# Multi-Class FairMOT with HOOT Dataset

Abdul Bhutta
*Electrical and Computer Engineering*
*Toronto Metropolitan University (TMU)*
Toronto, Canada
abdul.bhutta@torontomu.ca

***Abstract*—Object tracking is a complex task that has surged in popularity recently due to advancements in deep learning and increased computational power of graphical processing units (GPUs) in both the cloud and local environments. Many use cases of such trained models are becoming increasingly popular daily in real-world scenarios such as autonomous driving, surveillance, and medical areas. In recent years, Multi-Object Tracking has become the more popular area of research. FairMOT is one of the top performing algorithms that can track numerous unique objects of the same class. The algorithm is trained using the Heavy Occlusion in Object Tracking Benchmark (HOOT) dataset, which includes a wide range of occlusion scenarios, where 68% of the frames have some occlusion. The model performance and accuracy after training were tested using the test set. The overall test data results show a Multi-Object Tracking Accuracy (MOTA) score of 24.2% and a Multi-Object Tracking Precision (MOTP) of 0.276. Although the accuracy of tracking the object of some objects was good, it needed to be more accurate as it could not maintain the unique ID of the objects throughout the video and kept updating the identifier to a new value each frame, as seen by the total identity switches of 242.**

## I. Introduction

Multi-Object Tracking has been a challenging task in computer vision and one of the most active research areas in the field. The area is derived from Single Object Tracking (SOT), where the main task is to select the object of interest and track it throughout the video. Some popular SOT algorithms include SiamRPN, SiamMask, and NanoTrack [1]. Multi-Object Tracking (MOT) is an extension of SOT, where the task is to track multiple objects of the same class in each frame. A few popular MOT algorithms include DeepSORT, FairMOT, and ByteTrack. Multi-Object Tracking has made Single-Object Tracking somewhat obsolete, as most real-world scenarios require tracking multiple objects in the video. At the same time, new algorithms have been designed to be modified to track single objects if needed. Although it has made progress in recent years and shows promising results, it still has much room for improvement. Tracking the identity of an object is complex and challenging as many factors come into play, such as occlusion, scale variation, and appearance changes. The main challenge is occlusion, where other objects partially or fully occlude the tracked object in the video, which resembles real-world scenarios. Occlusion can cause the tracker to lose the object being tracked and re-identify it as a new object, which can cause many issues and errors in real-world scenarios. FairMOT is a state-of-the-art MOT tracker that tracks a single object class with multiple instances of the object. Although, with a few changes, the algorithm can be modified to track multiple object classes by changing the head of the model to match the number of classes in the dataset [3]. In this project, the FairMOT model is trained to track multiple object classes (20 Classes) using the High Occlusion Object Tracking (HOOT) dataset and evaluated using the MOT metrics.

## II. HOOT Dataset

Fig. 1. Various objects in HOOT dataset

One of the most recent benchmarks released for Single Object Tracking is the Heavy Occlusion for Object Tracking (HOOT) dataset, introduced in 2023 by G. Sahin et al. at the University of Southern California. The dataset includes 581 HD videos with an average of 22 seconds per video and approximately 436K frames. The dataset is specifically designed using manually annotated bounding boxes for each frame and has approximately 68% of the frames with some occlusion [2]. It consists of 74 object classes and is generalized for everyday objects such as apples, backpacks, bottles, etc, as shown in Figure 1. It contains various high occlusion scenarios where the object being tracked is partially or fully occluded by another object or not visible in the video compared to other datasets such as MOT, which contains less than 10% of occlusion scenarios and modified to almost perfect tracking scenarios, which is why most models can achieve high accuracy in tracking. It does not consider real-world scenarios where other objects in real-time can obstruct the object being tracked. The dataset includes four different occluders in the benchmark taxonomy, shown in Table 1.

