An Hybrid CPU-GPU Parallel Multi-tracking Framework for Long-Term Video Sequences

Juan P. D'amato[1,2(✉)], Leonardo Dominguez[1,2], Franco Stramana[1,3], Aldo Rubiales[1,3], and Alejandro Perez[1,3]

[1] PLADEMA, Universidad Nacional del Centro de la Provincia de Buenos Aires, Tandil, Argentina
[2] National Council Scientific Technical Research, CONICET, Buenos Aires, Argentina
[3] Comisión de Investigaciones Científicas, CICPBA, Buenos Aires, Argentina

Abstract. The automatic evaluation of video content is today one of the biggest challenges in computer Vision. When the purpose is to work with static surveillance cameras, where most of the time the scenes do not change, a full Convolutional Network (CNN) approach seems to require too much CPU effort, specially when the objects are slightly moving between different frames. On the other side, visual tracking has seen great recent advances in either speed or accuracy but still remain scarce when have to deal with long videos where new objects constantly come into the scene and others disappear.

In this paper, we present a parallelization scheme to handle multiple instances of object tracking. The main purpose is reduce overall processing time. The idea is to use already pre-trained CNNs for discovering objects and a parallel multi-tracker for following them, using both CPU and GPU devices.

Our multi-tracker framework consists of three main components, a movement detector, an object classification and a tracker. The detector is used as an initialization for trackers. When there are plenty of objects in the scene, the other two components are incorporated for reducing CPU effort. The first one is a *scheduler* than prioritizes tracking those objects that seems more relevant than the others. The second one, is a GPU memory handler, that lets adapt the framework to different hardware configuration. We evaluate this framework in different cases and cameras configurations, reaching reasonable speed-up and confidence.

Keywords: Video processing · GPGPU · Object tracking · CNN

1 Introduction

Visual object tracking plays a crucial role in computer vision and a critical role in visual surveillance [1]. Despite great successes in recent decades, robust real-time tracking is still a challenge, as in visual tracking still exists many factors including object deformation, occlusion, rotation, illumination change, pose variation,

J. C. Figueroa-García et al. (Eds.): WEA 2021, CCIS 1431, pp. 263–274, 2021.
https://doi.org/10.1007/978-3-030-86702-7_23

among others. The new algorithms trend toward improving tracking accuracy using deep learning-based techniques (e.g., [2]).

Such algorithms, unfortunately, often suffer from high computational effort and hardly run in real-time or requires a lot of power consumption, situation that not seems suitable for sustainable smart cities. Researchers have been proposing efficient visual trackers like [3,4], represented by the series of algorithms based on correlation filters. While easily can run at real-time, these trackers are usually less robust than deep learning-based approaches. In this context, it is a likely to have the better of both worlds, having a trade-off between speed and accuracy.

Additionally, today for smart cities, a bunch of new different hardware is available, having a reasonable CPU at less than 50 dollars, compared to a high performance processing server that could cost at least 2,000 dollars. This cheap equipment are very interest and help to think in a massive distributed processing network, as propose in [5], but have different limitations that have to be over passed in order to be useful for Computer Vision.

Finally, other challenges arise in surveillance systems:

1. In long video scenes, new objects constantly come into the image and other ones leave. In these cases, it is essential to have algorithms that handle when new objects are discovered.
2. In general, many different class of objects are present (people, vehicles, animals) in every image. Each one, moves at different speed, depending its class. In general, people moves smoothly, in contrast, vehicles could move faster. These cases typically require to be handled by different tracking strategies, adapted to movement speed.
3. Convolutional Networks (CNNs) are not perfect. Even tough they can detect with high precision different objects at different scales, they suffer when objects are constantly displacing and changing its orientation.
4. GPGPU capabilities limitations. Multi-thread and GPGPUs computing has already benefited computer vision system. There are plenty of CNN implementations that uses too much vRAM, that restricts its portability against different platforms (ARM, intel). The idea here is to use a multi-platform approach based on OpenCL.

