This is a pre-print of the following original article:

Title: On the Role of Smart Vision Sensors in Energy-Efficient Computer Vision at the Edge

The final version of this paper appears in 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), 2022, pp. 497-502

DOI: 10.1109/PerComWorkshops53856.2022.9767380.

Publisher: IEEE

On the Role of Smart Vision Sensors in Energy-Efficient Computer Vision at the Edge

Alberto Ancilotto[§], Francesco Paissan[§], Elisabetta Farella
aancilotto@fbk.eu, fpaissan@fbk.eu, efarella@fbk.eu
E3DA Unit, Digital Society center, *Fondazione Bruno Kessler (FBK)*, Trento, Italy

***Abstract*—The increasing focus of the research community towards lightweight and small footprint neural network models is closing the gap between inference performance in cluster-scale models and tiny devices. In the recent past, researchers have shown how it is possible to achieve state-of-the-art performance in different domains (e.g. sound event detection, object detection, image classification) with small footprints and low computational cost architectures. However, these studies lack a comprehensive analysis of the input space used (e.g. for images) and present the results on standard RGB benchmarks. In this manuscript, we investigate the role of smart vision sensors (SVSs) in deep learning-based object detection pipelines. In particular, we combine the motion bitmaps with standard color spaces representations (namely, RGB, YUV, and grayscale) and show how SVSs can be used optimally for an IoT end-node. In conclusion, we report that, overall, the best-performing input space is grayscale augmented with the motion bitmap. These results are promising for real-world applications since many SVSs provide both image formats at low power consumption.**

***Index Terms*—smart vision sensors, edge computing, tinyML, embedded computer vision**

I. Introduction

In recent years, many efforts have been made to port artificial intelligence to edge devices, typically composed of microcontroller-based embedded systems, with limited available resources [1]–[3]. Several literature approaches adopt techniques such as model pruning [4], [5] or knowledge distillation [6], [7] intending to decrease the complexity of computationally expensive models that otherwise cannot fit on resource-constrained units. These techniques have shown significant results in network compression by enabling the porting of state-of-the-art models on MCUs with a minor impact on the final performance (2-3% accuracy drop). However, current approaches do not always examine the impact of the type of camera used on an embedded platform. They also do not consider scaling with different input image formats, which is crucial when implementing computer vision algorithms on microcontroller units (MCUs).

Nowadays, more and more smart vision sensors (SVS) are being developed to ease image processing pipelines by performing in-hardware compression and signal processing. A significant advantage of SVSs is that they are computationally efficient since they generally exploit parallelization to process the whole image on a single clock cycle [8]. When bringing intelligence on edge devices, it is crucial to respect the stringent computational constraints deriving from the target platforms, which usually have less than a MB of flash and a similar amount of RAM. Thus, relying on a pre-processed input, for example, by using an SVS, can significantly lighten the computational load on the MCU. This benefits the runtime performance of the applications and the resulting power consumption of the devices, a key aspect for always-on Internet of Things devices.

Among the many variations of SVS present in literature [9] (ranging from skin detection [10]–[12], background subtraction [13], color tracking and face detection [14]), the most relevant for urban scenarios are SVS integrating a background removal algorithms at the sensor level. These provide, in a power-efficient manner, high-level information that can be used to ease subsequent processing steps. It has already been proved how such sensors are a good fit when paired with traditional image processing techniques [15], [16], leading to lower system power requirements. In this work, we want to investigate the effects of using information generated by similar systems when performing an object detection task, and, in particular, we will focus on the power-performance trade-off of such pipelines. To do so, we will emulate the information generated by these sensors by using different approaches for background subtraction. Using an object detector based on the novel PhiNets architecture family, we will argue how the motion bitmaps generated can be used to develop neural network pipelines for object detection.

The work is structured as follows:

- Section II covers some examples of SVS and pipelines developed to take advantage of the data generated by these sensors;
- Section III presents our experimental setup, focusing on the training data used, the object detection pipeline employed, and the approaches used to emulate a possible output of a background subtracting SVS;
- Section IV presents the two proposed experiments that we carried out to analyze the effects of background removal data on object detectors using different processing techniques and different network complexities. Moreover, we show empirically how the motion bitmaps affect the generated feature maps during an inference pass.

