

Practical People Counting Algorithm

Timur Mamedov^{1,2}, Denis Kuplyakov^{1,2} and Anton Konushin^{1,3}

¹Lomonosov Moscow State University, 1, Leninskie Gory, Moscow, 119991, Moscow, Russia

²Video Analysis Technologies, 7, Sculptora Mukhina, 119634, Moscow, Russia

³NRU Higher School of Economics, 11, Pokrovsky Bulvar, 109028, Moscow, Russia

Abstract

In this paper, we consider the problem of people counting in video surveillance. This is an important task in video analysis, because this data can be used for predictive analytics and improvement of customer services, traffic control, etc. The proposed methods are based on object tracking and are able to work on sparse frames, which allows them to work faster and requires minimum computing resources. We use the algorithm from [1] as a baseline, which based on object tracking by head detections. Head tracking in baseline is proved to be more robust and accurate as the heads are less susceptible to occlusions. But this approach has two disadvantages: the height of people is different, which means that people's heads are in different planes, so the raised signal line doesn't look so clear, and also because of this, the accuracy of people counting may decrease. In baseline, this problems were solved using head-to-body linear regression, which had to be retrained for each scene, but this complicates the use of the algorithm for practical purposes. In this paper, we propose a new neural network head-to-body regressor, which allows us to solve the mentioned problems at once. Also in this paper, we use a new visual tracking algorithm that allowed us to speed up our solution. In this work, we introduce two methods – distributed modified baseline algorithm with high people counting accuracy and a solution that can run on a single processor core. Our experimental evaluation showed that the proposed modifications are consistent.

Keywords

Computer Vision, Video Analytics, Tracking, People Counting

1. Introduction

Counting people passing through certain zones of a public infrastructure, such as pedestrian crossings, sidewalks, squares, etc., is a practically important task. Since the solution of this problem has to deal with the processing of large data streams, it is necessary to automate the process of counting people. There are many solutions to this problem, one of them is object tracking. The task of object tracking is to create tracks for each person. Track is uniquely specified with the person and contains his location on every frame where he is visible. In order to count people a signal line is usually specified in the frame (see fig. 1). If the track crosses the signal line, we can say with confidence that the person also crossed it.

But similar solutions have one big drawback – the integration of modern tracking algorithms into real people counting systems is economically unprofitable in practice. This is due to the

GraphiCon 2021: 31st International Conference on Computer Graphics and Vision, September 27–30, 2021, Nizhny Novgorod, Russia

✉ timur.mamedov@graphics.cs.msu.ru (T. Mamedov); denis.kuplyakov@graphics.cs.msu.ru (D. Kuplyakov); anton.konushin@graphics.cs.msu.ru (A. Konushin)

🆔 0000-0001-6554-7988 (T. Mamedov); 0000-0002-2957-3297 (D. Kuplyakov); 0000-0002-6152-0021 (A. Konushin)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).



CEUR Workshop Proceedings (CEUR-WS.org)

