A First Sensitivity Study of Multi-object Multi-camera Tracking Performance

Miguel Ramos, Carlos Pereira, Luis Almeida

A first sensitivity study of multi-object multi-camera tracking performance

Miguel Ramos[1 [0000−0001−7805−4939]], Carlos Pereira[2 [0000−0001−6153−7555]], and Luis Almeida[1,3 (✉) [0000−0002−9544−3028]]

[1] Faculty of Engineering, University of Porto, Portugal
{up201808904,lda}@fe.up.pt
[2] NOS Comunicações, Senhora da Hora, Portugal**
dee12014@fe.up.pt
[3] CISTER Research Center on Real-Time & Embedded Computing Systems, Portugal

Abstract. Computer Vision is becoming widely used for a myriad of purposes, e.g. people counting and tracking. To execute this application in real-time, a relatively complex algorithm processes intensive data streams to identify people in a visual scenario. Although such algorithms frequently run in powerful servers on the Cloud, it is also common that they have to run in local commodity computers with limited capacity. In this work we used the Multi-Camera Multi-Target algorithm of the recent OpenVINO™ toolkit to detect and track people in small retail stores. We ran the algorithm in a common personal computer and analyzed the variation of its performance for a set of different relevant scenarios and algorithm configurations, providing insights into how these affect the algorithm performance and computational cost. In the tested scenarios, the most influential factor was the number of people in the scene. The average frame processing time observed varied around 200ms.

Keywords: Computer Vision · Machine Learning · object detection · people counting and tracking.

1 Introduction

Computer Vision (CV) became widely disseminated since the first decade of 2000s with the inception of Deep Learning (DL) [14] and the capacity to execute it in computationally efficient ways. DL is a field within Machine Learning (ML) that consists on training computers so they can learn empirical models like humans do, with numerous applications, such as speech recognition, forecasts and image identification. DL uses Neural Networks (NN) with multiple intermediate layers of neurons [2] allowing representations of arbitrary complex data, much like the human brain works. Some of the most common model architectures used in DL are R-CNN [4], YOLO [15], SSD [13] and RetinaNet [12].

** This work was developed while this author was at NOS Comunicações.

Nowadays, a common application of DL is object detection. Extending the detection of objects from a single frame to a set of consecutive frames enables tracking objects motion, too. Applied to people this is commonly called people counting and tracking.

Our specific motivation is to use security cameras already deployed in small retail spaces to obtain anonymized information from customers, namely their paths in real-time within the store, enabling new forms of on-the-fly market analysis. For this purpose, the video streams from the cameras have to be transported to a server for real-time processing. Using a server in the Cloud brings significant advantages in processing time, but also implies significant Internet communication with its overhead and exposure to the threat of access to private data. For this reason, this application requires processing in the premises of the store. In turn, this brings up the challenge of executing the counting and tracking algorithm in a local server, with limited resources.

In this paper we study the performance of people counting and tracking using multiple cameras with overlapping fields of view and an already trained DL system running on an ordinary personal computer. We selected the recent OpenVINO^{TM} toolkit [9] since it is currently the only open framework to offer a Multi-Camera Multi-Target algorithm. Then we carried out a sensitivity study to determine the parameters that impact the algorithm performance the most. We considered the number of people in the scenario, the speed of their movements, the level of overlap between the cameras and their frame-rate. The performance was assessed with the MOTA (Multi-Object Tracking Accuracy) metric and the average frame processing time. To the best of our knowledge, this is the first such study carried out on the Multi-Camera Multi-Target algorithm of OpenVINO^{TM}. The next section discusses related work, Section 3 presents the experimental setup while Section 4 discusses the results. The conclusion appears in Section 5.

2 Related work

Object detection aims at determining the objects location in an image and assigning a semantic category to each one, together with an identifier. Tracking aims at assigning the same identifier to the same object as it moves around in the scene. Tracking can be rather challenging when there are objects' occlusions, significant changes in illumination and objects' posture, and when objects exit and reenter the scene [5].