[§] These authors contributed equally to this research. This work was partially funded by the EU H2020 project MARVEL (project number: 957337).

II. Related works

There are several SVS performing on-board background subtraction in literature. The goal of these devices is to produce motion bitmaps from the acquired images directly in-sensor, in a power-efficient manner. The main difference with state-of-the-art approaches is the background subtraction algorithm implemented, which directly impacts the device's power consumption. For example, a 64×64 CMOS sensor based on frame difference is proposed in [17]. It consumes only $0.4\,\mathrm{mW}$ at 100 fps. In [18], the authors present an image sensor with motion-triggered feature extraction capabilities, thus delegating the processing to external units (like MCUs). However, this approach limits the analysis that can be performed on the produced features. In [19], the authors proposed a novel algorithm for background removal based on two adaptive thresholds. As can be seen, there are several different approaches employed by SVSs for providing a high-level interpretation of the input to the subsequent steps of the image processing pipeline. This results in a wide variety of different motion bitmaps that can be obtained using various sensors in the same scene. Given the high variance of these motion bitmaps, in this work, we will compare three different methods: a K-nearest neighbour (KNN) based approach [20], a mixture of Gaussians (MOG) approach [21] and a simple frame difference, for training the deep-learning-based object detection.

In the last decade, many techniques to process binary images have been developed [22], [23] and have recently been applied to the output of SVSs [15], [16], [24]. However, many constraints prevent from easily adapting these techniques to new problems, mainly due to the hand-crafted nature of the selected features. The end-to-end nature of deep learning offers a better option since we can adapt the processing pipeline to a new task by re-training the neural network. In this work, we adopt the novel PhiNets architecture family [25], a scalable backbone for object detection and tracking on MCUs. We chose PhiNets because their scalability principles enable the correlation of different input types (motion bitmap, RGB, grayscale) and the optimal neural architectures (and thus computational cost).

III. Experimental setup

A. Data preparation

To simulate the outputs of different SVS and correlate them to network performance, we relied on the well-known MOT17 dataset [26]. This benchmark contains a collection of annotated videos with different properties (e.g. illumination, the density of targets, static or moving camera). Given the restricted application domain of SVSs, we only used video sequences acquired from stationary cameras, allowing us to benchmark different background subtraction algorithms. Moreover, this subset also represents the typical SVS application scenario (e.g. fixed to a pole in urban applications). We generated five square crops for each frame in the original sequences to have enough data to train a network from scratch. The crops are randomly located in the original frame and have a size between 300 and 700 pixels. For data augmentation, we flipped crops with a 50% chance. With this approach, we generated 10875 crops containing ≈ 55000 bounding boxes. We split the frames in train and test randomly, with a 90-10 ratio. All crops belonging to the same frame are inserted together either in the training or test sets.

The bounding boxes are assigned to each crop if a fraction > 0.35 of an object is inside the crop and are ignored otherwise. In the same way, occluded objects are accounted for only if the same fraction of the object is visible.

In this work, we used this data to evaluate some object detectors in terms of Mean Average Precision (*mAP*) and investigate the impact of SVSs input in such a pipeline. Mean Average Precision has always been computed with an Intersection Over Union (IOU) threshold of 0.5.

B. Object Detector

As the primary use case of SVSs is low-power machine intelligence, we used the *Phinets* architecture described in [25] as a processing algorithm for the sensor's output. It is a low operation count scalable backbone, composed of a sequence of B (in our case we used the default value of 7) inverted residual blocks, designed to be adapted to different embedded platforms and MCUs. Scalability is achieved by using three parameters that can optimize the resource requirements for the networks to target them to various hardware configurations. These parameters are:

- α: width scaling parameter, used to regulate the operation count of the network;
- β: shape parameter, used to optimize the static memory requirements for the architecture;
- t_0: base expansion factor, used to optimize the RAM required by the network.

This backbone is coupled to a *YOLOv2* [27] detection head to provide an end-to-end object detector. We performed minimal modifications of the architecture to accommodate for input with varying channel count. These changes are supported in the original implementation[1].

