

# Architecture exploration of an embedded multi-processor for video content analysis

Julien A. Vijverberg  
VDG Security B.V., Zoetermeer,  
Eindhoven University of Technology, Eindhoven  
The Netherlands  
Email: j.a.vijverberg@tue.nl

Peter H.N. de With  
CycloMedia Technology B.V., Waardenburg  
Eindhoven University of Technology, Eindhoven  
The Netherlands

**Abstract**—This paper discusses the design of embedded multi-processor architectures for pixel-level video-content analysis applications in video sequences. First, a number of object segmentation computing tasks are analyzed. In order to come to an efficient proposal for a content-analysis processing platform, we have taken five different representative applications and mapped those on various platform configurations. Since the computational requirements of video content analysis are significant, the experiments show that even up to 8 C30-like DSP cores are not sufficient to realize real-time performance. Therefore, we propose an architecture which is based on a combination of powerful DSPs and (re-)configurable hardware to improve performance and maintain algorithmic flexibility.

**Index Terms**—Video-Object Segmentation, Hardware Acceleration, Multi-processors, Content Analysis

## I. INTRODUCTION

Advances in computer vision technology and IC design support the rapid growth of intelligent video cameras for applications in surveillance and monitoring. For efficiency reasons, the trend is to make the cameras smarter, such that the attention of the operator is automatically focused on one particular event captured by one of a set of cameras, so that in a more advanced form the vision system can respond autonomously. In addition, consumer products are increasingly equipped with smart(er) vision algorithms, such as image-stabilization and face-detection in photo-cameras, sensor networks in houses and video-based driver assistance in cars. These developments motivate our research for an efficient implementation of a platform suited for Video Content Analysis (VCA), that can cover a broad set of analysis applications.

For efficient computing in signal processing and multimedia coding, it is known that Application-Specific ICs (ASICs) and Processors (ASIPs) provide cost-effective solutions in combination with general-purpose computing. Unfortunately, designing ICs is costly and there are currently no solutions to automate the design of efficient ASICs, starting from an application specification. Moreover, verification of the designed hardware takes a considerable effort. For this reason, reusable components and System-on-Chip (SoC) are becoming increasingly popular.

Current embedded multi-processor systems are limited to execute simple VCA applications and designing special vision hardware is too costly. Speed-ups and programmability can

be achieved by clever design of a platform for many vision applications. Until now, most architectures are mostly fixed for a single application or for a single type of tasks. This paper discusses various possibilities for more generic (hence reusable) hardware acceleration for executing video-analysis tasks on embedded multi-processor platforms.

### A. State-of-the-Art Processor Architectures

Several commercial SoCs have been implemented on silicon. A very popular image-processor type is the SIMD array. For example, the Xetal-II architecture [1] represents an SIMD array with 320 processing elements. However, the limitations of large SIMD arrays have been described in literature: many image-operation primitives cannot exploit data parallelism to a high extend. In [2], an SIMD array of 8 elements is presented and the applicability to a number of execution kernels is tested. Indeed, it seems wise to add a number of small SIMD arrays (or add SIMD instructions to the already present processors). The author does not evaluate image-processing functions with more irregular data access patterns than ARPS motion estimation.

The most popular DSP architecture for image processing is probably the TMS320C64xx. Arth *et al.* present the implementation of a vehicle detection and tracking algorithm on the TMS320C6414 in [3]. However, the proposed application was not very complex and the DSP was not found suitable for more complicated algorithms. In [4], an SoC based on an ARM and a VLIW core for 3DRS motion estimation was presented, based on a short-list of tasks for acceleration [5]. In addition to the VLIW and ARM cores, specific modules for different applications relying on fast and accurate motion estimation were implemented. Since not all applications are based on motion estimation, this core does not seem to be a very generic solution.

The Acadia Vision Processor from [6] is a special image computer, which contains several special vision cores, including two image-warping cores, two filter cores for pyramidal construction, a global correlator core, and a global motion-estimation core, which are connected via a cross-point switch. This proposal does not contain general-purpose or DSP cores and the specific cores are limited to a small number of computationally intensive subtasks. Another vision IC targeted

for car-driver assistance applications is presented in [7]. This IC has a  $4 \times 4$  multi-layer matrix with two ARM9 cores and four special computer-vision cores: Window Process & Filter Engine, Tracker Engine (for image warping and tracking), Classifier Engine, and Lane Detection Engine. Even a global architecture of these specific cores is not disclosed, such that there is no way to benchmark the design.