TABLE I
OCCLUDERS IN HOOT DATASET

| Occluder | Description |
|---|---|
| Solid | Completely blocks view of the object |
| Sparse | Formed of sparsely distributed solid but some view of the object is visible |
| Semi-Transparent | Fully covers the object but some view of the object is visible |
| Transparent | Fully covers the object but the object is visible through the occluder |

Furthermore, it also includes occlusion attributes per frame, as shown in Table 2. These attributes describe the object's occlusion level. They can be beneficial in training the model to learn from various occlusion scenarios and improve its tracking performance.

TABLE II
OCCLUSION ATTRIBUTES IN HOOT DATASET

| Occlusion Attribute | Description |
|---|---|
| Absent | No occlusion in the frame |
| Full Occlusion | The object is fully occluded by another object |
| Cut-By-Frame | The object is partially cut by the edge of the frame |
| Partial Occlusion | The object is partially occluded by another object |
| Occluded-By-Similar-Object | The object is occluded by another object of the same class |
| Occluded-By-Multiple-Occluder-Types | The object is occluded by multiple occluder types |

## III. FAIRMOT ALGORITHM

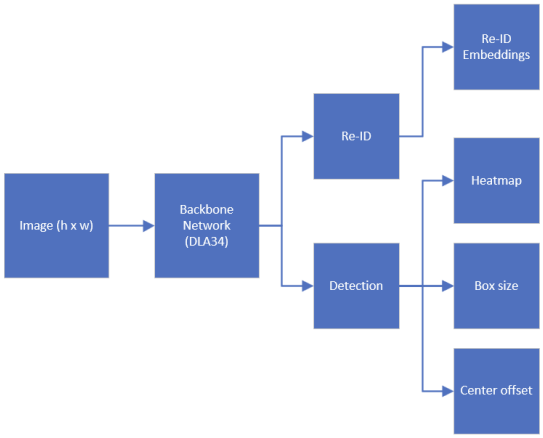


Fig. 2. FairMOT Architecture

The FairMOT algorithm, designed for Multi-Object Tracking, is structured around two main branches: Detection and Re-Identification, as depicted in Figure 2. The backbone used in the algorithm is DLA-34, an encoder-decoder neural network used for extracting features in each image. It provides the extracted features to the two branches, and the output of the network is a feature map in the format of c x h x w, where c is the total number of channels, h and w are the height and width of the feature map. The network downsamples the input image by a factor of 4 and impacts the feature map to be reduced by 1/4.

### A. Detection Branch

The detection branch uses the CenterNet algorithm to detect objects in each frame. It contains three parallel heads that are appended to the feature maps produced by the DLA-34 network: heatmap, box size, and center offset. The final target output is obtained by a 3x3 convolutional layer with 256 channels and a 1x1 convolution layer in the last layer.

*1) Heatmap Head:* The heatmap head estimates the object's center in each frame, and the dimension of the tensor output is $1xhxw$, where h and w are the height and width of the feature map. The expected output of a center point in the heatmap will be close to one but decays exponentially as the distance from the object's center increases, leading to a value closer to 0.

*2) Box Size Head:* The box size head estimates the width and height of the bounding box for each detected object in the frame. The dimensions of the head tensor output are $2xhxw$, where h and w are the height and width of the feature map. The width and height of the ground truth bounding box are calculated by taking the top left coordinates and subtracting them from the bottom right coordinates.

*3) Center Offset Head:* The center offset head estimates the object's center more precisely and overcomes the problem of downsampling, which impacts the object's ground truth and predicted center. When an image is passed through the network, the dimension of the feature map is $cxh/4xw/4$. For instance, in a 128 by 128 image with an object at the center of a bounding box at location (55, 55), the output of a feature map is 32 by 32. The object's center within the feature map is at location (55/4) = 13.75, where the tensor only accepts integer values and the floor of the result is taken, which is 13. This indicates the object's center is at location (13, 13), which is inaccurate. The center offset is computed by calculating the difference between the feature map center and the floor value, which results in a value of 0.75. This difference and the tensor values are used to predict the object's center more precisely and accurately while overcoming the issues with downsampling.