The main purpose in this work is to present a multi-tracker framework that could handle complex scenes with many different objects in real time. This framework takes long-video sequences from surveillance systems, registering moving objects on a distributed platform. With some similar ideas presented in [6], the idea is to reach a high frame rate (above 20FPS) using both parallel CPUs and GPUs. The proposed framework has three main components: the detection module, the scheduler and the trackers. The detector itself is based on CNNs framework that will identify all potential objects in an image with their corresponding class. Despite obtaining excellent performance, the application of such algorithms is severely restricted by the high cost of extracting features in each frame. To deal with such limitation, the idea is to apply a fast tracker for each new detection, that updates objects position and confirms its prediction with a new detection. When the amount of objects is rather high (more than 20), this

task could be even more expensive than the detection itself. For that reason, a task scheduler is required to distribute the available computation power.

As many times the scenes remain unchanged (there are no movement), a fast Background Subtracion Algorithm (BGS) could also be used as an activation or deactivation of the pipeline. All these parts will interact through synchronized images queues. To process all these tasks, both the CPU and GPU of a computer will be exploited. The framework is illustrated in Fig. 1.

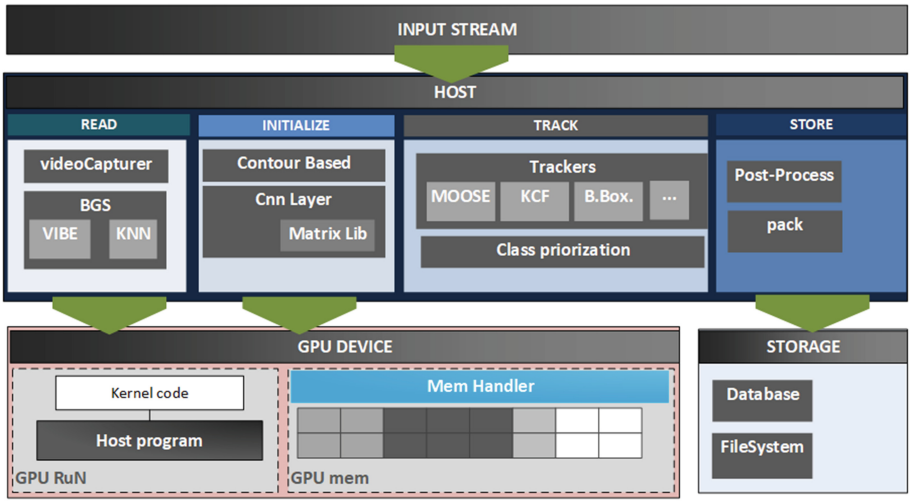

Fig. 1. Architecture of the framework.

2 Related Work

The techniques of detection of events used in Video-Surveillance systems are mainly based on quickly discriminating of movement from a fixed video camera. Research works as [7], describe the most common algorithms to detect and track objects. In [8], there is a special comparison between the different algorithms of basic detection, concluding that their combination may be absolutely useful to diminish false positive rates; while keeping the time and rate of true positives. The background subtraction commonly used classify shadows as a part of the objects, what alters both size and shape of the objects, affecting consequently the efficacy of such algorithms [9]. Also, it hinders some other procedures which need the result of the detection of objects, such as classification or tracking of the trajectory and also, analysis of certain behavior: loitering, vandalism, traffic law braking, among others. This problem also affects those techniques based on characteristics [10].

By contrast, trackers usually are based on searching similar regions respect to the target. For this task, various object appearance modeling approaches have been used such as [11], other inspired by deep features in visual recognition like

[12,13]. Recently, correlation filters have drawn increasing attention in visual tracking. [14] propose a filter tracker as sum of squared error (MOSSE), achieving a high frame rate but punishing accuracy. [4] introduce kernel space into correlation filter that result in the well-known Kernelized Correlation Filters (KCF) tracker.

In all these cases, the idea is to first try to detect the objects in the scene based on a inference algorithm and some time later confirm that the same objects are still present. This idea is not new in tracking. A good example is presented in [2], in which tracking results are validated per frame to decide which detection shall progress.

The proposal in our work is to combine tracking algorithms with re-identification ones, based on neural networks. As it is possible that many objects are present in the scene, and tracking algorithms could be computationally expensive, the idea is to include some metric that could dynamically helps to choose the best tracking parametrization, which means, choosing an algorithm and tracking frequency. This combination can helps to reach high performance, while mostly all the scene is constantly evaluated with good accuracy.

In [15] the formulate different tracking strategies in different hardware configuration. They evaluate multiple algorithms application of such trackers for cars and propose some combination that could run according to the capabilities. With a similar purpose, our solution takes into account the objects that should be detected and choose dynamically the most convenient tracking algorithm.