### B. Contribution

In this paper, the effect of adding co-processors to an embedded multi-processor system and its influence on the resulting frame rate and required memory bandwidth of pixel-level content-analysis applications will be discussed. To this end, the execution time and external memory bandwidth of five benchmark applications mapped on multi-core architectures will be computed, using the results of an assembly-level analysis of the tasks in the benchmark applications. This type of analysis has been shown to give reasonably accurate estimations of the complexity [8]. Our explorations give insight in the complexity of several tasks and applications, while indicating the preferred platform solution. It will be shown the best way to implement VCA applications appears to be using powerful DSPs combined with (re-)configurable hardware. due to the variability in processing tasks and memory-access patterns.

### C. Overview

The sequel of this paper is organized as follows. In Section II, the benchmark applications will be introduced and some typical and resource-intensive tasks will be described. In Section III, a list of frequently occurring and/or resource-intensive functions is composed using recently published papers on resource consumption in video-object segmentation and data from a detailed evaluation of algorithms from Section II. In Section IV, this data is used to evaluate the influence of several platform configurations on the obtained frame rate and external memory bandwidth for the execution of the benchmark applications. Section V presents conclusions.

## II. BENCHMARK APPLICATIONS

For benchmark applications, we have studied VCA literature and finally concentrated on five applications, which are summarized in Table I. Generic work on video segmentation is presented in [9], involving a  $5 \times 5$  multi-pixel significance test and Markov Random Fields (MRFs) on a  $3 \times 3$  neighborhood. For background estimation, a temporal median filter is added. For tracking, a mean-shift tracker with a double exponential smoothing filter is used. As a second application, the same algorithm is used with additional tasks to cope with segmentation in video sequences with moving cameras. First, features which are generated by a Harris corner detector, are matched within two consecutive frames. Then the parameters of an eight-parameter camera model are estimated using RanSaC, which are finally refined using direct global motion estimation. Kim *et al.* [10] designed an algorithm using watershed and region merging for segmenting persons in teleconferencing sequences. These tasks are known to have

a very irregular memory-access pattern. Lao *et al.* [11] designed an abnormal human behavior detection application, which consists of a computationally inexpensive background estimation and subtraction technique. The processing also involves mean-shift tracking in the hue colorspace. Arth *et al.* [3] have presented an algorithm, that can detect both license plates and vehicles on a highway. The Viola-Jones detector in this applications was concluded to be the most complex task. These five applications are rather different and involve a large number of content-analysis techniques at the pixel-level.

The above-described applications are rather broad and involve many different computing tasks of which most are compute- and/or memory-intensive. Therefore, we have focused on the segmentation and tracking tasks at the pixel-level.

## III. TASK ANALYSIS

### A. Task-Analysis Method

Jaspers [8] has made a detailed evaluation of the resource consumption of several multimedia-processing algorithms. He estimated the resource consumption of the image-processing algorithms under a list of assumptions. We address this list in somewhat more detail, because we will use it as a starting point for our explorations. The assumptions are that: (A1) the execution time is determined by the number of native DSP operations during the execution of a loop (other operations are neglected), (A2) processors compute addresses in parallel to data processing while accessing data with *a-priori* known patterns with a limited locality, (A3) load and store are one-cycle operations, (A4) scratch pad memory or cache is available and decently operating to have sufficiently low amount of cache misses, and (A5) all arithmetic operations (including multiplication) require one clock cycle. In addition, we impose extra assumptions that describe memory usage in more detail so that we are able to analyze our memory requirements sufficiently accurate. These extra assumptions are: (A6) the memory requirements are dominated by the largest data structures such as line buffers and queues, while other data structures can be neglected, and (A7) the fetching of instructions does not require data memory or bandwidth, since the instructions are available in a local instruction memory.

Steux and Abramson [5] made a coarse analysis of the relevance and computational complexity of several pixel-level tasks for content analysis. Their conclusions were that the algorithms should be accelerated in the following order (only the top most important tasks are mentioned here): Arithmetic operations, basic morphological operations, connected component labeling, blob analysis, histogram computation, linear kernel filtering. We have found similar results and will provide accelerators for some of those tasks in Section IV-B.