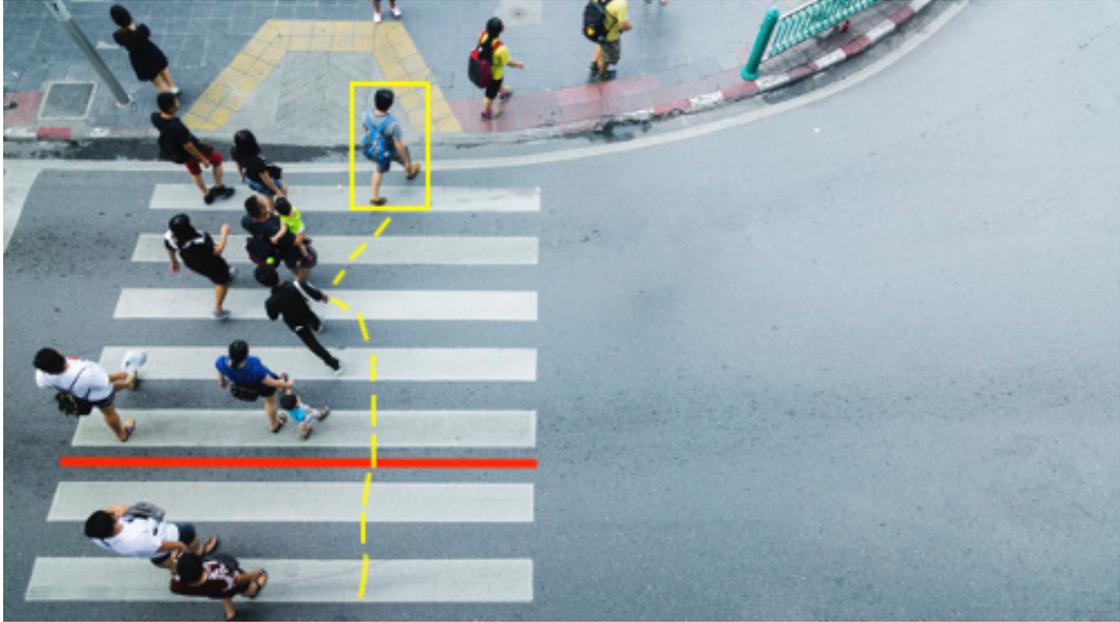


Figure 1: An example of signal line for estimating the load of a crossing.

need to use a large number of GPUs, which are very expensive, especially after the growing popularity of cryptocurrency mining. That's why, in our previous works [1, 2] we introduced distributed algorithms, that allow us effectively count people in real-world scenarios by dividing GPU resources (see fig. 3), taking into account the mass processing of video streams.

However, our latest work [1], which has the best quality in the problem of counting people among our algorithms, has two drawbacks that prevent it from being used in practice:

- this algorithm is based on object tracking by head detections, but the height of people is different, which means that people's heads are in different planes, so the raised signal line doesn't look so clear (see fig. 2), and also because of this, the accuracy of people counting may decrease. Moreover, the automatic method of raising the signal line proposed in the previous work [1] requires calibration for the scene, since information about the average height of a person on a particular scene is used to calculate the height of the signal line;
- the previous problems were solved using head-to-body linear regression in the paper [1]. But this linear regression needs to be retrained for each scene, which also complicates the use of the algorithm in real scenarios.

In this paper, we solve the problems with setting up for the scene described above, and also propose an additional method that is able to count people on a single processor core without GPUs. We introduce a fully automatic people counting algorithms in a video sequence shot by a stationary camera. Algorithms take as input a video stream $\{F_i\}_{i=1}$ of frames captured by stationary camera and signal line that specified by an ordered pair of points (L_a, L_b) on the frame. The output of algorithms is a set of events $\{E_i\}_{i=1}$ that represented by triples of values $E_i = (k_i, r_i, d_i)$, where k_i is the frame index in which the signal line was crossed, r_i specifies



Figure 2: Imagine for yourself where to draw a signal line at the level of the head.

the coordinates of the bounding box and the last value d_i indicates the direction of the signal line intersection.

As it was said earlier, our solutions are an extension of the algorithm from [1]. In this article, we propose the following changes designed to speed up the algorithms and allow them to be used for practical purposes:

- novel neural network body regression by head, which allow us to use our algorithms on any scene without retraining;
- using Staple visual tracking algorithm [3], which allows us to significantly speed up our solution while maintaining the quality of counting;
- using a simpler head detector, which allows us to run our algorithm on a single processor core and maintain an acceptable quality of people counting for practical purposes.

2. Related Work

The modern methods of tracking people are based on tracking through detection. They can be divided into two groups depending on the type of detection used: detection of body [4, 5, 6], detection of head [1, 7]. The first solution is the most popular because there are many datasets and ready-made solutions.

The head-tracking approach is well suited to track people in a crowd: usually video surveillance cameras are installed above the height of the person, where heads in a crowd can be seen better than full bodies. Heads are more resistant to overlapping than bodies. The number of ready-made solutions and data for training is less than for bodies. There are methods that use