These challenges are tackled by image object detectors, which can be split into two main groups: i) two-stage detectors and ii) one-stage detectors [11]. The former uses a region proposal generator to propose the presence of objects in the image, inserting them in a bounding box, then the features from the candidate boxes are extracted and, finally, the classification and regression procedures are executed for each of the candidates boxes. The most representative two-stage detector is the Faster R-CNN [4]. On the other hand, one-stage detectors propose the candidate boxes directly, without using a generator. The most well-known

one-stage detectors are YOLO [15] and SSD [13]. The two-stage detectors provide an accurate localization and identification of the objects and the one-stage detectors can deliver high processing speed results. Faster R-CNN models, nevertheless, can be as fast as the others if specific configurations are made, but better accuracy can only be attained by sacrificing speed [6].

While a good detector is key for improving single-camera performance, good appearance features are crucial for multi-camera performance [17]. Moreover, specific algorithm configurations, e.g., threshold for occlusion detection, play an important part on the overall performance of the re-identification task. Throughout the last decade, several studies proposed custom algorithms and implementations to solve the multi-camera multi-target challenge, but limitations on flexibility and openness hinder reproducibility and improvements [19, 16]. To circumvent this, open frameworks such as OpenCV and Caffe have been gaining adoption as they set a common ground for all developers and researchers [10, 3]. Still, until the release of OpenVINO™, and to the best of our knowledge, no free open-source framework capable of addressing multi-camera multi-target tracking and counting was available. Additionally, the toolkit is tuned to take advantage of Intel hardware, yielding considerable performance improvements when comparing to either OpenCV or Caffe [7].

3 Experimental setup

We organized the workflow of the target sensitivity study as follows. First, we chose an algorithm for people counting and tracking that is real-time capable. Second, we defined the variables' spaces to be explored and the performance metrics. Third, we carried out experiments to cover the variables space, assessing the defined performance metrics. From this step we could identify the variables that exert a stronger impact on people counting and tracking.

We ran the algorithm on a Thinkpad 580s laptop computer with Windows 10 Pro, featuring a 7th generation Intel® Core™ i5-7300U processor (2.60 GHz, up to 3.50 GHz with Turbo Boost, 2 Cores, 4 Threads, 3MB Cache), integrated Intel® HD Graphics 620 accelerator and 8 GB of DDR4 2133MHz RAM.

3.1 OpenVINO™ toolkit

As we referred before, the OpenVINO™ toolkit [9] is currently the only free and open framework to offer a Multi-Camera Multi-Target algorithm. It includes a comprehensive set of algorithms and tools to develop ML-based applications using state-of-the-art artificial neural networks, including convolutional, recurrent and attention-based networks, optimized for Intel® hardware to maximize performance. The toolkit also includes an extensive set of pre-trained models, the Open Model Zoo, and demo applications [8] that show how to use the toolkit Inference Engine in specific use-cases.

The Multi-Camera Multi-Target algorithm allows tracking a predefined class of objects in video streams captured by several cameras. It uses two models in

the Intermediate Representation format, namely Object Detection and Object Re-Identification. The first one is responsible for identifying the target class of objects in the video frames, and the second one assures that each correctly detected object is assigned a unique identifier (ID). As input, the algorithm can receive paths to pre-recorded video files or indexes of cameras for real-time operation. The algorithm workflow consists on reading tuples of frames, one from each source, and for each frame in the tuple it runs the object detector and then, for each detected object, it extracts embeddings using the re-identification model. Then, all the extracted embeddings are passed to the tracker which assigns an ID to each object. Finally, the algorithm displays the resulting bounding boxes and unique object IDs assigned during the tracking process.

3.2 OpenVINO™ models

Inside the OpenVINO™ group of pre-trained models there are multiple options for each specific task. To help deciding which one suits a specific purpose best, the toolkit provides a set of characteristics about each model. For object detection, we chose the ***person-detection-retail-0013*** [9] model given its superior Average Precision (AP) metric. This model is based on MobileNetV2-like backbone that includes depth-wise convolutions to reduce the amount of computation for the 3x3 convolution block [18]. Concerning the re-identification models, we chose ***person-reidentification-retail-0277*** [9] for the same reason as the object detection model. In this case, we observed the superior Rank@1 accuracy and mean AP metrics on the Market-1501 dataset. This is a person re-identification model for a general scenario. It uses a whole body image as input and outputs an embedding vector to match a pair of images by the cosine distance. The model is based on the OmniScaleNet backbone with Linear Context Transform blocks developed for fast inference.

