Optimized Parallel Model of Covariance Based Person Detection

Nesrine Abid[1(✉)], Kais Loukil[1], Walid Ayedi[1], Ahmed Chiheb Ammari[2,3], and Mohamed Abid[1]

[1] Laboratory of Computer and Embedded Systems, National School of Engineering of Sfax, Sfax University, Sfax, Tunisia
nesrineabid88@gmail.com
[2] MMA Laboratory, National Institute of the Applied Sciences and Technology, Carthage University, Carthage, Tunisia
[3] Renewable Energy Group, Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia

Abstract. Covariance descriptor has good performance for person detection systems. However, it has high execution time. Multiprocessors systems are usually adopted to speed up the execution of these systems. In this paper, an optimized parallel model for covariance person detection is implemented using a high-level parallelization procedure. The main characteristics of this procedure are the use of Khan Process Network (KPN) parallel programming model of computation, and the exploration of both task and data levels of parallelism. For this aim, a first KPN parallel model is proposed starting from the block diagram of the covariance person detection application. This model is implemented through the Y-Chart Application Programmers Interface (YAPI) C++ library. To ensure the best workload balance of the optimized model, communication and computation workload analysis are considered. Based on these results, both task merging and data-level partitioning are explored to derive an optimized model with the best communication and computation workload balance. The optimized parallel model obtained has three times lower execution time in comparison with the sequential model.

Keywords: Covariance descriptor · Person detection · Mpsoc · KPN · Parallel model

1 Introduction

Person detection is exploited as a key operation in video surveillance, and in many other fields. The large variability of the target, which is subjected to occlusions, pose, appearance and shape variations, presents a hard issue. To solve these problems, many solutions have been proposed. Among these solutions, a classifier that operates by means of a sliding window over the image is used. This classifier can be fed with heterogeneous features, e.g. Wavelet-based features [1], Haar-like characteristics [2], Histograms of oriented gradients (HOG) based features [3].

V. Murino and E. Puppo (Eds.): ICIAP 2015, Part II, LNCS 9280, pp. 287–298, 2015.
DOI: 10.1007/978-3-319-23234-8_27

The experiments in [3] show that the HOG descriptor is an excellent alternative of many others antecedent descriptors. In [4] authors proposed a region covariance descriptor technique for person detection that outperforms the previous approaches. Authors in [5] show that person detection with covariance (COV) even outperforms the HOG based solution. The covariance descriptor combines location, shape and color information such as pixel coordinates intensity, gradients, etc. and is invariant to color, luminance and pose. Recently, COV approaches have received the attention of many researchers [6] that are looking for detectors capable to achieve high accuracy. Nevertheless, in video surveillance the processing speed required for person detection is also a primary issue. To achieve real time detection, multiprocessing approaches are motivated to enable for sharing the system execution time between several processing elements. Any processing element can be either a processor that is executing a software (SW) task or any hardware accelerator that is implementing the needed task directly in hardware [7]. Now, prior to any multiprocessing implementation, a parallel specification of the system application is required. This parallel model has to have the best characteristics in terms of process workload and inter-process communication balances. In this paper, we propose an optimized parallel model for the covariance person detection using a high-level independent target-architecture parallelization procedure. The main characteristics of this procedure are the use of Khan Process Network (KPN) parallel programming models of computation [8], the exploration of task and data levels of parallelism with the minimal communication granularity, and the analysis of both communication and computation workloads for the best balance of the parallel model under investigation. For this aim, a first KPN parallel model is proposed starting from the block diagram of the covariance person detection application. This model is implemented through the Y-Chart Applications Programmers Interface (YAPI) C++ library [9]. Both communication and computation workloads analysis are then considered. Based on these results, task merging and data-level task splitting are explored to get the best communication and computation workload balance.

The paper is organized as follows: the following section 2 presents the covariance based person detection algorithm. In section 3, we will discuss the different steps of the parallelization procedure implemented to get the optimized parallel model. Finally, conclusions of the paper are given in Section 4.

2 Person Detection Based on Covariance Descriptor

The person detection system we are targeting is based on a covariance descriptor [4] followed by a Support Vector Machine (SVM) classification [10].

