Gaussian Mixture Background Modelling Optimisation for Micro-controllers

Claudio Salvadori[1], Dimitrios Makris[2], Matteo Petracca[3,1], Jesus Martinez del Rincon[2], and Sergio Velastin[2]

[1] Real-Time Systems Laboratory, Scuola Superiore Sant'Anna, Pisa, Italy
[2] Digital Imaging Research Centre, Kingston University, London, United Kingdom
[3] National Laboratory of Photonic Networks, CNIT, Pisa, Italy

Abstract. This paper proposes an optimisation of the adaptive Gaussian mixture background model that allows the deployment of the method on processors with low memory capacity. The effect of the granularity of the Gaussian mean-value and variance in an integer-based implementation is investigated and novel updating rules of the mixture weights are described. Based on the proposed framework, an implementation for a very low power consumption micro-controller is presented. Results show that the proposed method operates in real time on the micro-controller and has similar performance to the original model.

1 Introduction

Background modelling is a fundamental task for many computer vision applications such as motion detection, object tracking and action recognition. Among the background modelling methods that have been proposed in the literature (see [1], [2], [3] for extended surveys), the Gaussian Mixture Model (GMM) approach [4] has proved very popular. However, its high computational and memory requirements restrict the type of processors in which the GMM algorithm can be deployed. On the other hand, with the increasing demand on the number and cost-effectiveness of video monitoring devices (driven by security concerns but also by applications such as gaming and intelligent spaces), it is highly desirable to explore new computer vision algorithms designed for embedded systems. For this work we have selected a particular micro-controller: the Microchip PIC32 [5], which is typical of the type of low-cost low-power devices used in embedded systems. This family of devices lack a dedicated hardware floating-point processing unit (FPU) and consequently must rely on floating point software libraries or fixed-point/integer algorithmic variants. This limitation represents a major challenge for the implementation of most computer vision and signal processing algorithms.

The goal of this work is to explore ways in which the GMM algorithm could be used on micro-controllers with low power consumption. In particular, an integer precision approach is proposed to reduce both the computational cost and the memory footprint of the algorithm. Nevertheless, this algorithm has to

The final publication is available at http://link.springer.com/chapter/10.1007%2F978-3-642-33179-4_24

be able to approximate the original implementation (double precision) without significant loss of robustness under changing light conditions and accuracy for foreground/background segmentation.

Considering micro-controller based architectures, different approaches for background modelling have been proposed in the literature. In [6] and [7] the background is modelled using the *median filter* while [8] use *running averages*, and in [9] a combination of both methods is used. These approaches have low computational requirements, because they use only one parameter to model the background. However, the usage of only one parameter implies also a limitation since it does not explicitly model the foreground and it is likely to underperform in comparison to GMM, specially for complex scenes.

The rest of the paper is organised as follows: in Sec. 2 the methodology to approximate the GMM updating rules is described, given the micro-controller constraints and the operative limits of this approach. In Sec. 3 the implementation constraints imposed by the chosen hardware are discussed. In Sec. 4 the performance evaluation of the implementation using two Gaussians is presented, while conclusions follow in Sec. 5.

2 Methodology

Gaussian Mixture Model is a method to model the background view for a static camera so as to facilitate the separation between foreground and background, on the assumption that over time a pixel is mostly background. One of its most significant characteristic is the capability to adapt to illumination changes at the expense of the inclusion of stationary objects into the background. Each pixel of the background model is separately described as a mixture of G Gaussians, where each one of them is represented by three parameters: mean-value μ, variance σ^2 and a weight w. In Table 1 the ranges of the three parameters are shown. It is particular important to consider the range of the variance: for this parameter a lower bound σ^2_{min} and an upper bound σ^2_{MAX} are defined to prevent the degeneration of the Gaussian distribution into a *Dirac delta* or a *uniform* distribution respectively.

