading to visible errors.

8. Conclusion

In this paper, we introduced a scale-invariant trajectory simplification method to minimize the annotation cost for semi-automated bounding box trajectory collection. The proposed method selects keyframes for each object in the video, such that only the keyframes needs manual review and correction. The experiments conducted on three popular tracking datasets demonstrate that our method can generate high-quality annotation data while requiring correction of significantly fewer frames. We also conducted ablation studies that showed that using a scale-invariant error metric is crucial for the task of simplifying bounding-box tracking data. Future work can consider extending these formulations to other vision trajectory tasks, such as pose tracking.

References

[1] Computer Vision Annotation Tool (CVAT). https://github.com/openvinotoolkit/cvat/, 2018. 2

[2] Richard Bellman. On the Approximation of Curves by Line Segments using Dynamic Programming. *Communications of the ACM*, 4(6):284, 1961. 2

[3] Amran Bhuiyan, Yang Liu, Parthipan Siva, Mehrsan Javan, Ismail Ben Ayed, and Eric Granger. Pose Guided Gated Fusion for Person Re-identification. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2675–2684, 2020. 1

[4] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking. *arXiv preprint arXiv:2203.14360*, 2022. 1, 2, 6

[5] Weiquan Cao and Yunzhao Li. DOTS: An Online and Near-optimal Trajectory Simplification Algorithm. *Journal of Systems and Software*, 126:34–44, 2017. 2, 3, 5

[6] Minjie Chen, Mantao Xu, and Pasi Franti. A Fast $o(n)$ Multiresolution Polygonal Approximation Algorithm for GPS Trajectory Simplification. *IEEE Transactions on Image Processing*, 21(5):2770–2785, 2012. 2, 3, 4, 5, 6, 7

[7] Anthony Cioppa, Silvio Giancola, Adrien Deliege, Le Kang, Xin Zhou, Zhiyu Cheng, Bernard Ghanem, and Marc Van Droogenbroeck. SoccerNet-Tracking: Multiple Object Tracking Dataset and Benchmark in Soccer Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3491–3502, 2022. 1, 2, 6

[8] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. MOT20: A Benchmark for Multi Object Tracking in Crowded Scenes. *arXiv preprint arXiv:2003.09003*, 2020. 2, 6

[9] David H Douglas and Thomas K Peucker. Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature. *Cartographica: the International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. 2, 3, 4

[10] Yunhao Du, Yang Song, Bo Yang, and Yanyun Zhao. StrongSORT: Make DeepSORT Great Again, 2022. 1

[11] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: Exceeding YOLO Series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 2, 6

[12] Ross Greer, Jason Isa, Nachiket Deo, Akshay Rangesh, and Mohan M. Trivedi. On Salience-Sensitive Sign Classification in Autonomous Vehicle Path Planning: Experimental Explorations With a Novel Dataset. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pages 636–644, January 2022. 1

[13] John Hershberger and Jack Snoeyink. Speeding Up the Douglas-Peucker Line-Simplification Algorithm. Technical report, University of British Columbia, CAN, 1992. 4

[14] Kutalmis Gokalp Ince, Aybora Koksal, Arda Fazla, and A Aydin Alatan. Semi-Automatic Annotation For Visual Object Tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1233–1239, 2021. 2

[15] Atsushi Kawasaki and Akihito Seki. Multimodal Trajectory Predictions for Autonomous Driving Without a Detailed Prior Map. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3723–3732, January 2021. 1

[16] Trung-Nghia Le, Akihiro Sugimoto, Shintaro Ono, and Hiroshi Kawasaki. Toward Interactive Self-Annotation For Video Object Bounding Box: Recurrent Self-Learning And Hierarchical Annotation Based Framework. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 3231–3240, 2020. 2

[17] Yang Liu, Luiz G Hafemann, Michael Jamieson, and Mehrsan Javan. Detecting and Matching Related Objects with One Proposal Multiple Predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4520–4527, 2021. 1

[18] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A Higher Order Metric for Evaluating Multi-Object Tracking. *International Journal of Computer Vision (IJCV)*, pages 1–31, 2020. 3, 6

[19] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting with a Single Convolutional Net. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3569–3577, 2018. 1

[20] Nirvana Meratnia and Rolf de By. Spatiotemporal Compression Techniques for Moving Point Objects. In *International Conference on Extending Database Technology*, pages 765–782. Springer, 2004. 2, 4, 5, 6, 7

[21] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. MOT16: A Benchmark for Multi-Object Tracking. *arXiv preprint arXiv:1603.00831*, 2016. 3, 6

[22] Jerome Revaud and Martin Humenberger. Robust Automatic Monocular Vehicle Speed Estimation for Traffic Surveillance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4551–4561, October 2021. 1

[23] Ryan Sanford, Siavash Gorji, Luiz G Hafemann, Bahareh Pourbabaee, and Mehrsan Javan. Group Activity Detection from Trajectory and Video Data in Soccer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPR)*, pages 898–899, 2020. 1

[24] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. CrowdHuman: A Benchmark for Detecting Human in a Crowd. *arXiv preprint arXiv:1805.00123*, 2018. 6

[25] Specker, Andreas and Moritz, Lennart and Cormier, Mickael and Beyerer, Jürgen. Fast and Lightweight Online Person Search for Large-Scale Surveillance Systems. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) Workshops*, pages 570–580, January 2022. 1

[26] Sultani, Waqas and Chen, Chen and Shah, Mubarak. Real-World Anomaly Detection in Surveillance Videos. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. 1

[27] Peize Sun, Jinkun Cao, Yi Jiang, Zehuan Yuan, Song Bai, Kris Kitani, and Ping Luo. DanceTrack: Multi-Object Tracking in Uniform Appearance and Diverse Motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20993–21002, 2022. 2, 6, 7

[28] Moumita Roy Tora, Jianhui Chen, and James J Little. Classification of puck possession events in ice hockey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 147–154. IEEE, 2017. 1

[29] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently Scaling up Crowdsourced Video Annotation. *International Journal of Computer Vision (IJCV)*, 101(1):184–204, 2013. 2

[30] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 1, 2, 6

[31] Yifu Zhang, 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 (IJCV)*, 129(11):3069–3087, 2021. 1

[32] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-IoU loss: Faster and better learning for bounding box regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 12993–13000, 2020. 3, 7

[33] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. *IEEE Transactions on Cybernetics*, 2021. 7