The block diagram of the covariance person detection system is shown in Fig.1. It is composed by four modules. The first module extracts features from the image. The second module exploits the fact that covariance may be computed by adopting integral representations under the form of first-order and second-order P and Q tensors [4]. These tensors are exploited by the third module. A sliding window scans

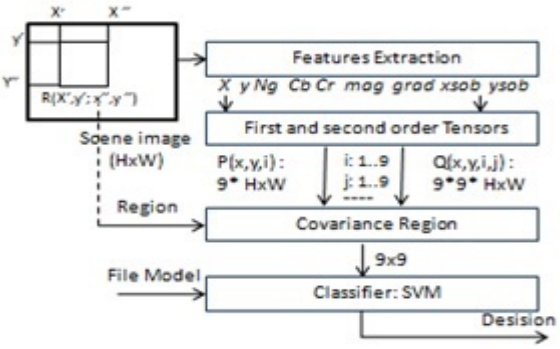

Fig. 1. Schematic of the covariance person detection application

the entire image to calculate a covariance matrix for each window. Final the fourth module builds the entire object model to perform the classification. Next, more details are given to better explain the role of each module and our first optimization for the sequential code.

2.1 Features Extraction

On the scene image we perform feature extraction, sequentially for each pixel, in a row wise fashion. The choice of the desired features and their number (d = 9), is based on Michael et al.study [11] where F(x,y) feature is extracted from each pixel such that: $F(x,y)=[x \quad y \quad Ng \quad Cb \quad Cr \quad mag \quad grad \quad xsob \quad ysob]^T$ where :

x: x location; y: y location; Ng: Grayscale intensity value; Cr: Read component value; Cb: Bleu component value; xsob and ysob: Norm of the first order derivatives in x and in y; grad: Sobel Gradient; mag: Sobel Magnitude.

Given the feature extraction principle, the x and y locations, the Grayscale intensity, and the Cr and Cb color component values can be computed in parallel. Once the Grayscale intensity value is extracted, the first order derivatives (xsob and ysob) can be computed in parallel. Using the first order derivatives, the second-order derivatives (grad and mag) can also be computed in parallel. The Sobel filters are designed to produce pixel by pixel output, as long as consecutive local areas. Pixels processed by this module will be forming a feature vector to be sent to the next module.

2.2 First and Second Order Tensors

In this task, the first-order integral tensor P and the second-order integral tensor Q are computed. P is a 9-dimensional vector encoding the sum of each feature defined by equation (1). Q is 9x9 array defined by equation (2) as given next.

$$P(x', y', i) = \sum_{x<x', y<y'} F(x, y, i) \qquad i=1..9 \qquad (1)$$

$$Q(x', y', i, j) = \sum_{x<x', y<y'} F(x, y, i)\, F(x, y, j) \qquad i,j=1..9 \qquad (2)$$

Where (x', y') is the upper left coordinate and (x", y") is the lower right coordinate of the rectangular region R(x',y';x",y") presented in Fig 1.

For each pixel from the image scene, nine features values are generated. The tensor processing will be using the feature extraction and thus can not start performing unless the first feature vector is available. Taking advantage from the major "symmetric characteristic" advantage of the covariance descriptor, we propose first to reduce the number of Q elements from 81(=9x9) to 45 (=9* (9 +1) / 2) as shown in Fig.2. To speed up processing, a coarse-grained task level parallelism can be implemented to separately compute the P and Q elements. In particular, the 45 elements of Q can be calculated in parallel with the 9 elements of P. At the end, a total of 54 vector elements for each pixel will be sent to the following modules.

Original code	*Our optimized code*

For(y=0;y<**9**;y++)
  For (x=0;x<**9**;x++)
  Q[y][x]=integral(F[x],F[y])
	For(y=0;y<**9**;y++)
	  For (x=0;x<**5**;x++)
	  Q[y][x]=integral(F[x],F[y])

Fig. 2. First optimization

2.3 Covariance Region

For each sliding widow region R (x', y'; x", y"), a covariance matrix of 9x9 elements is computed as follows:

$$\mathrm{Cr}(x',y';x",y") = \frac{1}{n-1}[Q_{x",y"} + Q_{x',y'} - Q_{x",y'} - Q_{x',y"} - \frac{1}{n}(P_{x",y"} + P_{x',y'} - P_{x",y'} - P_{x',y"}) \quad (3)$$

where n = (x' − x")·(y' − y").