### B. Re-Identification Branch

The Re-ID branch generates essential or unique features that can help distinguish between different objects of the same class, even when they leave or enter the frame. The branch extracts the feature embeddings from each detected object, which can be achieved by applying a 128-channel 3x3 convolution layer to the feature map produced by DLA-34 at each location. The reidentification feature vector of an (x, y) location is extracted from the feature map with 128 channels at the corresponding locations.

### C. Loss Function

The FairMOT algorithm uses multiple loss functions, each with a specific task, to help train the model. Each head has its own loss function, while box size and center offset use the same loss function. The $total\_loss$ is used during training and computed by summing all the loss functions and utilizing the uncertainty loss function. This allows the model

to automatically adjust the weights for both tasks, which are trained simultaneously. Table 3 summarizes the loss functions used during training, and additional information can be found in FairMOT's official paper [4].

TABLE III
LOSS FUNCTION FOR EACH BRANCH

| Branch | Loss Function |
|---|---|
| Heatmap | $L_{\text{heat}} = -\frac{1}{N}\sum_{xy}\begin{cases}(1-\hat{M}_{xy})^{\alpha}\log(\hat{M}_{xy}), & M_{xy}=1;\\ (1-M_{xy})^{\beta}(\hat{M}_{xy})^{\alpha}\log(1-\hat{M}_{xy})\end{cases}$ |
| Box Size | $L_{\text{box}} = \sum_{i=1}^{N}\left\|o^i-\hat{o}^i\right\|_1 + \lambda_s\left\|s^i-\hat{s}^i\right\|_1$ |
| Center Offset | $L_{\text{box}} = \sum_{i=1}^{N}\left\|o^i-\hat{o}^i\right\|_1 + \lambda_s\left\|s^i-\hat{s}^i\right\|_1$ |
| Re-ID | $L_{\text{identity}} = -\sum_{i=1}^{N}\sum_{k=1}^{K} L^i(k)\log\left(p(k)\right)$ |
| Detection | $L_{\text{detection}} = L_{\text{heat}} + L_{\text{box}}$ |
| Total Loss | $L_{\text{total}} = \frac{1}{2}\left(\frac{1}{e^{w_1}}L_{\text{detection}} + \frac{1}{e^{w_2}}L_{\text{identity}} + w_1 + w_2\right)$ |

## IV. PREPROCESSING OF HOOT DATASET

The model training process requires the annotations for each frame in the video to be in a specific format, or the model will perform poorly. The HOOT dataset underwent various preprocessing steps using a custom script. The script, *gen_labels_hoot.py*, allows the user to control its functionality by passing in an argument (1 or 2). When 1 is used, it generates labels and annotations for the bounding box for each frame and writes the data to a text file in the FairMOT format, as shown below,

$$class\_id, track\_id, normalized\_x, normalized\_y, normalized\_width, normalized\_height$$

Where the $class\_id$ is the class of the object being tracked (e.g., apple, backpack, etc.), $track\_id$ is the unique identifier for the object being tracked. For the current dataset, all the track IDs are set to 1 for each frame. The $normalized\_x$, $normalized\_y$, $normalized\_width$, and $normalized\_height$ are the normalized coordinates at the object's center and dimensions of the object.

If two is passed as the argument, the script creates a text file containing the path for each image required for training. The HOOT dataset provides the annotations in the anno.json file, which includes the information for each frame and the bounding box coordinates given in two axes: $aa\_bb$ and $rot\_bb$. The $aa\_bb$ is used for axis-aligned bounding boxes, and the $rot\_bb$ is used for rotated bounding boxes where the bounding box is rotated at an angle. The axis-aligned coordinates are used for training and testing as the FairMOT algorithm only accepts those coordinates. The aa_bb is a list of tuples where the tuple contains the coordinates of the bounding box. The first tuple contains the coordinates for the top left corner of the bounding box, the second tuple includes the coordinates for the top right corner, and so on, given in the following format:

$$[(x1, y1), (x2, y2), (x3, y3), (x4, y4)]$$

Given all the coordinates for each corner of the bounding box, the width and height can be computed using the minimum and maximum values of the x and y coordinates. This method provides the best accurate representation of the size of each bounding box and can be calculated using the following equation,

$$width = max_x - min_x, \quad height = max_y - min_y \tag{1}$$

Furthermore, the center of the bounding box can be calculated using equation 2,

$$center\_x = \frac{min_x + max_x}{2}, \quad center\_y = \frac{min_y + max_y}{2} \tag{2}$$

Figure 3 provides an example of a bounding box with two min-circles drawn. The blue circle represents the top left corner of the bounding box, and the red circle represents the center.

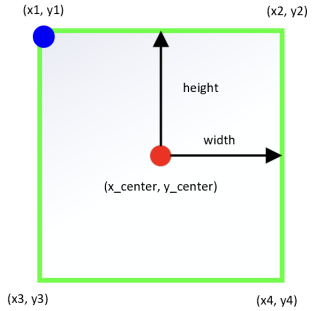


Fig. 3. Bounding Box Example

The bounding box center and dimensions were drawn on each frame to visualize the annotations using the CV2 library to ensure and verify the calculated values were accurate, as shown in Figure 4. The process took approximately 50 minutes to complete for 20 classes using the HOOT dataset.

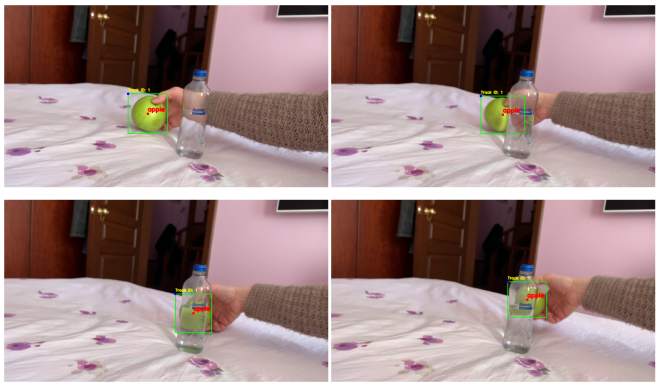

Fig. 4. Annotation Visualization for Apple

The width and height of each video are extracted from the meta.info file, which is also in JSON format. These values are required to calculate the normalized coordinates and allow

the algorithm to adapt to various image sizes for diverse datasets. The normalized coordinates for the bounding box center, width, and height are calculated by dividing by the width and height of the image (1280 by 720). Furthermore, if the current frame contains no bounding boxes or the width or height dimensions are less than 15 pixels, it is skipped, and the next frame is processed. The annotations in FairMOT format are written to a text file for each frame and saved in the $labels\_with\_ids$ directory for every object. Once the labelling for each frame is complete, another text file ($hoot.train$) is created to provide the path to each text file and passed to the algorithm so the annotations for each frame and location of the image are known. An example of a text file containing a path to an image is,

