



Real Time Multi Object Detection & Tracking on Urban Cameras

Rifkat Minnikhanov¹(✉), Maria Dagaeva^{1,2}, Timur Aslyamov¹,
Tikhon Bolshakov¹, and Emil Faizrakhmanov¹

¹ “Road Safety” State Company, Kazan, Russia
its.center.kzn@gmail.com

² Kazan National Research Technical University - KAI named
after A. N. Tupolev, Kazan, Russia

Abstract. Processing video from urban video surveillance cameras requires the use of algorithms for multi-object detection and tracking on video in real-time. However, existing computer vision algorithms require the use of powerful equipment and are not sufficiently optimized to process multiple video streams simultaneously. This article proposes an approach to using the tracker in conjunction with the YoloV4 object detector for real-time video processing on medium-power equipment. Paper also presents the solution for difficulties that arise during work with optical flow. The results of the comparison of the accuracy and speed of image processing of the applied approach with such trackers as IOU17, SORT, KCF, and MOSSEE are also presented.

Keywords: Deep learning · Object detection & tracking · Video surveillance · Computer vision

1 Introduction and Problem Definition

Currently, one of the most relevant areas in the field of intelligent transport systems (ITS) is the task of detecting and tracking objects in real-time on video streams of closed-circuit television systems (CCTV). Existing algorithms for detecting and tracking objects in the video image allow to determine the position of the object, its trajectory and class with high accuracy. However, when processing video images from multiple video surveillance cameras in a data centre, there is a problem associated with the need to accommodate many expensive graphics processor units (GPU). The development of our solution for object detection and tracking allows real-time analysis of video images on medium-power equipment with sufficient accuracy that allows using of video surveillance cameras as an optimal and operational source of information.

2 Related Works

Today, most existing tracking solutions follow the tracking-by-detection paradigm. The paradigm involves splitting the tracking process into 2 stages: detecting all objects in the image (frame) and linking the corresponding detected objects to form a trajectory.

According to the features of the application, existing trackers can be divided as follows:

1. Trackers that require a marked-up training dataset (Braso and Leal-Taixe 2020; Fang et al. 2018; Sadeghian et al. 2017; Leal-Taixe et al. 2014). This feature creates difficulties in obtaining a marked-up dataset, which makes their use difficult.
2. Trackers that perform processing with insufficient FPS (an indicator of video processing, denoted as the number of frames processed per second), which makes real-time processing impossible (Bergmann et al. 2019; Karthik et al. 2020).
3. Trackers that perform independent tracking of individual objects, without the use of a detector on each frame (Chu et al. 2019; Bolme et al. 2010). This group of trackers has a problem related to the insufficient quality of tracking objects.
4. Trackers that provide acceptable tracking quality and image processing speed (hereinafter referred to as FPS), provided that an effective detector is used (Bewley et al. 2016; Bchinski et al. 2017).

Although, the problem, outlined in the article, should be solved by trackers from the 4-th group, to date, there are no sufficiently effective, pre-trained detectors for detecting objects on video from surveillance cameras in the public domain. The exception is the YoloV4 detector (Bochkovskiy et al. 2020), which is the best in FPS/accuracy ratio. It is capable of providing real-time image processing on Nvidia GeForce 2070 graphics cards with an input image size of 320*320 pixels. However, its use on medium-power video cards (GeForce 960, 1050 Ti, 1060, 1070 and 1080) is not possible due to the low FPS rate. At the same time, the use of a lightweight version of this detector – YoloV4 – tiny (Bochkovskiy et al. 2020) is not sufficiently effective in detecting objects. The Table 1 shows a comparison of the FPS output of YoloV4 on different GPUs.