### B. Task-Analysis Results

The computational analysis technique described above was applied to the tasks in the benchmark applications. In Table II, the key features of the computing for the most important functions are shown. The benchmark tasks are the temporal median with 10 frames [9], watershed [10], region merging

TABLE I  
THE PROCESSING PROPERTIES OF THE SUBTASKS FOR EACH BENCHMARK APPLICATION.

Application/Function	#Operations	Memory Bandwidth	Memory Access Pattern	Neighborhood
<b>Farin2005a</b>				
Temporal Median	High	High	Regular	pixel
MPST	Low	Medium	Regular	5 lines
MRF-ICM	Medium	Medium	Regular	3×3 lines
Blob extraction	Medium	Medium	Irregular	large table
<b>Farin2005b</b>				
Harris 3x3	Medium	Medium	Regular	3 lines
Feature matching	Medium	Medium	Semi-regular	8×8 pixels
Ransac GME	Very High	Medium	Regular	motion vectors
Direct GME	Very High	Very High	Regular	pixel
<b>Kim2006</b>				
3x3 Open-Close	Low	Medium	Regular	3 lines
Watershed	Medium	Medium/High	Irregular	pixel queue
Region merging	Medium	Medium/High	Irregular	large matrix
<b>Lao2007</b>				
Static-object Detection	Medium	Medium	Regular	pixel
Blob extraction	Medium	Medium	Irregular	large table
Mean-shift	Medium	Medium	Semi-regular	large region
<b>Arth2006</b>				
Approximate Median	Low	Medium	Regular	pixel
Viola-Jones Detector	High	Medium	Regular	blocks

TABLE II  
ESTIMATED PROCESSING REQUIREMENTS AND TASK PROPERTIES OF FUNCTIONS WITH A HIGH RESOURCE CONSUMPTION FOR A FRAME OF SIZE 352×288. THE LAST TWO COLUMNS - DESCRIBE THE GRANULARITY OF PARALLELISM FOR TASK AND DATA PARALLELISM.

Function	#Ops MOps/fr.	Loc.Mem. KB	Ext.Mem. KB.	BW up KB/fr	BW down KB/fr.	Task	Data
Temporal Median (10 frames)	50	0	3,000	3,000	600	high	high
Watershed	9	700	0	200	100	low	low
Region Merge (180 to 8 regions)	4	2,000	70	100	200	none	low
Union Find	2	100	100	200	10	low	medium
MeanShift (80x75 object)	16	17	100	100	100	none	high
Full-search ME (8 × 8, ±16)	500	11	200	200	100	high	high
RanSaC GME (1024 vectors)	450	11	0	0	10	none	medium
DGME (20 iterations)	200	400	200	0	4,000	none	high
GMC-8 + bilinear interpolation	8	0	200	300	1,200	high	high

with 180 input regions and 8 output regions [10], *union find* connected component labeling, mean-shift tracking, full-search motion estimation with  $8 \times 8$  blocks and  $\pm 16$  pixels search range, RanSaC GME[9] with 1,000 iterations, Direct GME (DGME)[9] with Levenberg-Marquardt optimization (20 iterations) and global motion compensation. The global motion model is the eight-parameter model from [9].

Not surprisingly, the global and local motion-estimation techniques are among the most complex functions. The temporal median is also a very compute- and memory-intensive task and a candidate for acceleration or for replacement by a less complex technique. The sorting of pixels can be reused for spatial median filtering and other tasks. Table II also shows the high memory requirements for the watershed, region-merging and region-labeling tasks, which all need at least the memory for one frame. Computing histograms and moments for tracking combine the undesirable properties of irregular data access and a high number of cycles. Operations on neighborhoods are among the most frequent tasks and can be computationally intensive for large neighborhoods.

The results of the analysis are not very detailed, but have shown a proven value [8] to be sufficiently accurate for architectural analysis without biasing to a specific platform.

#### IV. PLATFORM EXPLORATION

This section evaluates design decisions such as changing the number of DSPs and including co-processors for exploring several mappings of the five applications onto various platform configurations. The required external memory bandwidth and the resulting frame rate are computed using the estimated number of operations, local memory size, and number of bytes to communicate per task iteration under the assumptions described in Section III-A.

