Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 710685, 15 pages
http://dx.doi.org/10.1155/2014/710685

Research Article

Complexity Analysis of Vision Functions for Comparison of Wireless Smart Cameras

Muhammad Imran, Khursheed Khursheed, Naeem Ahmad, Mattias O'Nils, Najeem Lawal, and Malik A. Waheed

Department of Electronics Design, Mid Sweden University, Holmgatan 10, 851 70 Sundsvall, Sweden

Correspondence should be addressed to Muhammad Imran; muhammad.imran@miun.se

Received 31 May 2013; Revised 7 December 2013; Accepted 12 December 2013; Published 9 January 2014

Academic Editor: Hongkai Xiong

There are a number of challenges caused by the large amount of data and limited resources such as memory, processing capability, energy consumption, and bandwidth, when implementing vision systems on wireless smart cameras using embedded platforms. It is usual for research in this field to focus on the development of a specific solution for a particular problem. There is a requirement for a tool which facilitates the complexity estimation and comparison of wireless smart camera systems in order to develop efficient generic solutions. To develop such a tool, we have presented, in this paper, a complexity model by using a system taxonomy. In this model, we have investigated the arithmetic complexity and memory requirements of vision functions with the help of system taxonomy. To demonstrate the use of the proposed model, a number of actual systems are analyzed in a case study. The complexity model, together with system taxonomy, is used for the complexity estimation of vision functions and for a comparison of vision systems. After comparison, the systems are evaluated for implementation on a single generic architecture. The proposed approach will assist researchers in benchmarking and will assist in proposing efficient generic solutions for the same class of problems with reduced design and development costs.

1. Introduction

Vision systems implemented on wireless smart cameras have recently been the focus of research for many applications including surveillance [1], recognition [2], traffic monitoring, personal care [3], and industrial monitoring [4]. Often, a number of wireless smart cameras are spread over an area to form a network called a Wireless Vision Sensor Network (WVSN). In the WVSN context, the individual wireless smart camera is referred to as a Wireless Vision Sensor Node (VSN). Each VSN consists of an image sensor, an embedded processing platform, memory, battery or an alternative energy source, and a wireless link. Designing a VSN on an embedded platform is a challenging task because resources are limited compared to those for general purpose platforms. General purpose platforms offer greater design and implementation flexibility; however, these systems are often considered as being unsuitable for real time applications. Therefore, our focus is on vision systems implemented on embedded platforms. When designing a VSN for a particular application, the designers must firstly investigate the complexity of the design, and a failure to do this may result in both increased design costs and a longer developmental cycle. The vision functions can be implemented on microprocessors, dedicated hardware such as Application Specific Integrated Circuits (ASICs), or on programmable hardware such as Field Programmable Gate Arrays (FPGAs).

The microprocessors are widely used as General Purpose Processors (GPPs) and Application Specific Instruction-set Processors (ASIPs). The general purpose processors are focused for average performance and greater flexibility while the ASIPs, such as the Digital Signal Processor (DSP), are focused for specific performance and specific flexibility. The GPPs and ASIPs are software based and it is easy to be certain of the correctness of the code and the system performance by means of simulations in the software. The performance of

microprocessor based platforms is often limited as compared to that for an ASIC and reconfigurable hardware platforms because, in general purpose processors, every instruction must be retrieved and decoded before execution [5].

An ASIC is customized for a specific application; therefore it will provide a better performance and low power consumption but the design cost is high due to nonrecurring engineering (NRE) and manufacturing costs associated with small volumes. Moreover, the ASIC solutions are customized for specific applications. Programmable logic devices (PLDs), which include Field Programmable Gate Arrays (FPGAs), are replacing traditional logic circuits by offering advantages such as small size, low power, and high performance in relation to the disadvantages associated with custom ASICs. The reconfigurable computing allows the user to program at a low level and supports general purpose computing by virtue of reconfigurability [5]. The choice of processing platform for a particular system is dependent on the performance requirement and constraints of the particular application. The features including smaller cost for low volume products, reconfigurability, and parallelism make FPGA a suitable platform for wireless smart cameras. Therefore, in this work we will consider complexity estimation for wireless smart camera functions on FPGA based platforms.