Table 1. Comparison of YoloV4 FPS on different GPUs

Nvidia GPU	Framework	An input image size	FPS
GeForce 850M	OpenCV 4.4.0 for Windows	416 * 416	5.4
GeForce 1050 Ti		416 * 416	14.3
GeForce 1050 Ti		608 * 608	9.1
GeForce 1060 Ti		416 * 416	20.4
GeForce 1060 Ti		608 * 608	16.1
Jetson Nano		TensorRT FP16	416 * 416
Jetson Nano	608 * 608		1.9
GeForce 1080 Ti	TensorRT FP32	416 * 416	27.3
GeForce 1080 Ti		608 * 608	18.2
Tesla V100 16 Gb	OpenCV 4.5.0 – pre for Linux	416 * 416	62.5
Tesla V100 16 Gb		608 * 608	37

3 Proposed Solution

In this paper, we propose a new tracker for tracking multiple objects based on the pyramidal implementation of the allowed optical flow KLT (Bouguet 2001).

Our tracker independently predicts the position of the object bounding box several frames ahead, and every few frames the YoloV4 detector is used (with the size of the input image - 608*608 pixels). This is necessary to correct the predicted values of the object frame position and track new objects. For effective prediction, the tracker uses statistical characteristics of the distribution of the calculated allowed optical flow KLT to predict the current position of the frame from its previous location. The association of objects between frames is carried out by solving the assignment problem with a greedy algorithm according to the following principle: if the centroid of a newfound object falls into one of the existing frames, then it is considered that it is the same object. If the centroid of the newfound object falls within more than one of the existing frames, then it is associated with the one whose distance to the centroid is less. In this case, an object is considered lost if the detector has not confirmed its existence a certain number of times in a row.

3.1 Predicting the Position of the Object Frame with the Tracker

To predict the position of an object's frame, it is enough to know the position of the object in the current frame and the contour or mask of the object in the previous frame. Also, given that the object is defined by a bounding rectangle rather than a contour, it is sufficient to determine how much the frame's centroid must be offset for the frame to continue to reflect the object's position. The displacement of the centroid can be determined using a statistical characteristic of the distribution of displacements of points of the object (mathematical expectation, mode or median of the optical flow of points of the object). However, there are some problems when calculating the optical flow.

1. Fast detectors do not return the object mask, but its bounding box, which does not allow you to simply accept new frame coordinates after calculating the optical flow for the coordinates of points in the current frame, for a number of reasons: some points fall on a fixed background; the object mask may be partially covered by a fixed background object and for overlapping points; the optical flow is cleared incorrectly.

This problem was solved in the following way. The absolute majority of points for which the zero optical flux is calculated will belong either to the background or to a static occluder (an object that overlaps a moving object). Points associated with the background or static occluder appear gradually in the object frame and, having a sufficiently large area, retain their position relative to the object in neighbouring frames. Excluding "standing points" (points with the near-zero optical flow) from consideration, we will most likely exclude all points related to the background or static occluder. An example of excluding points can be seen in the Fig. 1.

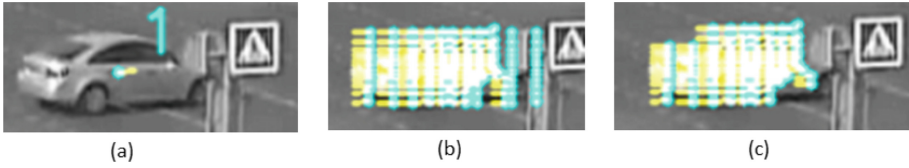


Fig. 1. (a) – the tracked object (car) moving behind a static occluder (road sign) indicating its direction of movement; (b) - display of the optical flow of a number of points inside the frame; (c) - display of the optical flow of points, except for those whose optical flow was calculated incorrectly, or for the second or more consecutive times turned out to be near-zero.

2. Calculating the optical flow over all points of an object is a very expensive operation if there are many objects or they are large.