C. Network Training

The different architectures tested were trained on the artificial dataset generated as described before, where each crop has been normalized between $[-1; 1]$ on each channel before sending it as input to the network. The Adam [28] optimizer was used, with a cosine decaying learning rate starting at 10^{-2}, for 300 epochs. A constant 10^{-2} learning rate was used for the first three epochs.

D. Experiments

To improve the coverage of our analysis, we emulate SVSs employing different background subtraction techniques. In particular, we applied algorithms based on:

- Frame difference. This method computes the pixel-wise intensity difference between two consecutive frames.

[1] https://github.com/fpaissan/phinet_pl

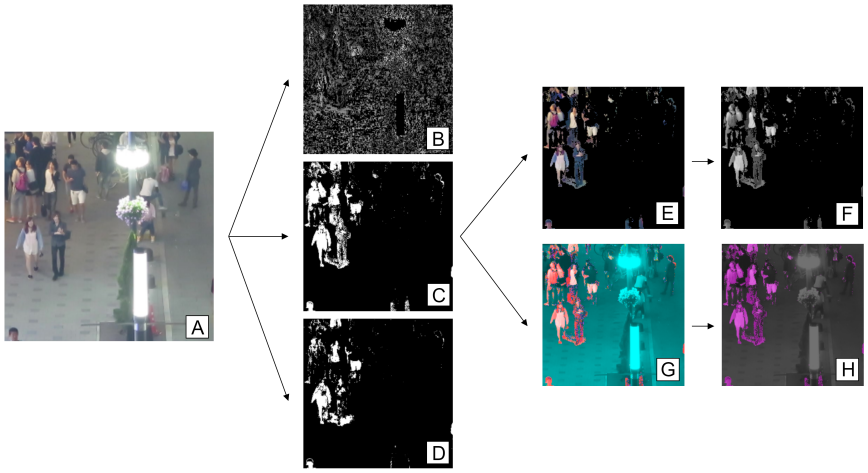

Fig. 1. The different pre-processing steps applied to a frame crop to emulate different SVS. A- default RGB frame crop. B, C, D - output of the different background subtractors. E, F - grayscale and RGB masked inputs. G, H - grayscale and RGB images with additional motion bitmap channel

After this, a threshold is applied on the output map to binarize the image and create the foreground and background masks. Example results are shown in Figure 1B.

- MOG: Mixture of Gaussian based algorithm, as described in [29] and [30]. The default `OpenCV` implementation is used (Figure 1C).
- KNN: K-nearest neighbor-based algorithm, as described in [31]. The default `OpenCV` implementation is used (Figure 1D).

To identify the best input format for our detection application, we tested the object detector combining image data and motion data in different ways:

- RGB, G: Input is composed of the image data in 3 channel color and grayscale (A in Figure 1);
- MB: Input is composed of the motion bitmap obtained with the algorithms shown above (B, C, D in Figure 1);
- RGBA, GA: Input is composed by the color and grayscale image, respectively, with an additional input channel containing the motion bitmap data (G, H in Figure 1). It represents a concatenation between image channels and motion bitmap along the channel axis;
- RGBM, GM: Input is composed by the color and grayscale image respectively, masked using the motion bitmap. The output image is equal to the original image in pixels where motion is detected and is zero otherwise (E, F in Figure 1). This kind of input represents the dot product between the original image and the binarized motion bitmap.

IV. Results

We split the experimental analysis into two steps. At first, a single network will be evaluated using different input formats to identify the best ways to use the additional data coming from SVS. The second experiment will then analyze the behavior of networks of different complexities when dealing with the various input configurations to understand if lower complexity networks can take advantage of different inputs compared to higher ones.

A. Benchmarking of different inputs

The first experiment analyzed the impact of the different types of input on the performance of a single network. The parameters used to generate the architecture are shown in Table I. This is the largest default architecture presented in [25].

