Study on Small Object Tracking with OBB Annotations: A Comparative Analysis of ByteTrack and StrongSORT

Jiwoo Choi^O, Seonjoo Kim
Yonsei University
jiw55@ewha.ac.kr, seonjookim@yonsei.ac.kr

Abstract

The purpose of the study is to conduct a comparative analysis of ByteTrack[3] and StrongSORT[4] with OBB annotations for Small Object Tracking. Previous studies predominantly relied on Horizontal Bounding Box (HBB) annotations, which often included inaccurate margins around target objects, and resulted in false detection and tracking errors. This study leverages the Oriented Bounding Box (OBB) annotations from the SAT-MTB dataset[1] to accurately capture the rotation and orientation of target objects. We trained the detection model YOLOv8[2] with the OBB annotations and integrated it with ByteTrack and StrongSORT, respectively. This study aims to contribute to improving the performance of Small Object Tracking by analyzing the accuracy of these tracking models.

1. Introduction

Small object detection and tracking models are employed in various fields such as satellite object tracking, drone footage analysis, and microscopic image analysis to identify and analyze extremely small targets. These studies aim to detect and track targets in low-resolution image frames captured from high altitudes, such as those taken by drones or satellites. Unlike general object detection and tracking, small object detection and tracking typically identify targets that occupy approximately 1–5% of the pixel area in an image, making it difficult to extract detailed visual information. A major challenge in Small Object Tracking is the small size of the object, which makes it difficult to extract features and easy to be obscured by the complex background of these datasets.

In object detection and tracking research, Previous tracking studies typically analyze images using multi-scale approaches. They attempt to extract and associate object features across resolutions of different scales. Furthermore, they have predominantly used Horizontal Bounding Box (HBB) annotations. An HBB represents a rectangular bounding box surrounding an object and is defined by the coordinates (xmin, ymin, xmax, ymax). Detection models traditionally rely on HBB to identify and predict objects.

However, HBB has a critical limitation: they are always aligned with horizontal and vertical axes, leading to the inclusion of unnecessary margins. As shown in Figure 1(a), when objects are tilted and densely packed, the HBB is overlapped to enclose the object, leading to false detections, increased computational overhead, and challenges in accurate detection and tracking. It also fails to provide rotational information for rotating objects during tracking. Thus, when an object rotates, it can easily overlap with other objects due to the limitations of HBB.

To address this issue, it is necessary to introduce **Oriented Bounding Box (OBB)** annotations, which incorporate rotational information and minimize unnecessary areas. By switching from Figure 1(a) to Figure 1(b), the detection model can be trained using rotational information, thereby enhancing its ability to process objects effectively. Using OBB annotations and trained detection models, we aim to advance Small Object Tracking and conduct a comparative performance analysis of **ByteTrack[3]** and **StrongSORT[4]** in this context.

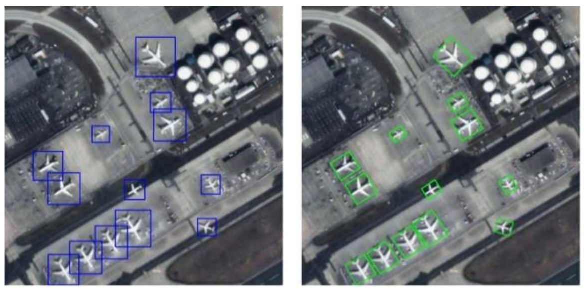

Figure 1. The left shows HBB annotations, while the right shows OBB annotations in the same image.

The reason for choosing the two trackers for Small Object Tracking is that they take different approaches to this task. ByteTrack reuses low-confidence score boxes and performs IOU-based matchings twice, while StrongSort extracts the appearance of objects and utilizes it for association. ByteTrack tends to accurately detect and track objects but is less consistent in maintaining object IDs. In contrast, StrongSort excels at maintaining consistent object IDs but has a higher FP score, indicating instances of incorrect object detection. ByteTrack generally performed better than StrongSort across the five metrics.

2. Related Works

2.1 Object detection

Object detection refers to algorithms that detect and localize objects in images or videos. It has been widely utilized in various fields, such as autonomous driving and medical image analysis.

Detection models are generally classified into two types: two-stage detection models and one-stage detection models. Two-stage detection models involve an object proposal stage, where candidate regions likely to contain objects are first generated. Based on these proposals, the model then performs classification, which extracts the class and bounding box of the object. Representative examples include R-CNN[5], Fast R-CNN[6], and Faster R-CNN[7]. In contrast, one-stage detection models simultaneously perform object proposal and classification in a single step. A representative example is the YOLO series, which stands for "You Only Look Once." Such detectors are the basis of Multi Object Tracking (MOT).