The problem of slow calculation of the optical flow is eliminated by selecting a representative sample of points belonging to the object frame. It was assumed that the frames returned by the detector are well centred relative to the object, that is, the centroid of the frame is close to the centroid of the object and if the object is solid, then the points closer to the centre of the frame will be more likely to fall on the object, and not on the background. Obtaining a representative sample was carried out by creating a “template for placing control points” on a hypothetical object that has a side size of one length. Such a template is a tensor of size $H \times W \times 2$, where H is the number of points in the sample in height, W is the number of points in the sample in width. The tensor contains the coordinates of the sample control points for the hypothetical object under consideration. The value of the tensor elements is determined by the following formulas:

$$\left\{ \begin{array}{l}
 \begin{array}{l}
 template_{i,j,0} = 0.5 + \frac{10^{(0.5-\frac{1}{W}) \cdot (j-\frac{W}{2})}}{1 + 0.25 \left(10^{(0.5-\frac{1}{W}) \cdot (w-1-\frac{W}{2})} - 10^{(0.5-\frac{1}{W}) \cdot (w-2-\frac{W}{2})} \right)}, j > \frac{W}{2} \\
 template_{i,j,0} = 0.5 - \frac{10^{(0.5-\frac{1}{W}) \cdot (\frac{W}{2}-j)}}{1 + 0.25 \left(10^{(0.5-\frac{1}{W}) \cdot (w-1-\frac{W}{2})} - 10^{(0.5-\frac{1}{W}) \cdot (w-2-\frac{W}{2})} \right)}, j < \frac{W}{2}
 \end{array} \\
 \begin{array}{l}
 template_{i,j,1} = 0.5 + \frac{10^{(0.5-\frac{1}{H}) \cdot (\frac{H}{2}-i)}}{1 + 0.25 \left(10^{(0.5-\frac{1}{H}) \cdot (H-1-\frac{H}{2})} - 10^{(0.5-\frac{1}{H}) \cdot (H-2-\frac{H}{2})} \right)}, i > \frac{H}{2} \\
 template_{i,j,1} = 0.5 - \frac{10^{(0.5-\frac{1}{H}) \cdot (i-\frac{H}{2})}}{1 + 0.25 \left(10^{(0.5-\frac{1}{H}) \cdot (H-1-\frac{H}{2})} - 10^{(0.5-\frac{1}{H}) \cdot (H-2-\frac{H}{2})} \right)}, i < \frac{H}{2}
 \end{array} \\
 \begin{array}{l}
 template_{i,j,0} = 0.5, j = \frac{W}{2} \\
 template_{i,j,1} = 0.5, i = \frac{H}{2}
 \end{array}
 \end{array} \right. \quad (1)$$

Next, a similar pattern is applied to the frame of the object by adding to it the offset of the object relative to the coordinates (0, 0), as well as multiplying the corresponding coordinates by the width and height of the object. An example of such a template superimposed on an object with a size of 64*128 pixels is shown in the Fig. 2.

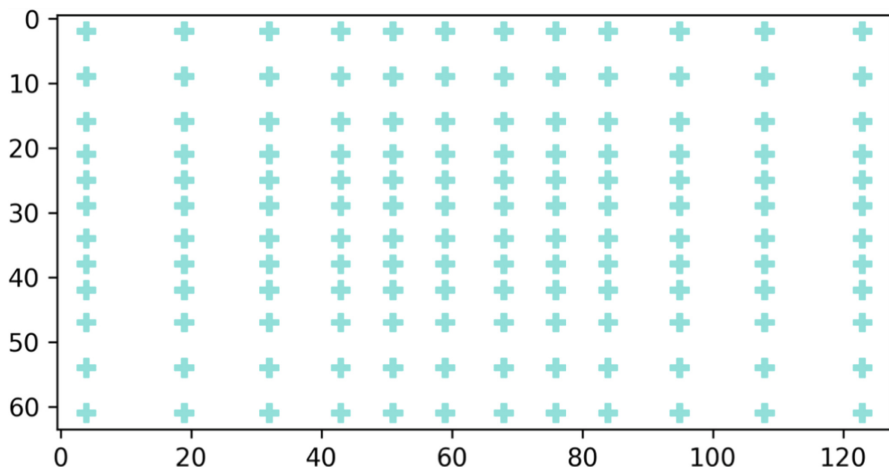


Fig. 2. “Template for placing control points” on objects defined by a $12 \times 12 \times 2$ tensor superimposed on a hypothetical object with a size of 64×128 pixels

To determine whether the object is moving or not, we calculate the proportion of control points that are standing for the first time, from all control points that are not standing for the second or more times. To do this, the optical flow of each object is calculated at the points corresponding to the previously described pattern superimposed on the object frame.

