Parallel processing applied to object detection with a Jetson TX2 embedded system

Jesús Benito-Picazo[1,2], Jose David Fernández-Rodríguez[1,2] Enrique Domínguez[1,2], Esteban J. Palomo[1,2], and Ezequiel López-Rubio[1,2]

[1] University of Málaga, Bulevar Louis Pasteur, 35, 29071 Málaga, Spain
{jpicazo, josedavid, enriqued, ejpalomo, ezeqlr}@lcc.uma.es
[2] Biomedical Research Institute of Málaga (IBIMA), Málaga, Spain

Abstract. Video streams from panoramic cameras represent a powerful tool for automated surveillance systems, but naïve implementations typically require very intensive computational loads for applying deep learning models for automated detection and tracking of objects of interest, since these models require relatively high resolution to reliably perform object detection. In this paper, we report a host of improvements to our previous state-of-the-art software system to reliably detect and track objects in video streams from panoramic cameras, resulting in an increase in the processing framerate in a Jetson TX2 board, with respect to our previous results. Depending on the number of processes and the load profile, we observe up to a five-fold increase in the framerate.

Keywords: deep learning, embedded systems, object detection, multiprocessing

1 Introduction

Computer vision has been one of the most revolutionary technologies with a very promising field of study for researchers, and strong foundations supporting the industrial activity of many of the most successful and consolidated companies.

In this work, we describe an improved deep learning-based automatic video surveillance system for panoramic cameras, specifically optimized to be deployed on a Jetson TX2 board. A survey related to vision-based human action recognition can be found in [16], where authors address different challenges when building these kinds of systems. A large number of related works have been published in the literature, such as a real-time video surveillance system for detection and tracking of people in outdoor environments [9], or systems equipped with person identification modules based on effective features representation [7].There are times when people are not the only target of a video surveillance system. In fact, sometimes it is important that video surveillance systems are ready to alert from the presence of a certain object in the scene that is under vigilance, and for this a background modeling algorithm [12]is needed, such as a deep learning-based background subtraction model for flexible foreground segmentation [18].

Deep learning is a machine learning technique that excels in accuracy and performance, especially in the field of computer vision. An overview of various

applications of convolutional neural networks (CNNs) for solving inverse problems in image processing, such as deconvolution, denoising, medical image reconstruction, and superresolution, is presented in [14]. Deep convolutional networks have been proposed to be used in the construction of crack detection systems on asphalt pavement surfaces [2], or in post-disaster inspection [13].

Choosing an appropriate camera is critical in designing any video surveillance system, as this device will be tasked with providing the image input of the cited system. Pan-tilt-zoom (PTZ) cameras are powerful, yet affordable devices. Their versatility and motion capabilities have made them widely accepted in the design of video surveillance systems [15], and in the development of methods for salient motion detection in non-stationary videos [6], or even in specially designed background subtraction algorithms [17].

Computer vision, and more specifically deep learning-based video surveillance systems, typically have high computational requirements that in many cases can only be met by expensive, high-power-consuming devices, severely limiting their autonomy and versatility. Thus, there is a need for low-power automatic video surveillance systems with acceptable performance. Along this line, some works have been developed in the last years, such as the design of a system, which can be deployed into unmanned aerial vehicles (UAV), for detecting moving objects [1], a system composed by a Raspberry-Pi board and a RaspiCam camera for tracking [8], or a deep learning-based automatic video surveillance system based on a Raspberry-Pi microcomputer for panoramic cameras [3].

The system developed in this work is based on a previously published deep-learning system [4], and relies on a novel multiprocess-based potential detection generator that performs up to 5 times faster than its predecessor.

The rest of the paper is organized as follows. Section 2 presents the mathematical model of the proposal. Section 3 describes the system architecture. Experimental results are provided in section 4. Finally, some conclusions and further lines are presented in section 5.

2 Methodology

In this section, our proposed method aimed to detect anomalous objects is presented. We consider environments where anomalous objects are present. This implies that these objects belong to classes that are rarely found in the scene. Provided that an anomalous object is detected, the system must prompt an alarm.

Our strategy is based on the analysis of the object detections that have occurred in the recent past. Those detections are stored in a set and are called active detections. This set contains the objects associated with the most recent detections from the video camera. Let us note a detection as (π_i, x_1, x_2, x_3), which is a real-valued vector with the following four components:

- π_i is the *a priori* object probability.
- (x_1, x_2) are the vertical and horizontal coordinates of the object, relative to a panoramic coordinate system that is intrinsic to the surveillance camera.