3.3 Multi-object tracking metrics

To assess the algorithm performance we chose one well-known tracking metric, namely MOTA [1], together with the average frame processing time.

MOTA is defined in Eq. 1 where ϕ represents the number of fragmentations (number of times a tracking ID switches) and T is the overall number of detections. FP and FN are the sums of f_{p_t} (false positives) and f_{n_t} (false negatives) over all frames t from all cameras. False positives correspond to situations in which the tracker detects a target in a frame where there is none in the ground-truth. Conversely, false negatives are cases in which the tracker failed to detect targets that existed in the ground-truth.

$$\text{MOTA} = 1 - \frac{FN + FP + \phi}{T} \tag{1}$$

3.4 Variables space

For the targeted sensitivity analysis we organize the variables to study in two groups, namely algorithm variables and video variables. Algorithm variables rep-

resent the 14 algorithm configuration parameters in OpenVINO. To reduce this number we carried out a few preliminary sensitivity tests (Table 1). We assigned three different values to each parameter covering the respective scales and assessed the variation of the MOTA metric. Then, we picked just the four that caused the strongest metric variation (column R in Table 1).

Table 1. Preliminary sensitivity tests to choose algorithm variables

Variables		Scale	MOTA (%)			R Average Interval (%)	Selected Parameters
			R1	R2	R3		
mct_config	time_window	[1, 20, 80]	93.5	93.3	93.3	0.1	
	global_match_tresh	[0.01, 0.2, 0.8]	61.0	93.3	71.7	27.0	x
sct_config	time_window	[0.5, 10, 80]	49.0	93.3	93.3	22.2	x
	continue_time_thresh	[10, 50, 90]	93.2	93.3	93.3	0.1	
	track_clear_thresh	[5, 30, 80]	72.2	93.3	93.6	10.7	
	match_threshold	[0.1, 0.375, 0.8]	93.3	93.3	93.3	0	
	merge_thresh	[0.05, 0.15, 0.8]	93.2	93.3	71.6	10.9	
	n_clusters	[1, 10, 20]	93.3	93.3	93.3	0	
	max_bbox_velocity	[0.05, 0.2, 0.8]	93.3	93.3	72.3	10.5	
	detection_occlusion_thresh	[0.1, 0.7, 0.9]	56.7	93.3	55.7	37.1	x
	track_detection_iou_thresh	[0.1, 0.5, 0.9]	77.2	93.3	59.7	24.9	x
	interpolate_time_thresh	[1, 10, 80]	93.2	93.3	93.6	0.2	
	detection_filter_speed	[0.1, 0.6, 0.9]	91.0	93.3	86.5	4.6	
	rectify_thresh	[0.1, 0.5, 0.9]	93.3	93.3	93.3	0	

Video variables are those related to the videos visual content, namely number of people in the store, the speed of their motion, the frame-rate of the cameras and the overlap of their fields of view. For convenience, both overlap and speed are classified in a few qualitative groups. For overlap we considered the case in which the cameras all point to the center of the room (maximum overlap) and when the side cameras point to the sides reducing the overlap to about 50% of the image (minimum overlap). The speed was considered the same for all people in the room and its value was consistent with usual motion inside small stores, with the maximum at about 1m/s. Overall, the values used for the video variables are shown in Table 2 and cover realistic scenarios in the target spaces. The table also contains the final values decided for the algorithm variables, tuned after the preliminary experiments reported in the Table 1.

Table 2. Variables excursion for the sensitivity study

Video Configurations

Number of People	People Speed	Frame-rate (fps)	Overlap Level
{1, 2, 3}	{Low, Medium, Fast}	{4, 8, 15, 25}	{Minimum, Maximum}

Algorithm Configurations

global_match_thresh	time_window	detection_occlusion_thresh	track_detection_iou_thresh
{0.15, 0.5, 0.85}	{0, 40, 70}	{0.1, 0.5, 0.9}	{ 0.1, 0.5, 0.9 }

3.5 Video samples and results collection

In our setup we use an indoors space representing a small retail store equipped with three cameras (Fig. 1). In this setup we defined all needed operational scenarios to carry out the sensitivity study. For this purpose, we expanded all possible combinations of the Video variables values (Table 2) resulting in 72 different scenarios. For each of these scenarios we recorded a 20s video clip with the streams from the three cameras synchronized, for the sake of reproducibility.