$$hoot/images/train/apple/002/000000.png$$

## V. Training & Testing Dataset

The current training environment used a single GPU (Nvidia GeForce RTX 2080Ti) with 11 GB of memory, a batch size of 4, and a learning rate of 7e-5. It took approximately three days to complete the training for 30 epochs, and every ten epochs was saved as a fail-safe mechanism. This allowed the training to continue from the last checkpoint in case of a crash or if the model did not converge. Additionally, multi-scale training was implemented to train on 23 different image sizes. The image sizes were randomly scaled with a factor of 32 between 640 by 320 and 1216 by 672. The model also incorporated transfer learning through pre-trained weights from the CenterNet model zoo (ctdet_coco_dla_2x.pth), which was trained on the COCO dataset for 230 epochs. It utilizes keypoint estimation to locate the object's center and calculate the bounding box around the detected object [5]. This differs from the traditional method of anchor boxes and allows the model to extract features with a higher accuracy rather than starting the training from scratch, which may lead to significantly increased training time. The model was configured to include multiple classes from its default single-class configuration. The head was modified and changed to $'hm' : 20,' wh' : 2,' id' : 128,' reg' : 2$, where the hm is the heatmap for the object detected, wh is the width and height of the bounding box, id is the unique identifier for the object, and reg is the regression offset for the bounding box. The heatmap was set to 20 as the number of classes used for training in the HOOT dataset. The unique identifier and regression offset were set to their default values of 128 and 2, respectively. The testing dataset consisted of 12 classes with 16 videos, and each object class used for testing is shown in Table 4. The test data required preprocessing before assessing all the videos. Each video was processed using the $gen\_labels\_hoot\_test.py$ script to generate annotations for the ground truth in MOT format. The script receives information from the $anno.json$ and $meta.info$ files and converts it into MOT format. It saves each video annotation in a single text file labelled $gt.txt$. The newly created text files are moved to the test directory for each object class. This allows the model's performance to be evaluated using the mot-metrics library from evaluation metrics such as MOTA, MOTP, IDF1, IDP, IDR, and many more. The annotation format must be in the following format,

$$frame\_num, object\_id, bb\_left, bb\_top, bb\_width, bb\_height, confidence, x, y, z$$

Where $frame\_num$ is the current frame number, $object\_id$ is the unique identifier for the object, $bb\_left$ and $bb\_top$ are the top left coordinates of the bounding box, $bb\_width$ and $bb\_height$ are the width and height of the bounding box, $confidence$ is the confidence score of the object detected, and $x$, $y$, $z$ are the 3D coordinates of the object.

TABLE IV
Object Classes for Training and Testing Set

| Object | Class ID | Training Videos | Testing Videos |
|---|---|---|---|
| apple | 0 | 8 | 1 |
| avocado | 1 | 14 | 2 |
| backpack | 2 | 5 | 0 |
| banana | 3 | 4 | 0 |
| bear | 4 | 1 | 0 |
| bicycle | 5 | 1 | 0 |
| bird | 6 | 4 | 0 |
| book | 7 | 22 | 1 |
| bottle | 8 | 7 | 1 |
| bowl | 9 | 6 | 1 |
| camel | 10 | 3 | 1 |
| carrot | 11 | 7 | 2 |
| cat | 12 | 3 | 2 |
| clock | 13 | 16 | 1 |
| coaster | 14 | 3 | 1 |
| coat | 15 | 3 | 2 |
| coin | 16 | 4 | 0 |
| crocodile | 17 | 4 | 0 |
| cup | 18 | 17 | 0 |
| deer | 19 | 5 | 1 |
| **Total** | **20** | **130** | **16** |

## VI. Results

### A. Training Dataset Results

Table 5 summarizes the results for the training dataset after 30 epochs. The optimal weights were achieved in the last epoch (30) with a loss of -20.6021. The $total\_loss$ shows a negative value due to the uncertainty loss function, which the FairMOT author confirms can be expected to have a negative value. In contrast, the other loss functions are approaching a value of zero.

TABLE V
Training Results

| Epoch | Loss | HM Loss | WH Loss | Off Loss | ID Loss | Img Size |
|---|---|---|---|---|---|---|
| 1 | 9.9133 | 4.3180 | 7.7293 | 0.2542 | 0.0000 | 1152x640 |
| 5 | -8.1380 | 0.6602 | 2.7863 | 0.2342 | 0.0000 | 1088x608 |
| 10 | -17.4712 | 0.3505 | 2.1852 | 0.2257 | 0.0000 | 992x544 |
| 15 | -19.4380 | 0.1718 | 1.7014 | 0.2139 | 0.0000 | 1088x608 |
| 20 | -20.4724 | 0.1418 | 1.6053 | 0.2102 | 0.0000 | 1152x640 |
| 25 | -20.5932 | 0.1273 | 1.5553 | 0.2079 | 0.0000 | 832x448 |
| 30 | -20.6021 | 0.1211 | 1.5409 | 0.2068 | 0.0000 | 704x384 |