– x_3 is the pixel count corresponding to the bounding box associated with the detected object.

Furthermore, let us note α the forgetting rate. This rate controls the update of the a priori probability parameter π_i. Whenever a detection is no longer visible, the associated element of the detection set is marked as inactive.

Let us note $\mathbf{x} = (x_1, x_2, x_3)$ for simplicity. This allows expressing the domain of possible values for $\mathbf{x}$ as follows:

$$\mathcal{V} = [1, N_{rows}] \times [1, N_{cols}] \times [S_{min}, S_{max}] \subset \mathbb{R}^3 \tag{1}$$

where $N_{rows} \times N_{cols}$ is the image size of the incoming video frame expressed in pixels. Consequently, the lower and upper limits for the size of a bounding box are given by S_{min} and S_{max}, respectively.

Given the above considerations, probabilistic modeling is chosen to manage the objects present in the scene. In particular, the probable locations of the objects are modeled as follows:

$$p(\mathbf{y}) = qU_{\mathcal{V}}(\mathbf{y}) + (1-q)\frac{1}{M}\sum_{i=1}^{M}\pi_i K(\mathbf{y}, \mathbf{x}_i, \sigma) \tag{2}$$

where $U_{\mathcal{V}}(\mathbf{y})$ stands for the uniform probability distribution on $\mathcal{V}$, $K(\mathbf{y}, \boldsymbol{\mu}, \sigma)$ denotes a suitable multivariate probability distribution which has a mean vector $\boldsymbol{\mu}$ and a constant spread parameter σ, M is the count of active detections, $q \in (0, 1)$ is the mixing parameter to combine the two probabilistic mixture components, and σ is the spread parameter associated with the multivariate distribution.

Three distinct multivariate probability distributions have been chosen to be integrated into the above-described probabilistic mixture model, namely: Gaussian, Student-t, and triangular. Their definitions are provided in Table 1. Please note that $\|\cdot\|$ stands for the Euclidean norm of a vector. On the other hand, ν denotes the degrees of freedom parameter for the Student-t distribution. Also, it is worth noting that both the Gaussian and Student-t distributions contain a spread parameter σ which is the standard deviation.

The multivariate mixture component of our probabilistic mixture is aimed to model the possible object occurrences close to the most recently detected objects. On the other hand, the uniform mixture component of the probabilistic mixture models object occurrences that might arise in other regions of the incoming video frames. The above-presented algorithm to detect anomalous objects can be summarized as follows:

1. Initialize the current detection set $\mathcal{A}$ to the empty set.
2. Obtain the current video frame from the hardware.
3. Apply the forgetting rate α to recompute the a priori probabilities π_i for all the elements of the active detection set. If an object is no longer visible because it has gone out of $\mathcal{V}$, then it is erased because it is now inactive.
4. Draw M samples at random according to the multivariate distribution (2). Identify the bounding box associated with each sample. Then modify its size

$$K_{Gaussian}\left(\mathbf{y},\boldsymbol{\mu},\sigma\right)=\left(2\pi\right)^{-\frac{3}{2}}\sigma^{-3}\exp\left(-\frac{1}{2\sigma^{2}}\left\|\mathbf{y}-\boldsymbol{\mu}\right\|^{2}\right) \tag{3}$$

$$K_{Student}\left(\mathbf{y},\boldsymbol{\mu},\sigma\right)=\frac{\Gamma\left(\frac{\nu+3}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)\nu^{\frac{3}{2}}\pi^{\frac{3}{2}}\sigma^{3}}\left(1+\frac{1}{\nu\sigma^{2}}\left\|\mathbf{y}-\boldsymbol{\mu}\right\|^{2}\right)^{-\frac{\nu+3}{2}} \tag{4}$$

$$K_{Triangular}\left(\mathbf{y},\boldsymbol{\mu},\sigma\right)=\prod_{j=1}^{3}k_{Triangular,j}\left(y_{j},\mu_{j},\sigma\right) \tag{5}$$

$$k_{Triangular,j}\left(y_{j},\mu_{j},\sigma\right)=\begin{cases}0 & \text{for } y_{j}<\mu_{j}-\sigma\\ \frac{y_{j}-\mu_{j}+\sigma}{\sigma^{2}} & \text{for } \mu_{j}-\sigma\leq y_{j}<\mu_{j}\\ \frac{1}{\sigma} & \text{for } y_{j}=\mu_{j}\\ \frac{\mu_{j}+\sigma-y_{j}}{\sigma^{2}} & \text{for } \mu_{j}<y_{j}\leq\mu_{j}+\sigma\\ 0 & \text{for } y_{j}>\mu_{j}+\sigma\end{cases} \tag{6}$$