2.1 GPU Management

To reduce memory consumption in single-GPU detection, there are some existing ideas and work like Leveraging host RAM as a backup to extend GPU device memory. CUDA, for example, enables Unified Memory with Page Migration Engine so that unified Memory is not limited by the size of GPU memory. Of course, this method can bring a severe performance loss, as GPUs are more focused on transferring data than processing it.

Another way is using a run-time memory manager to virtualize the memory offloading the output of each layer and pre fetching it when necessary, which can only be applied to the layer-wise CNN models, not to sequence models. A Memory-efficient RNN is proposed by [16]. However, for those sequence models with attention mechanism, the attention layer actually requires much more memory space than LSTM/GRU.

There are also some other optimization methods, such as Memory-Efficient DenseNets [17] with significantly reducing memory consumption. However, this is applicable for special cases. In this paper, a general based approach "swap-out/in" handler is proposed, which is targeted for any kind of neural network. To pursue more memory optimizations, memory-efficient attention algorithm are designed.

3 Proposal

As was named at the beginning, the proposed framework is composed by three main components:

1. an identifier: identifier tries to identify objects present in the frame with their corresponding class. Detected objects are used as input to the trackers.
2. a set of trackers: responsible of locating and updating the position of each object in each frame.
3. an (adaptable) scheduler: responsible for distributing computation priority among available trackers

In this work, it is considered that tracking is much faster than identifying and are not fully dependent, so both components could work asynchronously. The identifier tries to find a set of potential objects from an image and a tracker is created and assigned to each object. Such mechanism triggers a temporal tracking drift that is corrected during processing. Also, each tracked object is independent of the other, so a second level of parallelism could be exploited.

To improve performance, several separated threads are allocated for the different software components, like video reading, tracking, classifying and storing. The components are connected each other through several synchronized queues.

An input queue is first used to read frames from a file or stream. A frame is chosen from this queue, and passed through the next processing one. Once this frame is processed and objects detected, both structures are passed to the tracker component that follows up every object until they leave the scene. Finally, but also important is storing. The fourth thread is responsible for taking all objects and stored someway, either in the local file system or in remote databases. For dealing these cases, a storage task runs asynchronously collecting all data obtained from the multi-tracker. Objects are encoded in a format proposed in [18].

As it is expected to be a very flexible framework, some important designing marks should be taken in consideration

1. The inference network and tracking algorithms should be selected according to application (it is no the same a traffic video that a retailing one) and available computational resources.
2. The trackers should be robust to small light changes and to occlusion.
3. The objects detection step can be implemented in many ways, from fast object moving segmentation to CNN inference. When the scenes are not crowded, a simple movement segmentation should be more effective.

Each item is detailed in the following sections.

3.1 Initialization and Tracking

Initialization task has the responsibility to take a frame from an input queue and return a set of potential objects with their corresponding location or bounding

box. Objects could be classified in that moment (determining their class, as vehicle, person, animal among other) using a CNN or just identifying them as a set of connected components in the foreground mask and deferring the classification step once they are stored. These both strategies has their corresponding advantages and disadvantages that will be discussed later.

In any case, the input for the trackers are a set of bounding box indicating the objects that have to be followed through the consequent frames until they leave the scene. For each frame, each tracker must update their corresponding object state and compute the new location. If during this step the tracking fails, it is possible that the object was occluded or has to be removed. In both cases, a parameter indicating that the object is "alive" is decreased. Once this parameter reach to ZERO, it should be considered as "dead" and moved to the store task.

Contour Based Initialization. In a scene taken from a static surveillance camera, static objects has non-value. For that reason, a background subtraction algorithm could easily segments pixels in movement from the static ones. Once these pixels are obtained, they are merged into "blobs"; grouping pixels that are in contact with their neighbours. A contour based algorithm to be used for this step. If blobs are too small, they could be discarded.

CNN Initialization. Using a trained network, a set of objects are extracted from the input image. Very depth networks, such as Yolo or Resnet, are more precise and could detect almost all objects in the scene (both static or movable ones). The CNN predicts the location and bounding box of the objects with their corresponding class, that is used as input for the next stage.

3.2 GPU Management

With the purpose of having an adaptable multi-platform framework, we develop a memory handler that dynamically allocates and organizes the amount of available memory. Every time a GPU operation requires memory, the Handler checks if it is already allocated. In case not, a copy from host to device is carried out. In other case, the handler returns the actual buffer. If the amount of memory is not enough, the handler removes those buffers that are not used in that moment. After several operations, the handler could clear less used buffers and keeps the most used ones.