It also summarizes the training results after each 5th epoch with the corresponding loss values for the four other loss functions (HM, WH, Off, ID). The $ID\_Loss$ are all zero as each image contains only one object of the same class, while the last column displays the image size used during training for each epoch. In Figure 5, the loss curve for each loss function throughout the epochs is shown. The graph indicates that the loss decreases as the number of epochs increases, which suggests the model is learning during the training phase.

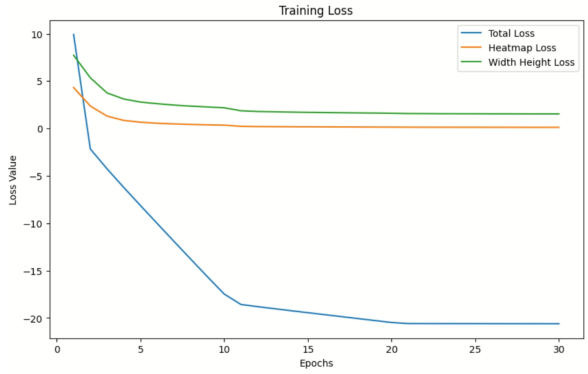


Fig. 5. Loss Curve for Training Dataset

### B. Test Dataset Results

Table 6 summarizes the test dataset results of 12 classes and 16 video sequences. It displays each object class's MOTA and MOTP scores and the overall result. The overall results show a MOTA score of 24.2% and an MOTP score of 0.276. It also includes the number of false positives, false negatives, and identity switches for each object class. The test set contains a video sequence for avocado (010), in which the model performed exceptionally well with a MOTA score of 81.1% and a few frames with the tracking are displayed in Figure 6. The avocado is seen moving across the frames but with minimal occlusion where the basket occluding the avocado is not entirely blocking the view.

TABLE VI
TEST RESULTS

| Object | Sequence | MOTA | MOTP | IDF1 | IDP | IDR | Rcll | FP | FN | IDs |
|---|---|---|---|---|---|---|---|---|---|---|
| apple | 001 | 32.9% | 0.267 | 22.7% | 35.5% | 17.2% | 44.7% | 33 | 264 | 23 |
| avocado | 003 | 20.0% | 0.312 | 17.9% | 26.9% | 13.4% | 38.6% | 45 | 248 | 30 |
| avocado | 010 | 81.1% | 0.230 | 51.0% | 50.4% | 55.5% | 99.7% | 206 | 4 | 6 |
| book | 001 | 17.1% | 0.284 | 9.4% | 22.3% | 5.9% | 23.6% | 36 | 914 | 42 |
| bottle | 003 | -0.3% | 0.290 | 3.6% | 19.7% | 2.0% | 5.6% | 29 | 625 | 10 |
| bowl | 003 | 26.3% | 0.324 | 15.5% | 32.1% | 10.2% | 30.9% | 4 | 284 | 15 |
| camel | 002 | 73.2% | 0.289 | 86.4% | 87.7% | 85.5% | 85.5% | 39 | 46 | 0 |
| carrot | 001 | 30.1% | 0.347 | 9.1% | 11.9% | 7.4% | 48.4% | 176 | 654 | 55 |
| carrot | 008 | -16.8% | 0.315 | 8.8% | 11.0% | 7.4% | 26.1% | 314 | 558 | 10 |
| cat | 001 | 68.4% | 0.237 | 45.7% | 52.2% | 40.7% | 73.3% | 18 | 103 | 1 |
| cat | 003 | 0.2% | 0.391 | 0.3% | 100.0% | 0.2% | 0.2% | 0 | 1219 | 0 |
| clock | 007 | 36.1% | 0.135 | 27.6% | 47.1% | 19.5% | 39.1% | 14 | 349 | 3 |
| coaster | 003 | 3.9% | 0.263 | 7.5% | 100.0% | 3.9% | 3.9% | 0 | 74 | 0 |
| coat | 003 | -3.1% | 0.286 | 5.9% | 16.1% | 3.6% | 10.8% | 103 | 797 | 21 |
| coat | 005 | 15.6% | 0.324 | 22.4% | 46.0% | 14.8% | 25.5% | 34 | 372 | 15 |
| deer | 002 | 23.0% | 0.382 | 19.7% | 24.9% | 17.2% | 50.0% | 99 | 204 | 11 |
| **Overall** | | **24.2%** | **0.276** | **22.4%** | **35.4%** | **16.5%** | **37.2%** | **1150** | **6715** | **242** |