TABLE I
HYPERPARAMETERS OF THE NETWORK USED IN EXPERIMENT 1

Resolution	α	B	β	t_0	**MAC**	**Parameters**
128×128	0.35	7	1	6	9.5 to 9.8 M	55.4 K

For each input type, the network has been trained three times for 300 epochs over the training dataset to minimize performance variance. Table II shows the average mAP reached for each of the different input configurations, together with the operation count for the network (which varies slightly due to the different number of input channels used).

TABLE II
ACHIEVED MAP FROM THE DIFFERENT INPUT TYPES ANALYSED IN EXPERIMENT 1

Input type	**Background subtractor**	**MAC**	**avg mAP**	σ
RGB	-	9.70M	52.4	10.1
RGBA	KNN	9.79M	70.7	1.51
RGBM	KNN	9.70M	47.1	2.66
G	-	9.49M	68.3	3.60
GA	**KNN**	**9.60M**	**71.6**	**2.31**
GA	**FD**	**9.60M**	**71.5**	**3.94**
GM	KNN	9.49M	60.0	5.42
MB	KNN	9.49M	56.3	3.95
MB	FD	9.49M	44.9	1.35
MB	MOG	9.49M	54.8	2.76
YUV	KNN	9.79M	70.6	2.31

As shown in Table II, the quality of different background subtraction approaches has been evaluated by training the

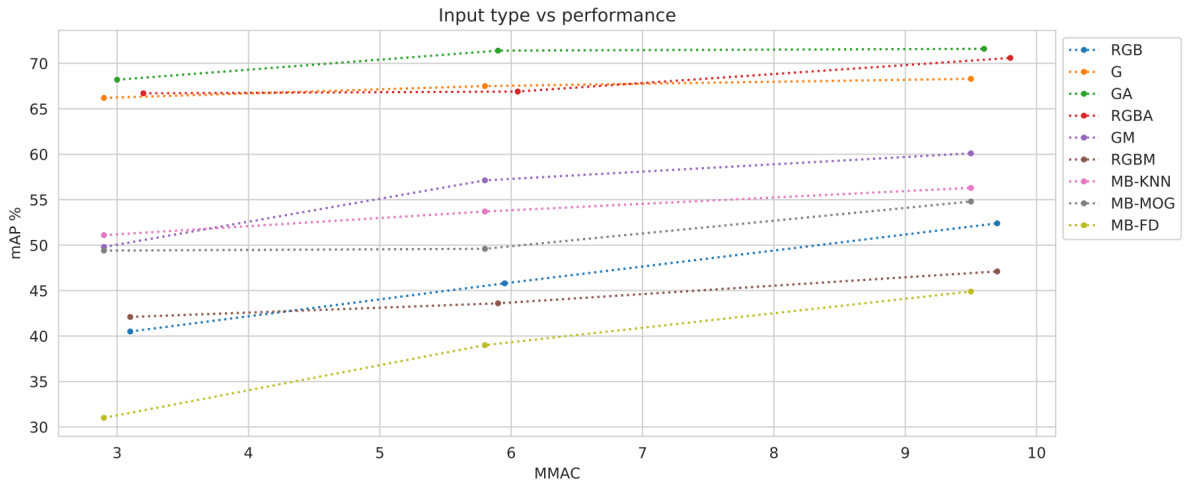

Fig. 2. Results of varying network sizes for different input types

network on a single channel image containing only the motion bitmap data (1 channel, binarised input). This has shown how the K-nearest neighbor-based background subtractor outperformed other approaches, but performance is not too far from using a MOG-based subtraction algorithm. The naive frame difference implementation is, as expected, beaten by more advanced methods but has still provided enough information for the network to detect objects in the frame. This experiment also shows that training the network on a more straightforward grayscale input consistently outperformed using a more feature-rich RGB implementation. This could be due to the fact that the network tested is relatively tiny, so having a feature-rich input composed of a higher number of channels leads to a more underfitting model with respect to using a more basic input image. Given this observation, we also tested the network on an image coded in YUV format. In this way, the complete information of the RGB data can still be used by the CNN while providing an easily interpretable luminance channel. However, the performance demonstrated in this configuration has been similar to using the original RGB color space, thus confirming the underfitting hypothesis.