3.3 Multi-tracking Scheduling and Update

Every frame, the list of active tracker should evaluate if the object is moving and update their position. As it was named, updating each tracker has a relative high computation effort. For instance, updating the position of a KCF tracker, takes about 20 ms for each frame for each. When there are many objects in the scene (above 10), the total computation effort is not enough to attend to each tracker each frame.

On the other side, every object is not moving at same speed or not moving at all. Even more, due to perspective effect, further objects seems to move slower in pixels space. For those reasons, we consider that is not strictly necessary to update the objects' position on every frame but if a tracker is not updated thoroughly, the observed object could be lost. To manage these situation, the movement of the objects is defined by three terms:

- the Object class (vehicles move faster than persons)
- the position in the space (further objects seems to move slower)
- its own behaviour (object could be stop or in movement).

To avoid tracker starvation, a fourth parameter is included, modeling the time since last update. With these information, we propose a criteria for distributing the available computation capabilities among the active trackers: the higher the priority, the more probable to be updated. The criteria is described in 1:

$$Priority(O) = \alpha * class(O) + \beta * pY(O) + \gamma * rS(O, MK) + \epsilon * timeSinceU(O) \quad (1)$$

Where pY is a function dependent of the camera perspective. The farther the object is, the slower its movement. $Class$ is a function that gives a score to each class. rS indicates if the object is moving or not. Using the movement mask MK and the object bounding box R, with 2

$$moving(O, MV) = \begin{cases} 1 : \text{n}onBlank(MV[R]) \geq \text{thresholdMove} \\ 0 : \text{in}OtherCase \end{cases} \quad (2)$$

For handling multi-tracking, we implemented a tracker scheduler. This scheduler computes the $Priority$ of each pair tracker-object and sort them in decrease order. Using a backpack strategy, those that has the higher priorities are first assigned to be update. If there is available computation time, the next trackers are included.

To avoid starving objects, the more time the tracker awaits to update, $timeSinceU$, the higher the priority. On each new frame, the scheduler computes the priority and updates only those trackers that have a higher one. With these scheme, objects that are not moving but were detected during the initialization, have very low priority and should not be tracked ever.

On each frame, it is checked whether there are new objects or if they are the same that are being tracked from previous frames. To support this, a main history list of objects is kept. After a new detection, the list is updated. If there are new objects, a tracker is assigned to each one. On the other side, trackers update the position. If objects could not be tracked during last frames, they are removed from the list. The algorithm is presented here:

4 Experiments

Our framework is implemented in C++ and it can runs on several hardware configurations, supporting Intel, NVidia and AMD GPUs. As each GPU memory allocation request in the framework requires to pass through the Handler,

Algorithm 1. Multi-tracking update

```
procedure UpdateTrackersState(newObjects, histObjects)
    for d ∈ new do
        t ← initTracker(d.box)
        keep ← push(t)
    for d ∈ histObjects do
        if d.life ≤ 0 then
            keep ← remove(d)
    return keep
procedure Update                          ▷ this runs on a synchronized thread
    History ← new()
    while true do
        Objects, frame ← dequeue(queueTrack)
        active ← UpdateTrackersState(Objects, History)
        for frame.index ≤ Window do                          ▷ Process next frames
            computePriorities(active)
            while t ← active do
                if t.priority > threshold then
                    couldUpdate ← t− > update(R, frame) ;
                    if couldUpdate = True then
                        t.setNewPosition(t.objectRectangle)
                    else                  ▷ Probably, the object has left the scene
                        d− > life− = 1
```

we implemented an own OpenCL version of YOLO and VIBE BGS. Other third party libraries, like the tracking one, that could also used GPUs are not considered in this work.

For evaluating the capabilities of this framework, several t studies that were carried up:

1. Memory management configurations: different memory limitations are studied. As the idea is to have a deeper network, only those CNNs that used more of memory are evaluated.
2. Priorization parameters on long sequences. For evaluating how the framework adapts to different situations, different parameters are considered, such as the different initialization schemes.

We have selected the pre-trained Yolo v4 networks for inference [19]. Other traffic objects segmentation networks are also useful.