Table 1: Gaussian parameters ranges

μ	σ^2	w
[0, 255]	$[\sigma^2_{min},\sigma^2_{MAX}]$	[0, 1]

In a mixture, the Gaussians are ordered accordingly to a *sorting rule*, based on the value of ρ [4] (see Eq. (1) below), and splitted into two subsets: the *background sub-set* and the *foreground sub-set*. GMM allows the labelling of every pixel in every image accordingly so as to discriminate the foreground from the background.

$$\rho = \frac{w^2}{\sigma^2} \tag{1}$$

This section presents a general method to deploy the GMM algorithm on micro-controllers with computation and memory constraints (see Sec. 3). Due

to these two constraints an integer precision implementation is required to fit the algorithm on a chosen micro-controller. In particular, the updating rules for the Gaussian parameter must be addressed. Inevitably, an integer implementation restricts the range and/or the granularity of the Gaussian parameters. For instance, the number of bits that are allocated to the mean-value and variance of each Gaussian directly affects their range and granularity. The range of learning-rates of GMM algorithm is also affected, however in a more complex way. Specifically, each approximation (mean-value, variance, weight) introduces a lower bound for the learning-rate (see Eq. (2), where α^{w}_{min}, α^{μ}_{min} and α^{σ}_{min} are the lower bounds generated by the updating rules limitation of the weight, the mean-value and the variance, respectively).

$$\bar{\alpha}_{min} > MAX\{\alpha^{w}_{min}, \alpha^{\mu}_{min}, \alpha^{\sigma}_{min}\} \tag{2}$$

In this way, a learning-rate *operating range* is defined as $[\bar{\alpha}_{min}, 0.99]$.

2.1 Mean-value and variance updating

In the GMM algorithm the mean-value and the variance of a given Gaussian of the mixture are updated if and only if the value of the pixel *"belongs"* to that Gaussian: consequently a *belonging criterion* is defined. In this case a pixel, represented by its value $p_{i,j}$, "belongs" to a Gaussian if the condition shown in Eq. (3) is satisfied.

$$(\mu_{g,(i,j)} - p_{i,j})^2 < T\sigma^2_{g,(i,j)} \tag{3}$$

where $\mu_{g,(i,j)}$ and $\sigma^2_{g,(i,j)}$ are respectively the mean value and the variance of the g-th Gaussian of the mixture and T is a threshold fixed a priori.

To redefine the mean-value and the variance updating rules, and to make them more appropriate for an embedded system, the concepts of *granularity* and *updating step* are introduced. Let assume a sequence of ordered natural numbers $\mathbf{B} = \{b_i \in \mathbb{N} | b_{i+1} - b_i = \gamma\}$, the *granularity* of this set is the value γ. The parameter *updating step* ξ is defined as a function of the *granularity* ($\xi = \mathcal{F}(\gamma)$), so as to characterize a *generalised-round* operation *G_ROUND* as shown in Eq. (4).

$$b = G\_ROUND(a, \xi, \gamma) = \begin{cases} \left\lfloor \frac{a}{\gamma} \right\rfloor * \gamma + \gamma & \text{if } \left(a - \left\lfloor \frac{a}{\gamma} \right\rfloor * \gamma\right) \geq \xi \\ \left\lfloor \frac{a}{\gamma} \right\rfloor * \gamma & \text{if } \left(a - \left\lfloor \frac{a}{\gamma} \right\rfloor * \gamma\right) < \xi \end{cases} \tag{4}$$

where a is the number to be rounded. In the case considered here, the parameter *updating step* is assumed to be half the *granularity* ($\xi = \frac{\gamma}{2}$). Consequently, the updating rules of the mean-value and the variance are defined as shown respectively on Eq. (5) and Eq. (6).

$$\mu_{i+1} = G\_ROUND(\mu_i + k_i d_i, \xi_\mu, \gamma_\mu) \tag{5}$$

$$\sigma^2_{i+1} = G\_ROUND(\sigma^2_i + k_i(D_i - \sigma^2_i), \xi_\sigma, \gamma_\sigma) \tag{6}$$

where $k_i = \frac{\alpha}{w_i}$, p_i is the current value of the pixel, $d_i = p_i - \mu_i$ and $D_i = d^2_i$.

