FORT: Fisheye Online Realtime Tracking with an Improved Kalman Filter

Nathan Odic, Benoit Faure, Baptiste Magnier

FORT: Fisheye Online Realtime Tracking with an Improved Kalman Filter

Nathan Odic, Benoit Faure, and Baptiste Magnier
EuroMov Digital Health in Motion, Univ Montpellier, IMT Mines Ales, Ales, France
nathan.odic@mines-ales.org, benoit.faure@mines-ales.org, baptiste.magnier@mines-ales.fr

***Abstract*—The goal of human tracking is to detect people in a scene and assign them a unique identifier that the tracker will follow across multiple frames. Our tracker, FORT, implements deep learning solutions such as, YOLOv7 for detection, ResNeXt-50 for feature extraction and re-identification and an adapted Kalman filter for tracking. The goal is to present a real time tracking solution for the complex environment of top-view, fisheye images. The proposed solution is then compared with the BoT-SORT and StrongSORT trackers on a custom fisheye Multiple Object Tracking (MOT) Challenge dataset.**

***Index Terms*—fisheye, MOT, people tracking, re-identification, tracking-by-detection, SORT, lightweight**

I. Introduction

Human tracking is a still current challenging task [1]. In a video, people change direction, pass next to each other, are occluded, their detected positions can be unreliable and as such, it is easy to lose or not be able to find a person. These problems are emphasised in distorted contexts such as in top view fisheye cameras. Indeed, a fisheye camera increases the focal distance and embraces a wide field of view [2], [3], making people look inclined and distorted in the image. Consequently, people have irregular movements in the image, their aspect ratio, size and orientation can change drastically as they move. This context is extensively studied for detection [4]–[9] and tracking tasks [10]–[13] as fisheye cameras give 180° of vision and as such find applications in many fields such as security, autonomous driving and aeronautics.

In this paper, a tracker adapted to top view fisheye images is proposed. The goal is for the algorithm to be a reliable, real time solution that could be integrated in embedded systems, while still incorporating deep learning solutions. The tracker will consist of a detector from the YOLO (You Only Look Once [14]) suite of algorithms and a tracker inspired by the SORT (Simple Online Realtime Tracking [15]) suite of trackers. A reduced view of the algorithm's architecture is presented in Fig. 1. To test this tracker, the MOT (Multiple Object Tracking) Challenge will be used: a well-known benchmark for state of the art trackers [16]. However, the datasets included in MOT only contain videos from perspective cameras; therefore we developed our own dataset to work with MOT Tools. Our tracker and two other state of the art trackers will be tested on this dataset. With this comparison, we aim to demonstrate the effectiveness of our proposed approach for human tracking in challenging top view fisheye image contexts.

The next section presents the related trackers, which are compared with the proposed method. The Sec. III is dedicated to the developed method for tracking people in fisheye sequences. Then, evaluations and results are reported in the Sec. IV, with comparison of recent methods. Eventually, Sec. V will conclude this paper and propose some perspectives.

II. SORT and related trackers

A. SORT – Simple Online and Realtime Tracking

People tracking consists in following through time the position of a person. This is the point of the SORT algorithm [15] which is based on the combination of statistics and state estimation. It retrieves position data from a detection algorithm. To know if two detections in different frames target the same person, SORT combines the *IoU* (Intersection over Union) and the Hungarian algorithm: an association algorithm. The position prediction is done using a Kalman filter and is based on several statistics: the position of the target, the scale of the detection box, their respective velocities and its aspect ratio.

B. Upgraded SORT

DeepSORT is a more advanced version of SORT [17]. It adds the aspect ratio velocity to the statistics used for the state estimation. In addition, it implements the use of the Mahalanobis distance and the cosine distance to obtain better association results. Moreover, the biggest breakthrough is the use of deep learning to extract *features* from the image and re-identify more easily the appearance of a target to better associate it.

StrongSORT is an improvement of DeepSORT. For the association part, it adds the Exponential Moving Average which allows for a better combination of re-identification *features* as well as camera movement compensation [18].