4.1 Considerations

As in [6], different classification interval initialization interval I may affect both the accuracy and efficiency. A smaller I means more frequent verification, which requires more computation and thus degrades the efficiency. A larger I, on the contrary, may cost less computation but could make loosing targets while their appearance change quickly. The first initialization interval was set to 20.

At the same time, in the priority metric proposed, the component based on object position varies mainly on perspective. As each scene is observed in different point of view, the function should be rectified before using it. Before applying the algorithm, the camera has to be calibrated. For this purpose, the class and normalized Y position of each object in the image is evaluated. For several frames, the size and location of the objects is obtained and represented, as can be shown in Fig. 2. After this study is carried for each object class, it was determined that the slope of the mean line is used as input in the Eq. 1.

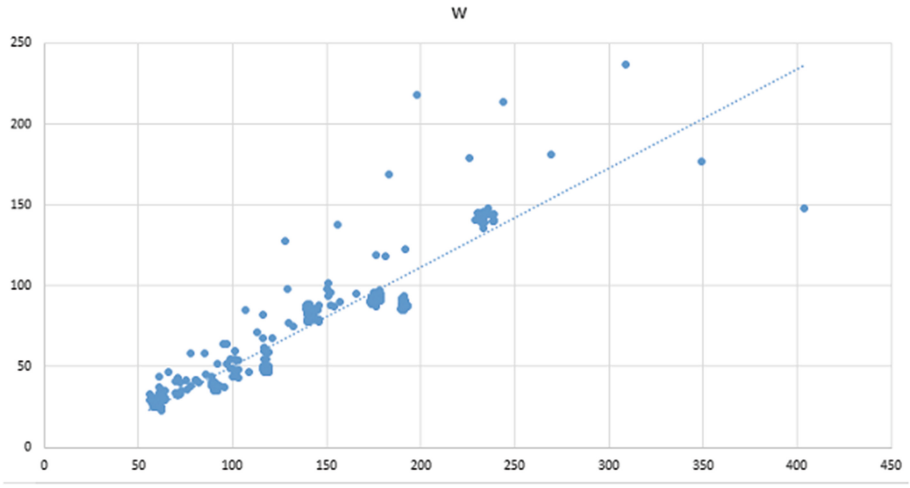

Fig. 2. Car speed variation respect to Y coordinate.

Respect to objects trackers, it was first consider MIL [20] and KCF [4]. Compared to MIL, KCF is more efficient while less accurate in short time interval. Though KCF runs faster than MIL, it performs less accurately in short time, hence significantly increasing the amount of lost objects. By contrast, MIL is more accurate in short time, and finally leads to efficiency in computation.

4.2 Scheduling Parameters

In this framework, it is required to be efficient and accurate most of the time, even in hard cases. To show the effects of managing the multi-tracks, different scenes with a particular configuration were evaluated. In such cases, the scheduling algorithm should give more importance to vehicles rather than people, also considering perspective correction. For this purpose, we evaluate several frames from different moments taken on a street corner or routes, either during night or early morning. The used cases are shown in Fig. 3.

Fig. 3. Different scenes.

In general, scenes with too few elements are named "simple". On the other side, city center sequence is called "complex", as there many objects to be tracked. The amount of objects per frame or *density* is considered to classify each scene.

To evaluate how good are the methods, it is used the $T_{Accuracy} = \frac{TP}{(TP+FP)}$ metric, measuring the amount of objects true detected objects (TP) and missing objects (FP) compared to ground-truth using overlapping bounding boxes. $T_{Accuracy}$ is computed in both cases, applying and not the scheduling method. Also, the amount of frames processed per second are computed (FPS). It is important to remark that tracker and network accuracy are not evaluated, because they were already evaluated in the cited works.

Finally, we present in 4.2 a resume of different scenarios, measuring processing speed, scene complexity and detection accuracy.

		Non scheduled		Scheduled	
Scene	Density	FPS	Accuracy	FPS	Accuracy
1 (street)	1.3	2.9	0.64	**7.26**	0.508
2 (street)	1.5	6.4	0.93	**8.84**	0.84
3 (corner)	0.45	9	0.81	**12.03**	0.76
4 (route)	0.2	26	0.95	**28**	0.95
5 (corner)	0.1	38	1.0	**38**	1.0
6 (crowded corner)	1.17	3.1	0.82	**5.27**	0.79
7 (avenues)	0.75	5.1	0.79	**6.85**	0.77
8 (route)	1.17	12	0.9	**15**	0.89

It is observed that when density is bigger than 0.5, the scheduled algorithm always improves the performance, about a 50%. In some situation, like scene 1, it punishes the accuracy. That case should be evaluated in depth.