The condition that the object is worth will be:

$$\frac{\sum stop_{mask_{s,n}} - \sum (stop_{mask_{s,n}} \circ stop_{mask_{s-1,n}})}{H * W - \sum (stop_{mask_{s,n}} \circ stop_{mask_{s-1,n}}) - \sum st_{s,n}} > 0.75 \quad (2)$$

Taking into account that:

- $st_{s,n}$ - binary matrix $H \times W$ of the correctness of calculating the optical flow for the n -th object on the s -th frame, in which 1-means that the optical flow is calculated incorrectly, and 0-that it is calculated normally;
- $stop_{mask_{s,n}}$ - binary matrix $H \times W$ of immobility of control points for the n -th object on the s -th frame, in which 1 is the optical flow for the point near zero, 0 is significantly different from zero;

The coefficient 0.75 was determined empirically.

The matrix $st_{s,n}$ is determined automatically when calculating the offsets of control points by the optical flow. The matrix $stop_{mask_{s,n}}$ is filled in by comparing the coordinate differences of all control points for which the optical flow was calculated with a certain threshold, which is calculated based on the size of the object frame. If it was determined that the object is stationary, then the frame offset for it is not calculated. In addition, all elements of the matrix $stop_{mask_{s,n}}$ for this object are set to 0, so that on the next frame, if this object remains stationary, there is no “parasitic” jitter for its frame. All fixed control points and control points for which the optical flow has not

calculated offsets are removed from the calculation of the final offset of the object frame. Thus, it becomes possible to calculate some statistics on the offsets of control points with a high probability related to the object itself, and not to the background or occluders. All fixed control points and control points for which the optical flow has not calculated offsets are removed from the calculation of the final offset of the object frame. Thus, it becomes possible to calculate some statistics on the offsets of control points with a high probability related to the object itself, and not to the background or occluders. In the first version of our tracker (Makhmutova et al. 2020), the median was used as a statistical measure for determining the displacement of the object frame. Its advantage is that in most cases it perfectly copes with “outliers” in the sample of control point offsets, but its calculation requires sorting at least half of the sample. Sorting is a rather expensive operation that is poorly suited for performing it in parallel mode.

On the other hand, mathematical expectation can be calculated quickly and conveniently, but it will be a biased estimate if there are serious outliers. To fix this problem, you can recalculate the mathematical expectation after removing the control points from consideration, the offset of which differs from the already calculated mathematical expectation by a certain coefficient that depends on the standard deviation. Moreover, such an operation can be carried out iteratively the required number of times, which will consistently reduce the offset of the mathematical expectation. The visual effect of the iterative recalculation of the mathematical expectations can be seen in the Fig. 3.