##### A. Basic Platform Description

Parts of the platform have been synthesized with XST for the Virtex-4 FPGA (XV-4). An example of the platform configuration is shown in Fig. 1. The host CPU (a RISC processor) is used to issue tasks to the other tiles on the platform and perform operating-system tasks. It is not considered in the analysis, since it is assumed that communication

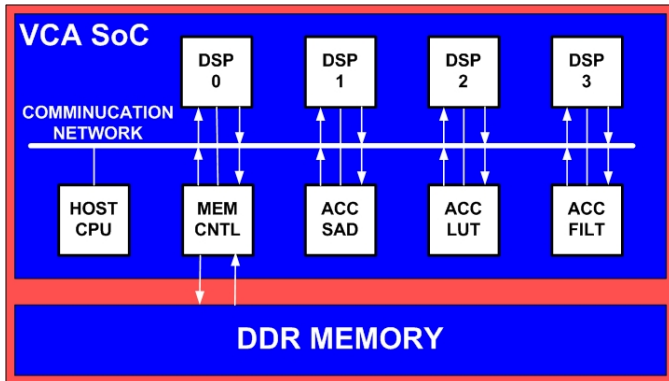


Fig. 1. Block diagram of a proposed architecture with the following components: host CPU, four DSPs, video interface, accelerators and external memory controller.

with the host CPU does not interfere with the high-bandwidth communication between other tiles. The C30-like [12] DSP tiles operate at 100 MHz and are used to execute several video-processing tasks. The local memory size is 32 kB per DSP tile and each DSP tile is also equipped with a communication assist, that handles the communication between the DSP’s local memory and the communication network. In our platform exploration, the number of DSPs will be a variable design parameter.

We have not fixed bandwidth parameters for the DDR memory, because the effectively required external memory (DDR) bandwidth is computed. For on-chip communication, we have assumed point-to-point communication using FSL buffers, since this was initially the only way of communication in the initial version of the platform. Since the FSL buffers operate at 300 MHz, which is three times faster than the DSPs, the results of the analysis do not take internal communication into account.

### B. Accelerator Description

In this section, we describe a number of accelerators which are used in one of the explored platform configurations. Examples of such accelerators are a histogram-computing engine and a systolic array for accelerating neighborhood operations. The properties of the accelerators are shown in Table III.

The first accelerator is a Histogram-computation accelerator (HIST). The Look-Up-Table tile (LUT) consists of a block of memory which can be filled by one of the DSPs or the CPU, after which the 8-bit input pixels are used as an address for the memory block. A Sum-of-Absolute-Differences tile (SAD) is also included, which can compute the SAD of four 8-bit pixels and accumulate the results of the SADs over multiple packets of pixels.

We have also defined a systolic-array accelerator, which is essentially a  $3 \times 3$  array of processing elements (PEs), consisting of three parallel pipelines (first-level PEs) together with two line buffers. The results of these first-level PEs are forwarded to PEs connected in a tree structure, which all have the same instruction pipeline (instructions selected from AND,

TABLE III

ESTIMATED IMPLEMENTATION PROPERTIES OF THE ACCELERATORS USING THE RESOURCES OF THE XV-4. THE TABLE ALSO SHOWS WHETHER THE ACCELERATOR IS USED IN EACH OF THE BENCHMARK APPLICATION. THE ABBREVIATION “N.A.” MEANS NOT AVAILABLE. THE THROUGHPUT OF THE SYSTOLIC MODULE IS EXPLAINED IN THE TEXT.

Accelerator	HIST	LUT	SAD	SYSTOLIC
4-input LUTS	272	43	n.a.	2,900
DSP-slices	0	0	n.a.	5
BlockRAMS	2	2	1	0
Clock Freq.(MHz)	200	200	100	150
Throughput (pix/cc)	1	1	4	*
Arth2006	-	-	-	USED
Farin2005a	-	-	-	-
Farin2005b	-	-	USED	-
Kim2006	-	-	-	USED
Lao2007	USED	USED	-	USED

OR, MAX, MIN, or ADD), to compute the final result. The line buffers are mapped on two Block-RAMs (BRAMs) of 18 kbits, thereby enabling sharing of the array by multiple tasks simultaneously for image widths lower than 1,024 pixels, albeit at a lower throughput.

### C. Application Evaluation

The frame rate and bandwidth were computed for several platforms and images at CIF resolution ( $352 \times 288$ ). The obtained results are shown in Table IV. The platform with one DSP (P1) executes one task at a time and writes the results back to the external memory. For both the platforms with 4 DSPs without accelerators (P4) and 8 DSPs (P8), two different execution schedules have been evaluated: one with maximal data-parallelism, high frame rate and high bandwidth requirements, and another schedule with moderate frame rate and much lower bandwidth requirements.