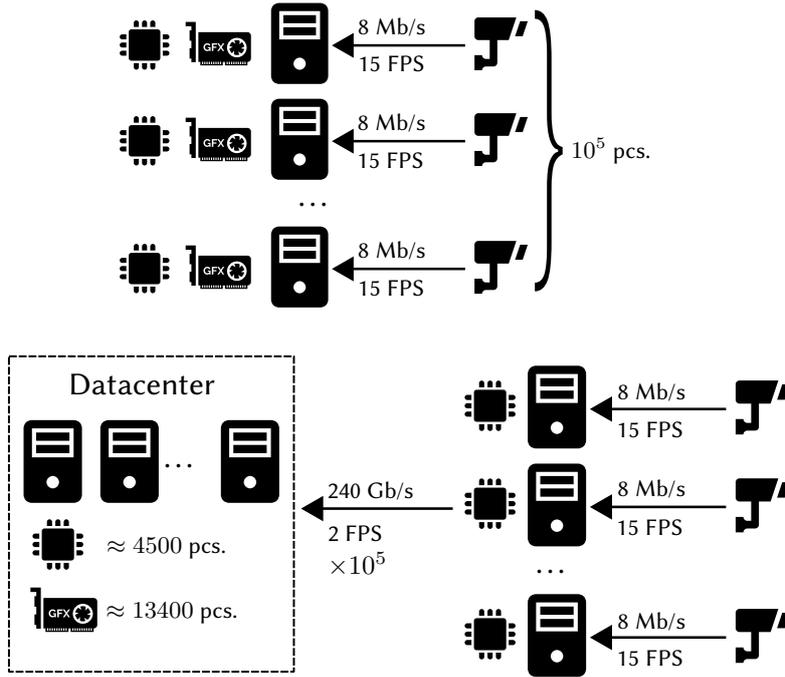


Figure 3: Resources (CPUs, GPUs) required for large-scale video surveillance system (10^5 cameras) for (top) traditional tracking algorithms; (bottom) baseline algorithm with 2 Hz detection frequency.

body parts detectors for tracking [8], key points of human pose [9], combined solutions (body and head) [10], and detector ensembles [11].

After detection we need to bind all the detections to tracks. As in the detection task, there are a lot of methods. The first group of algorithms for creating tracks is greedy algorithms. In most online algorithms tracks are constructed frame by frame, each frame creates a matrix of the cost of matching new detections and existing tracks, then the problem of matching is solved by a greedy algorithm (searching for the maximum in each row/column) [12, 13] or by a Hungarian algorithm [14] [4, 5, 6]. Sometimes MCMC is used to bind all the detections to tracks [7, 15].

Recently, neural networks have been used more often in tracking. For example, in paper [16] authors suggest using detector to obtain new detection by regression of detection on previous frame. However, this method has disadvantages. For example, it is able to work well only at a high frame rate and it also increases the load on the detector.

3. Proposed Method

We use the solution from [1] as the baseline, which is an extension of SORT tracking algorithm [4]. We choose this algorithm as it is capable to work by detection on a sparse set of frames. This significantly reduces the amount of computational resources required for large-scale video surveillance systems (see fig. 3).

Baseline works in online mode and use the Hungarian algorithm [14] to match detections.

To improve results at a low detection rate ASMS visual tracking [17] is used to evaluate a speed of people between frames. The same approach with visual tracking speed estimation is used in [12].

The baseline algorithm consists of the following steps: (1) detection; (2) evaluation of the speed of detections using visual tracking; (3) prediction of the position of tracks by the Kalman filter; (4) matching; (5) extrapolation of the tracks; (6) detection of signal line crossing events.

In this paper, we propose improvements in the following steps in the baseline: detection, evaluation of the speed of detections using visual tracking, detection of signal line crossing events. Proposed improvements are described below.

3.1. Detection

Since heads are seen better in the video and are less prone to occlusions, we continue the idea of using the head detector instead of the body detector for object tracking, proposed in the baseline. Another advantage of the head detector is that neural network body detectors can combine nearby people into one bounding box, which is less frequent for heads. Therefore, in this paper we use the head detector based on SSD [18]. In this work, we use two backbones for the head detector – ResNet50 [19] for distributed modified baseline algorithm with high people counting accuracy and MobileNet0.5 [20] for solution that can run on a single processor core. The detector were trained on CrowdHuman [21] public dataset and on the dataset collected by Video Analysis Technologies.