Table 1. Considered multivariate probability distributions.

to match the window size required by the image classification deep neural network. Then the window associated with the bounding box is fed to the deep network. In case the network informs of detection, the corresponding sample is inserted into $\mathcal{A}$. Also, the sample is marked with the reliability of the detection, which is taken as an estimation of the probability that the detection is correct.

5. Go to step 2.

3 System architecture

Automatic object detection and classification in digital video streams is a very common task nowadays. However, it is also a very complex procedure that requires the use of deep learning techniques in order to be enough robust and accurate. Any system involving the use of deep learning techniques usually requires high amounts of computing power. This is more remarkable when processing video streams coming from panoramic 360° cameras as they handle even larger frames. On a regular basis, the building of a deep learning-based object detection system requires the use of expensive and high-power demanding hardware that, at the same time, requires a series of external components that difficult its integration into autonomous devices.

One solution for building deep learning-based object detection systems for autonomous devices is to deploy them into embedded systems as they are small, low power consuming and they barely require external components to work. But, in the case of embedded systems-based video stream processing, computational resources are especially valuable as they tend to be scarce. Thus, it is desirable to utilize system architectures that optimize computation processes so they can be performed by low-profile pieces of hardware whose computing power is reduced,

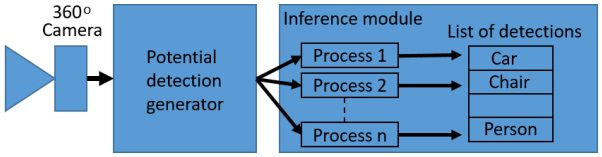

Fig. 1. Diagram of the software architecture

in order to achieve a low power consumption and a higher level of autonomy and versatility.

As mentioned in section 1, this work pursues the development of a standalone object detection and classification system for 360° camera video streams with low power consumption and reduced size. Consequently, the architecture of the system will be divided into two well-differentiated parts, namely the *software architecture*, and the *hardware architecture*. Both of them will be precisely detailed below.

3.1 Software architecture

The software architecture developed in this work is committed to the objective of optimizing the system's operation in order to be deployed in a low-power-consuming hardware device without experiencing a strong loss in performance. With this target in mind, the architecture consists of three different modules that operate concurrently in a producer-consumer configuration (Figure 1).

The first module is a video stream acquisition process that is in charge of receiving the frames from any 360° video source, namely a panoramic camera. Frames are supplied to the second module which implements the potential detection generator this system relies on. Contrary to what systems such as Faster-RCNN do, the potential detection generator takes advantage of the information learnt from past frames by selecting a certain number of areas in the current frame where a Convolutional Neural Network (CNN) is going to check if there is an identified object or not. The position and size of these areas will be selected by using one of the three multivariate homoscedastic probability distributions presented in section 2, over the position and size of the objects the system has already found in the video stream.

The third module is the one presenting the highest novelty in this work and is the inference module. The cited module is in charge of performing the identification of the objects appearing in the portions of the frame supplied by the potential detection generator. It consists of an array of several inference parallel processes each one of them performing the inference task which will determine whether there is an identified object in that area according to the accuracy obtained. Every parallel process will add the result of the inference, i.e., the position and category of the identified object, to a common list of confirmed detections. This list of confirmed detections is going to be updated by the main process according to the algorithm proposed in Section 2.

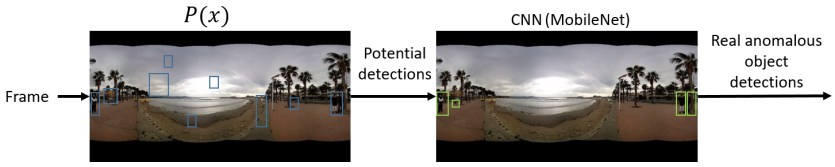

Fig. 2. Schematic of the system's workflow.