Thus, the mean-value and the variance updating rules introduce lower bound limits on the range of learning-rate values. These limits are required to emulate similar behaviours in both integer and double precision implementations. These lower boundaries are defined by detecting the worst possible case for a given condition. Thus, the worst case for the mean-value updating rule is considered by choosing a background pixel ($w \simeq w_{max} = 1$, consequently $k \simeq k_{min} = \alpha$, and $\sigma \simeq \sigma_{min}$) with its value close to the borders of the interval defined by the *belonging criterion* ($p = \lfloor \mu \pm \sigma_{min}\sqrt{T} \rfloor$). The condition described by Eq. (7) imposes this lower bound limit.

$$\alpha^{\mu}_{min} = \frac{\xi_{\mu}}{\left\lfloor \sigma_{min}\sqrt{(T)} \right\rfloor} \tag{7}$$

On the other hand, the condition reported in Eq. (8) imposes on the variance updating rule the restriction of a lower bound σ^2_{min} in updating the parameter:

$$\alpha^{\sigma}_{min} = \frac{\xi_{\sigma}}{\sigma^2_{min} + 1} \tag{8}$$

2.2 Weight updating

Eq. (9) shows the updating rule when the intensity of a pixel k belongs to Gaussian g ($g \in [1, G]$). If the intensity of the pixel does not belong to the Gaussian, the weight value will be decreased as it is shown on Eq. (10).

$$w^g_k(s+1) = (1-\alpha) * w^g_k(s) + \alpha \tag{9}$$

$$w^g_k(s+1) = (1-\alpha) * w^g_k(s) \tag{10}$$

where g is the Gaussian index ($g \in [1, G]$), k is the pixel index and s is the counter that indicates how many times the value of the k-th pixel of an image belongs the g-th Gaussian during its temporal evolution, and α is the learning-rate.
Assuming that the intensity of pixel k belongs to Gaussian g for s consecutive frames and solving the iterative Eq. (9) leads to Eq. (11).

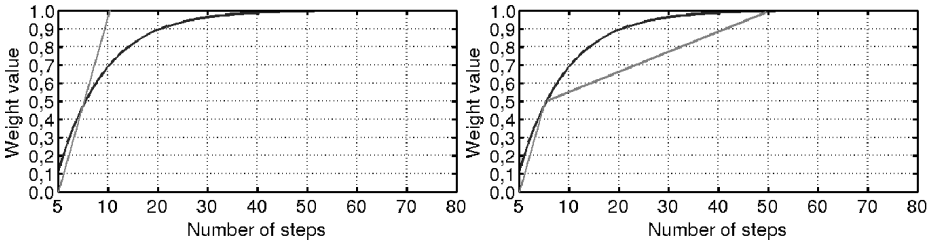
$$w^g_k(s) = 1 - (1-\alpha)^s \tag{11}$$

Finally, solving Eq. (11) and retrieving the value s ($\equiv S(w, \alpha)$), Eq. (12) is obtained. Using this formula it is possible to calculate the number of iterations needed to reach a certain value of weight w.

$$S(w, \alpha) = \frac{\log_{10}(1-w)}{\log_{10}(1-\alpha)} \tag{12}$$

In this situation it is possible to simplify the weight updating rule as an operation of the increase or decrease of an integer counter, which implies a significant gain on performance and a reduction of hardware requirements.

However, Eq. (11) and Eq. (12) can not be directly implemented due to the particular micro-controller features, such as the lack of floating-based capabilities. Consequently two linearly approximated solutions are defined to retrieve the



(a) Single-line approximation. (b) Poly-line approximation.

Fig. 1: The weight trend and its approximations.

weight value from the integer counter. Fig. 1a and 1b depict both approaches (gray curves) and its comparison with the logarithmic relation of the weight described on Eq. (12) (black curves).