Currently, the trend in WVSN is to propose specific solutions and each solution requires a great deal of design and development effort and cost on FPGA based platforms. Proposing generic solutions which fulfills the needs of different solutions would reduce these efforts and cost. In this way, the efforts being utilized for individual solutions can be diverted to development of single optimized solutions. For proposing generic solutions for existing different individual solutions, the investigated solutions need to be implemented on each other architectures. This requires a lot of design and development efforts. Our goal is to propose a mechanism in the form of complexity model and system taxonomy which can assist the comparison of existing solutions and development of generic solutions without the need for actual implementation. To the best of our knowledge, no model exists which can facilitate the researchers/designers in comparison and generalizations of different smart camera solutions.

The comparison of vision solutions is necessary for the improvement of current research and for proposing generic solutions within this field. In relation to the complexity analysis, the arithmetic complexity, memory requirement, and suitability of vision functions for vision systems are investigated. To illustrate the use of a complexity model, a number of actual systems have been classified with the assistance of the system taxonomy and the resources are then estimated by using a complexity model. After the complexity estimation, the vision systems are compared and evaluated for implementation on a single generic architecture. It is worth mentioning that this paper is extended version [6]. Following this, Section 2 presents related work, Section 3 describes the problem, Section 4 presents the complexity model, Section 5 presents the case study, Section 6 shows the comparison of the vision systems, and Section 7 illustrates the future challenges while Section 8 provides the conclusion.

2. Related Work

Before describing the problem area, some of the related work published in the literature is presented in this section. Different solutions are proposed by researchers for WVSN problems. Currently, the focus is on particular solutions for particular problems in WVSN. Hengstler and Aghajan [3] proposed an application oriented design methodology for a VSN. However, the authors consider only the specific case of tracking objects using a single camera and a stereovision VSN. Dieber et al. [7] presented a formulation and approximation method for the camera selection and task assignment in order to achieve the required monitoring activities. The tradeoff between surveillance quality and resource utilization has been investigated. The design aspects and software modelling for a ubiquitous real time camera system are investigated by Lin et al. [8]. The authors divide the design aspects into two categories, namely, the general and the application specific. Taxonomy for a VSN is proposed by Rinner et al. [9] based on platform capabilities, the degree of distribution processing, and system autonomy. Some of the existing vision systems implemented on VSNs are described and classified according to the proposed taxonomy. In addition, some of the challenges associated with VSNs are described. Tilak et al. [10] classified the wireless microsensor networks from a communication protocol perspective. Different types of communication functions, data delivery models, and network dynamics are discussed for wireless sensor networks; however, there is no discussion in relation to camera based sensor systems. Generally, the camera based sensors produce two-dimensional data which requires greater processing capabilities, greater memory, high power consumption, and a high communication bandwidth as compared to traditional sensor networks. Therefore, the requirements of visual sensor networks are different.

3. Problem Statement and State of the Art

An investigation of the related work shows that the focus is on particular aspects of the vision systems. There is no common mechanism for complexity estimation and for comparison of different vision systems, which is necessary for proposing generic solutions and the improvement of research within this field. When comparing two different vision systems, each of the vision system is required to be implemented on the other's architecture as depicted in Figure 1. This requires a great deal of effort in relation to both the design and its development. In some cases, a researcher does not have access to another researcher's architecture. As the number of systems increases for comparison, it becomes less feasible to implement all the vision systems. This necessitates the building of a common tool, which can facilitate the researchers in both the benchmarking and development of different classes of vision systems. In order to meet this demand, we have proposed an abstract model for complexity analysis with the assistance of system taxonomy.