5 Conclusions

In this work, a novel parallel tracking and framework that combines correlation kernel-based tracking and deep learning-based classification is presented. We propose a metric that taking into account objects behaviour, can choose which objects should be observed more frequently. The videos have been processed and visualized in real time since the algorithm has relative computational low cost. The solution shows promising results on thorough experiments. Moreover, it is worth noting that it is a very flexible framework and our implementation is far from optimal. We believe there are great rooms for future improvement and generalization.

Future works will attempt to incorporate the automatic adaptation of the weights of each tracker, considering the variation of the intensity of the movement when processing the video, and adding some other considerations, like darker areas where the algorithm has difficulty and in general lost objects. Another challenge aims to achieve that the parameters of the algorithm automatically adapt to different times of the day.

Acknowledgement. This work was partially supported by the National Council Research of Argentina (PICT 2016 - 0236) and (PICT 2020 - 005).

References

1. Kruegle, H.: CCTV Surveillance: Video Practices and Technology. Butterworth-Heinemann, Newton (2014)
2. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-learning detection **34**(7), 1409–1422 (2012)
3. B. Babenko, Yang, M.-H., Belongie, S.: Robust object tracking with online multiple instance learning **33**(8), 1619–1632 (2011)
4. Henriques, J.F., Caseiro, R., Martins, P., Batista, J.: High-speed tracking with kernelized correlation filters. CoRR abs/1404.7584 (2014)
5. D'Amato, J.P., Dominguez, L., Perez, A., Rubiales, A., Stramana, F.: Generación de servicios digitales en ciudades inteligentes a partir de las capacidades de los sistemas de cámaras. RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao (27), 566–578 (2019)
6. Fan, H., Ling, H.: Parallel tracking and verifying: a framework for real-time and high accuracy visual tracking. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 5487–5495 (2017)
7. Legua, C.: Seguimiento automático de objetos en sistemas con múltiples cámaras (2013)
8. Shaikh, S.H., Saeed, K., Chaki, N.: Moving object detection using background subtraction. In: Moving Object Detection Using Background Subtraction. SCS, pp. 15–23. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07386-6_3
9. Azab, M., Shedeed, H., Hussein, A.: A new technique for background modeling and subtraction for motion detection in real-time videos. In: IEEE International Conference on Image Processing, pp. 3453–3456 (2010)
10. Hadi, R., Sulong, G., George, L.: Vehicle detection and tracking techniques: a concise review. Sig. Image Process Int. J. (SIPIJ) **5** (2014)

11. Bao, C., Wu, Y., Ling, H., Ji, H.: Real time robust l1 tracker using accelerated proximal gradient approach. In: CPVR, vol. 3 (2012)
12. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, vol. 25. Curran Associates, Inc. (2012)
13. Nam, H., Han, B.: Learning multi-domain convolutional neural networks for visual tracking. CoRR abs/1510.07945 (2015)
14. Bolme, D.S., Beveridge, J.R., Draper, B.A., Lui, Y.M.: Visual object tracking using adaptive correlation filters. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. (2010) 2544–2550
15. Greco, A., Saggese, A., Vento, M., Vigilante, V.: Vehicles detection for smart roads applications on board of smart cameras: a comparative analysis **20**(99), 1–13 (2021)
16. Wang, Z., Lin, J., Wang, Z.: Accelerating recurrent neural networks: a memory-efficient approach. IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **25**(10), 2763–2775 (2017)
17. Pleiss, G., Chen, D., Huang, G., Li, T., van der Maaten, L., Weinberger, K.Q.: Memory-efficient implementation of DenseNets. CoRR abs/1707.06990 (2017)
18. Stramana, F., D'amato, J.P., Dominguez, L., Rubiales, A., Perez, A.: Object extraction and encoding for video monitoring through low-bandwidth networks. In: Figueroa-García, J.C., Garay-Rairán, F.S., Hernández-Pérez, G.J., Díaz-Gutierrez, Y. (eds.) WEA 2020. CCIS, vol. 1274, pp. 431–441. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-61834-6_37
19. Redmon, J.: Darknet: Open source neural networks in C. http://pjreddie.com/darknet/ (2016)
20. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 983–990 (2009)