Similarly to Sec. 2.1, both solutions introduce a lower bound on the learning-rate range. If $\Sigma_1(\alpha)$ is the maximum number of iterations needed by the chosen approximation and learning-rate α ($\Sigma_1(\alpha)$ is the number of steps to reach the value 1, the maximum in the case of weight), the relation shown in Eq. (13) holds, where the *maximum value* L that can be represented is $L = 2^l$, for a given number l of bits required to represent the integer counter.

$$\Sigma_1(\alpha^w) \leq L \tag{13}$$

Assuming that the function $\Sigma_1(\alpha)$ is known, it is simple to extract the learning-rate lower bound (α^w_{min}) from Eq. (13) (see Eq. (14)).

$$\alpha \geq \alpha^w_{min} \tag{14}$$

Although this method is general and extensible to any number of Gaussians per pixel, given the hardware constrains of the micro-controller chosen for this work, the number of Gaussians is $G = 2$ and every plot and figure shown in the next sections is related to this case.

Single-line approximation. This is based on the approximation of the logarithmic curve described on Eq. (12) by means of a line passing through the origin and the point $P_0 = (S(w_0, \alpha), w_0)$ (with $w_0 = 0.50$) as shown in Fig. 1a and described in Eq. (15), where $m = 2S(w_0, \alpha)$, and the value of steps to increment and to decrement the counters are both equal to 1. The choice of the point P_0 is because around this the swap between the foreground and the background sub-sets happens. Therefore, it is a critical point for achieving a correct emulation of double precision implementation behaviours.

$$w(s) = \frac{s}{m} \tag{15}$$

Consequently, knowing that $\Sigma_1 = 2S(w_0, \alpha)$ the relation shown in Eq. (16) holds.

$$2S(w_0, \alpha^w_{min}) \leq L \tag{16}$$

Finally from Eq. (16) and from Eq. (12), the lower bound of the learning-rate is defined as shown in (17)

$$\alpha^w_{min} = 1 - (1 - w_0)^{\frac{2}{L}} \tag{17}$$

Poly-line approximation. Let the two ordered Gaussians be G_1 (the background) and G_2 (the foreground). The relation shown in Eq. (18) and the consequent condition shown in Eq. (19) are valid:

$$w_{max} + w_{min} = 1 \tag{18}$$

$$\begin{cases} w_{max} \geq \frac{1}{2} \\ w_{min} \leq \frac{1}{2} \end{cases} \tag{19}$$

From the condition shown on Eq. (19), it is possible to derive a two-line poly-line approximation of the logarithmic weight trend (see Fig. 1b). The first line of the poly-line (l_1) passes through the origin and the point $P_l = (S(w_l, \alpha), w_l)$ (with $w_l = 0.50$) and maps the lowest weight Gaussian. The second one (l_2) passes through the point P_l and the point $P_h = (S(w_h, \alpha), w_h)$ (with $w_h = 0.99$) and maps the larger weight Gaussian.

Because of Eq. (18), it is possible to represent the two weights (and consequently the two lines) using only one of them. As shown in Fig. 2, line l ($\equiv l_2$) is selected and translated on the origin (line l_0, see Eq. (20), where $m = S(w_h, \alpha) - S(w_l, \alpha) = b - a$, and $\bar{s}$ and $\bar{w}$ are respectively the number of steps and the weight on the translated domain) to simplify the computation.