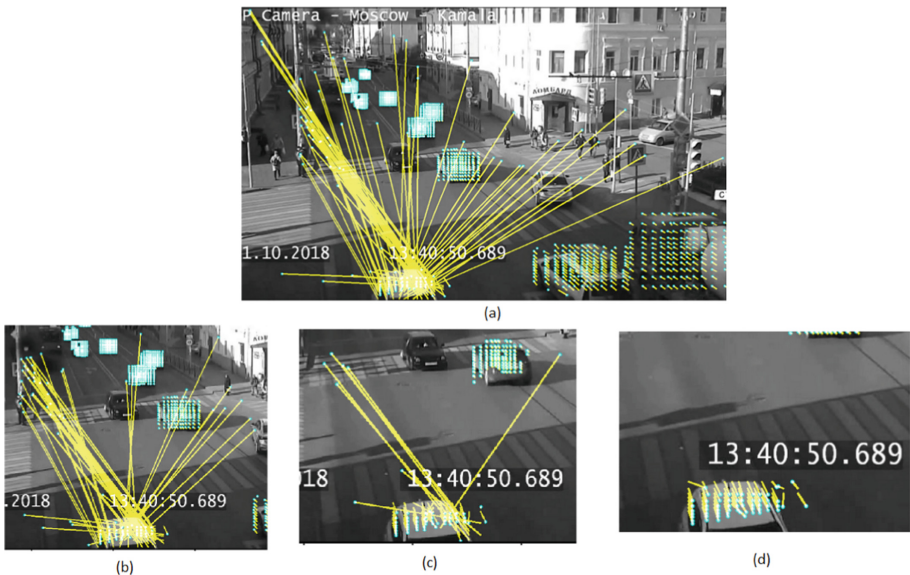


Fig. 3. Comparison of different ways to remove outliers. (a) - displaying offsets of control points without any filtering; (b) – filtering outliers with a cut-off of $\pm 1.5 \sigma$; (c) – filtering outliers with a cut-off of $\pm 1.1 \sigma$; (d) - filtering outliers with 2 cut-offs of $\pm 1.5 \sigma$ with a recalculation of the expectation value.

At each iteration, the corresponding binary mask $korr$ with size $H*W$ is calculated for the points:

$$korr_{c,s,n,i,j} = \begin{cases} 1 * korr_{c-1,s,n,i,j}, & |x_{s,n,i,j} - m_{c-1,s,n}| < K * \sigma_{c-1,s,n} \\ 0 * korr_{c-1,s,n,i,j}, & |x_{s,n,i,j} - m_{c-1,s,n}| > K * \sigma_{c-1,s,n} \end{cases} \quad (3)$$

where:

- $x_{s,n,i,j}$ is the offset of the control point (i, j) for the n-th object on frame s;
- c - the outlier elimination iteration number;
- $m_{c,s,n}$ is the expected value of the control point offsets on iteration c for the n-th object on frame s;
- $\sigma_{c,s,n}$ is the standard deviation of the control point offsets on iteration c for the n-th object on frame s.

The matrix $korr_0$ is initially initialized as:

$$korro_0 = \overline{stop_{mask}} \circ \overline{st} \quad (4)$$

The expectation value $m_{c, s, n}$ and the standard deviation $\sigma_{c, s, n}$, in this case, are determined by the formulas:

$$m_{c,s,n} = \frac{\sum x_{s,n}^o korr_{c,s,n}}{\sum korr_{c,s,n}} \quad (5)$$

$$\sigma_{c,s,n} = \frac{\sum_{i=1,j=0}^{H,W} (x_{s,n,i,j} * korr_{c,s,n,i,j} - m_{c,s,n,i,j})^2}{\sum korr_{c,s,n}} \quad (6)$$

The expectation value, together with the application of this method of eliminating outliers, is a fairly accurate and easily calculated measure that was used to determine the final offset of the object frame from a set of control point offsets.

3. Optical flow for the “hidden” points are calculated has been grossly inadequate, and the fact of such calculations can often be very difficult to detect automatically.

The optical flow is calculated inadequately if the control point has disappeared behind the occluder or has gone beyond the image. With the gradual overlap of the object, static occluder offset checkpoints before occluders computed optical flow is much less than the real displacement of the object frame. If the occluder has small dimensions relative to the object, these differences do not make a significant contribution to the final frame displacements, being compensated by the displacements of the remaining control points. At the same time, the frame begins to lag behind the object horizontally if a significant part of the columns of the control point placement template is blocked, and vertically if a significant part of the rows is blocked. We solved this problem by adding artificial acceleration to objects horizontally if a certain percentage of the columns of the object’s control point placement pattern is significantly overlapped, and vertically if a certain percentage of rows are significantly overlapped.

The problem of inadequate calculation of the optical flow for control points located near the image boundaries can be solved by tracking how much the width and height of the object frame have decreased due to going beyond the image boundaries relative to the original ones. Then, according to the calculated shares, exclude from consideration the corresponding shares of rows and columns from the corresponding sides of the template for placing control points of the object. In addition, if the frame of an object has reduced its area due to going beyond the image boundary, reduce the coefficient before σ for this object, as well as increase the number of iterations of recalculating the mathematical expectation with clipping by the standard deviation by one.