3.2 Hardware architecture

Multiprocess-based implementations of Deep learning-based detection systems require larger amounts of memory than monoprocess implementations. The reason is that, since data structure sharing between different processes is quite limited, sometimes is necessary to store some data structures in the memory of every process involved. Hence, it was critical to choose a piece of hardware that not only was small and low power consuming but also has enough memory and computing capabilities. Therefore, the device selected to support the system developed in this work was the Jetson TX2 board. This device is a consolidated system for deep learning tasks and it has a 256-core NVIDIA Pascal GPU, 8GB of RAM/VRAM memory and a power consumption of 7.5 watts.

4 Experimental results

In order to test the object detection and classification system for panoramic video streams developed in this work, a complete tests series has been developed by implementing the software architecture described in section 3.1, and deploying it in the hardware platform described in 3.2. This implementation consists of a program designed in Python language that analyses, frame by frame, a 360° video simulating the video stream provided by a 360° camera. In the opinion of the authors of this work, this method is more convenient for testing the performance of the system since it avoids the intrinsic issues produced by the interaction with the camera, resulting in more reproducible experiments and more accurate measurements. The video used is a 360° video supplied by Stanford University's *Virtual Human Interaction Lab* [11]. The system workflow is shown in figure 2.

According to this, in the first place, a frame from the 360° video indicated above is fed to the system. Next, this frame is processed by the potential detection generation engine whose operation was described in section 3.1. This module will generate a set of potential detections that in practice, is a set of areas or windows from the current frame whose position and size will be calculated by using one of the three probability distributions explained in section 2. The program will feed the potential detections to the inference module, where a set of n parallel processes will use a CNN to determine whether there are any objects of the category set recognized by the CNN in the areas enclosed by the potential detections. If one process determines that a certain window contains any object, this one will be added to the list of confirmed detections and incorporated into the knowledge base of the system.

As this system is an improvement over the one published in [4], the experiments section in this work is mostly oriented to illustrate the increase achieved

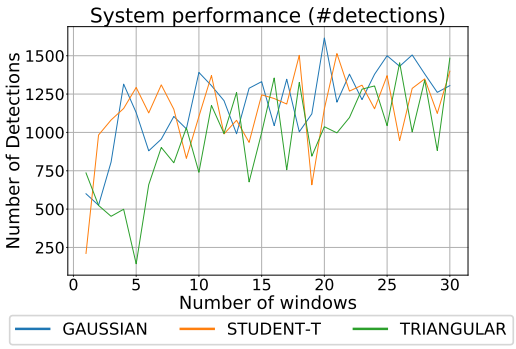

Fig. 3. Number of detections performed by the system in 612 frames setting up the potential detection generator with the three multivariate distributions (see Section 2).

in the system speed by introducing parallel computing in the inference module. So, the experiments consisted of feeding the 612 frames of the 360° video cited above to the system and checking how many of the objects which are actually present in the frames it can detect and how fast it can do it. This experiment has been repeated for all three multivariate probability distributions presented in section 2, for a number of potential detections that goes from 1 to 30 and for different amounts of parallel inference processes that goes from 1 to 4. The reason for using up to 4 parallel inference processes is that this was the maximum number of processes the Jetson TX2 was capable of managing without running out of memory.

The video from [11] which has been used to perform the tests, was manually tagged localizing all the appearances of objects from four categories of the Pascal VOC 2012 dataset. These categories are "person", "dog", "car" and "motorcycle". The program can check whether the position of a detection generated by the system really contains the object identified in this detection. This way the system can count the number of positive detections of objects in each frame. The Convolutional Neural Network used in the inference process is the MobileNet [10] implementation from the Pytorch framework properly trained with the widely used Pascal VOC 2012 dataset. The reason for using the MobileNet in our inference module is its balance between accuracy, inference speed, and memory consumption.

In order to ensure the reproducibility of the experiments it is also important to indicate the values of the different parameters affecting the multivariate probability distributions in which relies the potential detection generator. These values are $\sigma = 0.3$ and $\alpha = 0.1$ for all three distributions. In the case of q, it is $q = 0.4$ for the Gaussian mixture, $q = 0.2$ for the Student-t mixture, and $q = 0.7$ for the Triangular mixture. It is important to remark that the parameter values are selected after a supercomputer-driven process of optimization by using as a validation dataset two different videos from the [5] dataset.