Regarding the different possible ways to provide motion bitmap data to the network, it is notable how the addition of a dedicated input channel allows the network to make better use of this data, for a very low increase in computational cost (around 1%), as will be investigated in Section IV-C.
Another interesting result is the relatively close performance of the network working on grayscale data merged with the best performing background subtractor and the simplest frame-difference implementation. This shows that, while the standalone motion bitmap obtained through frame difference presents heavy artifacts that challenge the detection performance, it still contains meaningful information that can be extracted by the network to increase performance when coupled with the original image.

B. Network size variation

The following experiment aimed at understanding how different input formats influence the performance of networks of diverse computational complexities. Multiple networks were tested, of a varying number of operations between $3M$ and $10M$ multiply-accumulate operations, to understand if smaller networks would benefit more from less complex representations of the input data. As before, the different architectures were all trained for 300 epochs 3 times, and the average of the achieved mAP over the three runs has been reported.
Table III shows the generating hyperparameters for the architectures tested, taken from [25].

TABLE III
HYPERPARAMETERS USED TO GENERATE THE DIFFERENT CONFIGURATIONS TESTED

Resolution	α	B	β	t_0	**MAC**	**Parameters**
128×128	0.35	7	1	6	9.5 to 9.8 M	55.4 K
128×128	0.25	7	1	6	5.8 to 6.1 M	34.7 K
128×128	0.2	7	1	5	2.9 to 3.2 M	20.1 K

As shown in Figure 2, the input types offering the best performance across all network complexities are those joining the motion bitmap data to the captured frame by using an additional channel. This shows how SVSs data can significantly improve detection quality for scenarios where object detection from a static camera is the target - but the data coming from these sensors must be completed with image data to provide a complete representation to the network. Even for very low operation count nets ($\approx 3MMAC$), object detection from merged frame data and motion data is a much simpler task than detection from only the motion bitmap, no matter how accurate this may be. This is especially true for scenarios where partially occluded or overlapping objects need to be detected, as the additional input information, in this case, can allow the network to discern between a single blob and multiple superimposed targets (as can be seen from Figure 5 and Figure 6). Notably, using a single

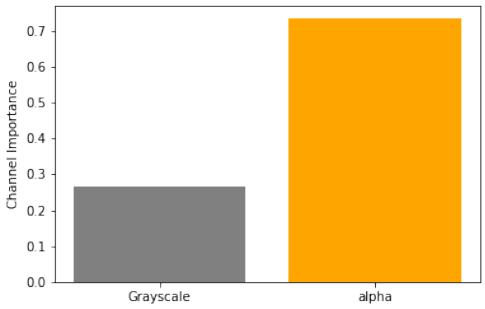

Fig. 3. Channel importance for the most extensive grey-scale configuration.

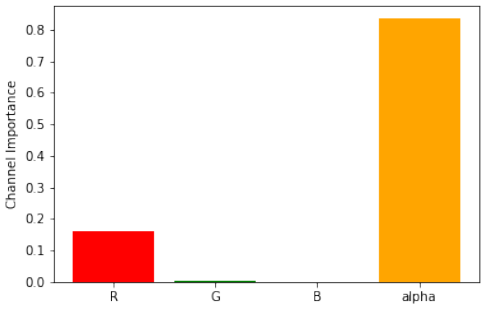

Fig. 4. Channel importance for the most extensive RGB configuration.

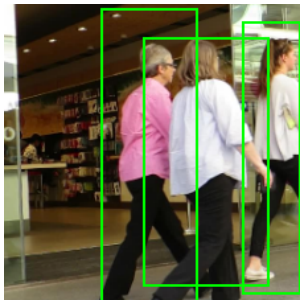

Fig. 5. All objects are identified with an RGB input.

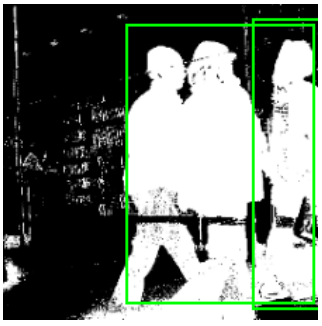

Fig. 6. Occluded objects are merged using only the motion bitmap.

channel, grayscale input provides performance close to the one obtained using these more comprehensive approaches. At the same time, while scaling network size, accuracy increases very slowly, as this type of input does not provide rich enough information content that more extensive networks can use.