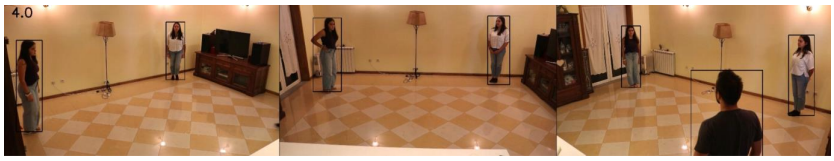

Fig. 1. Cameras perspectives (maximum overlap case)

After collecting the 72 video samples we feed them to the Multi-Camera Multi-Target algorithm, one at a time. For each video all algorithm configurations are tested. The outcome of each experiment is saved in a *detections* file, saved in JSON format, containing all the information about the obtained detections. In this step we also observe the execution time of the algorithm.

In parallel with the algorithm execution we build a ground-truth file that holds the correct detections for every experiment. This file is generated using the CV Annotation Tool (CVAT) that allows uploading a video file and then manually draw the bounding boxes around people in each frame with the respective assigned ID. After creating the ground-truth annotations with CVAT, we export them in an XML file that will be used as reference. In each experiment, once we have the detections and the annotations files, we feed them as input to the ***run_evaluate.py*** script of OpenVINO™ that will compare their similarity and output the MOTA metric. The whole process is shown in Fig. 2.

4 Results and Analysis

In this section we present and discuss the results achieved when going through the whole variables space defined in Sec. 3.4. According to Table 2, we have 72 different combinations of video variables, each corresponding to a different scenario and recorded in a separate video clip. Then we have 81 different algorithm configurations that are applied to each of the 72 scenarios. This results in a total of 5832 experiments. In the following figures, we present the results in box-plots, with each box representing one specific video configuration and the variations incurred by the 81 algorithm configurations.

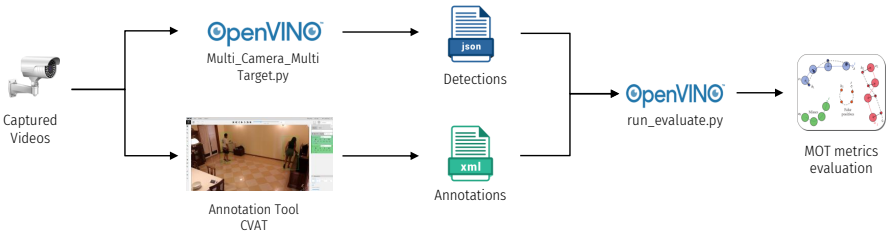

Fig. 2. Detection and evaluation process

4.1 Frames processing time

We measured the frames processing time so that all involved I/O overheads are included. Thus, we computed the total time lapsed from reading the first frame to the end of processing the last frame and then divided by the total number of frames in the video. This is a lump value, thus we carried out a simple experiment to observe the behavior of individual frame processing times considering possible interference from other software executing in the computer, e.g., Operating System (OS) services. We ran the tracking algorithm for a scenario of one person, moving at fast speed, with cameras capturing 25fps and with maximum overlap. We collected three independent samples, each with 37989 measurements of the individual frame processing time. The results show a reasonable statistical consistency. The average frame processing time varied between 180 and 200ms, the median varied between 184 and 194ms and the quartiles varied between 177 and 184ms for the 1st and 188 and 204ms for the 3rd. Thus, when observing the average lump measurements we show next, we need to keep in mind that they are still affected by this level of variation.

4.2 Number of People and Speed Variation

The results for the impact of the number of people in the scene and their speed on the MOTA are shown in Fig. 3. The strongest variation is observed for the number of people in the scene. For the three speed scenarios, the median values of MOTA decrease approximately 3% for low speed and 5% for medium and high speeds, for each new person added to the scene (for all speed values together, we saw a MOTA median decrease of 4% per new person added to the scene). This inverse relationship between the number of people in the scene and the MOTA results can be explained by the fact that as we introduce new target elements in the video frames, the possibility of occurring occlusions, ID switches, motion blur and unpredictable behaviors also increases. These issues hamper the task of identifying and tracking people. The more elements in the video frames, the more complex the image analysis will be.