BoT-SORT represents the latest improvement (2022) of SORT by merging cosine distance with re-identification for better

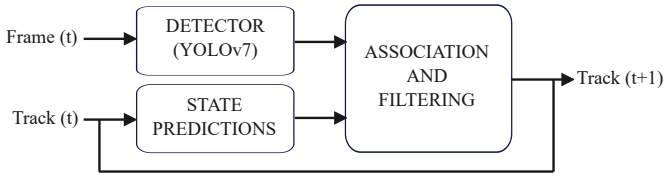

Fig. 1. Reduced architecture of the FORT algorithm for a frame at time t. Detections and tracked objects are associated and their states updated at every frame.

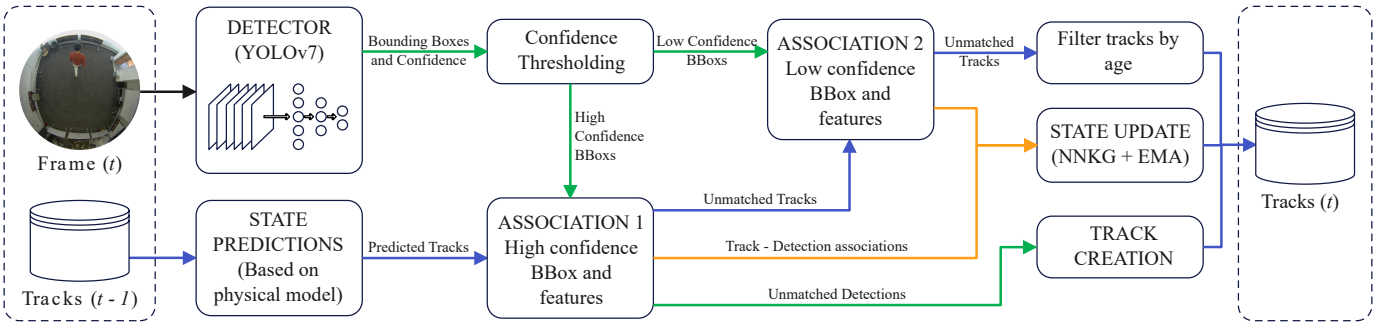

Fig. 2. Global architecture of the FORT algorithm for a frame at time t. Detection BBoxes are created from the frame and are split into two groups using associated confidence. Tracks from the previous frames have their new state predicted and are then associated with high confidence BBoxes. The remaining Tracks are associated with low confidence BBoxes. Unmatched tracks are filtered depending on their age. Unmatched high confidence BBoxes are created into new tracks. Finally, Track-BBox associations are used to update the state of the track using an adapted Kalman filter and the characteristic of each track are updated using Exponential Moving Average (EMA).

associations as well as a new camera motion compensation module [19]. This algorithm uses YOLOX [20] or YOLOv7 [21] as a detector.

III. FORT – Global Architecture

The proposed method, FORT (Fisheye Online Realtime Tracking), is divided into 3 steps: *(i)* Detection, *(ii)* Tracking and *(iii)* Association; all detailed in this section.

A. Detection

The first step in tracking is the detection. To this end, we compared three of the most recent, efficient and fast algorithms to determine which would be more reliable in a fisheye context:

- YOLOv5 [22], version L, update 6.2,
- YOLOX [20], version M,
- YOLOv7 [21], version X, called YOLOv7X.

These algorithms were then trained and tested on a dataset composed of both Mirror Worlds[1] images and images we acquired (2 828 images for training and 491 for validating, 29% from Mirror Worlds and 71% from our acquisitions) which were resized to 640×640 pixels. Training was done for 10 epochs. Three statistics were collected during these tests, interference time, the mean average precision $mAP_{0.5}$ and $mAP_{0.5:0.95}$ are used to quantify the detection accuracy:

$$\begin{cases} mAP_{\mathcal{H}} & = \dfrac{Number\ \ of\ \ detections\ \ where\ IoU \geqslant \mathcal{H}}{Number\ \ of\ \ detections}, \\ mAP_{0.5:0.95} & = \frac{1}{10} \cdot \sum_{k=0}^{9} mAP_{(0.5+0.05 \cdot k)}, \end{cases} \tag{1}$$

with $IoU = \dfrac{Intersection\ area\ of\ both\ boxes}{Union\ area\ of\ both\ boxes}$ and $\mathcal{H} = 0.5$.

