東京大学　関本研究室 / Sekimoto Lab. IIS, the University of Tokyo.

CNN Based Robust Pedestrian Counting for Helicopter Footage

Gergely Csönde (The University of Tokyo Department of Civil Engineering)

Yoshihide Sekimoto (The University of Tokyo Institute of Industrial Science)

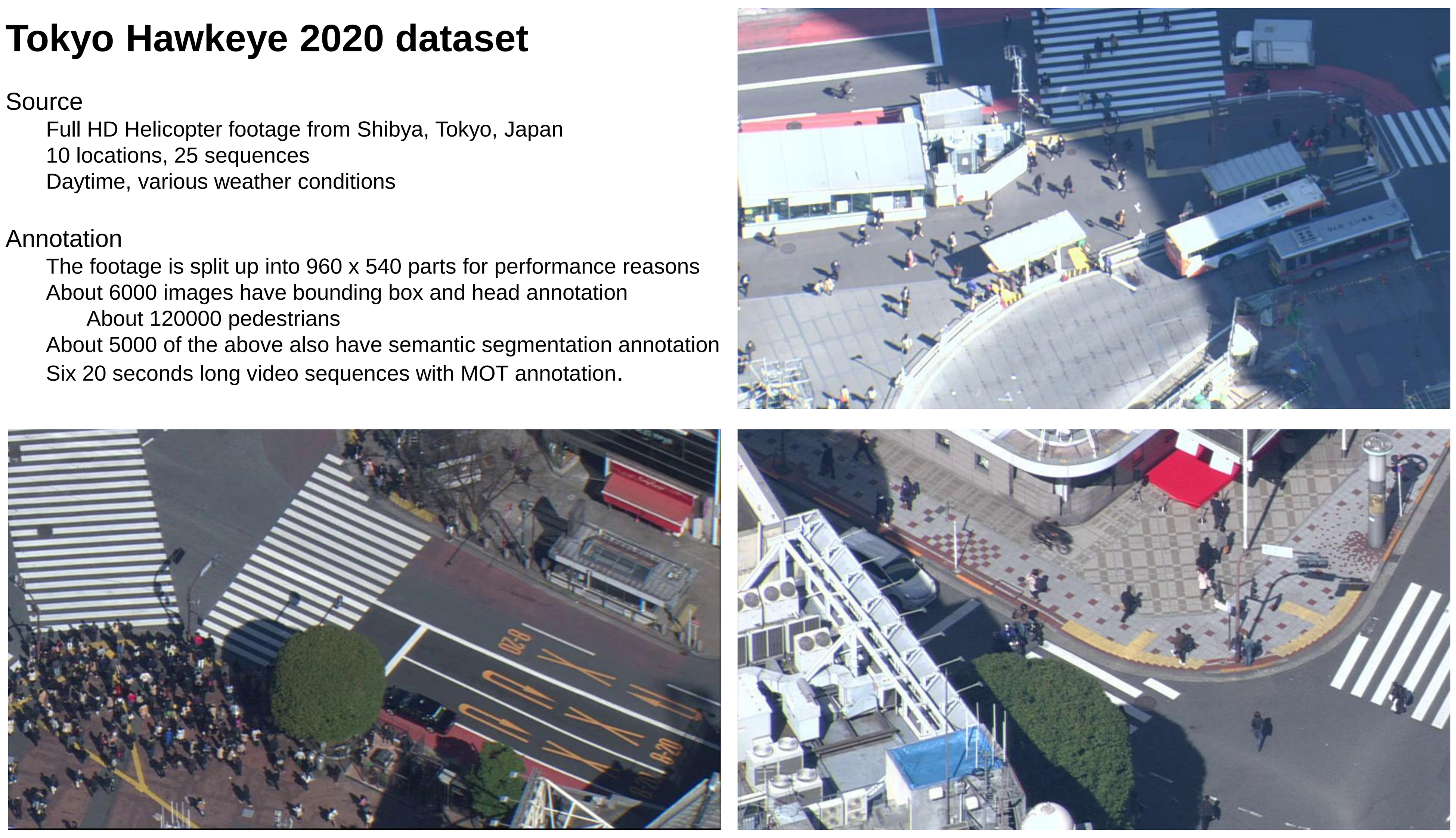
Takehiro Kashiyama (The University of Tokyo Institute of Industrial Science)