On the other hand, the variation with the speed of the targets in the scene is residual. Still, we can observe a subtle decrease in the MOTA results as the

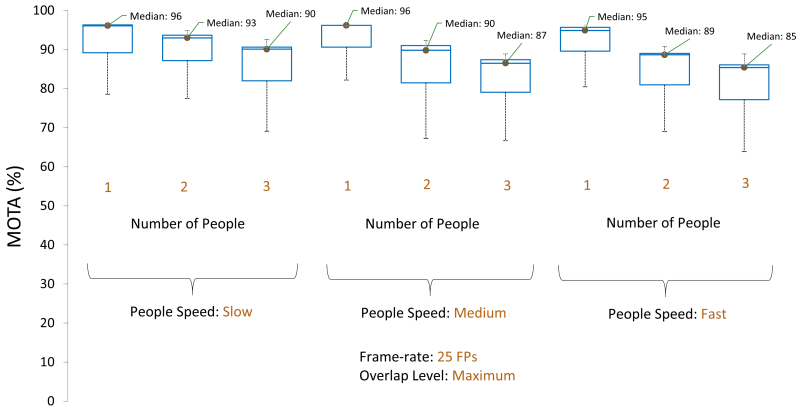

Fig. 3. MOTA results when varying the number of people in the scene and their speed

speed of the targets (people) in the frames increase. For each step in speed, we observe a decrease of the MOTA median value of approximately 1% for 1 person and 2% for 2 and 3 people (for all numbers of people together, we saw a MOTA median decrease of 2% per step in speed). Again, this inverse relationship is expected since a slower speed leads to smaller variations of the targets position between frames, leading to more consistent and accurate detections.

For the same scenarios, the measured average execution time per frame is shown in Fig. 4. Again, the impact of the number of people in the scene dominates the measurements, with a clear increase as more people are added. This is expected since more target elements (people) in the scene require more processing to analyze the video frames and output the correspondingly more detections and identifications. The figure shows an increase of approximately 34ms from 1 person to 2 people and 50ms from 2 to 3 people (across all speed classes).

In turn, the variation of the frame processing time with the speed of the targets in the scene is again residual, particularly between slow and medium speeds. Between medium and high speeds we can already observe a mild decrease of 10ms in the execution time median (for all numbers of people). We believe this effect emerges from the same reason that causes a similar variation in MOTA results. Faster speeds imply larger changes between consecutive video frames, leading to lower accuracy, thus less detections and lower computational demand.

4.3 Overlap Level Variation

The impact of the level of overlap between the images captured by the three cameras revealed to be rather small on both MOTA and frame processing time, thus we do not show them here, due to space limitations. The MOTA median reduced approximately 3% when the overlap decreased from Maximum to Minimum. This variation can be due to the algorithm using the video frames from

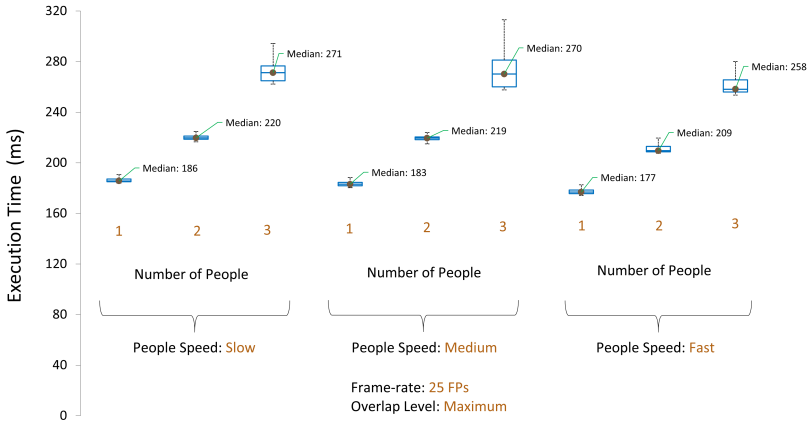

Fig. 4. Execution time results when varying number of people and their speed

different cameras to support its decision about a specific ID assignment. If we have a high level of overlap, the frames from the different cameras will be more consistent, facilitating identification assignment. Thus, detections will be more robust. The same happens with the average frame processing time, which decreases when the overlap decreases, again due to less accuracy of lower overlap that leads to less detections, thus less processing to do.