The mechanism that has been adopted in order to develop this model is shown in Figure 2. The problem space is identified, which is to propose a mechanism or at least an

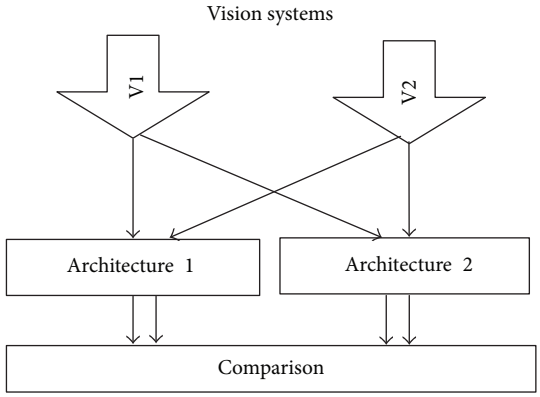


Figure 1: Comparison of vision systems.

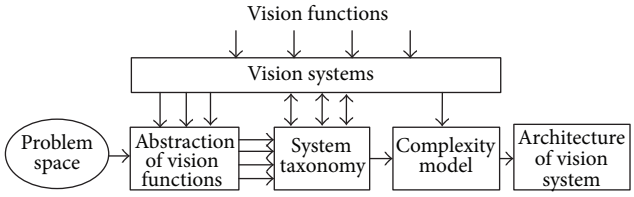


Figure 2: Problem statement and taxonomy formulation.

abstract model for comparison of different wireless smart camera systems. In relation to this, a number of published vision systems, wired and wireless, individual standalone vision systems, and vision nodes in Wireless Vision Sensor Networks (WVSNs) have been surveyed and a large number of functions were extracted. Similar functions were grouped together to make an abstraction of vision functions. The abstracted vision functions are then used to develop our proposed system taxonomy. The vision functions are used for building the taxonomy because the majority of vision applications in wireless smart cameras focus on target detection, analysis, and the recognition of objects present in the field of view [11]. The taxonomy building is an iterative process and it can be modified to accommodate future developments within the field so there are back and forth arrows. The taxonomy may not be exhaustive, but it does cover both the fundamental and the common vision functions which are required for a wide range of VSN vision systems. The study [12] showed that our proposed taxonomy covers 95 percent of the investigated systems. After the system taxonomy, the complexity in terms of arithmetic operations and memory of vision functions are investigated for the complexity model. The complexity model together with the taxonomy can be used for comparison and development of a generic architecture for different classes of VSN. The proposed taxonomy is shown in Figure 3. In this, some of the functions are labeled by capital letters, which are further expanded in Figures 4, 5, 6, 7, 8, 9, and 10.

The system taxonomy is grouped into 9 levels including data triggering type, data sources, data depth, learning, storage requirement, vision processing, postprocessing, output type, and feedback. A VSN is categorized into three types, based on how the system is triggered to start processing. In time driven systems, the processing is performed after a certain time duration. In event/query driven systems, processing starts when a system is triggered by a certain event or query. Periodic systems start processing after a fixed duration of time. The system can receive data using three types of sources including: area scan, line scan, or from another system. Images can be captured in binary, grey scale, or in colour format. Conversion from one format to another would require additional resources. Therefore, a better strategy is to capture the image in the relevant format. For some applications, the systems learn about the environment in order to adapt to it, while, in some applications, there is no need for learning. Similarly, there could be a requirement for storage in some systems in order to store frames, for example, for subtraction, for temporal filtering, or for template matching, while in others, there is no need of storage.

The vision processing level in Figure 3 shows the abstraction of vision functions which, while not exhaustive, include typical processing functions required for VSN. After the processing level, postprocessing occurs, which includes functions for the reformatting of data, that is, compression algorithms in order to make the output data suitable for communication purposes. Note that, in general, functions have an alternative path, which is able to circumvent them, in order to show that they might not be required for some VSNs. A VSN is also characterized by the output type it produces. The output can be a matrix, vector, scalar, or flag and can be sent directly to the user or can be used for feedback. The dashed line represents the system's flow for one experimental system, presented in Section 5 for illustration purposes. Following on from the system taxonomy, the complexity analysis of vision functions used in the system taxonomy is presented.