With our experimental setup, the single-channel grayscale input showed excellent performance across the tested complexity range. Therefore, using preprocessing steps such as masking the input image does not seem to make sense from a performance standpoint. It is still interesting to note the performance trend for RGB images as, in some applications, color information could result in a more significant performance advantage than with our setup. This type of input shows an exciting trend in the analyzed complexity range where, with simpler networks, using a masked version of the input demonstrates better performance. On the other hand, larger networks benefit more from the higher amount of data in the full image. Using a binarised motion bitmap mask as input to a standard CNN does not provide notable advantages in this application, as performance is lower than using different input types. Additionally, smaller networks still show a significant performance loss compared to larger ones as the binarised input does not provide a direct interpretation. Instead, larger networks offer a better ability in reconstructing the semantic content of the frame in the case of partially occluded objects. Advantages of this kind of input may be more notable when using binarised neural networks or different processing techniques.

C. *Does the SVS matter?*

Models trained with the motion bitmaps as an additional channel of the input image are the best-performing ones. Therefore, we investigate how the trained networks weigh each input channel to validate the idea. We expect that, if the foreground mask is the key to boost the detection performance, the feature map generated by this channel (that can be studied given the architectural design of PhiNets) should have a considerably higher activation rate. To prove this idea, we assumed to have a random tensor as input of the network (random matrix with entries sampled by a uniform distribution between $[-1, 1]$). We computed the value of the activations for each channel over 500 different random inputs. After this, we calculated the covariance matrix and considered the diagonal elements (after dividing the matrix by the trace) as a metric of how meaningful every channel is. Results are summarised in Fig. 3 and Fig. 4 for the most computationally expensive network among the ones proposed in the manuscript. With this analysis, we confirm our hypothesis about the importance of the motion bitmap in the object detection pipeline and conclude that the training performed on the target datasets benefited from using the information of the SVS.

V. Conclusions

In this paper, we investigated the role of SVSs for object detection pipelines based on deep learning. In particular, we showed how an embedded-friendly detection pipeline - which exploits PhiNets - can be boosted by providing additional information based on in-sensor processing of the images. Overall, we conclude that the information contained in the motion bitmap is really valuable and helps the detector gain a 5% improvement in mAP without compromising the computational cost of the pipeline.

REFERENCES

[1] Gianmarco Cerutti, Rahul Prasad, Alessio Brutti, and Elisabetta Farella, "Neural network distillation on iot platforms for sound event detection.," in *Interspeech*, 2019, pp. 3609–3613.

[2] Robert Forchheimer and A Astrom, "Near-sensor image processing: A new paradigm," *IEEE Transactions on Image Processing*, vol. 3, no. 6, pp. 736–746, 1994.

[3] Angelo Garofalo, Manuele Rusci, Francesco Conti, Davide Rossi, and Luca Benini, "Pulp-nn: accelerating quantized neural networks on parallel ultra-low-power risc-v processors," *Philosophical Transactions of the Royal Society A*, vol. 378, no. 2164, pp. 20190155, 2020.

[4] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek, "Pruning by explaining: A novel criterion for deep neural network pruning," *Pattern Recognition*, vol. 115, pp. 107899, 2021.

[5] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, "Thinet: A filter level pruning method for deep neural network compression," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5058–5066.

[6] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[7] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh, "Improved knowledge distillation via teacher assistant," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 5191–5198.

[8] Michele Benetti, Massimo Gottardi, Tobias Mayr, and Roberto Passerone, "A low-power vision system with adaptive background subtraction and image segmentation for unusual event detection," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 11, pp. 3842–3853, 2018.

[9] Bulent Tavli, Kemal Bicakci, Ruken Zilan, and Jose M Barcelo-Ordinas, "A survey of visual sensor network platforms," *Multimedia Tools and Applications*, vol. 60, no. 3, pp. 689–726, 2012.