3.2. Visual Tracking

As it was said earlier, ASMS visual tracking is used in the baseline. In this paper, we use Staple visual tracking algorithm [3], which allows us to significantly speed up our solution while maintaining the quality of counting. Staple consist of two models working in parallel. The first model is invariant to the change of lighting, but depends on the change in the shape of the object, and the second model is invariant to the transformations of the object, but depends on the lighting. Therefore, using two models together, it is possible to achieve the best result.

3.3. Signal Line Crossing

The baseline is based on object tracking by head detections. Head tracking in baseline is proved to be more robust and accurate as the heads are less susceptible to occlusions. But this approach has two disadvantages: the height of people is different, which means that people’s heads are in different planes, so the raised signal line doesn’t look so clear, and also because of this, the accuracy of people counting may decrease. In baseline, this problems were solved using head-to-body linear regression (because people’s feet are always in the same plane), which had to be retrained for each scene, but this complicates the use of the algorithm for practical purposes. In this paper, we propose a new neural network head-to-body regressor, which allows us to solve the mentioned problems at once. Our neural network regressor must be trained once and can be used for any scenes.

The proposed head-to-body regressor consist of two steps:

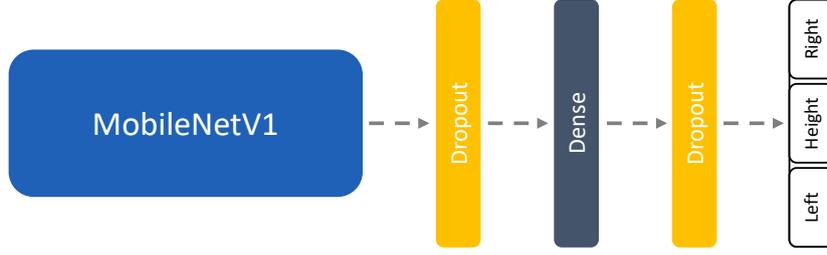


Figure 4: The architecture of the proposed neural network for head-to-body regression. We use MobileNetV1 [20] as the backbone of our neural network.

1. using heuristics, we find the approximate position of the body;
2. using neural network regression, we clarify the position of the body.

The First Step In our heuristics we are using the following fact from anatomy: the width of a person’s body is on average three times the width of a person’s head, and the height of a person’s body is on average eight times the height of a person’s head. Experimentally, these proportions were slightly corrected:

$$body_{left} = head_{left} - head_{width}, \quad (1)$$

$$body_{top} = head_{top}, \quad (2)$$

$$body_{width} = 3 \cdot head_{width}, \quad (3)$$

$$body_{height} = 8.5 \cdot head_{height}, \quad (4)$$

where $(head_{left}, head_{top}, head_{width}, head_{height})$ – the coordinates of the head bounding box and $(body_{left}, body_{top}, body_{width}, body_{height})$ – the coordinates of the body bounding box.

The Second Step At the second step we specify the exact position of the body bounding box, obtained in the previous step. To do this, we developed neural network to regress three coordinates: the left and right extreme points of the human body ($body_{regLeft}$ and $body_{regRight}$, respectively) and the height $body_{regHeight}$ of the human body. The fig. 4 shows the architecture of the proposed neural network for head-to-body regressor. Our regression neural network has a simple architecture, because we need our algorithm for counting people to work in real time. Despite the simple architecture, the proposed neural network has a good regression quality. As a result, we have the final coordinates of the body bounding box: $(body_{regLeft}, body_{top}, body_{regWidth}, body_{regHeight})$, where $body_{regWidth} = body_{regRight} - body_{regLeft}$.

In our solution, we use head detection for object tracking, but we implement the detection of signal line crossing events using body detection, regressed by the proposed head-to-body neural network regressor. That is, each detection of the head is associated with the detection of the body. Also, when we add a new body detection to the track, we adjust the width and height

of the new body detection (due to occlusions, we may have regression errors that can lead to false crossings of the signal line) as follows:

$$bodyNew_{width} = \alpha \cdot bodyNew_{width} + (1 - \alpha) \cdot bodyPrev_{width}, \quad (5)$$

$$bodyNew_{height} = \alpha \cdot bodyNew_{height} + (1 - \alpha) \cdot bodyPrev_{height}, \quad (6)$$

where $bodyNew_{width}$, $bodyNew_{height}$ – the width and height of the body bounding box that we are adding, $bodyPrev_{width}$, $bodyPrev_{height}$ – the width and height of the last body bounding box in the track, and α – is a specially selected constant.