3.2 Parameters of the Tracker

By applying all the above methods together, you can significantly improve the efficiency of mathematical expectation as an estimate of the displacement of the object frame with a slight increase in the time of its calculation. We have empirically selected the following parameters for our tracker:

- Percentage of standing points from the total number of points except for standing the second or more times to recognize the object as standing – 75%
- number of points in the template for placing control points in width and height – $12*12$
- Number of iterations to recalculate expectation with sigma - 2
- The coefficient in front of σ for drop emissions – 2
- The rate of acceleration of objects horizontally due to the overlapping of a considerable number of columns in the pattern of placement of control points – 1.6
- The rate of acceleration of objects vertically due to the overlapping of a considerable number of lines in the pattern of placement of control points – 1.1
- The share of the points standing in the column/row placement pattern of control points, to make a column/line has been blocked – 0.6
- The proportion of overlapped columns/rows to start the working mechanism of acceleration – 0.2
- The multiplier factor σ for discarding feature points, the part of the framework which went beyond image – 0.85 (Fig. 4)

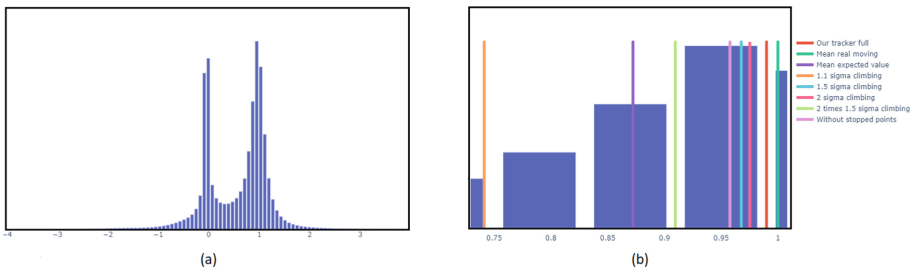


Fig. 4. Estimates of the distribution of control point offsets relative to the real displacement of the object frame: (a) - a histogram of the distributions of control point offsets relative to the real displacement of the object frame; (b) - various estimates of the real displacement of the object frame based on the distributions of control point offsets.

4 Evaluation

The effectiveness of our tracker was evaluated on 3 datasets: training data 2DMOT2015 (Leal-Taixe et al. 2015), training data MOT16 (Milan et al. 2016) and our own dataset consisting of various videos from surveillance cameras of the city of Kazan. Our own dataset is a set of 5 videos from CCTV Kazan city roads. Their description is presented in the Table 2.

Table 2. Description of the dataset for trackers evaluation.

Videofile's name	FPS	Resolution	Length (Duration)	Tracks	Number of bounding boxes	Density	Description of the video
Bulak	30	1280 * 720	1 980 (1:06)	132	34 183	17,3	Footage of busy intersection in the centre of Kazan, clear weather
Bulak_accident	30	1280 * 720	1 483 (0:49)	89	26 926	18,2	Footage of an accident at one of the intersections with rainy weather
Bulak_night	30	1280 * 720	460 (0:15)	24	3 391	7,5	Footage at one of the intersections in the centre of Kazan at night with defocusing of the camera
Kamala	30	1280 * 720	2 010 (1:07)	64	24 935	12,4	Footage where there are a large number of static occluders
Nesmelova	30	1280 * 720	1 285 (0:42)	57	17 230	13,4	Footage where there is a significant difference in light
Total	–	–	7 208	366	106 665	14, 8	–

The sets provided by the authors were used as the detection set for the MOT Challenge sets. For our dataset, we used detections made using yolov4 with a 608x608 window.

For comparison with our tracker, 3 trackers were selected: SORT (Bewley et al. 2016), IOU17 (Bchinski et al. 2017), KCF MOSSE. For SORT and IOU17, implementations provided by the authors of these trackers were used as part of the official publication of the results of the MOT Challenge performed by these trackers. For KCF and MOSSE, implementations from OpenCV 4.4.0 were used.

The main key point is the maximum reduction in the cost of processing video from a single camera, this can be achieved by increasing the intervals between the use of the detector while maintaining an acceptable detection efficiency of objects. Also, in the case of increasing the intervals by reducing the FPS of the video stream, there is an extremely undesirable loss of information. Therefore, one of the points of our study was a comparative analysis of the behaviour of trackers with a decrease in the frequency of use of the detector. The comparison results are shown in the Fig. 5.