[10] Michela Lecca, Massimo Gottardi, Elisabetta Farella, and Bojan Milosevic, "Always-on low-power optical system for skin-based touchless machine control," *JOSA A*, vol. 33, no. 6, pp. 1015–1024, 2016.

[11] R. Etienne-Cummings, P. Pouliquen, and M. A. Lewis, "Single chip for imaging, color segmentation, histogramming and pattern matching," in *2002 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.02CH37315)*, 2002, vol. 1, pp. 38–442 vol.1.

[12] F. Boussaid, D. Chai, and A. Bouzerdoum, "On-chip skin detection for color CMOS imagers," in *Proceedings International Conference on MEMS, NANO and Smart Systems*, 2003, pp. 357–361.

[13] Nicola Cottini, Massimo Gottardi, Nicola Massari, Roberto Passerone, and Zeev Smilansky, "A 33uw 64×64 pixel vision sensor embedding robust dynamic background subtraction for event detection and scene interpretation," *IEEE Journal of Solid-State Circuits*, vol. 48, no. 3, pp. 850–863, 2013.

[14] Anthony Rowe, Adam Goode, Dhiraj Goel, and Illah Nourbakhsh, "Cmucam3: An open programmable embedded vision sensor," *Carnegie Mellon Robotics Institute Technical Report, RI-TR-07-13*, 01 2007.

[15] Manuele Rusci, Davide Rossi, Elisabetta Farella, and Luca Benini, "A sub-mw iot-endnode for always-on visual monitoring and smart triggering," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1284–1295, 2017.

[16] Francesco Paissan, Gianmarco Cerutti, Massimo Gottardi, and Elisabetta Farella, "People/car classification using an ultra-low-power smart vision sensor," in *2019 IEEE 8th International Workshop on Advances in Sensors and Interfaces (IWASI)*. IEEE, 2019, pp. 91–96.

[17] Bo Zhao, Xiangyu Zhang, and Shoushun Chen, "A cmos image sensor with on-chip motion detection and object localization," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2011, pp. 1–4.

[18] Jaehyuk Choi, Seokjun Park, Jihyun Cho, and Euisik Yoon, "A 3.4 μw cmos image sensor with embedded feature-extraction algorithm for motion-triggered object-of-interest imaging," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*. IEEE, 2013, pp. 478–479.

[19] Manuele Rusci, Davide Rossi, Michela Lecca, Massimo Gottardi, Elisabetta Farella, and Luca Benini, "An event-driven ultra-low-power smart visual sensor," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5344–5353, 2016.

[20] Zoran Zivkovic and Ferdinand Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, 2006.

[21] Zoran Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* IEEE, 2004, vol. 2, pp. 28–31.

[22] Stephen B Gray, "Local properties of binary images in two dimensions," *IEEE Transactions on computers*, vol. 100, no. 5, pp. 551–561, 1971.

[23] Satoshi Suzuki et al., "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[24] Francesco Paissan, Massimo Gottardi, and Elisabetta Farella, "Enabling energy efficient machine learning on a ultra-low-power vision sensor for iot," *arXiv preprint arXiv:2102.01340*, 2021.

[25] Francesco Paissan, Alberto Ancilotto, and Elisabetta Farella, "Phinets: a scalable backbone for low-power ai at the edge," *arXiv preprint arXiv:2110.00337*, 2021.

[26] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian D. Reid, Stefan Roth, and Laura Leal-Taixé, "Motchallenge: A benchmark for single-camera multiple target tracking," *CoRR*, vol. abs/2010.07548, 2020.

[27] Joseph Redmon and Ali Farhadi, "YOLO9000: better, faster, stronger," *CoRR*, vol. abs/1612.08242, 2016.

[28] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[29] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, 2004, vol. 2, pp. 28–31 Vol.2.

[30] Zoran Zivkovic and Ferdinand van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognition Letters*, vol. 27, no. 7, pp. 773–780, 2006.

[31] Thomas Pock, Martin Urschler, Christopher Zach, Reinhard Beichel, and Horst Bischof, "A duality based algorithm for tv-l1-optical-flow image registration," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2007*, Nicholas Ayache, Sébastien Ourselin, and Anthony Maeder, Eds., 2007.