2.2 Tracking by detection

Object tracking refers to algorithms that track targets across sequential frames in a video. Tracking models aim not only to detect targets but also to consistently associate their IDs across frames. It has become essential in fields such as autonomous driving, sports video analysis, and medical imaging.

Detection-based tracking models utilize bounding boxes predicted by detection models and the similarity with tracklets for tracking. By leveraging tracklets, these models can predict the paths of targets and reconstruct their overall trajectory, even when they are occluded or undetected. However, they are highly dependent on the performance of the detection model. The number of missing detections and very low scoring detections cause tracking models' low accuracy.

3. Method

The overall flow of our approach involves connecting a detector with a tracker. We used OBB annotations that incorporated the angles of objects into HBB annotations. OBB annotations and video frames are fed into the detector to accurately identify the target object's location, and the detection results are then linked to a detection-based tracker to track the target object's movement. For the detector, YOLOv8[2] is chosen as it is composed of Feature Pyramid Networks (FPN) and Path Aggregation Networks (PANet), which are expected to effectively detect small targets across multiple scales. We trained YOLOv8 to accurately detect moving objects, including those with rotational information, using OBB annotations. YOLOv8 was used as a detector, and the performance of ByteTrack[3] and StrongSort[4] as trackers was compared and analyzed for object tracking.

3.1 Tracker

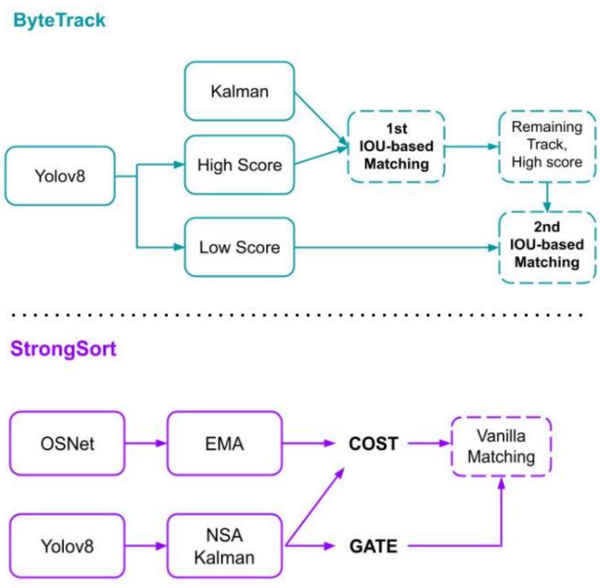

Figure 2. Architecture of Bytetrack and StrongSort.

Figure 2 depicts the architecture of the ByteTrack[3] and Strongsort[4], both of which are training-free models. ByteTrack reuses low-confidence score boxes by classifying the detector's outputs into high-confidence score boxes and low-confidence score boxes, performing two rounds of matching in total. StrongSort uses features of appearance extracted from video frames to match objects for tracking. Details of these trackers are explained below and summarized in Table 1.

To train the two trackers with **OBB annotations**, the data loader and Kalman filter structure are modified to handle angle information. Additionally, since traditional IOU calculations have been based on HBB, which uses axis-aligned intersection areas, we replace it with Polygon Intersection to compute the intersection areas for rotating objects. This study analyzes which of the two trackers performs better for Small Object Tracking. This study analyzes which of the two trackers trained with OBB annotations is more suitable for Small Object Tracking.

Table 1. Comparison of ByteTrack and StrongSort.

	ByteTrack	**StrongSort**
Filter	Kalman Filter	NSA Kalman Filter
Matching	IOU-based	appearance based & IOU-based
mechanism	Reuses High& Low score boxes	Gate, Cost mechanism

ByteTrack. first divides the boxes predicted from YOLOv8 into three categories based on high and low thresholds, discarding the category with the lowest confidence. Next, it utilizes the Kalman filter algorithm[9] to predict the positions of tracklets as a dynamic model. It required 4 dimensional measurement vectors (x, y, a, h), where (x,y) is the center position, a is the aspect ratio, and h is the height of the bounding box. However, we modified the filter to add angle information θ, extending it to 5-dimensional measurement vectors (x, y, a, h, θ) for OBB annotations. The Kalman filter algorithm consists of a state prediction stage and a state update stage. The state prediction stage is as:

$$x'_{t+1} = F_{t+1}x_t \tag{1}$$
$$P'_{t+1} = F_{t+1}P^{T}{}_{t} + Q_{t+1} \tag{2}$$

where x_t and P_t are the mean and covariance of the state at time step t, x'_{t+1} and P'_{t+1} are the estimated state at time step t+1. F_{t+1} and Q_{t+1} are the transition matrix and a covariance of process noise. In the state update stage, Kalman gain K is used to weight the estimated state and the observation at same time, updating the final state.