4.4 Frame-rate variation

The variation of the frame-rate of the cameras is one dimension that deserves special attention for two reasons. First, as referred in Sec. 4.1, the frame processing times are affected by interference, e.g., from the OS. Second, we are handling video clips off-line, processing their frames, one after the other, independently of the time it takes to process them. This is why we can process video clips that were acquired at any frame-rate. However, when processing in real-time, as desired in the target application scenario, this is clearly not possible. Altogether, we have observed frame processing times roughly in the range of 180 to 330ms. This means that, with the computer we used in our experiments, the maximum frame-rate that can be sustained in real-time varies approximately between 3 and 6fps. The results we show for higher frame-rates correspond to what would be achieved with sufficiently fast hardware.

Fig. 5 shows the MOTA results for different frame-rates and number of people. The latter are consistent with those in Fig. 3 (fast scenario), showing a decreasing trend with more people, but with a stronger decrease from one person to two people than from two to three people. The variation with the frame-rate is significantly more pronounced. The values of MOTA increase sharply with low frame-rates and more softly with higher rates, seeming to reach a saturation around 95% for the single person scenario. Overall, for the three scenarios

of number of people, MOTA increased approximately 26% from 4 to 8fps, 19% from 8 to 15fps and 8% from 15 to 25fps. This type of variation is expected, too, since higher frame-rate means more frames collected by the source camera per video second, thus smoother variations between frames leading to more robust tracking. The striking observation, here, is that MOTA drops sharply for a frame-rate of 4fps, which may compromise the use of OpenVINO in real-time with commodity hardware.

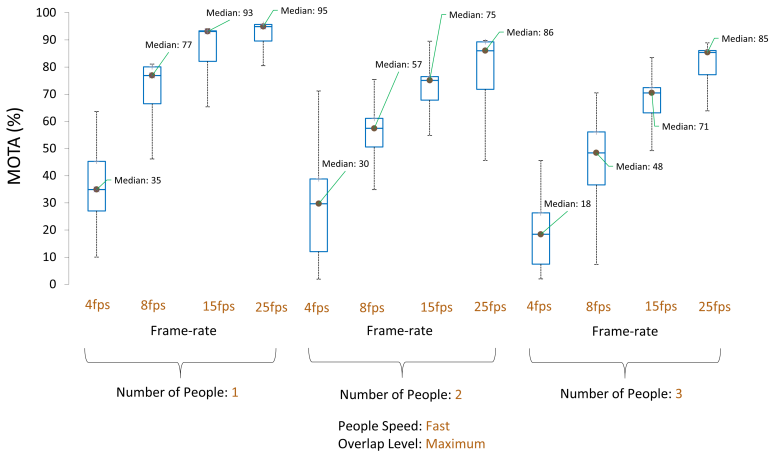

Fig. 5. MOTA results when varying frame-rate and number of people

The results for the frame processing time are shown in Fig. 6. In principle, this time should not vary with the frame-rate when processing the videos offline, except for the interference referred in Sec. 4.1. The variations we observe within each scenario with a defined number of people prompt us to find an explanation. We believe this is due to extra cache delay incurred when reading the first frames. This would appear once during each experiment, as an offset in the lump measurements, thus being consistent with a higher impact for lower frame-rates, i.e., with less frames in the video file, the impact per frame is higher.

4.5 Other variations

Finally, remember that the variations represented by the previous box plots arise from the 81 combinations of the algorithm variables. Curiously, whatever the algorithm configuration, the impact in MOTA is just moderate, frequently less than 10%, and even less in the frame processing time. In this case, there is a small but visible increase in this variation as the number of people in the scene increases.