Fig. 2 shows the ratio between the obtained frame rate and the external memory bandwidth. This should give a global indication of how much an application can be accelerated by higher DSP clock frequencies and more powerful instructions without approaching or even exceeding the maximum external memory bandwidth. If the height of a plotted bar is below a certain threshold, the ratio between the frame rate and the external memory bandwidth is so low, that the application cannot be executed in real-time by increasing the DSP clock frequency, since it will reach the maximum external memory bandwidth.

## V. DISCUSSION AND CONCLUSIONS

In the preceding sections, an application-driven architecture design approach and a theoretical evaluation for pixel-level content-analysis tasks for execution on embedded multi-processors have been presented. First, several applications have been analyzed to obtain properties such as the number of operations per execution, the communication bandwidth, and memory usage. Secondly, the frame rate and required external memory bandwidth for five reference applications have been computed for different platforms with and without accelerators.

TABLE IV

COMPUTED FRAME RATES (FPS) AND EXTERNAL MEMORY BANDWIDTH (MB/S) OF BENCHMARK APPLICATIONS FOR DIFFERENT PLATFORMS AND SCHEDULES. THE NUMBER OF DSPS AND THE PRESENCE OF CO-PROCESSORS DEFINE THE PLATFORM. THE BANDWIDTH (BW) HAS BEEN ROUNDED TO AN INTEGER VALUE.

Platform	P1		P4				P4A		P8			
#DSPs	1		4				4		8			
Co-Processors	No		No				Yes		No			
	Fast		Fast		BW-cheap		BW-cheap		Fast		BW-cheap	
	Rate	BW	Rate	BW	Rate	BW	Rate	BW	Rate	BW	Rate	BW
Arth2006	2.3	44	9.3	176	7.6	15	8.1	17	18.4	349	15.1	31
Farin2005a	2.2	10	8.6	40	8.4	21	12.5	31	16.6	77	16.4	41
Farin2005b	0.2	4	0.8	15	0.8	12	0.8	12	1.6	29	1.6	24
Kim2006	5.5	11	11.8	24	11.3	10	13.8	12	18.6	38	17.9	16
Lao2007	5.0	24	19.8	96	18.2	34	27.1	51	39.6	193	36.3	68

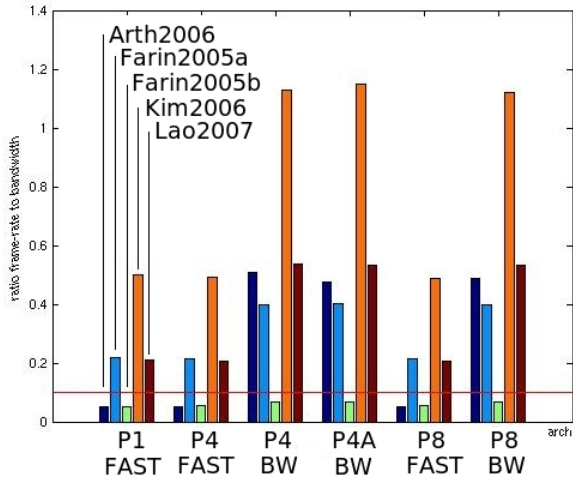


Fig. 2. Bar plot showing the ratio between the frame rate and the external memory bandwidth for all platform configurations and applications. On the x-axis, the combinations between applications, platform configurations and different types of mappings are shown. On the y-axis, the ratio between the frame rate (fps) and external memory bandwidth (MB/s) is drawn. The crossing horizontal line is set to 0.1 which corresponds for example to 30 fps and 300-MB/s bandwidth.

From the obtained results in Table IV it can be concluded that the computational power of the proposed platforms is not sufficient. At CIF resolution, only Lao's application operates faster than 25 fps and higher resolutions are probably required. Higher clock frequencies and task-specific accelerators seem to be a good solution at first sight, but the external memory bandwidth will rapidly become the critical factor, as illustrated in Fig. 2 by the small application bars indicating insufficient frame rate. This is particularly the case for the Farin2005b application, of which the frame rates are so low, that acceleration by more computing (more than a factor of 20) will increase the bandwidth beyond the capabilities of the complete reference platform. A general platform for VCA applications seems to hardly benefit from task-specific co-processors, due to the large variability in task operations. In Table IV, this is illustrated by the frame rates for platforms P4 and P4A (with the exception of applications Farin2005b and Lao2007, for which the frame rate is dominated by a single task).