Taking advantage from the symmetric characteristic we propose also to compute only 45 elements of the covariance as shown in Fig.3.

Original code	*Our optimized code*

For each image region:
  For(y=0;y<**9**;y++)
  For (x=0;x<**9**;x++)
  cov[y][x]=Cr(x',y';x",y")
	For each image region:
	  For(y=0;y<**9**;y++)
	  For (x=0;x<**5**;x++)
	  cov[y][x]=Cr(x',y';x",y")

Fig. 3. Second optimization

Each descriptor is transformed to Euclidean space to be fed to the classifier.

2.4 The Classifier

The SVM classifier classifies unlabeled descriptors based on their similarity with descriptors in their training sets. Two classes are considered one for person and the other class is for no person.

3 Parallelization Procedure

Implementing a covariance based person detection application for an embedded System-on-Chip is a big challenge. First, we start by the software implementation of this application using a mono processor architecture running on a stratixII_2s60_RoHS Altera FPGA. The computation has taken 44 s to detect persons from a 150X99 image. To achieve faster processing for real time detection, a multiprocessing approach is motivated and a parallel model of the application is required. In this case, we will be using the KPN parallel programming models of computation. In addition, an appropriate partitioning analysis is performed to ensure for the best computation and communication balance of the proposed parallel model.

3.1 Kahn Process Networks

Prior to any multiprocessor implementation, a parallel specification is required to functionally describe the studied application as a set of processes exchanging data according to an appropriate model of computation. Many dataflow process network models of computation have been used in several parallelization studies of signal processing applications. Examples include Kahn process networks(KPN) [9], dynamic data flows [12], synchronous data flows [13], etc. Among all these models, the KPN is the most often used for dataflow oriented applications [14]. The KPN model of computation assumes a network of concurrent autonomous processes that communicate over first-in-first-out (FIFO) channels using particular blocking-read and write synchronization primitives. Several frameworks for designing multi-processor systems are based on the KPN model, such as C-heap, Cic [15], DaedalusR [16], Space Codesign [17], Dal [18],[19]. For all these frameworks, system level design tools are used to facilitate the mapping of a behavioral application specification to an architecture platform model. The applicability of these frameworks to the multimedia domain comes essentially from the use of KPN model of computation in which parallel processes, implemented in a high-level language, communicate with each other via unbounded FIFO channels. It is demonstrated that the execution of a KPN model is deterministic, meaning that for a given input always the same output is produced and the same workload is generated, irrespective of the execution schedule.

To execute the KPN in a parallel fashion, several implementations are provided. Since the most language chosen for writing image processing programs is the C/C++, we choose to implement the KPN process using the C++ library YAPI [10].

3.2 The Initial Parallel Model

The application blocks diagram presented in Fig.1 is used for extracting the maximum task-level parallelism. The application is decomposed in separate blocks. Each block defines one single task or process that runs a separate stage of an algorithm. The sequential covariance person detection algorithm is thus split into separate concurrent blocks that will be executed as independent processes. Next, the inter-process communication is established using message passing KPN primitives. Going through this

procedure, the Initial proposed model shown in Fig.4 is obtained and then implemented using the YAPI multi-threading runtime environment.

This model performs as follows: the “Ng”, “Cb” and “Cr” processes collect image data from the input file. The “X_sob”, “Y-sob”, “mag” and “grad” processes calculate the sobel first and second derivatives. The outputs of these processes represent the features. Each feature is sent to “Pi” and “qij” (i: 0..8; j:0..8) to calculate first and second Tensors. The obtained Tensors are forwarded to “ComputCovregion” process to get covariance descriptors. Finally each descriptor is classified by “SVM” process.