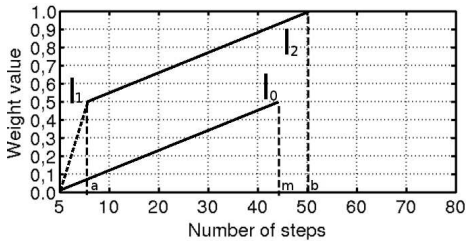


Fig. 2: Poly-line approximation, using a single line.

$$l_0 : \bar{w} = \frac{\bar{s}}{2m} \tag{20}$$

In this configuration, the line l_0, that maps the counter $\bar{s}$, has the same slope of l_2 (the greatest value weight Gaussian), up to some translation constants. Consequently, in the case where the current pixel belongs to the Gaussian related with l_2, the weight updating rule consists on increasing the counter by 1. On the other hand, when the pixel belongs the Gaussian with l_1 the weight updating rule consists on subtracting from $\bar{s}$ a constant derived from the mapping of the line l_1 over l_0. The above mentioned constant is the ratio between the slope of l_1 ($\frac{1}{2a}$) and the slope of l_0 ($\frac{1}{2m}$) (see Eq. 21).

$$STEP = \left\lceil \frac{m}{a} \right\rceil \tag{21}$$

Moreover, from Eq. (20), the relationship between the weight values (w_{max} and w_{min}) and the value $\bar{s}$ (see Eq. (22) and Eq. (23)) can be derived:

$$w_{max} = \frac{\bar{s} + d}{2 * d} \tag{22}$$

$$w_{min} = \frac{d - \bar{s}}{2 * d} \tag{23}$$

Usually the greatest weight Gaussian is labelled as G_1. However, because the sorting rule is based on the values of ρ (see Eq. (1)), in some cases the above mentioned assumption is not valid. For this reason one bit of the weight representation is used as a descriptor for associating the counter correctly to the Gaussian with the largest weight. In this case the learning-rate lower bound is defined by the condition in Eq. (24). Solving Eq. (24), the relation shown in Eq. (25) is retrieved.

$$S_1 = S(w_h, \alpha^w_{min}) - S(w_l, \alpha^w_{min}) \leq L \tag{24}$$

$$\alpha^w_{min} = 1 - \left(\frac{1 - w_h}{1 - w_l}\right)^{\frac{1}{L}} \tag{25}$$

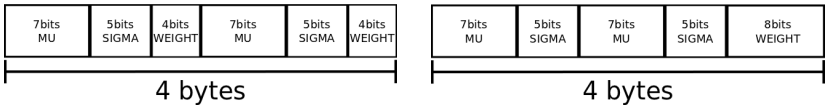
3 Implementation constraints

Because of the use of micro-controllers such as PIC32MX795F512L [5] (80MIPS, 128 KBytes of RAM), gray-levels images with 8-bits depth and resolution QQ-VGA (160x120) are considered. As described on the previous sections, two integer precision approaches are used, to reduce the computational cost of the algorithm processing (PIC32 family micro-controllers have no FPU and the floating-point numbers algorithms [10] are implemented using software libraries), and the algorithm memory footprint. As we can see in Table 2 both floating-point representations exceed the available memory footprint of the micro-controller (128 KBytes of RAM). However, even the integer representation (uint8_t) uses a large amount of memory and it does not permit the implementation of any other additional image processing algorithm or any communication stack. To address this problem, a two bytes per Gaussian approach is implemented. Consequently using QQ-VGA images and mixtures of two Gaussians, the algorithm memory footprint is 76,800 Bytes.

Table 2: Algorithm footprint (QQ-VGA images and mixture of 2 Gaussians)

Precision	Bytes per parameter	Bytes per Gaussian	Footprint (Bytes)
Double	8	24	921,600
Float	4	12	460,800
uint8_t	1	3	115,200

In Fig. 3 the two bytes per Gaussian representations are shown: the first one, shown in Fig. 3a, is used to fit the single-line approximation approach and the second one, shown in Fig. 3b, to fit the poly-line approximation.



(a) Single-line representation. (b) Poly-line representation.

Fig. 3: 2 bytes per Gaussian representations

3.1 The learning-rate operating range

In this section the *learning-rate operating range* is computed taking into account the different learning-rate lower bounds resulting from the constraints related to the integer implementation of the Gaussian parameter updating rules. Because of $T \in [15, 35]$, the worst case is chosen ($T = 15$). Because both implementations differ only on weight representation, the learning-rate lower bounds dictated by the mean-value and the variance are the same (see Table 3).