4. Complexity Model

The complexity analysis is a challenging task due to the large number of influencing factors such as the specific requirements of an application, the number of vision functions and the external environment. There is no standard definition in relation to measuring the vision system complexity.

However, in order to provide an abstract model, we have investigated both the arithmetic complexity and the memory requirements of the vision functions, employed in different classes of VSN systems with the assistance of the system taxonomy. The complexity analysis for some of the vision functions depends both on the situation and on the incoming data from the previous stage. Therefore, the absolute parameter for complexity measurements is a challenging task and it is not intuitive to draw quantitative conclusions. In these cases, we have analyzed and discussed the suitability of the functions for VSNs. Other parameters, such as registers and latency, are design dependent. The abstract model of complexity analysis is provided in Table 1. It is worth mentioning that for some tasks there are a number of algorithms, each with varying complexity. In this paper, we have investigated the complexity of functions that are commonly used in wireless smart camera systems [12]. However, in this case, the system taxonomy needs to be updated periodically in order to make it more exhaustive for the current systems.

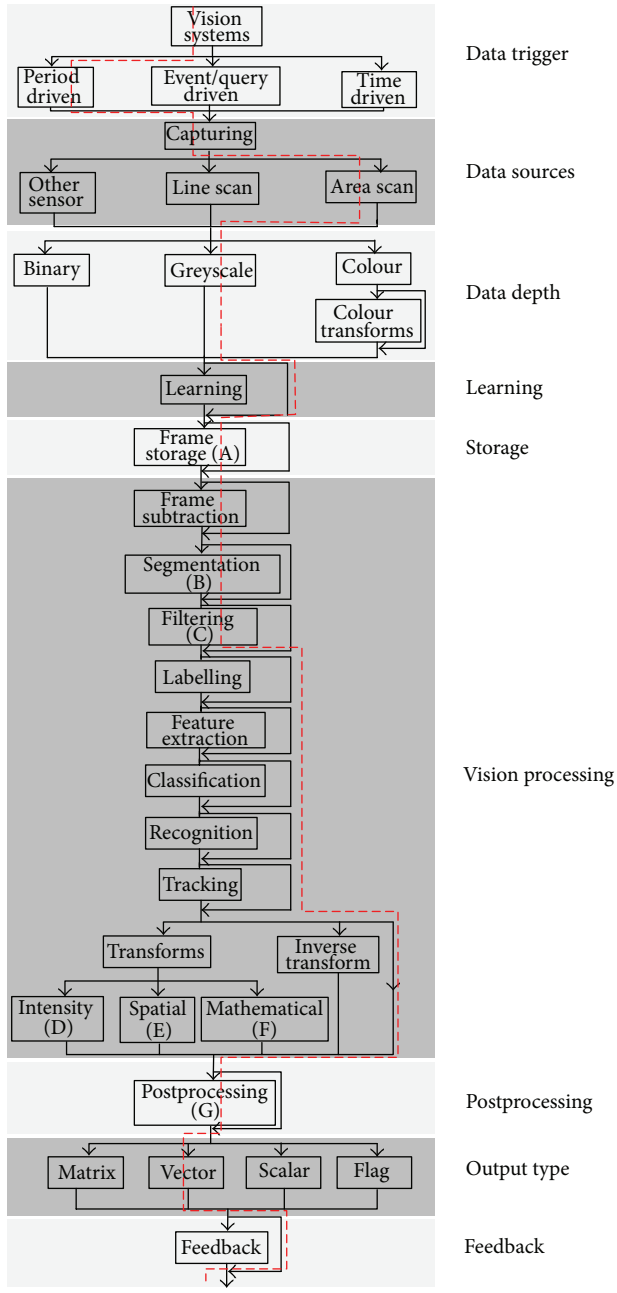


Figure 3: Taxonomy of vision system.