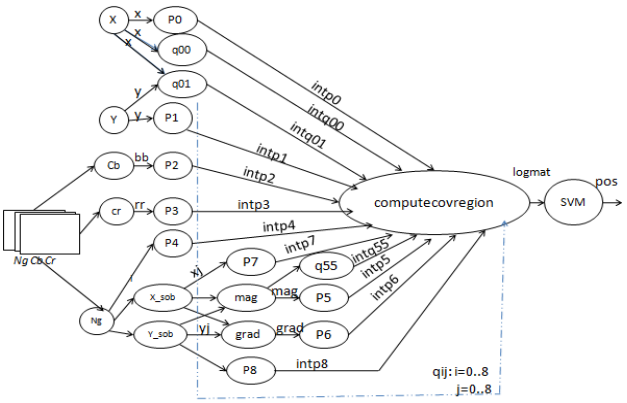

Fig. 4. Initial parallel model of the covariance person detection application

For YAPI implementation purposes, we started with the sequential code of the covariance based person detection application. This code is modified and restructured manually to describe the KPN in C++. Each KPN process is described by a set of associated functions extracted from the original C code. Appropriate FIFO communication channels are then added to enable for data exchange and communication between processes. The implemented parallel KPN model of Fig.5 is validated by high level functional simulation. First, using the same test benches for the sequential and parallel code, a same detection results are obtained. This gives a proof of correctness of the parallel code. Second, for performance evaluation of the proposed parallel model, the amount of data exchange over FIFO channels and the processing time are evaluated using communication and computation workload profiling. These characteristics define the concurrency properties of the model and measure the efficiency of the computation division over the different processes. In fact, a parallel model with good concurrency proprieties should have a balanced computation workload for all the network processes together with a balanced communication workload over its different FIFO communication channels.

The Communication Workload. The obtained communication workload for an image of 640x422 resolution is shown in Fig.5. The Initial model has 171 FIFO channels, 169 of these channels are transmitting 270080 Tsize elements (Tsize= 4 bytes) making the total number of bytes communicated over each one of these 169 channels equal to 270080*4*1 bytes. This amount of data is generally received through read

instructions and then transmitted back over a write instruction (“Wtokens” = "Rtoken"). This represents a very big amount of data to be processed which will require the connected processes to spend a lot of the time dealing with communication. In addition, some other channels are exchanging only few amounts of data. All this makes the communication workload of this starting model completely unbalanced and thus we should be looking for a better model with a communication behavior using data level parallelism using task level splitting or merging techniques.

Communication Workload:

	size	Tsize	Wtokens	Wcalls	T/W	Rtokens	Rcalls	T/R
Covdetect.xx[0]	270080	4	270080	270080	1	270080	1	270080
Covdetect.bb[0]	270080	4	270080	270080	1	270080	1	270080
Covdetect.xi[0]	270080	4	270080	270080	1	270080	1	270080
Covdetect.yj[0]	270080	4	270080	270080	1	270080	1	270080
Covdetect.i[0]	270080	4	270080	270080	1	270080	1	270080
Covdetect.ii	270080	4	270080	270080	1	270080	270080	1
Covdetect.X[0]	270080	4	270080	270080	1	270080	270080	1
Covdetect.Y[0]	270080	4	270080	270080	1	270080	270080	1
Covdetect.mag[0]	270080	4	270080	270080	1	270080	1	270080
Covdetect.grad[0]	270080	4	270080	270080	1	270080	1	270080
Covdetect.isxp0	270080	4	270080	1	270080	270080	270080	1
Covdetect.isxq00	270080	4	270080	1	270080	270080	270080	1
Covdetect.o	128	4	62937	???	81	62937	???	81
Covdetect.logmat	128	4	62937	???	81	62937	???	81
Covdetect.pos	128	4	1554	???	2	1554	???	2

Fig. 5. Communication workload of the Initial parallel model

Computation Workload Analysis. Generally, different tasks need different amount of processing time. For this purpose, a computational workload analysis is considered using the “Gprof” GNU [20] profiling tool. The obtained results are represented in Fig 6 in terms of the CPU time percentage spent in the execution of each process.