It is worth noting that for all trackers except IOU17, the position of the frames for objects between detector applications was determined based on tracker predictions. IOU17 does not have a mechanism for predicting the position of the object frame on the next frame, so the only way available for it to reduce the frequency of using the detector is to reduce the FPS of the video itself, by discarding all frames between detections. For this reason, the evaluation results for this tracker are somewhat inflated compared to the rest.

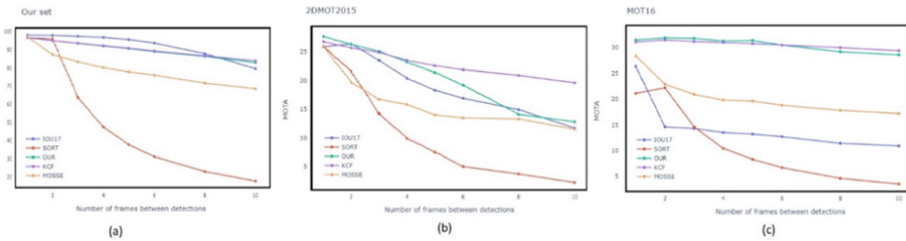


Fig. 5. The dependence of the Multi-Object Tracking Accuracy (MOTA) on the frequency of use of the detector: (a) - for our dataset; (b) - for the 2dMOT2015 dataset; (c) - for the MOT2016 dataset.

The results show that our tracker is slightly behind in efficiency and speed SORT and IOU17, when applying the detector on each frame, greatly surpasses them in effectiveness in reducing the frequency of application of the detector 3 or more times, allowing us to confidently move objects by reducing the frequency of application of the detector to 5 times. When compared with KCF, our tracker shows similar efficiency, while reducing the frequency of use of the detector by 4–5 times.

The effect of reducing the frequency of detector application on the final performance of the detector and tracker bundle was also measured. As a detector, YoloV4 was used with a window of 608x608 pixels on hardware with GPU: NVIDIA GeForce 1050ti 4 GB, CPU: Intel Core i5 8300H and 8 GB RAM. The results are shown in the

Fig. 6. The figure shows that with this configuration of equipment, real-time video processing is achieved only when the frequency of using the detector is reduced by 4 times for SORT and IOU17, 5 times for our tracker by 7–8 times for MOSSE. For KCF, real-time processing cannot be achieved. At the same time, the only tracker that provides sufficient efficiency of tracking objects in real-time is the tracker with the suggested approach.

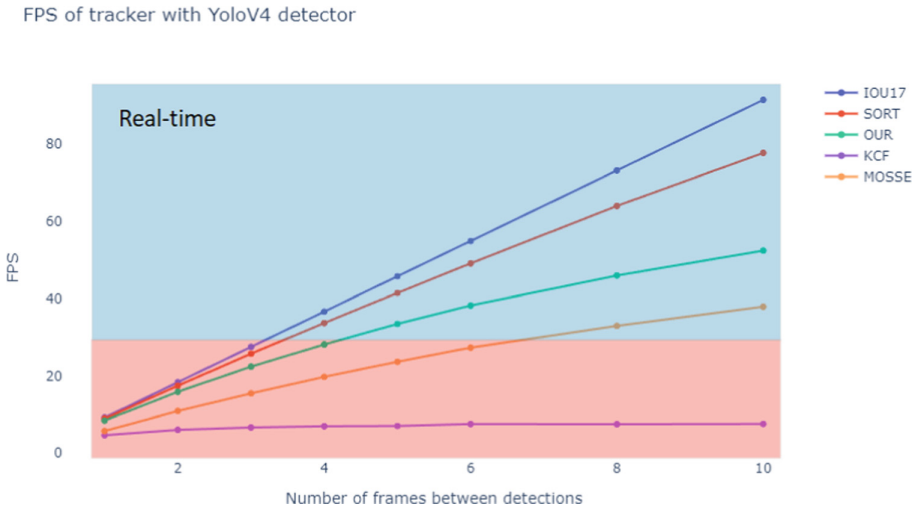


Fig. 6. The dependence of the FPS of the detector and tracker bundles on the decrease in the frequency of use of the detector.