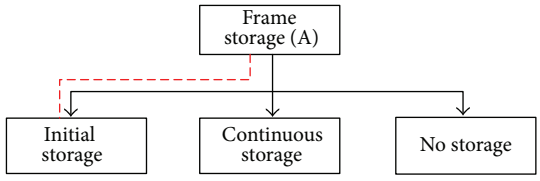


Figure 4: Storage in vision systems.

Nonetheless, this complexity model can be used together with the system taxonomy for classification, comparison, and complexity estimation of vision systems, implemented on VSNs. Equation (1) is used in Table 1 with the Hough transform, while (2) is used with labelling:

$$\rho(x, y) = x\cos\theta + y\sin\theta, \quad (1)$$

$$\text{mem}_{\text{total}} = \text{mem}_{\text{BUF}} + \text{mem}_{\text{LOOKUP}} + \text{mem}_{\text{DATA}}, \quad (2)$$

where $\text{mem}_{\text{BUF}} = (\log_2(\text{CC}_{\text{max}} + \text{CC}_{\text{col}} + 1)) \cdot C$, $\text{mem}_{\text{LOOKUP}} = \text{mem}_{\text{DATA}}$, and $\text{mem}_{\text{DATA}} = (2 \cdot (\log_2(R)))$.

In (2), C represents column, R represents Row, CC_{max} is the maximum number of connected components, CC_{col} is the number of label collisions, and +1 is because the 0 is a preoccupied label [13]. Following this, the complexity on actual hardware is discussed.

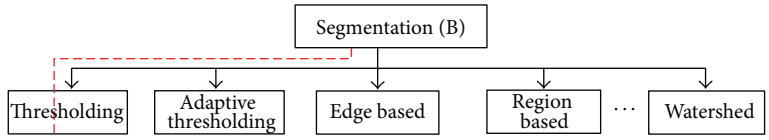


Figure 5: Segmentation in vision systems.