$$K = P_{t+1}H^{T}{}_{t+1}(H_{t+1}P_{t+1}H^{T}{}_{t+1} + R_{t+1})^{-1}, \tag{3}$$
$$x_{t+1} = x'_t + K(z_{t+1} - H_{t+1}x'_{t+1}), \tag{4}$$
$$P_{t+1} = (I - KH_{t+1})P'_{t+1} \tag{5}$$

where H_{t+1} is the observation model, z_{t+1} is the measurement. The Kalman filter algorithm updates the mean and covariance by correcting predictions based on the observed location of the object.

Afterwards, the first IOU-based matching associates high-confidence score boxes with tracklets predicted by the Kalman filter algorithm. In the second IOU-based matching, Unmatched high- and low-confidence score boxes are matched again with the remaining tracklets.

StrongSort. is an advanced model that improves the architecture of the DeepSort[8]. StrongSort performs vanilla matching by combining the cost computed using appearance features and the gate determined by the NSA Kalman filter algorithm[10]. StrongSort extracts appearance features of targets, which we used the pretrained OSNet, and then an exponential moving average (EMA) updates the appearance state for the tracklet to improve the matching.

The NSA Kalman filter algorithm outperforms the vanilla Kalman filter in handling low-quality detections and effectively incorporates information on the scales of detection noise. We modified it to also provide rotational information (x, y, a, h, θ), just like the Kalman filter does too. It proposes a formula to calculate the noise covariance:

$$\tilde{R}_{t+1} = (1 - c_{t+1})R_{t+1} \tag{6}$$

where c_{t+1} is the detection confidence score. According to formula 3, R_{t+1} can be replaced with $\tilde{R}_{t+1}$ It means to adaptively provide weight, which can improve the accuracy in the state update stage. The motion information as Motion cost A_m is calculated by the Mahalanobis distance of the NSA Kalman filter algorithm.

Appearance cost A_a is computed based on the cosine similarity between the appearance state and the predicted tracklet. It is then combined with motion cost A_m to form the cost matrix C, which is also used as the gate. The cost matrix C is defined with a weighted sum of appearance cost A_a and motion cost A_m:

$$C = \lambda A_a + (1 - \lambda)A_m \tag{7}$$

where λ is the weight factor. Vanilla Matching matches cost and gate, eliminating the matching cascade algorithm in DeepSort.

In summary, due to the differing approaches of the two trackers, this study aims to examine how each method impacts Small Object Tracking based on YOLOv8.

4. Experiments

4.1 Dataset

We preprocess OBB annotations of the SAT-MTB dataset[1] for input format. The SAT-MTB dataset consists of 249 satellite videos and includes the classes Airplane, Ship, Car, and Train. For the experiments in this study, only classes with OBB annotations are selected, resulting in the use of 108 satellite videos. Consequently, three classes—Airplane, Ship, and Train—are used, excluding the Car class. The satellite videos are recorded at 10 fps and cover an area of 11 km x 4.6 km. Additionally, each video varies in length, object shape, and the number of objects.

Out of the 108 videos, 58 videos are designated as the test dataset, while 50 videos are used as the train dataset. The train dataset is further split into train and validation sets for each class in a 6:4 ratio. YOLOv8[2] is trained for 100 epochs with a batch size of 12 using a single A5000 GPU. Trackers perform object tracking based on the detector. ByteTrack[3] and StrongSort[4], as a training-free model, do not require additional training. For ByteTrack, the thresholds for classifying high-confidence score boxes and low-confidence score boxes are set to 0.5 and 0.1, respectively. For StrongSort, a pretrained OSNet is used as the feature extractor for appearance features.