5 Conclusions

The essence of the work was a comparative analysis of existing trackers with the developed tracker algorithm to achieve real-time video processing on medium-power equipment and with acceptable object detection accuracy. The evaluation results showed that the presented algorithm of the tracker, under the mandatory condition of working in real-time, has a MOTA indicator close to the values of the trackers, which have a much lower FPS indicator. Thus, the tracker is balanced between the acceptable accuracy of object detection and the number of processed frames per second. The tracker provides reliable tracking of objects at intervals of up to 5 frames without the use of a detector, and, when working in conjunction with the YoloV4 detector, which detects once every 5 frames on input 608x608 images, produces 25 FPS on an GeForce 1050 Ti GPU and an Intel Core processor i5 8300H. The use of this tracker allows not only to provide tracking of objects in the video in real-time on relatively weak GPU, but also allows to process several video streams on one copy of YoloV4 on powerful ones: the detector performs object detection for one of the cameras, while for the rest it tracks objects without detections. Experiments show that using a similar technique, one copy of YoloV4 can serve up to 4–5 trackers on Nvidia Tesla V100 16 Gb.

References

- Bchinski, E., Eiselein, V., Sikora T.: High-speed tracking-by-detection without using image information. In: 14-th IEEE Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy (2017)
- Bergmann, P., Meinhardt, T., Leal-Taixe, L.: Tracking without Bells and Whistles. IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
- Bewley, A., Ge, Z., Ott, L., Ramox, F., Upcroft B.: Simple online and realtime tracking. In: IEEE International Conference on Image Processing (ICIP), Phoenix, USA (2016)
- Bochkovskiy, A., Wang, C., Liao, H.M.: YOLOv4: optimal speed and accuracy of object detection, [arXiv:2004.10934v1](https://arxiv.org/abs/2004.10934v1) [cs] (2020)
- Bolme, D., Beveridge, J.R., Draper, B., Lui, Y.: Visual object tracking using adaptive correlation filters. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, California (2010)
- Bouguet, J.-Y.: Pyramidal implementation of the affine Lucas Kanade feature tracker description of the algorithm. Intel Corporation, Microprocessor Research Labs (2001)
- Braso, G., Leal-Taixe, L.: Learning a neural solver for multiple object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 6247–6257 (2020)
- Chu, P., Fan, H., Tan, C., Ling, H.: Online multi-object tracking with instance-aware tracker and dynamic model refreshment (2019)
- Fang, K., Xiang, Y., Li, X., Savarese, S.: Recurrent autoregressive networks for online multi-object tracking. In: IEEE Winter Conference on Applications of Computer Vision, Lake Tahoe, USA (2018)
- Karthik, S., Prabhu, A., Gandhi, V.: Simple unsupervised multi-object tracking, [arXiv:2006.02609v1](https://arxiv.org/abs/2006.02609v1) [cs] (2020)
- Leal-Taixe, L., Milan, A., Reid, I., Roth, S., Schindler, K.: MOTChallenge 2015: towards a benchmark for multi-target tracking, [arXiv:1504.01942](https://arxiv.org/abs/1504.01942) [cs] (2015)
- Makhmutova, A., Anikin, I.V., Dagaeva, M.: Object tracking method for videomonitoring in intelligent transport systems. In: Proceedings of International Russian Automation Conference, RusAutoCon 2020, pp. 535–540 (2020)
- Milan, A., Leal-Taixe, L., Reid, I., Roth, S., Scindler, K.: MOT16: a benchmark for multi-object tracking, [arXiv:1603.00831](https://arxiv.org/abs/1603.00831) [cs] (2016)
- Sadeghian, A., Alahi, A., Savarese, S.: Tracking the untrackable: learning to track multiple cues with long-term dependencies. In: IEEE International Conference on Computer Vision (ICCV), Venice, Italy (2017)
- Leal-Taixe, L., Fenzi, M., Kuznetsova, A., Rosenhahn, B., Savarese, S.: Learning an image-based motion context for multiple people tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)