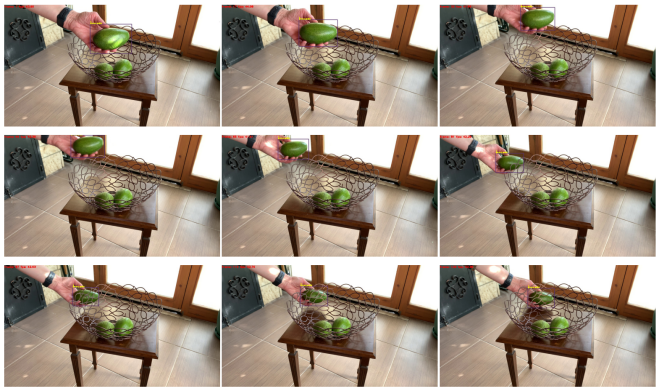

Fig. 6. Avocado Tracking

On the other hand, the bottle (003) object class had a MOTA score of -0.3%, where the model could not track nor detect the correct class. In Figure 7, the bottle is seen moving across the frames with similar bottles, and the model misclassifies the moving bottle numerous times while classifying it as a book or coat. The results show that the model had difficulty tracking objects occluded by other substances. This led to 242 identity switches for the test set, which contained only 16 videos with one object to track.

Fig. 7. Bottle Tracking

## VII. CONCLUSION

The HOOT dataset is a unique benchmark for evaluating object-tracking algorithms, particularly occlusion. The findings demonstrate that the FairMOT algorithm faces many challenges in maintaining the unique identifier for tracked objects in the presence of occlusion, resulting in a decrease in the MOTA and MOTP scores. This underscores the need for further research and time to determine the best approach to handle occlusion in Object Tracking. However, we can remain optimistic about the future as we witness the continuous advancements in object tracking algorithms. The solution to occlusion is within reach, and when achieved, it will mark a significant milestone in Computer Vision and Object Tracking.

## REFERENCES


[1] Honglin Chu, "Honglinchu/siamtrackers: (2020-2022) the PyTorch version of SIAMFC, SiamRPN, DaSiamRPN, UpdateNet, siamdw, SIAMRPN++, siammask, SIAMFC++, siamcar, siamban, ocean, lighttrack, TRTR, NanoTrack; visual object tracking based on Deep Learning," GitHub, https://github.com/HonglinChu/SiamTrackers.

[2] G. Sahin and L. Itti, 'HOOT: Heavy Occlusions in Object Tracking Benchmark', in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2023, pp. 4830–4839.

[3] Even, "Captaineven/MCMOT: Real Time One-stage multi-class & multi-object tracking based on anchor-free detection and Reid," GitHub, https://github.com/CaptainEven/MCMOT (accessed Aug. 17, 2024).

[4] Y. Zhang, C. Wang, X. Wang, W. Zeng, and W. Liu, 'Fairmot: On the fairness of detection and re-identification in multiple object tracking', International Journal of Computer Vision, vol. 129, pp. 3069–3087, 2021.

[5] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as Points," arXiv preprint arXiv:1904.07850, 2019.