Background

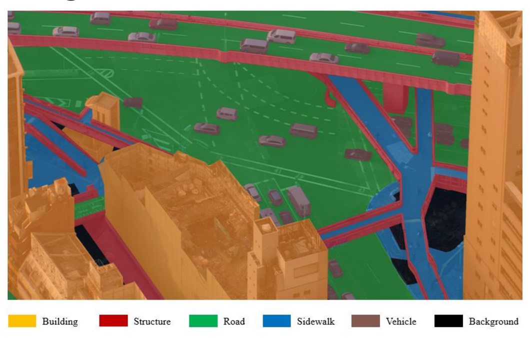
Tokyo Hawkeye 2020 dataset

Source
- Full HD Helicopter footage from Shibya, Tokyo, Japan
- 10 locations, 25 sequences
- Daytime, various weather conditions

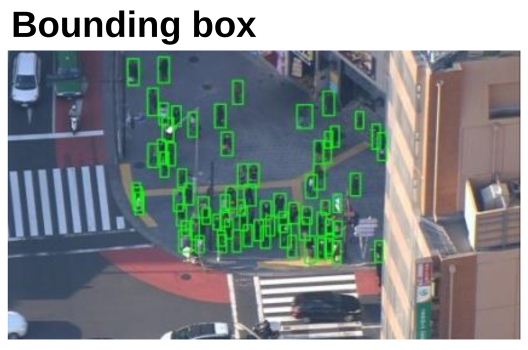
Annotation
- The footage is split up into 960 x 540 parts for performance reasons
- About 6000 images have bounding box and head annotation
  - About 120000 pedestrians
- About 5000 of the above also have semantic segmentation annotation
- Six 20 seconds long video sequences with MOT annotation.



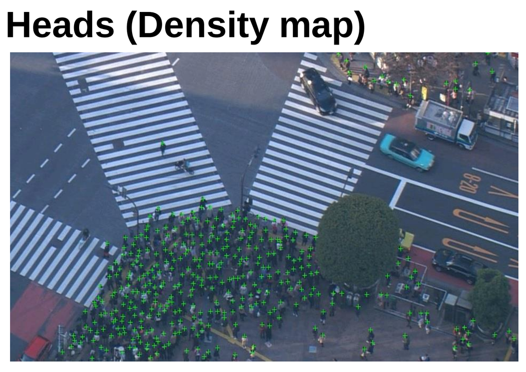
Segmentation



Bounding box



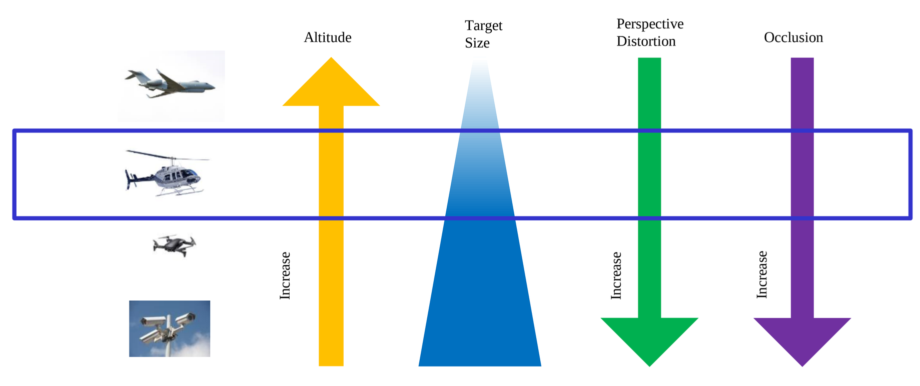
Heads (Density map)



As altitude increases:
- Target size gets smaller
- The same area can be covered from a steeper angle
  - Less perspective distortions
  - Less occlusions

Helicopter altitude at several hundred meters
- Detectable target size
- Nearly no perspective distortion
- Less occlusions
- Targets are unrecognizable, no privacy issues
- Helicopter footage is an excellent candidate for pedestrian counting

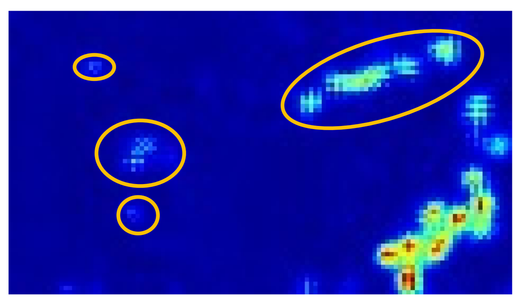
CNNs trained on conventional datasets are useless on such footage
- Important as training data



Crowd counting with semantic segmentation

Weakness of DM estimators
- Crowd density estimation methods lack semantic understanding
- Inanimate pedestrian-like objects (Billboard, statue) also counted
- Areas where pedestrians cannot be present often have nonzero density