4.2 Evaluation metric

YOLOv8[2] is evaluated for small object detection performance using the metrics Precision, Recall, mAP@0.5, and mAP@0.5:0.95. Trackers' performance is analyzed using the trained detector and the test dataset. Consistent with the SAT-MTB dataset paper[1], we set the IOU threshold to 0.3, considering the objects' size is too small. ByteTrack[3] and StrongSORT[4], based on YOLOv8, is analyzed using five performance metrics. The five metrics are defined as MOTA, IDF1, MT, FP, and FN. MOTA considers False Positives (FP), False Negatives (FN), and ID swaps (IDs) and is defined by the formula as:

$$MOTA = 1 - \frac{FP + FN + IDs}{GT} \quad (8)$$

MOTA represents the overall accuracy of successful tracking, with higher MOTA values indicating better tracking performance by the tracker. IDF1, or ID F1-Score, is calculated as the harmonic mean of Precision and Recall, reflecting ID consistency. A higher IDF1 value indicates that the tracker is effectively maintaining consistent IDs for the same objects. MT stands for Mostly Tracked, representing the proportion of objects that were well-tracked throughout the video, providing insight into how consistently the objects are tracked. FP and FN represent false positives and false negatives, respectively, where FP refers to incorrect object detections, and FN indicates objects that were not tracked. Lower FP and FN values indicate better tracking performance by the tracker.

4.3 Results

Table 2. Evaluation of Small Object detection on SAT-MTB dataset using YOLOv8

	Precision	**Recall**	**mAP @0.5**	**mAP @0.5:0.95**
ALL	0.666	0.654	0.613	0.436
Airplane	0.980	0.965	0.978	0.898
Ship	0.890	0.890	0.834	0.401
Train	0.107	0.0107	0.0258	0.00809

Table 3. Evaluation of Small Object Tracking on SAT-MTB dataset based on ByteTrack

	MOTA↑	**IDF1↑**	**MT↑**	**FP↓**	**FN↓**
ALL	86.6%	15.4%	30	3723	5450
Airplane	92.5%	17.8%	31	2197	1320
Ship	51.5%	5.8%	2	1537	3173
Train	22.5%	3.9%	2	0	968

In Table 2, the Airplane class outperforms other classes across all metrics due to its larger pixel size and richer visual features. In contrast, the performance for the Ship class is lower in mAP@0.5:0.95, while the Train class shows extremely poor performance, with scores below 0.01 in all metrics.

As mentioned in the SAT-MTB paper [1], the Ship class is challenging for the model to identify due to its small object size and the waves caused by object movement, which can easily confuse the model with the background. For the Train class, the extreme aspect ratios of the objects and the difficulty in providing rotational information make it challenging for the model to capture sufficient locality information. Furthermore, the Train class accounts for less than 10% of the entire dataset, resulting in uneven distribution and poor performance.

ByteTrack[3] is analyzes the performance of ByteTrack based on YOLOv8[2] in Table 3, the ALL class shows robust tracking with a MOTA of 86.6%. Since the ALL class includes more videos than individual classes, the FP and FN values are higher compared to other classes. The model shows relatively lower IDF1 performance overall, and qualitative results reveal that few videos maintain consistent objects' IDs. The Airplane class exhibits relatively good tracking performance, while the Ship and Train classes perform poorly in tracking. This is attributed to the fact that ByteTrack relies on the YOLOv8 detection model, and the Ship and Train classes, which had poor detection performance, also resulted in poor tracking performance.

Table 4. Evaluation of Small Object Tracking on SAT-MTB dataset based on StrongSort

	MOTA↑	**IDF1↑**	**MT↑**	**FP↓**	**FN↓**
ALL	60.4%	17.1%	32	24329	1637
Airplane	67.3%	21.4%	32	17201	182
Ship	32.1%	14.3%	6	6411	1216
Train	25.6%	40.4%	2	725	247

StrongSort[4] generally shows lower tracking performance compared to ByteTrack, but an interesting observation is that its IDF1 score is higher than ByteTrack. For the Train class, which represents the smallest dataset in the SAT-MTB dataset, StrongSort achieves a notable 40.4% in consistently tracking object IDs. A high FP score combined with a relatively low FN score indicates that objects are being misidentified yet still tracked.

Figure 3 illustrates the object ID consistency results for the two trackers, depicting the tracking of a ship's movement from frame 0 to frame 126. ByteTrack fails to assign an object ID at frame 126, indicating its inability to continue tracking the ship. In contrast, StrongSort successfully assigns the same ID to the same object at frame 126, demonstrating more consistent tracking performance. Regarding FP scores, Figure 4 shows that ByteTrack tracks only the correct airplane, while StrongSort erroneously tracks regions that are not airplanes. Additionally, StrongSort generates multiple overlapping boxes for a single object, leading to a higher FP score.