We observe from these results that these algorithms, whose sizes are relatively similar, do not obtain the same interference time. As reported in Tab. I, only YOLOv7 brings us guarantees in terms of precision in its results while keeping an interference time lower than the two other algorithms, this is why this algorithm was chosen for the detection process.

[1] www2.icat.vt.edu/mirrorworlds/challenge/

B. Tracking process

The output of the algorithm is stored in objects called tracks, which are characterised by their state, *features* and a unique identifier. Information on position and velocity are stored within the state variable. The flowchart presented in Fig. 2 gives a global overview of the tracking process.

Using the YOLOv7 detector, a list of detections is created for each frame. These detections are then sorted according to their confidence scores. Tracks carried over from the previous frame have their new state predicted based on our physical model. A first association is done between the high confidence detections and the tracks. This process returns a list of track-detection pairs, which are updated (see Sec. III-F), as well as a list of unmatched detections which are used to create new tracks. The remaining (unmatched) tracks are associated with the low score detections, here the unmatched detections are discarded, the unmatched tracks filtered and the track-detection pairs updated.

C. Association

The goal of the association algorithm is to match tracks left from the previous frame with detections from the current frame. This algorithm takes in state predictions for each track, bounding box (BBox) and corresponding sub image for each detection. The state predictions are created using a physical behaviour model based on the previous tracks' states and the elapsed time dt. As a first approach, we attempted to account for the curvature of the fisheye lens by using a pre-computed gradient. However, this approach required large amounts of memory to be efficient and created problems where the gradient diverged (image center and edge). Furthermore, it gave very small performance gains in places where the

TABLE I
Detection performance of different YOLO versions in the context of an Overhead fisheye camera (Tesla T4 GPU).

Version	$mAP_{0.5}$	$mAP_{0.5:0.95}$	Detection (ms)	Weights Size
YOLOv5l	0.716	**0.421**	31.3	147 MB
YOLOX	0.636	0.341	36.6	194 MB
YOLOv7x	**0.789**	0.393	**28.5**	137 MB

gradient was stable. This could be explained by the small tracking interference time dt (<0.1s); as a result, consequently, the space traveled in between the frames by the track can be considered to be linear.

1) IoU: Using the *IoU* between the predicted track states T^{ps} and the Detected BBoxes D^{bb}, the *IoU* cost matrix $\mathcal{A}$ is calculated. With n the number of detections and m the number of tracks:

$$\mathcal{A}_{i,j} = 1 - IoU(T_i^{ps}, D_j^{bb}), \quad \forall (i,j) \in [\![1,m]\!] \cdot [\![1,n]\!]. \tag{2}$$

2) Re-Identification (ReID): The *IoU* does not discriminate enough when two people are close together and also discriminates too much when the state prediction is far from the actual detection. To counter this, the characteristics of the people contained within the detections are compared with those of the people represented by each track. To extract these characteristics, the head of the popular classification algorithm ResNeXt-50 [23] is used, with weights pretrained on ImageNet [24]. After obtaining *features* in each BBox D^{fea} and using stored *features* in each track T^{fea}, the cosine distance d^{cos} is computed between the *features* of each group to obtain the *ReID* cost matrix $\mathcal{B}$:

$$\mathcal{B}_{i,j} = \max(0, d^{cos}(T_i^{fea}, D_j^{fea})), \quad \forall (i,j) \in [\![1,m]\!] \cdot [\![1,n]\!]. \tag{3}$$

3) Final Association: Using an *IoU* threshold θ_{IoU} and a *ReID* threshold θ_{ReID}, the *ReID* cost matrix is filtered. The goal is to penalise high *ReID* costs on BBoxes with low overlap with the expected BBox. The final cost matrix is composed of the lowest costs between the *IoU* cost matrix and the *ReID* cost matrix. Finally, a distance threshold θ_d penalises all detection-track pairs which are too far from each other. Once the final cost matrix has been computed, an assignment threshold (θ_{cost}) is set. Then, detection-track matches are created using the Hungarian algorithm [25] only considering detection-track pairs where the cost is below θ_{cost}. This process outputs detection-track matches, unmatched detections and unmatched tracks. The association process is summarized in Fig. 3.