Table 3: μ and σ^2 representation constraints

Parameter name	Definition range	N. bits	Granularity γ	Updating step ξ	α_{min} value
μ	[0, 255]	7	2	1	0.125
σ^2	[5, 36]	5	1	0.5	0.083

On the other hand, since the learning-rate lower bounds introduced by the weight depends on which approximation is chosen, the limits introduced by both representations are summarised in Table 4.

Table 4: Weight representation constraints

Approximation type	N. bits	Maximum value L	α^w_{min} value
Single-line	4	15	0.088
Poly-line	7	127	0.030

Finally, since the learning-rate lower bound is obtained from Eq. 2, *learning-rate operating range* is defined as $[0.15, 0.99]$ and all the performance tests shown on the next section will be evaluated over this range.

4 Performance evaluation

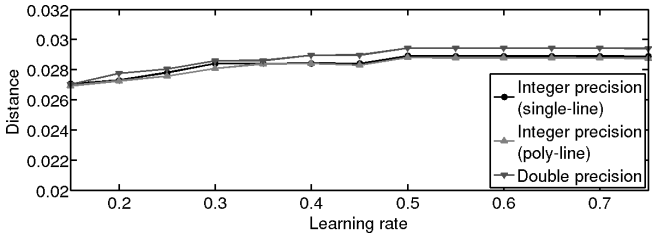
The performance of the proposed algorithm is presented in this section. For all the experiments, it is assumed that $T \in [15, 35]$ and $\alpha \in [0.15, 0.75]$.

4.1 Integer precision vs. double precision: performance comparison

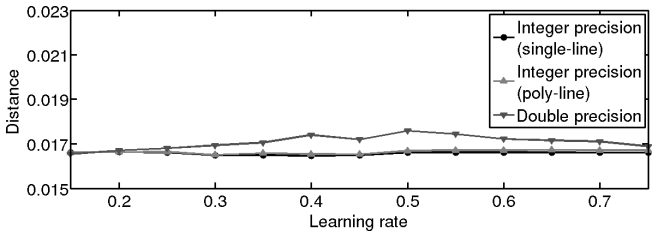
The comparison between the performance of the integer precision and the double precision is described by means of *precision-recall* diagrams. Particularly, the minimum distance between the P-R curve and the optimal point (the point (1,1)) is computed and plotted. In Fig. 4a and in Fig. 4b this minimum distance is plotted using the two different data-sets: the "IXMAS data-set" [11] (to represent situations with slow movement) and the "Fudan Pedestrian data-set" [12] (to represent situations with fast movement). As we can see in these plots, the performance in the two integer precision approaches is similar and comparable with the results coming from the double precision implementation experiments.

4.2 Integer precision: memory footprint and processing time

Both approaches have been implemented in the SEED-EYE [13] board, based on Microchip PIC32MX795F512L micro-controller. In the board, two different



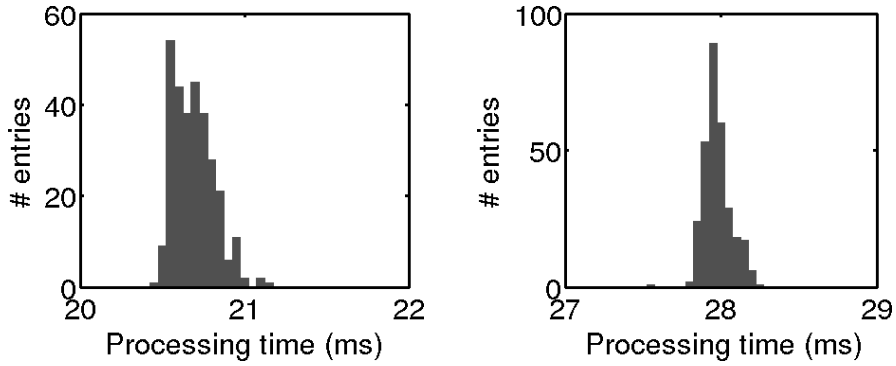
(a) Using "IXMAS" data-set.