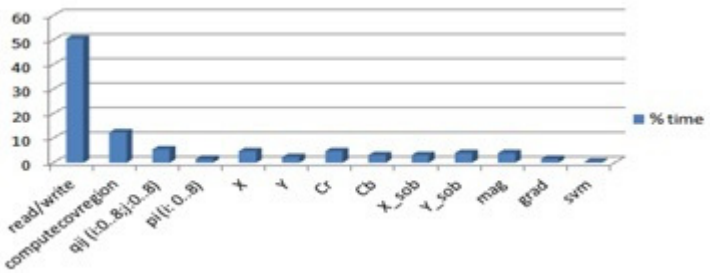

Fig. 6. Parallel computational profiling of the first proposed model

Given the profiling results of Fig.6, it is clear that the computational workload of this Initial model is too much unbalanced. More than 50% of the execution time is spent in reading and writing data. In addition, the majority of the implemented processes have negligible load. It is thus clear, using the obtained communication and

computation workload results, that the proposed Initial model has very poor concurrency properties and this outlines the potential of its optimization for a better computation and communication workload balance. For this aim, we propose to use data level parallelism and task level splitting or merging approaches.

3.3 The Optimized Parallel Model

This section presents the different steps that have been used to derive in a structured way a parallel implementation of the covariance person detection application that has a balanced workload and good communication behavior. Using the profiling results of Fig.5 and Fig.6, it is clear that the process “x_sob”, “y_sob”, “cb”, “cr”, “ng”, “grad”, “mag”, “X”, “Y”, the 9 processes “$p_{i(i:0..8)}$” and the 45 processes “$q_{ij(i:0..8;j:0..8),\ i<j}$” have negligible computations loads and exchange a large data structures. So we propose to merge “cb”, “cr”, “X” and “Y” in "feat_4" process and “x_sob”, “y_sob”, “grad”, “Ng”, and “mag” in "sobel" process. For more concurrency optimization, we propose also to perform data splitting for the “sobel” task thus splitting this process into two processes “sob1” and “sob2”. The nine processes “p_i” are merged into one process called “pregion” that computes the 9 vectors of the first order tensor. The 45 processes that calculate vectors of the second order tensor have been merged into 5 parallel processes. Each process calculates 9 vectors of the second order tensor. The “computecovRegion” process has initially too many FIFO input channels (54*H*W, where H= 422 and W= 640). To reduce the related communication workload, we integrate the computation of covariance matrix into “pregion” and “qiregion” $_{(i:\ 0..4)}$ processes. Fig.7 presents an example of calculation in “qregion” and “q1region” processes.

Original code	*Our optimized code*
For(x=0;x<**9**;x++) | Q[0][x]=integral(F[0],F[x]); **For each image region:** **For** (x=0;x<**9**;x++) | cov[0][x]=Q(x',y';x'',y'');	**For**(x=1;x<**9**;x++) | Q[1][x]=integral(F[1],F[x]); Q[8][8]=integral(F[8],F[8]); **For each image region:** **For** (x=1;x<**9**;x++) | cov[1][x]=Q(x',y';x'',y''); cov[8][8]=Q(x',y';x'',y'');

Fig. 7. Example of proposed processes

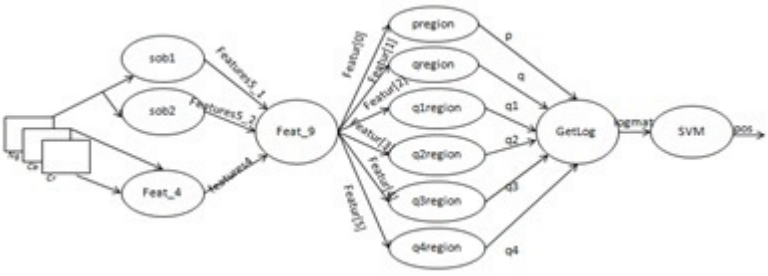

Fig. 8. Proposed optimized parallel KPN model of the covariance person detection application

The transmission of data between these processes is performed by an "image" type structure. “feature4” and “feature5” transmits respectively 4*h*w and 5*h*w image vector characteristics. “p” and “q_i” are exchanging respectively 9x9 and 1x9 data elements. In this case, all the communications channels associated with the exchange of large amounts of data (Fig 5) are removed.

Concurrency Results of the Optimized Model. The optimized parallel model obtained is given in Fig.8. This figure shows the data-partitioning “sob1” and “sob2” and also the task-merging of the “pregion”, “qregion”, “q1region”, “q2region”, “q3region”, “q4region”, and "feat4" processes. This model has been implemented and validated at the YAPI system level. The communication workload results are obtained and shown in Fig.9 for the 640x422 image. A computational “gprof” profiling is also performed and the obtained results are reported in Fig.10.