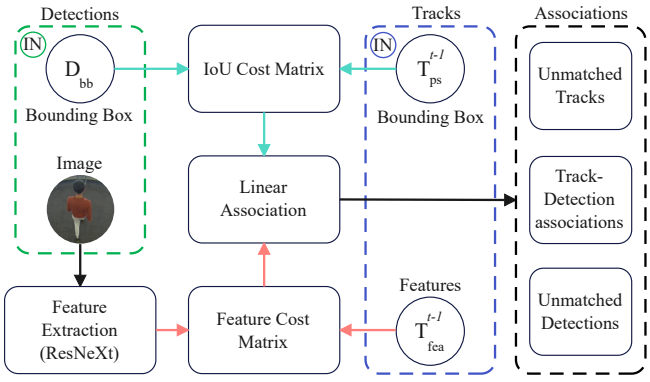

Fig. 3. Detections and tracks association method. First the IoU is caculated between each bounding box (BBox) and Track this creates the IoU cost matrix. Secondly, features are extracted from the image in each BBox using the Head of an image classifier (ResNeXt-50). The cosine distance is then calculated between each BBox and Track creating the Feature cost matrix. These two matrices are combined and a linear association is performed to create track detection associations and leaving unmatched tracks and detections.

D. Kalman Filter

To update the state of each track, the state predictions are combined with state measurements (*i.e.* detections). The combination was done using the well known Kalman algorithm [26]. This algorithm is, as of now, the optimal estimator for linear system models. This filter uses a prediction step to estimate the current state based on the previous state and a correction step to update the estimation based on the new measurement.

At time t, the prediction step gives the predicted new state x'_t and error covariance P'_t using the previous state x_{t-1} and error covariance P_{t-1} at time $t-1$:

$$\begin{cases} x'_t &= F_t \cdot x_{t-1} \\ P'_t &= F_t \cdot P_{t-1} \cdot F_t^T + Q_t, \end{cases} \tag{4}$$

where F_t is the state transition matrix and Q_t is the process noise covariance.

The correction step reduces the error of the predicted new stated and error covariance using the measurement y_t:

$$\begin{cases} K_t &= P'_t \cdot (P'_t + R_t)^{-1} \\ x_t &= (I - K_t) \cdot x'_t + K_t \cdot y_t \\ P_t &= (I - K_t) \cdot P'_t, \end{cases} \tag{5}$$

where R_t is the measurement noise covariance, K_t is the Kalman gain, x_t is the updated estimated state, and P_t is the updated error covariance. With this technique, noise from the detector can be reduced, however the matrix F_t indicates a linear state transition. Unfortunately, the time step between two consecutive states is too large to allow us to consider the relation between these two states to be linear.

A potential solution to this is the Extended Kalman filter [27] which linearises the model to solve the problem using the Kalman method. Due to unsuccessful attempts in calculating the gradient of the track state (see Sec. III-C), this algorithm was not implemented.

E. Kalman Filter Suitable for Fisheye Environment

The proposed solution was the implementation of a neural network to calculate the vector (Kalman Gain) [28], [29] used to interpolate between the detection and the state prediction. As of now, the network is a fully connected neural network with 5 inputs, 10 hidden neurons and 8 outputs with sigmoid activation functions on the hidden and output layers. As inputs, the network takes the detected position, the difference between the detected and predicted positions and whether or not the track had been lost the previous frame.

F. Tracks Update

Each track is defined by its *features* and current state. For a track-detection pair, the track's *features* are updated by interpolating between the track's *features* and the *features* obtained from the detection (Exponential Moving Average) as in [30]. Also, the track's state is updated by interpolating between the predicted state and the detected state using factors (Kalman Gain) outputted by the neural network detailed in Sec. III-E. For reference, the algorithm is detailed in the

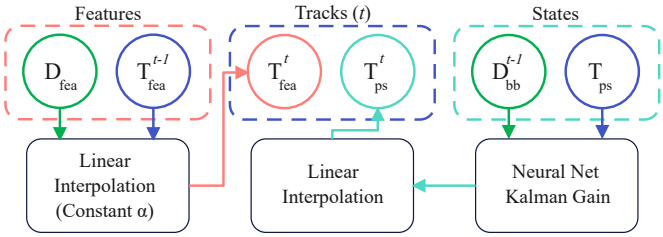

Fig. 4. Track-Detection update process. The track's and detection's state and features are combined. New track features are the linear interpolation between the extracted detection features and current track features using a constant α. New track state are the linear interpolation between the detected state and current predicted state using a value (Kalman Gain) calculated by a neural network (NN).