4. Experiments

4.1. Datasets

Head-to-Body Regression To train our regression neural network (see section 3.3) we used videos from 2DMOT2015 [22], MOT17 [23] challenges and data, collected by Video Analysis Technologies company. To obtain the data for training the neural network model from the video (frames are considered every 2 seconds), the following strategy was used:

1. using the head detector (see section 3.1), head detections were obtained for each frame;
2. using the linear regression from [1], body detections were obtained for each frame;
3. using our heuristic (see section 3.3), the approximate positions of the bodies were found for each frame. This data was used to make crops for training our neural network head-to-body regressor;
4. using body detections corresponding to the head detections, the left and right extreme points of the human body and the height of the human body were found for each frame. This data was used to train our neural network head-to-body regressor.

Counting People Algorithms For an experimental evaluation of our algorithms we need datasets filmed by static camera. Video sequences should be long enough to evaluate people counting quality. Most of the public datasets including popular MOTChallenge dataset [24] have short videos or filmed by moving camera. So we used 19 videos from the collection of the Video Analysis Technologies company and the Towncentre dataset [25] to test our algorithms. For all videos signal lines were manually drawn at ground level. The table 1 provides detailed information about each test video.

4.2. Metrics

As a quality metric we use the average error of counting the number of intersections (events) [2]. The resulting events can include both true and false ones. The false events have no correspondences in the reference labeling. We say that an event E_i in the input set of data matches the event \widehat{E}_i in the reference labeling if they correspond to the same person crossing the signal line at the same time. We match all events as described in the paper [2].

After the events have been matched we divide videos to segments with 10 reference events and calculate the following characteristics on them:

Table 1

Videos that we used to test the algorithms.

| Video name | Duration | Format | Number of events |
|------------------|----------|--------------|------------------|
| SignalineBase/02 | 00:17:40 | 1280x720@25 | 254 |
| SignalineBase/03 | 00:14:40 | 1280x720@25 | 134 |
| SignalineBase/04 | 00:05:20 | 1280x720@25 | 69 |
| SignalineBase/05 | 00:11:40 | 1280x720@25 | 172 |
| SignalineBase/06 | 00:12:00 | 1280x720@25 | 219 |
| SignalineBase/07 | 00:08:29 | 800x450@25 | 124 |
| SignalineBase/10 | 00:38:59 | 704x576@25 | 120 |
| SignalineBase/11 | 00:22:48 | 704x576@25 | 91 |
| SignalineBase/13 | 00:05:01 | 720x576@12 | 9 |
| SignalineBase/14 | 00:05:23 | 720x576@12 | 17 |
| SignalineBase/17 | 00:16:42 | 1280x1280@20 | 48 |
| SignalineBase/18 | 00:16:26 | 1280x1000@25 | 180 |
| SignalineBase/20 | 00:15:00 | 1024x768@20 | 144 |
| SignalineBase/21 | 00:19:40 | 800x600@25 | 149 |
| SignalineBase/26 | 00:10:29 | 1280x720@15 | 90 |
| SignalineBase/28 | 00:11:40 | 1280x960@25 | 72 |
| SignalineBase/29 | 00:40:00 | 704x576@25 | 159 |
| SignalineBase/31 | 00:29:10 | 1920x1080@25 | 381 |
| Towncentre | 00:03:00 | 1920x1080@25 | 162 |

- GT_{seg} is the number of reference events on the segment;
- FP_{seg} is the number of unmatched events from the algorithm on the segment;
- FN_{seg} is the number of unmatched events from the reference events on the segment;
- $E_{seg} = \frac{FP_{seg} - FN_{seg}}{GT_{seg}}$ is an error on the segment.

Then final error is calculated as $E = \sum_{i=1}^N \frac{E_{seg}}{N}$, where N is the number of the segments.