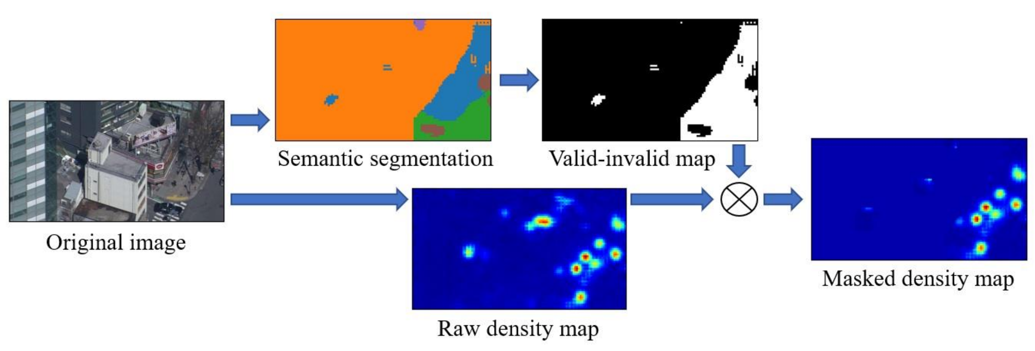
Most of the image is invalid area



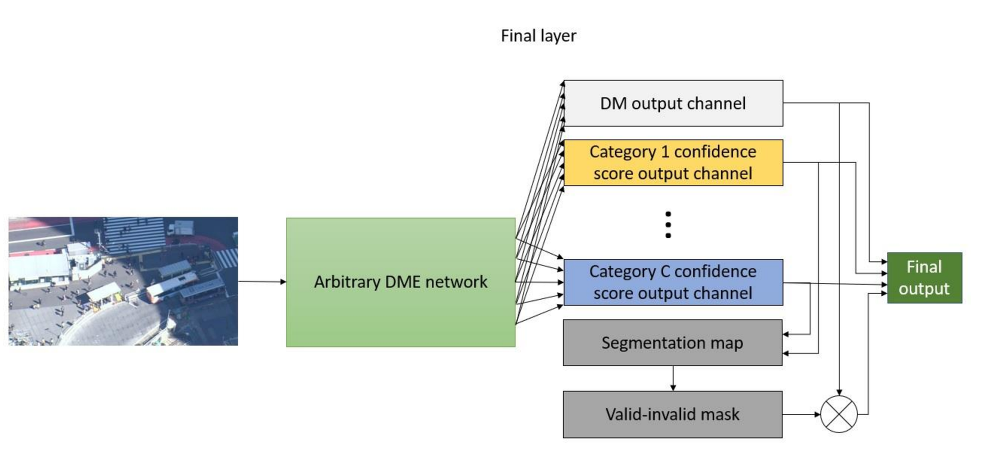
False positive detections in invalid areas

Solution
- Utilize semantic segmentation
- Good SS method can be used for masking the DM; E.g. false detections on building sides or roofs would be removed
- Incidentally, we show that simultaneous DM estimation and SS is possible with the same model.

Training Strategy 1
- Separate networks for DM and SM
- Separate training
- Combined inference
- Problems from negative examples can be avoided
- Degrades accuracy for underestimated DMs



Training Strategy 2
- Shared backbone for DM and SM
- Output is SM, raw DM and masked DM
- End-to-end training
- Two tasks are achieved in one task time
- We call this MultiTask Segmentation Method (MTSM)