Communication Workload:

	size	Tsize	Wtokens	Wcalls	T/W	Rtokens	Rcalls	T/R
detect.feature4	1080320	4	4	1	4	4	4	1
detect.feature5_1	1350400	4	5	1	5	5	5	1
detect.feature5_2	1350400	4	5	1	5	5	5	1
detect.featur[0]	2430720	4	9	1	9	9	1	9
detect.featur[1]	2430720	4	9	1	9	9	1	9
detect.featur[2]	2430720	4	9	1	9	9	1	9
detect.featur[3]	2430720	4	9	1	9	9	1	9
detect.featur[4]	2430720	4	9	1	9	9	1	9
detect.featur[5]	2430720	4	9	1	9	9	1	9
detect.p	82	4	???	???	1	???	???	1
detect.q	10	4	???	???	1	???	???	1
detect.q1	10	4	???	???	1	???	???	1
detect.q2	10	4	???	???	1	???	???	1
detect.q3	10	4	???	???	1	???	???	1
detect.q4	10	4	???	???	1	???	???	1
detect.logmat	128	4	???	???	1	???	???	1
detect.pos	128	4	???	???	1	???	???	1

Fig. 9. Communication workload of the optimized parallel model

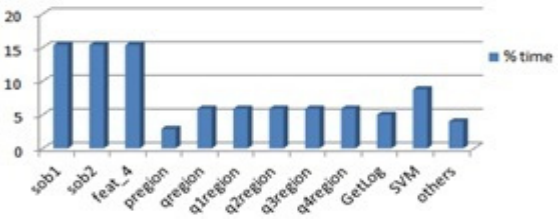

Fig. 10. Parallel computational profiling of the final model

It is clear from Fig.9 that the optimized proposed model has better communication behavior compared to the Initial model. The total number of tokens communicated from/to “feature extraction” and “covariance region” processes has been reduced. Effectively, the number of tokens transmitted over the “p”, “q”, “q1”, “q2”, “q3” and “q4” connecting the input of the “pregion” and “$q_i$region” $_{(i:0..4)}$ processes has been reduced to 777 tokens. The total bytes number communicated over these channel is 4*777*1 bytes. In addition, as indicated in Fig.10, merging tasks decreased the time processes spent in read/write data. Also this better distributes the complexity over processes. The final proposed model has obviously better communication and computational behavior compared to the Initial model.

Results Discussion. To evaluate the effectiveness of the used procedure and the proposed model, we compare the execution time performance of the proposed model with the initial model and the sequential model of 640x422 image resolution. Measurements have been done on the YAPI interface using Gprof tool.

Table 1. Comparative table using 640x422 image resolution

	execution time (s)	Speed-up
Sequential	15.782	-
Initial model	10.93	1,44
Optimized model	5.701	2,767

As shown in Table1, the obtained optimal model outperforms the two previous models. The only use of tasks partitioning mechanism in initial model improves execution time performance by decreasing processing time from 15.782 s to 10.93 s per image. However, using data splitting and task merging, the processing time of the optimized model is reduced to 5.70 s per image. This is three times accelerated execution time in comparison with the original sequential reference code (15.782 s). So depending on the computational and communication workload and the execution time, the initial model is spending a lot of time dealing with communication rather than computation.

To evaluate the feasibility of this procedure, we vary the resolution of the used image. So we apply this procedure to a standard image resolution, a lower resolution and a higher resolution. It is clear from Table 2 that the used procedure decrease significantly the execution time regardless of image resolution.

Table 2. Execution time of varied images resolutions

Resolution	**99x150**	**640x480**	**1024x676**
Sequential	0.21	16.1	41.342
Optimized model	0.03	5,967	18.737

Consequently, much more better performance is obtained using task level merging and data level splitting for the best workload balance of the covariance based person detection parallel model. In addition, the smaller resolution is used, the higher speedup is obtained. So at certain image resolution the use of this procedure accelerate the computation without achieving real time. This require the use of more performing processor or a hardware accelerator.

4 Conclusion

In this paper an optimized parallel model of a covariance based person detection system is developed. First, an Initial KPN parallel model is proposed using only task level parallelism. This model is then validated using the YAPI multi-threading environment. Analysis of the communication and computation workload results showed

the very poor concurrency properties of the initial model. In this context, data level parallelism and task level merging are applied to improve the concurrency properties. At the end, an optimized process network parallel model of the person detection system based on covariance descriptor is obtained. This model has fast processing time and considerable computation and inter process communication workload balance.