flowchart presented in Fig. 4. The detections which are not discarded are used to create new tracks, whose state and *features* are initialised based entirely on the detection state and *features*, with the state velocity set to 0. Tracks that are not associated have their age updated. Once a track is too old, it is discarded. The ones that remain have their state updated using their predicted state and their *features* are not modified.

IV. Evaluation and Results

State of the art tracking models are currently tested using the MOTChallenge [16] benchmark. The benchmark provides tools and annotated datasets. However, it does not provide a set containing fisheye videos. To remedy this we annotated a video (999 frames) which was appended to the Mirror Worlds Dataset (19 videos for 8751 frames). STRONGSort, BOT-Sort and FORT were then benchmarked on this dataset. The detector used was YOLOv7X as detailed in Sec. III-A, and default parameters were used across all trackers. On ReID tasks, FORT used ResNeXt-50 [23], STRONGSort and BoT-SORT used ResNet50 [31] all trained on ImageNet [24] dataset.

A. MOT Metrics

The MOTChallenge benchmark gives multiple metrics, the 3 most common are HOTA [32], MOTA [33] and IDF1 [34]. **MOTA** takes values below 100. It measures the quality of the position of the track relative to the ground truth and penalises occurrences of a person's identity switching with another.
IDF1 takes values between 0 and 100. It measures the fidelity of a track's identity relative to the ground truth identities.
HOTA aims to solve the shortcomings of MOTA and IDF1. It currently is the metric of reference for state of the art trackers.

TABLE II
Comparison of the metrics obtained with the different trackers on the entire dataset (Tesla T4 GPU).

Tracking Algorithm	Metrics			Interference time (ms)
	HOTA	MOTA	IDF1	
StrongSORT [18]	46,2	46,1	64,1	81,3
BoT-SORT [19]	45,6	53,9	**66,9**	99,1
FORT	**47,6**	**56,9**	66,7	**52,6**

B. Statistical results

The benchmark results are presented in Tab. II. FORT obtains the most promising results on HOTA. It is 2 points ahead of BoT-SORT and 1 point ahead of StrongSORT. The difference between the 3 algorithms gets bigger on the MOTA metric showing better concurrence of FORT tracking results with ground truth detections. On the IDF1 metric, BoT-SORT takes the lead, showing better re-identification capabilities, but FORT is only 0.2 points behind. Additionnaly, FORT shows interference times 1.5 times faster than StrongSORT and 1.9 times faster than BoT-SORT. The most complicated video comes from our acquisitions. The tracking results are detailed in Tab. III. Here, YOLOv7X creates many false positives and some false negative explaining the trackers' low scores (illustrated in Fig. 5, Video 3). However, the video is closer to a real life context as it contains a poor image quality, bright and dark areas, camera movement and a very cluttered environment, as such, we denote it *real conditions*. Here, FORT gives much better tracking results on all 3 metrics.

C. Visual results

Fig. 5 shows visual results on 3 different videos; for each 3 non-consequent frames are displayed. Without camera movement and in clear environments, the trackers do not encounter too many bad detection problems (video 1 and 2). Still, there are re-identification problems, StrongSORT creates 30 tracks up to frame (b) of Video 2. Furthermore, between frames (b) and (c) of Video 2, a man reaches over the table. In this instance, only FORT was able to keep a faithful detection and identification. However, in Video 3 (*real conditions*), the trackers do not perform as well. StrongSORT is not able to manage the low quality of the detections, meaning a lack of tracks on the people in the video and tracks of objects cluttering the space as seen in all 3 frames. This explains the very poor results of the trackers.

BoT-SORT and FORT have bboxes that enclose the people properly, but the trackers are not able to keep the indices constant during the entire video. This situation is even more accentuated for BoT-SORT whose IDs reach 40 in frame (c). A link to OneDrive is available to consult and compare the performance of all 3 trackers on each video of the dataset[2].