Results

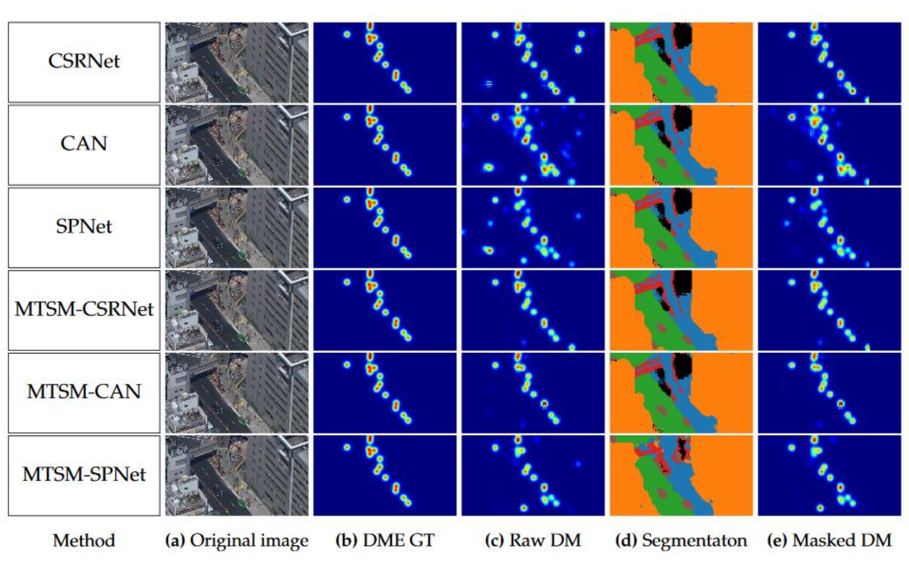


Figure 7. Sample results for comparison across methods. The segmentation for the regular DM methods was generated by MTSM-CAN.

Method		CSRNet MAE	CSRNet RMSE	CSRNet mIoU	CAN MAE	CAN RMSE	CAN mIoU	SPNet MAE	SPNet RMSE	SPNet mIoU
Simple DME		7.35	19.63		6.5	13.54		6.29	16.97	
Masked	Simple DME	7.26	19.9	0.7562	6.03	13.95	0.7562	6.14	17.15	0.7562
Masked	MTSM	6.84	20.94	0.7332	6.9	22.3	0.7562	6.58	19.56	0.5821
Raw	MTSM	6.84	20.85		6.82	22.2		6.56	19.56	

Table 1. Summary of crowd density estimation accuracies for models trained on TH2020. The mIoU column is only included to indicate the accuracy of the segmentation mask used.

Network	Building	Structure	Plant	Sidewalk	Vehicle	Road	Background	Mean	Upsampled mean
DeepLabV3	0.8494	0.3431	0.6552	0.7905	0.732	0.8632	0.6765	0.7014	0.7014
MTSM-CSRNet	0.9207	0.6263	0.6115	0.7571	0.7004	0.8584	0.6583	0.7332	0.7131
MTSM-CAN	0.9428	0.6693	0.642	0.7632	0.7072	0.8834	0.6858	0.7562	0.7358
MTSM-SPNet	0.9043	0.4162	0.4541	0.673	0.4674	0.8308	0.3288	0.5821	0.5778

Table 2. Per-category IoUs and means for DeepLabv3 and our MTSM models trained on TH2020.

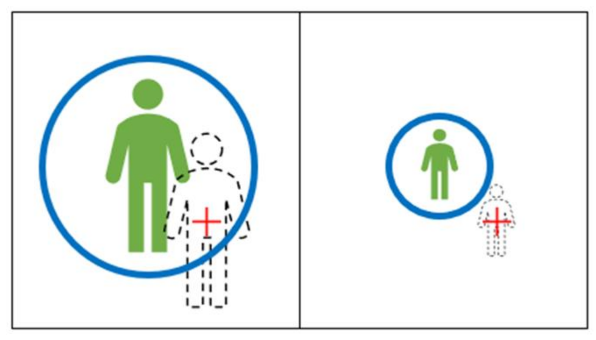
Network	Invalid (Building-Structure-Plant-Vehicle)	Valid (Road-Sidewalk-Background)	Mean
DeepLabV3	0.8916	0.8846	0.8881
MTSM-CSRNet	0.8999	0.8979	0.8989
MTSM-CAN	0.9112	0.9115	0.9113
MTSM-SPNet	0.8533	0.8508	0.852

Table 3. Valid–invalid IoUs and their means for DeepLabv3 and our MTSM models trained on TH2020.

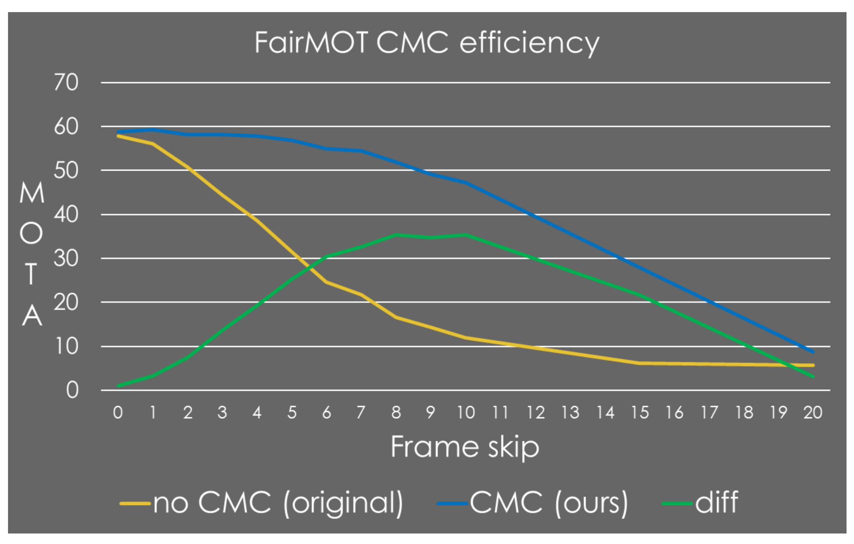
Pedestrian tracking with parallel detection and reidentification and simple camera motion cancellation