These observations demonstrate that ByteTrack and StrongSort operate differently and exhibit varying performance when applied to the same dataset.

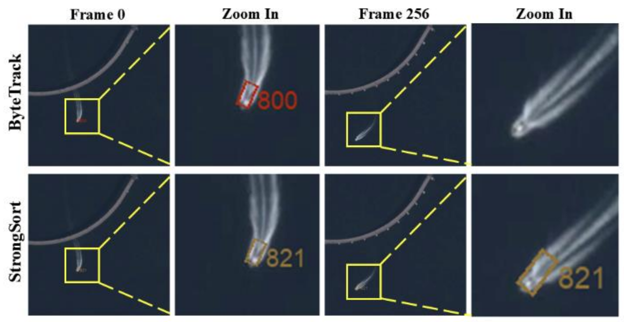

Figure 3. A qualitative comparison result of object ID consistency. StrongSort shows better ID consistency compared to ByteTrack.

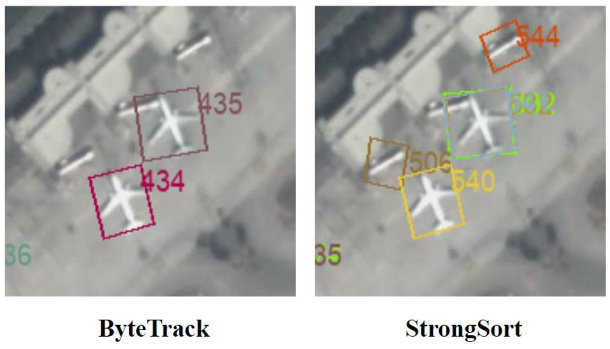

Figure 4. A qualitative comparison result of FP score. In contrast to StrongSort, ByteTrack tracks only the correct airplane.

5. Conclusion and Future Work

This study conducted experiments on Small Object Tracking in satellite videos and performed a comparative analysis of the performance of ByteTrack[3] and StrongSort[4] using OBB annotations. ByteTrack and StrongSort, both based on the YOLOv8 detector, exhibit interesting observations that ByteTrack shows lower FP scores than StrongSort, accurately identifying objects but with less consistent IDs. In contrast, StrongSort achieves higher IDF1 scores, indicating better ID consistency, but tends to misidentify objects. Consequently, ByteTrack can be regarded as the better model for detecting and tracking objects correctly in Small Object Tracking tasks.

As highlighted in the SAT-MTB dataset paper[1], limitations of the dataset and OBB annotations persist. Future research can build on these findings to improve dataset distribution and tracking performance. This study contributes to the development of precise small object detection and tracking systems, which can advance satellite and aerospace industries and research.

Acknowledgement

I would like to thank my mentors: Jinwoo Kim, Jinho Jeong, and Sangmin Han, for their helpful feedback and advice.

Reference

[1] Shengyang, L., et al., "A Multitask Benchmark Dataset for Satellite Video: Object Detection, Tracking, and Segmentation," *IEEE Transactions on Geoscience and Remote Sensing.*, vol. 61, pp. 1–21, 2023.

[2] Varghese, et al., "YOLOv8: A Novel Object Detection Algorithm with Enhanced Performance and Robustness," *IEEE International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)*, 2024.

[3] Zhang, Y., et al., "ByteTrack: Multi-object Tracking by Associating Every Detection Box," in *Proc. European conference on computer vision.*, Cham: Springer Nature Switzerland, 2022.

[4] Du, Y., et al., "StrongSORT: Make DeepSORT Great Again," *IEEE Transactions on Multimedia*, vol. 25, pp. 8725–8737, 2023.

[5] Girshick, R., et al., "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition.*, 2014, pp. 580–587.

[6] Girshick, R. "Fast r-cnn." *arXiv preprint arXiv:1504.08083* (2015).

[7] Ren, S., et al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE transactions on pattern analysis and machine intelligence.*, vol. 39, no. 6, pp. 1137–1149, 2016.

[8] Wojke, N., et al., "Simple Online and Realtime Tracking with a Deep Association Metric," in *Proc. IEEE International conference on image processing. (ICIP)*, 2017, pp. 3645–3649.

[9] Kalman, R. E., "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, vol. 82, pp. 35–45, 1960.

[10] Du, Y., et al., "GIAOTracker: A Comprehensive Framework for MCMOT with Global Information and Optimizing Strategies in VisDrone 2021," in *Proceedings of the IEEE/CVF International conference on computer vision.*, 2021, pp. 1230–1240.