Regarding to the results from the experiments, figure 3 shows how the number of detections increases as the number of potential detections grows up. This is expected, since the higher the number of potential detections generated by

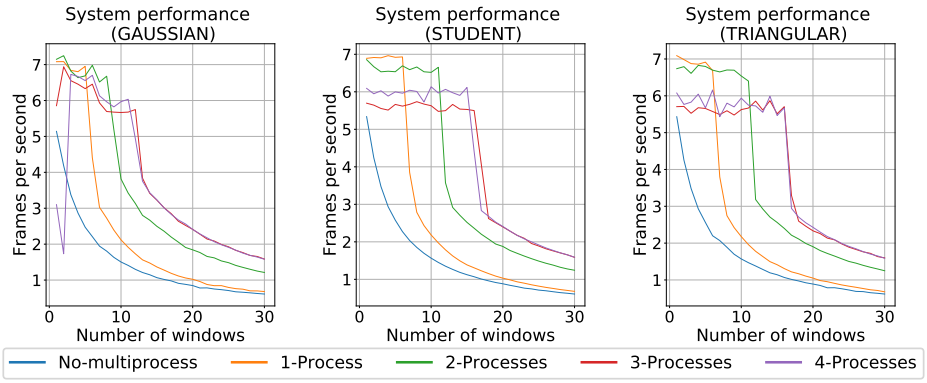

Fig. 4. Performance of the system in fps for all three multivariate probability distributions, from 1 to 4 parallel processes and the no multiprocess version of the system.

the system for each frame, the higher the number of regions of the image the system looks for possible objects simultaneously. It can also be observed that the number of correct detections performed by the system does not follow a smooth progression. Instead, multiple oscillations can be observed in the plot. The reason for this is that the probability distributions used for the system to know which region of the frame to observe in a certain frame have an important random component that will affect the initialization of the system, and consequently, its performance through the following frames. The last appreciation that can be extracted from this plot is that at first glance, the potential detection engine powered by the Gaussian distribution seems to perform slightly better than the other two, having the highest number of accumulated correct detections in one pass when using 20 potential detections.

As has been already anticipated in the paragraphs above, the experimental section is mostly dedicated to analyze the performance of this system when the inference module is implemented using a multiprocess parallel concurrent architecture. Consequently, for this section, it has been developed a series of experiments involving the evaluation of the system speed in frames per second when implementing the inference module with one, two, three, and four parallel processes on a Jetson-TX2 board for a number of potential detections spanning from 1 to 30 and for all three probability distributions described in Section 2. In order to illustrate the advantages of using multiprocessing, the tests have also been performed using the single process implementation described in [4]. Results of this series of experiments are shown in Figure 4.

The most important appreciation from Figure 4 is the speed increase in fps when using multiprocessing for almost every amount of potential detections. More precisely, the system speed increases from 1 to 5 fps depending on the number of parallel inference processes and the number of potential detections (windows) considered. This represents a system that in the best case can be up to 5 times faster than the non-multiprocessing version. It is remarkable that, for example, for one potential detection, the speed in fps is higher even when just one process in the inference module has been used. The reason for this is that

even using only one process in the inference module, this process works concurrently with the potential detection generator module in a producer-consumer architecture which makes image processing more efficient. The other important observation it can be obtained from these plots is that a high amount of parallel processes is not always equivalent to higher fps. This only happens when the number of potential detections is high enough to take advantage of the parallel processing architecture. When the number of potential detections is not high enough, the cost of managing multiple processes overrides the benefit of multiprocessing implementation of the inference module. So, it can be concluded that the multiprocessing architecture is more suitable for this system as the number of potential detections increases. It is also apparent from Figure 4 that the probability distribution used in the implementation does not seem to introduce a significant variation in the speed performance of the multiprocess implementation. This is also an expected behavior as the potential detection generator is placed in a monoprocess module.

5 Conclusion

In this paper, a novel anomalous object detection system embedded in a Jetson TX2 board is proposed. In our system, video streams taken from panoramic surveillance cameras feed a potential detection generator module based on a probability mixture model to detect anomalous objects. Then, these detected objects are classified in the inference module using a MobileNet model. The novelty of this proposal with respect to previous state-of-the-art works is the introduction of a multiprocess parallel concurrent architecture in the inference module to increase the processing framerate in a Jetson TX2 board.

Experimental results show that parallel processing increases the speed of the system from 1 to 5 fps depending on the number of parallel inference processes and the number of potential detections considered. In general, the higher the number of potential detections, the higher the number of parallel processes that can be used to increase the speed of the system. Finally, we can observe that the speed performance is not influenced by the probability distribution used in the potential detection generator.