Multi-Object Tracking (MOT) problem
- Most approach follows "Tracking by Detection" paradigm
  - Previously globally optimal solutions
  - Good accuracy, slow speed, not online
- Current focus is on image pair-wise optimization
  - Detection and re-identification with CNNs
  - Tracking with Kalman filter and Hungarian algorithm or LAPJV
  - Online capabilities

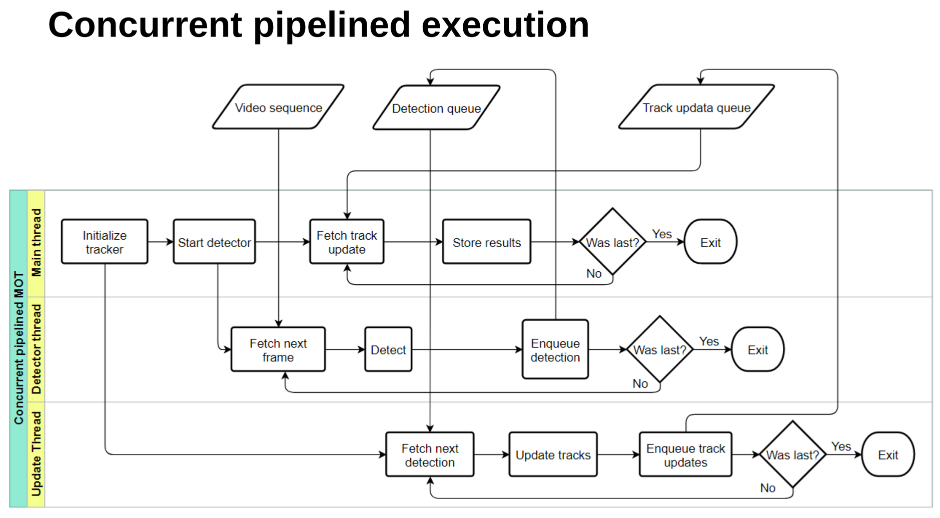
Weakness of MOT on helicopter footage
- Difference between street level and aerial footage
- State-of-the-art algorithms use Mahalanobis distance as gating metric
  - If dist > $dist_{gate}$ then association cannot be made
- The corresponding geometric gating distance is proportional to the target size
- The same camera rotation for far away targets results in larger displacement



Frame registration with key-points tracking and affine transformation
- Existing frame registration methods for wide area motion imagery assume:
  - The ground can be approximated with a flat surface
  - The ground plane is mostly stationary
- Neither of the above is true in our footage
- Only carefully selected well distinguishable objects can be used as key-points
- Robust key-point trackers have to be used
- We use CSRT [65] visual object tracking (VOT) algorithm to track key-points
  4 parameter affine transformation
  Scale, Rotation, Translation X,Y
- We are interested in real-execution therefore we check improvement with regards to the number of frames skipped.



Concurrent pipelined execution



Video sequence	02-SDP	04-SDP	05-SDP	09-SDP	10-SDP	11-SDP	13-SDP	Average
Average GT per frame	50.00	102.86	9.57	19.83	26.68	11.79	26.93	
Original FPS	22.94	19.15	26.6	26.32	24.38	25.89	24.87	23.71
Concurrent FPS (Ours)	**34.65**	**33.48**	**27.69**	**33.5**	**33.5**	**32.61**	**33.84**	**32.3**

Figure 7-7: Execution speed of the FairMOT algorithm on our target configuration with the original implementation and our concurrent one for MOT17 video sequences.

Video sequence	01B-1	01B-2	01B-3	01B-4	02D-1	02D-2	Average
Average GT per frame	10.14	14.93	15.12	15.30	22.31	26.40	
Original FPS	25.14	23.97	24.35	23.99	21.75	20.68	23.22
Concurrent FPS (Ours)	**27.97**	**27.3**	**27.75**	**26.86**	**26.79**	**26.02**	**27.11**

Figure 7-8: Execution speed of the FairMOT algorithm on our target configuration with the original implementation and our concurrent one for helicopter video sequences.