4.3. Experimental Results

In this section, we present experimental results for the proposed algorithms. We consider 3 types of experiments:

- **With Body Regression** – in this experiments, we used baseline from [1] and replaced the head-to-body linear regression to our neural network head-to-body regression (see section 3.3);
- **With Body Regression and Staple** – in this experiments, we used modification from the "With Body Regression" experiments and replaced ASMS visual tracking with Staple (see section 3.2);
- **Single Processor Core Algorithm** – in this experiments, we used modifications from previous experiments and used lightweight head detector with MobileNet0.5 backbone (see section 3.1).

Table 2

Experimental results for the proposed algorithm.

| Algorithm / Detection frequency | 5 Hz | 3 Hz | 2 Hz |
|---------------------------------|------|------|------|
| SORT [4] | 8.3 | 10.8 | 17.3 |
| Old Baseline [2] | 7.5 | 7.5 | 7.5 |
| Baseline [1] | 4.1 | 4.1 | 4.3 |
| With Body Regression | 4.8 | 4.7 | 5.1 |
| With Body Regression and Staple | 5.0 | 4.6 | 5.1 |
| Single Processor Core Algorithm | 8.9 | 9.3 | 9.1 |

The table 2 provides detailed information on the results of each of the above experiments.

The experimental evaluation shows that our distributed modified baseline algorithm has a high people counting quality, which significantly exceeds the results of our old algorithm from [2] and slightly lags behind the results of the algorithm from [1]. However, the proposed algorithm can be used for practical purposes, since our modifications allow us to use it without configuring it for a specific scene.

In addition, the table 2 shows that our solution, which can run on a single processor core, has a quality acceptable for practical purposes. Moreover, our lightweight algorithm bypasses the algorithm based on the classical SORT [4] tracking in terms of the quality of people counting.

4.4. Speed Estimation

Since our algorithm consists of several stages, its operating time is equal to the total operating time of its following components: detector, visual tracking and head-to-body regression. Our lightweight head detector runs ≈ 250 milliseconds on a Full HD frame (see section 3.1). Staple visual tracking [3] (see section 3.2) runs ≈ 5 milliseconds for a single detection (ASMS visual tracking [17], used in baseline [1] runs ≈ 20 milliseconds for a single detection). And the proposed neural network head-to-body regressor runs ≈ 130 milliseconds for 10 crops.

Given that the rest of the calculations are not so expensive, our algorithm, running on a single processor core, is able to work with a frequency of 2 Hz for 10 people per frame (≈ 500 milliseconds). All speed measurements were made on a single core of the Intel Core i5-9400 processor.

5. Conclusion

In this paper, we focused on the practical applicability of people counting algorithms and introduced two algorithms: distributed modified baseline algorithm with high people counting accuracy and a solution that can run on a single processor core. Also, in this work, we propose a new neural network head-to-body regressor, which allows us to solve drawbacks of the baseline from [1], related to configuring the algorithm for a specific scene. This modification allows us to use our algorithms for practical purposes.