[2]Tracking results link

TABLE III
Comparison of the metrics obtained with the different trackers on the *real conditions video* (Tesla T4 GPU).

Tracking Algorithm	Metrics		
	HOTA	MOTA	IDF1
StrongSORT [18]	1,6	-10,8	2,2
BoT-SORT [19]	19,2	32	31,3
FORT	**27,8**	**36**	**34,1**

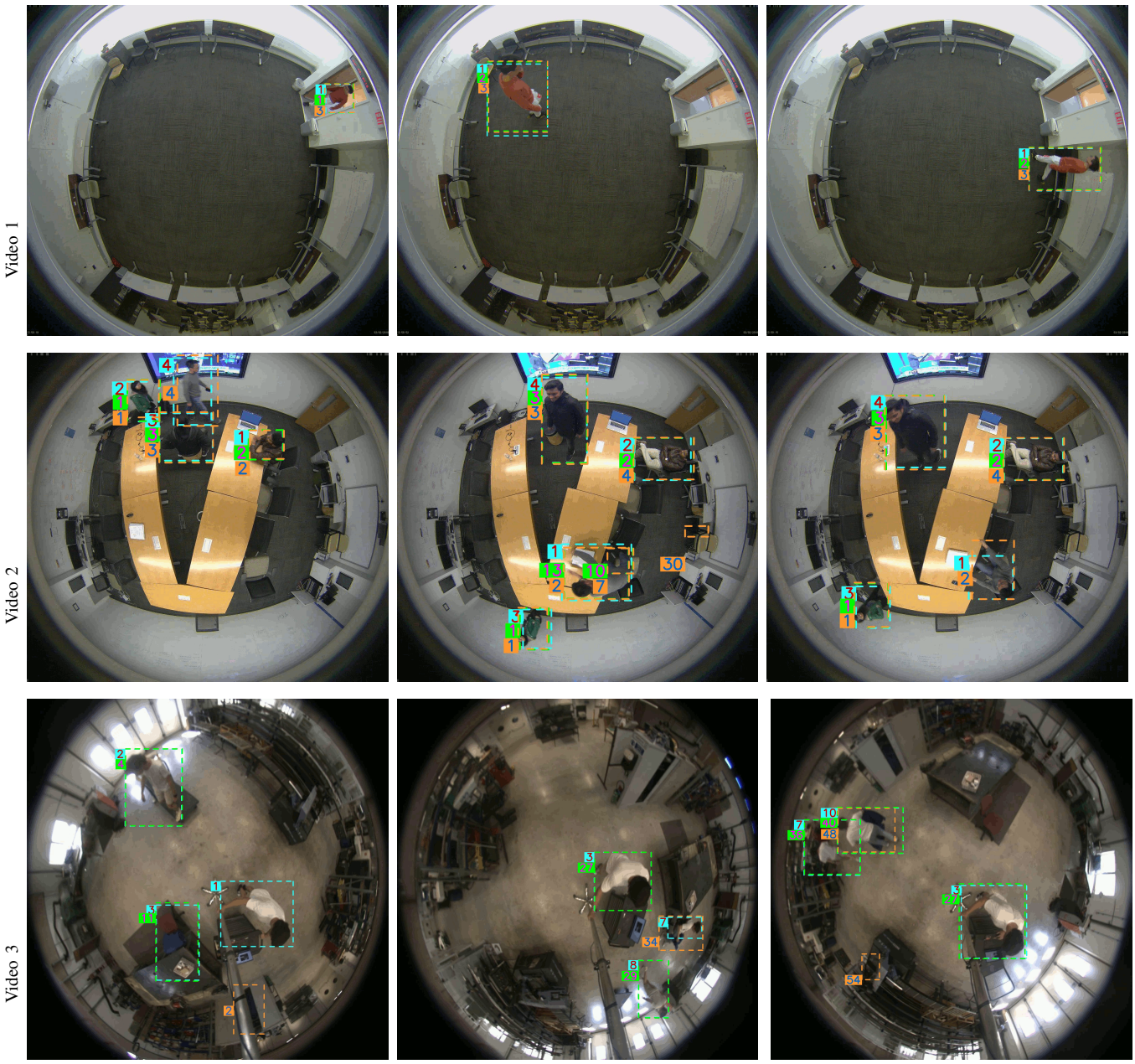

Fig. 5. People tracking on three top-view fisheye videos, orange: StrongSORT [18], green: BoT-SORT [19], and blue: FORT.