(b) Using "Fudan pedestrian" data-set.

Fig. 4: Distance from the P-R to the PR optimal point (1,1)

firmwares able to acquire, store and process images have been deployed. The memory footprint generated by these applications is the same (both are based on two bytes per Gaussian representation) and use 76% of the micro-controller memory. In Fig. 5 the processing times of the two approaches are shown. The average processing time for single-line approach (Fig.5a) is about 21ms, and for the poly-line one (Fig.5b) is about 28ms. For the type of videos that were considered, both approaches operate in real-time as the achieved frame-rate is more than 30fps (33ms/frame).



(a) Single-line approach (b) Poly-line approach

Fig. 5: Processing time distributions.

5 Conclusions

In this paper two integer precision embedded-oriented implementations of mixture of Gaussian background modelling have been described. Particularly a

learning-rate operating range is computed, arising from both an integer implementation and the limited number of bits assigned to each parameter. Instead the retrieved range is $\alpha \in [0.15, 0.99]$. Furthermore, the results coming from the integer precision implementations are compared with the results generated by a double precision one using the P-R curves. From this comparison it is possible to say that the two integer precision approaches have similar performance to the double precision one, in the above-mentioned learning-rate range. Finally, using the SEED-EYE board, the two implementations have a memory footprint of the 76% of the micro-controller memory and a mean processing time of about 21ms and about 28ms for the single-line approach and the poly-line one respectively, both of them less than the real-time constraints of 33ms (30fps).

References

1. Piccardi, M.: Background subtraction techniques: a review. In: IEEE International Conference on Systems, Man and Cybernetics. Volume 4. (2004) 3099–3104
2. Cheung, S.C.S., Kamath, C.: Robust techniques for background subtraction in urban traffic video. In: Visual Communications and Image Processing. Volume 5308. (2004) 881–892
3. Radke, R.J., Andra, S., Al-Kofahi, O., Roysam, B.: Image change detection algorithms: A systematic survey. IEEE Transactions on Image Processing **14** (2005) 294–307
4. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Volume 2. (1999) 2246–2252
5. Microchip Technology Inc.: PIC32MX3XX/4XX Family Data Sheet. ww1.microchip.com/downloads/en/DeviceDoc/61143E.pdf (2008)
6. McFarlane, N.J.B., Schofield, C.P.: Segmentation and tracking of piglets in images. Machine Vision and Applications **8** (1995) 187–193
7. Hung, M.H., Pan, J.S., Hsieh, C.H.: Speed up temporal median filter for background subtraction. In: Proceedings of International Conference on Pervasive Computing, Signal Processing and Applications. (2010) 297–300
8. Rahimi, M., Baer, R., Iroezi, O.I., Garcia, J.C., Warrior, J., Estrin, D., Srivastava, M.: Cyclops: In situ image sensing and interpretation in wireless sensor networks. In: Proceedings of ACM Conference on Embedded Networked Sensor Systems. (2005) 192–204
9. Iannizzotto, G., La Rosa, F., Lo Bello, L.: A wireless sensor network for distributed autonomous traffic monitoring. In: Conference on Human System Interactions. (2010) 612–619
10. IEEE Computer Society: IEEE Standard for Floating-Point Arithmetic. Technical report, Microprocessor Standards Committee of the IEEE Computer Society, 3 Park Avenue, New York, NY 10016-5997, USA (2008)
11. Weinland, D., Ronfard, R., Boyer, E.: Free viewpoint action recognition using motion history volumes. Computer Vision and Image Understanding **104** (2006) 249–257
12. Tan, B., Zhang, J., Wang, L.: Semi-supervised elastic net for pedestrian counting. Pattern Recognition (2011)
13. Scuola Superiore Sant'Anna and Evidence s.r.l.: SEED-EYE BOARD. http://www.evidence.eu.com/products/seed-eye.html (2011)