References

- [1] D. Kuplyakov, Y. Geraskin, T. Mamedov, A. Konushin, A distributed tracking algorithm for counting people in video by head detection, in: Proceedings of the 30th International Conference on Computer Graphics and Machine Vision, volume 2744 of *CEUR Workshop Proceedings*, M. Jeusfeld c/o Redaktion Sun SITE, Informatik V, RWTH Aachen, 2020, pp. 1–12. doi:10.51130/graphicon-2020-2-3-26.
- [2] D. A. Kuplyakov, E. V. Shalnov, V. S. Konushin, A. S. Konushin, A Distributed Tracking Algorithm for Counting People in Video, *Programming and Computer Software* 45 (2019) 163–170. doi:10.1134/S0361768819040042.
- [3] L. Bertinetto, J. Valmadre, S. Golodetz, O. Miksik, P. Torr, Staple: Complementary learners for real-time tracking, 2016. doi:10.1109/CVPR.2016.156.
- [4] A. Bewley, Z. Ge, L. Ott, F. Ramos, B. Upcroft, Simple online and realtime tracking, 2016 IEEE International Conference on Image Processing (ICIP) (2016). URL: <http://dx.doi.org/10.1109/ICIP.2016.7533003>. doi:10.1109/icip.2016.7533003.
- [5] N. Wojke, A. Bewley, D. Paulus, Simple online and realtime tracking with a deep association metric, 2017. arXiv:1703.07402.
- [6] F. Yu, W. Li, Q. Li, Y. Liu, X. Shi, J. Yan, Poi: Multiple object tracking with high performance detection and appearance feature, in: European Conference on Computer Vision, Springer, 2016, pp. 36–42.
- [7] D. Kuplyakov, E. Shalnov, A. Konushin, Further improvement on an mcmc-based video tracking algorithm, in: Proceedings of the 26th International Conference on Computer Graphics and Vision GraphiCon'2016, GraphiCon, 2016, p. 440–444.
- [8] G. Shu, A. Dehghan, O. Oreifej, E. Hand, M. Shah, Part-based multiple-person tracking with partial occlusion handling, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 1815–1821. doi:10.1109/CVPR.2012.6247879.
- [9] Y. Xiu, J. Li, H. Wang, Y. Fang, C. Lu, Pose flow: Efficient online pose tracking, CoRR abs/1802.00977 (2018). URL: <http://arxiv.org/abs/1802.00977>. arXiv:1802.00977.
- [10] R. Henschel, L. Leal-Taixé, D. Cremers, B. Rosenhahn, Improvements to frank-wolfe optimization for multi-detector multi-object tracking, CoRR abs/1705.08314 (2017). URL: <http://arxiv.org/abs/1705.08314>. arXiv:1705.08314.
- [11] R. Cobos, J. Hernandez, A. G. Abad, A fast multi-object tracking system using an object detector ensemble, CoRR abs/1908.04349 (2019). URL: <http://arxiv.org/abs/1908.04349>. arXiv:1908.04349.
- [12] E. Bochinski, T. Senst, T. Sikora, Extending iou based multi-object tracking by visual information, in: 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2018, pp. 1–6. doi:10.1109/AVSS.2018.8639144.
- [13] E. Bochinski, V. Eiselein, T. Sikora, High-speed tracking-by-detection without using image information, in: 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), 2017, pp. 1–6. doi:10.1109/AVSS.2017.8078516.
- [14] H. W. Kuhn, The hungarian method for the assignment problem, *Naval Research Logistics Quarterly* 2 (1955) 83–97. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109>. doi:<https://doi.org/10.1002/nav.3800020109>. arXiv:<https://onlinelibrary.wiley.com/doi/pdf/10.1002/nav.3800020109>.

- [15] D. Kuplyakov, E. Shalnov, A. Konushin, Markov chain monte carlo based video tracking algorithm, *Programming and Computer Software* 43 (2017) 224–229.
- [16] P. Bergmann, T. Meinhardt, L. Leal-Taixé, Tracking without bells and whistles, *CoRR abs/1903.05625* (2019). URL: <http://arxiv.org/abs/1903.05625>. arXiv:1903.05625.
- [17] T. Vojir, J. Noskova, J. Matas, Robust scale-adaptive mean-shift for tracking, in: J.-K. Kämäräinen, M. Koskela (Eds.), *Image Analysis*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 652–663.
- [18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A. C. Berg, Ssd: Single shot multibox detector, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), *Computer Vision – ECCV 2016*, Springer International Publishing, Cham, 2016, pp. 21–37.
- [19] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, 2015. arXiv:1512.03385.
- [20] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. arXiv:1704.04861.
- [21] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, J. Sun, Crowdhuman: A benchmark for detecting human in a crowd, 2018. arXiv:1805.00123.
- [22] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, K. Schindler, MOTChallenge 2015: Towards a benchmark for multi-target tracking, arXiv:1504.01942 [cs] (2015). URL: <http://arxiv.org/abs/1504.01942>, arXiv: 1504.01942.
- [23] A. Milan, L. Leal-Taixé, I. Reid, S. Roth, K. Schindler, MOT16: A benchmark for multi-object tracking, arXiv:1603.00831 [cs] (2016). URL: <http://arxiv.org/abs/1603.00831>, arXiv: 1603.00831.
- [24] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, L. Leal-Taixé, Mot20: A benchmark for multi object tracking in crowded scenes, 2020. arXiv:2003.09003.
- [25] B. Benfold, I. Reid, Stable multi-target tracking in real-time surveillance video, in: *CVPR 2011*, 2011, pp. 3457–3464. doi:10.1109/CVPR.2011.5995667.