In addition, the frame rate in the benchmark applications

scales almost linearly with the number of processors for a small given number of processors, except for application Kim2006. This application has resource intensive subtasks which are difficult to parallelize.

For four applications (all except Farin2005b), we predict based on Fig. 2 that more powerful programmable cores will increase the frame rate. An example of a core more powerful than the C30 core is the the C64x cores [13]. However, a new high-level analysis of the frame rates assuming a platform with four C64x cores (clock frequency: 100 MHz) reveals that frame rates for the applications Arth2006, Farin2005 and Kim2006 should approximately be 12, 16 and 17 fps respectively. Unless we allow the clock to operate at frequencies typical for the hardwired implementation of the C64x (300 to 600 MHz), this set-up will not achieve real-time execution of the benchmark applications.

As a main conclusion, it seems that the best platform solution should include both (design-time) reconfigurable logic for extensive hardware acceleration and powerful (*e.g.* SIMD-enhanced) programmable cores to increase the throughput and keep the external memory bandwidth low by exploiting more task parallelism. The last statement about task parallelism contradicts with a recent article on computer vision and homogeneous multi-cores [14] which concludes that it is better to exploit data-parallelism. The discrepancy between these concluding views is explained by the fact that the execution time of the application in [14] mainly depends on a single task (a particle filter) which is highly scalable in terms of data parallelism. In our case, the variation of expensive tasks is much higher

## REFERENCES

- [1] A. A. Abbo, R. P. Kleihorst, V. Choudhary, L. Sevat, P. Wielage, S. Mouy, B. Vermeulen, and M. Heijligers, "Xetal-ii a 107 gops, 600 mw massively parallel processor for video scene analysis," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 192–201, January 2008.
- [2] A. G. Boon, "A hybrid processor architecture for (ir)regular image processing on an fpga," Master's thesis, Eindhoven University of Technology, 2008.
- [3] C. Arth, H. Bischof, and C. Leistner, "Tricam - an embedded platform for remote traffic surveillance," in *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, 2006, pp. 125–133.
- [4] H. Peters, R. Sethuraman, A. Beric, P. Meeuwissen, S. Balakrishnan, C. Pinto, W. Kruijtzter, F. Ernst, G. Alkadi, J. van Meerbergen, and G. de Haan, "Application specific instruction-set processor template for motion estimation in video applications," vol. 50, no. 4, pp. 508–527, April 2005.
- [5] B. Steux and Y. Abramson, "Report on computer vision algorithms," Ecole des Mines de Paris, Tech. Rep. Camellia D3.1, July 2003.
- [6] G. van der Wal *et al.*, "The acadia vision processor," in *IEEE Proceedings of International Workshop on Computer Architecture for Machine Perception*, September 2000, pp. 31–40.
- [7] G. P. Stein, E. Rushinek, G. Hayun, and A. Shashua, "A computer vision system on a chip: a case study from the automotive domain," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 2005, pp. 130–134.
- [8] E. G. Jaspers, "Architecture design of video processing systems on a chip," Ph.D. dissertation, Eindhoven University of Technology, 2003.
- [9] D. S. Farin, "Automatic video segmentation employing object/camera modeling techniques," Ph.D. dissertation, Eindhoven University of Technology, 2005.
- [10] J. Kim, H.-J. Lee, T.-H. Lee, M. Cho, and J.-B. Lee, "Hardware/software partitioned implementation of real-time object-oriented camera for arbitrary-shaped mpeg-4 contents," in *Proceedings of the 2006 IEEE/ACM/IFIP Workshop on Embedded Systems for Real Time Multimedia*, October 2006, pp. 7–12.
- [11] W. Lao, J. Han, and P. H. N. de With, "Human motion analysis using simultaneous trajectory and body detection and modeling," in *Symposium on Information Theory in the Benelux*, vol. 1, May 2007, pp. 109–116.
- [12] *TMS320C3x User's Guide*, Texas Instruments, March 2004.
- [13] *TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide*, Texas Instruments, February 2008.
- [14] T. P. Chen, D. Budnikov, C. J. Hughes, and Y.-K. Chen, "Computer vision on multi-core processors: Articulated body tracking," in *IEEE International Conference on Multimedia and Expo*, July 2007, pp. 1862–1865.