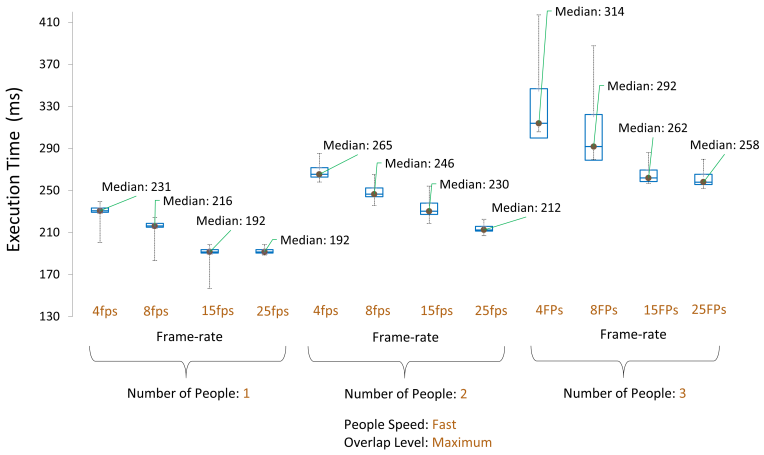

Fig. 6. Average frame processing time when varying frame-rate and number of people

5 Conclusion

In this paper we addressed the application of people counting and tracking in small retail stores using typical surveillance cameras and a common laptop computer. For counting and tracking we used the recent OpenVINO™ toolkit since it is open and optimized for execution on Intel-based computers. We used the Multi-Camera Multi-Target algorithm in the toolkit, which provides all the features needed for effective people counting and tracking, particularly re-identification. The core of the paper is a sensitivity study to assess the impact of video features and algorithm configurations on the quality of the counting and tracking and on the algorithm execution time. We defined a variables space reflecting realistic conditions, while being parsimonious in size. We found the number of people in the scene to be the most impactful variable. The average processing times indicate that real-time operation on the used hardware is limited to about 4fps. This is still a challenge because such low value implies a significant degradation in the quality of the counting and tracking process. In the future, we plan to validate the approach in a real retail store in real-time operation and compare the MOTA metric with the usefulness of the results for the purpose of the application.

Acknowledgments. This work was financially supported by: Base Funding - UIDB/04234/2020 of the Research Centre in Real-Time and Embedded Computing Systems - CISTER - funded by national funds through the FCT/MCTES (PIDDAC).

References

1. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: The clear mot metrics. EURASIP Journal on Image and Video Processing **2008** (01 2008). https://doi.org/10.1155/2008/246309
2. Bezdan, T., Bacanin, N.: Convolutional neural network layers and architectures. pp. 445–451 (01 2019). https://doi.org/10.15308/Sinteza-2019-445-451
3. Bradski, G., Kaehler, A.: Opencv. Dr. Dobb's journal of software tools **3** (2000)
4. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation (2013)
5. Gong, S., Cristani, M., Loy, C.C., Hospedales, T.M.: The re-identification challenge. In: Person re-identification, pp. 1–20. Springer (2014)
6. Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., et al.: Speed/accuracy trade-offs for modern convolutional object detectors. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7310–7311 (2017)
7. Intel: Parallel universe issue 34. `https://software.intel.com/content/dam/develop/external/us/en/documents/parallel-universe-issue-34.pdf` (2018), accessed: 2021-01-10
8. Intel® : Open model zoo demos. `https://docs.openvinotoolkit.org/latest/omz_demos_README.html`, accessed: 2020-11-10
9. Intel® : Openvino™ toolkit overview. `https://docs.openvinotoolkit.org/latest/index.html`, accessed: 2020-11-02
10. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on Multimedia. pp. 675–678 (2014)
11. Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., Qu, R.: A survey of deep learning-based object detection. IEEE Access **7**, 128837–128868 (2019)
12. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection (2017)
13. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector (2015)
14. Pitts, W., McCulloch, W.S.: How we know universals the perception of auditory and visual forms. The Bulletin of mathematical biophysics **9**(3), 127–147 (1947)
15. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2016). https://doi.org/10.1109/cvpr.2016.91, `http://dx.doi.org/10.1109/CVPR.2016.91`
16. Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: European conference on computer vision. pp. 17–35. Springer (2016)
17. Ristani, E., Tomasi, C.: Features for multi-target multi-camera tracking and re-identification. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 6036–6046 (2018)
18. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
19. Xiong, F., Gou, M., Camps, O., Sznaier, M.: Person re-identification using kernel-based metric learning methods. In: European conference on computer vision. pp. 1–16. Springer (2014)