Future work involves the improvement of the system's accuracy with the implementation of a new potential detection generator, based on a more specific probability distribution, and the using of NVIDIA's TensorRT technology in order to improve the system's speed performance.

Acknowledgements.

This work is partially supported by the Autonomous Government of Andalusia (Spain) under project UMA20-FEDERJA-108. It is also partially supported by the University of Málaga under grant. It includes funds from the European Regional Development Fund (ERDF). It is also partially supported by the University of Málaga (Spain) under grants B1-2019_01, B1-2019_02, B1-2021_20, B4-2022 and B1-2022_14. They also gratefully acknowledge the support of NVIDIA Corporation with the donation of a RTX A6000 GPU with 48Gb. The authors

also thankfully acknowledge the grant of the Instituto de Investigación Biomédica de Málaga y Plataforma en Nanomedicina-IBIMA Plataforma BIONAND.

References

1. Angelov, P., Sadeghi-Tehran, P., Clarke, C.: AURORA: Autonomous real-time onboard video analytics. Neural Comput. Appl. **28**(5), 855–865 (2017)
2. Bang, S., Park, S., Kim, H., Kim, H.: Encoder–decoder network for pixel-level road crack detection in black-box images. Computer-Aided Civil and Infrastructure Engineering **34**(8), 713–727 (2019)
3. Benito-Picazo, J., Domínguez, E., Palomo, E.J., López-Rubio, E.: Deep learning-based video surveillance system managed by low cost hardware and panoramic cameras. Integrated Computer-Aided Engineering **27**(4), 373–387 (2020)
4. Benito-Picazo, J., Domínguez, E., Palomo, E.J., Ramos-Jiménez, G., López-Rubio, E.: Deep learning-based anomalous object detection system for panoramic cameras managed by a Jetson TX2 board. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–7 (2021). DOI 10.1109/IJCNN52387.2021.9534053
5. Charles, P.L.S.: LITIV. http://www.polymtl.ca/litiv/en/ (2018). [Online; accessed 14-February-2018]
6. Chen, C., Li, S., Qin, H., Hao, A.: Robust salient motion detection in non-stationary videos via novel integrated strategies of spatio-temporal coherency clues and low-rank analysis. Pattern Recognition **52**, 410 – 432 (2016)
7. Dalwadi, D., Mehta, Y., Macwan, N.: Face recognition-based attendance system using real-time computer vision algorithms, *Advances in Intelligent Systems and Computing*, vol. 1141. Springer (2021)
8. Dziri, A., Duranton, M., Chapuis, R.: Real-time multiple objects tracking on raspberry-pi-based smart embedded camera. Journal of Electronic Imaging **25**, 041,005 (2016)
9. Haritaoglu, I., Harwood, D., Davis, L.S.: W4: Real-time surveillance of people and their activities. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8), 809–830 (2000). Cited By :2192
10. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications (2017)
11. Lab, V.H.I.: 360 video database. URL https://vhil.stanford.edu/
12. Li, L., Huang, W., Gu, I.Y.., Tian, Q.: Statistical modeling of complex backgrounds for foreground object detection. IEEE Transactions on Image Processing **13**(11), 1459–1472 (2004). Cited By :781
13. Liang, X.: Image-based post-disaster inspection of reinforced concrete bridge systems using deep learning with bayesian optimization. Computer-Aided Civil and Infrastructure Engineering **34**(5), 415–430 (2019)
14. McCann, M., Jin, K., Unser, M.: Convolutional neural networks for inverse problems in imaging: a review. IEEE Signal Processing Mag. **34**, 85–95 (2017)
15. Micheloni, C., Rinner, B., Foresti, G.: Video analysis in pan-tilt-zoom camera networks. IEEE Signal Processing Magazine **27**(5), 78–90 (2010)
16. Poppe, R.: A survey on vision-based human action recognition. Image and Vision Computing **28**(6), 976–990 (2010). Cited By :1484
17. Sajid, H., Cheung, S.C.S., Jacobs, N.: Appearance based background subtraction for PTZ cameras. Signal Processing: Image Communication **47**, 417 – 425 (2016)
18. Vijayan, M., Mohan, R.: A universal foreground segmentation technique using deep-neural network. Multimedia Tools and Applications **79**, 34,835–34,850 (2020)