V. Conclusion and Perspectives

By using deep learning methods and combining modules from multiple tracking models, FORT (Fisheye Online Realtime Tracking) is able to give reliable and fast tracking on top down fisheye images. We use features and position data of tracks and detections to associate them. Using the associations and the Kalman Filter we are able to accurately update each track and create new ones running in realtime (<35 ms).

Further improvements can be made, the Kalman filter network could gain from a LSTM (acronym of long short-term memory networks, see [35]) section and ResNeXt-50 *features* could be more precise if the its weights were trained on images pulled from fisheye contexts. Finally, the code for the FORT algorithm can be found on GitHub[3].

[3]https://github.com/BenoitFaureIMT/FORT

Additional Result

The Fig. 6 presents another example of people detection in fisheye images. In this example taken from our dataset, the tracking algorithms created to work on perspective images (StrongSORT [18] and BoT-SORT [19]) make tracking errors when the distortion of the fisheye image is too important. In this case, only the proposed algorithm, FORT, is able to track the person horizontally and the one with the head down while BoT-SORT only tracks the person in the center of the camera and StrongSORT does not track anyone.

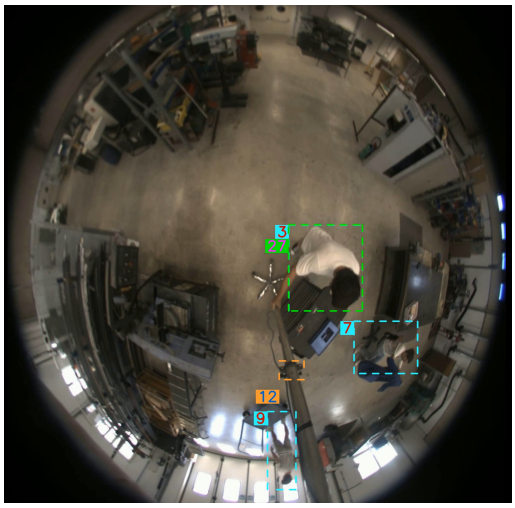

Fig. 6. Example of tracking, orange: StrongSORT [18], green: BoT-SORT [19], and blue: FORT (proposed tracking algorithm).

References

[1] O. Haggui, M. A. Tchalim, and B. Magnier, "A comparison of OpenCV algorithms for human tracking with a moving perspective camera," in *IEEE EUVIP*, 2021, pp. 1–6.

[2] J. Kumler and M. Bauer, "Fish-eye lens designs and their relative performance," in *Current developments in lens design and optical systems engineering*. SPIE, 2000, vol. 4093, pp. 360–369.

[3] S. Gao, K. Yang, H. Shi, K. Wang, and J. Bai, "Review on panoramic imaging and its applications in scene understanding," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–34, 2022.

[4] J. Yu, A. C. Perez Grassi, and G. Hirtz, "Applications of deep learning for top-view omnidirectional imaging: A survey," *arXiv preprint arXiv:2304.08193*, 2023.

[5] B. Faure, N. Odic, O. Haggui, and B. Magnier, "Performance of recent tiny/small YOLO versions in the context of top-view fisheye images," in *Intelligent Systems in Human and Artificial Perception*, 2022, pp. 246–257.

[6] O. Haggui, H. Bayd, B. Magnier, and A. Aberkane, "Human detection in moving fisheye camera using an improved yolov3 framework," in *IEEE MMSP*, 2021, pp. 1–6.

[7] D. Fuertes, C.R del Blanco, P. Carballeira, F. Jaureguizar, and N. García, "People detection with omnidirectional cameras using a spatial grid of deep learning foveatic classifiers," *DSP*, vol. 62, pp. 291–294, 2022.

[8] Q. N. Minh, B. Le Van, C. Nguyen, A. Le, and V.D. Nguyen, "ARPD: Anchor-free rotation-aware people detection using topview fisheye camera," in *IEEE AVSS*, 2021, pp. 1–8.