References

1. Strickland, R., Ilhahn, H.: Wavelet transform methods for object detection and recovery. IEEE Transaction. Image Processing **6**(5), 724–735 (1997)
2. Lienhart, R., Maydt, J.: An extended set of haar like features for rapid object detection. IEEE Proceedings. Image processing **1**, 900–903 (2002)
3. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. IEEE computer society. computer vision and pattern recognition **1**, 886–893 (2005)
4. Tuzel, O., Porikli, F., Meer, P.: Pedestrian Detection Via Classification on Riemannian Manifolds. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**, 1713–1727 (2008)
5. Paisitkriangkrai, S., Shen, C., Zhang, J.: Performance evaluation of local features in human classification and detection. IET Computer Vision **2**, 236–246 (2008)
6. Qin, L., Snoussi, H., Abdallah, F.: Adaptive covariance matrix for object region representation. In: SPIE Fifth International Conference on Digital Image Processing (2013)
7. Abid, N., Ayedi, W., Ammari, A.C., Abid, M.: SW/HW implementation of image covariance descriptor for person detection system. In: IEEE Advanced Technologies for Signal and Image Processing, pp. 115–119 (2014)
8. Kahn, G.: The semantics of a simple language for parallel programming. In: Proceedings of IFIP. vol. 74 (1974)
9. Kock, E., Essink, G., Smits, W., Wolf, P., Brunel, J.-Y., Kruijtzer, W.M., Lieverse, P., Vissers, K.A.: YAPI: application modeling for signal processing system. In: IEEE Procceeding Design Automation Conference, pp. 402–405 (2000)
10. Fradkin, D., Muchnik, I.: Support vector machines for classification. Mathematics subject classification (2000)
11. Metternich, M.J., Worring, M., Smeulders, A.W.: Color based tracing in real-life surveillance data. In: Shi, Y.Q. (ed.) Transactions on DHMS V. LNCS, vol. 6010, pp. 18–33. Springer, Heidelberg (2010)
12. Kangkook, J., Kemerlis, P., Keromytis, D., Georgios, P.: Shadowreplica: efficient parallelization of dynamic data flow tracking. In: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, pp. 235–246 (2013)
13. Canelhas, D., Stoyanov, T., Lilienthal, J.: SDF Tracker: A parallel algorithm for on-line pose estimation and scene reconstruction from depth images. IEEE Intelligent Robots and Systems 3671–3676 (2013)
14. Lee, E., Parks, T.: Dataflow Process Networks. IEEE Proceeding **83**(5), 773–801 (1995)
15. Kwon, S., Kim, Y., Jeun, W., Ha, S., Paek, Y.: A Retargetable Parallel-Programming Framework for MPSoC. ACM Trans. on Design Automation of Electronic Systems **13**, 39:1–39:18 (2008)
16. Bamakhrama, M., Zhai, J., Nikolov H., Stefanov, T.: A methodology for automated design of hard-real-time embedded streaming systems. In: Design, Automation Test in Europe Conference Exhibition, pp. 941–946 (2012)

17. Bailey, B., Martin, G.: Codesign experiences based on a virtual platform. In: ESL Models and their Application, Ser. Embedded Systems. Springer US, pp. 273–308 (2010)
18. Schor, L., Bacivarov, I., Rai, D., Yang, H., Kang, S.: Scenario-based design flow for mapping streaming applications onto on-chip many-core systems. In: International conference on Compilers, Architectures and Synthesis for Embedded Systems, pp. 71–80 (2012)
19. Corre, Y., Diguet, J.-P., Lagadec, L., Heller, D., Blouin, D.: Fast template-based heterogeneous MPSoC synthesis on FPGA. In: Brisk, P., de Figueiredo Coutinho, J.G., Diniz, P.C. (eds.) ARC 2013. LNCS, vol. 7806, pp. 154–166. Springer, Heidelberg (2013)
20. Arora, H.: Gprof Tutorial. How To Use Linux Gnu Gcc Profiling Tool (2012)