[9] Z. Duan, O. Tezcan, H. Nakamura, P. Ishwar, and J. Konrad, "Rapid: Rotation-aware people detection in overhead fisheye images," in *IEEE/CVF CVPR Workshops*, 2020, pp. 636–637.

[10] Y.-C. Huang, C.-W. Liu, and J.-H. Chuang, "Using fisheye camera for cost-effective multi-view people localization," in *IEEE ICIP*, 2021, pp. 3248–3252.

[11] O. Haggui, H. Bayd, and B. Magnier, "Centroid human tracking via oriented detection in overhead fisheye sequences," *The Visual Computer*, pp. 1–19, 2023.

[12] O. Haggui, M. Vert, K. McNamara, B. Brieussel, and B. Magnier, "Human tracking in top-view fisheye images with color histograms via deep learning detection," in *IEEE International Conference on Imaging Systems and Techniques (IST)*. IEEE, 2021, pp. 1–6.

[13] H. Talaoubrid, M. Vert, K. Hayat, and B. Magnier, "Human tracking in top-view fisheye images: Analysis of familiar similarity measures via HOG and against various color spaces," *J. of Imaging*, vol. 8, no. 4, pp. 115, 2022.

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *IEEE CVPR*, 2016, pp. 779–788.

[15] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *IEEE ICIP*, 2016, pp. 3464–3468.

[16] P. Dendorfer, A. Osep, A. Milan, K. Schindler, D. Cremers, I. Reid, S. Roth, and L. Leal-Taixé, "MOTchallenge: A benchmark for single-camera multiple target tracking," *IJCV*, pp. 1–37, 2020.

[17] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE ICIP*. IEEE, 2017, pp. 3645–3649.

[18] Y. Du, Y. Song, B. Yang, and Y. Zhao, "StrongSORT: Make deepsort great again," *arXiv:2202.13514*, 2022.

[19] N. Aharon, R. Orfaig, and B.-Z. Bobrovsky, "BoT-SORT: Robust associations multi-pedestrian tracking," *arXiv:2206.14651*, 2022.

[20] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding yolo series in 2021," *arXiv:2107.08430*, 2021.

[21] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv:2207.02696*, 2022.

[22] G. Jocher, A. Stoken, J. Borovec, A. Chaurasia, L. Changyu, A.V. Laughing, A. Hogan, J. Hajek, L. Diaconu, Y.K. Marc, et al., "ultralytics/yolov5: v5. 0-yolov5-p6 1280 models aws supervise. ly and youtube integrations," *Zenodo*, vol. 11, 2021.

[23] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE CVPR*, 2017, pp. 1492–1500.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, Andrej Karpathy, A. Khosla, M. Bernstein, and A.C. Berg, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[25] H. W Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[26] R.E. Kalman, "A new approach to linear filtering and prediction problems," *Trans. of the ASME - Journal of Basic Engineering*, vol. 82, pp. 35–45, 1960.

[27] S. Julier and J. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal processing, sensor fusion, and target recognition VI*. Spie, 1997, vol. 3068, pp. 182–193.

[28] G. Revach, N. Shlezinger, X. Ni, A.L. Escoriza, R. Van Sloun, and Y. Eldar, "KalmanNet: Neural network aided kalman filtering for partially known dynamics," *IEEE TSP*, vol. 70, pp. 1532–1547, 2022.

[29] J.A. Pérez-Ortiz, F. A. Gers, D. Eck, and J. Schmidhuber, "Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets," *Neural Networks*, vol. 16, no. 2, pp. 241–250, 2003.

[30] Z. Wang, L. Zheng, Y. Liu, Y. Li, and S. Wang, "Towards real-time multi-object tracking," in *ECCV*. Springer, 2020, pp. 107–122.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

[32] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe, "HOTA: A higher order metric for evaluating multi-object tracking," *IJCV*, pp. 1–31, 2020.

[33] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: the clear mot metrics," *EURASIP JIVP*, vol. 2008, pp. 1–10, 2008.

[34] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi, "Performance measures and a data set for multi-target, multi-camera tracking," in *ECCV*. Springer, 2016, pp. 17–35.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.