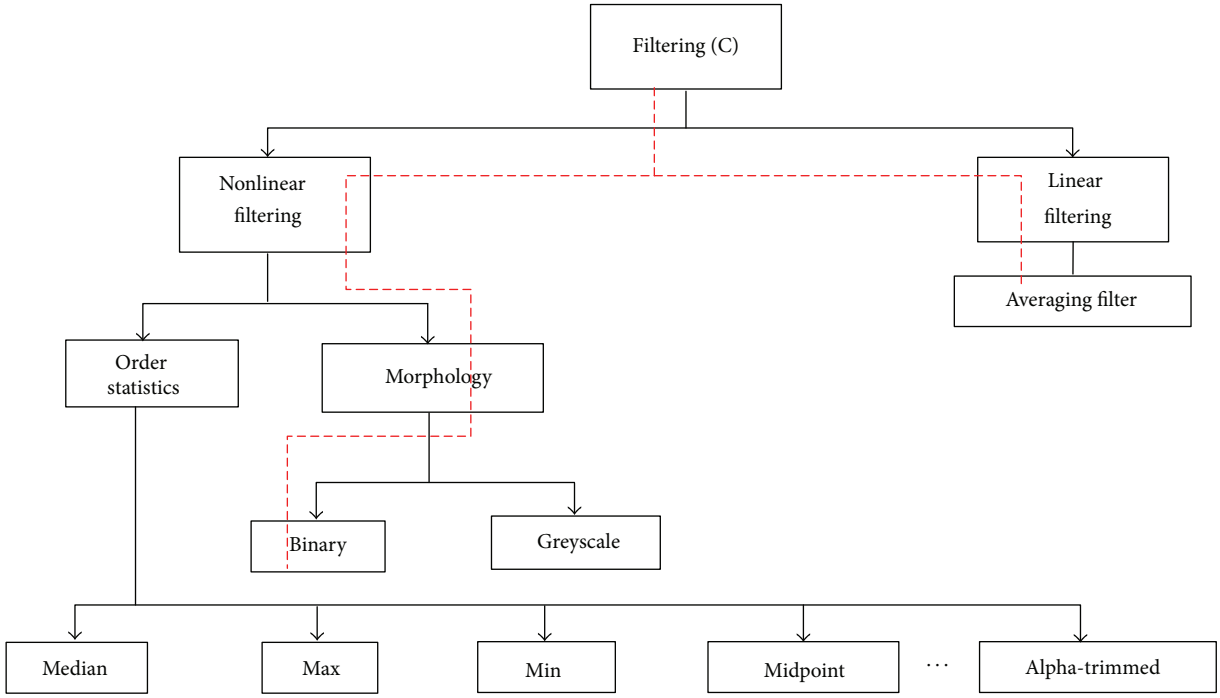


Figure 6: Filtering in vision systems.

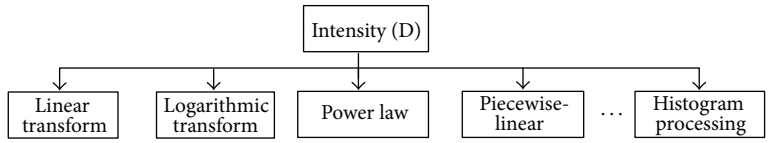


Figure 7: Intensity transformation.

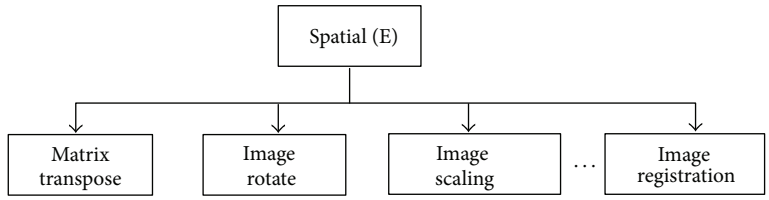


Figure 8: Spatial transformation.

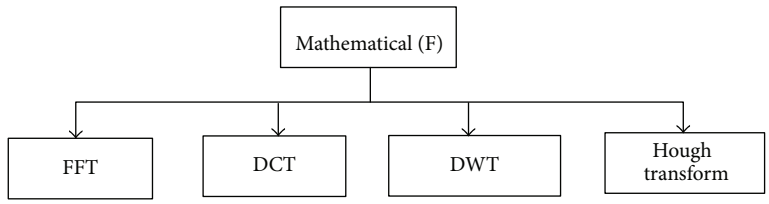


Figure 9: Mathematical transformation.

Table 1: An abstract complexity model of vision functions for VSN.

	Line memory	Frame memory	Arithmetic operations	Comments
Data triggering	N.A.	N.A.	N.A.	Depending on the application, triggering can be selected.
Data sources	W H	W H	W H	W is width and H is image height. Scaling of $W \times H$ affects mem. and arithmetic operation.
Data depth	b	b	b	b is bits per pixel. Scaling b affects the complexity.
Colour transforms	mem = bits × 9, where bits are coefficient bits width and 9 are coefficients.	N.A.	RGB to YCrCb conversion require $W \times H \times 9$ multiplications and $W \times H \times 6$ additions/subtractions.	Depending on the application, images can be captured in binary, grey scale, or colour format.
Learning	Application and algorithm dependent.	Application and algorithm dependent.	Application and algorithm dependent	Offline classifier with online training is preferable [1].
Frame storage	N.A.	mem = $W \times H \times b$ bits.	N.A.	Memory size depends on the pixel and image resolution.
Frame subtraction	Size depends on the background modelling technique.	Frame storage is required.	Technique dependent. Simple frame subtraction of size $W \times H$, require $W \times H$ additions/subtractions.	Static background or generated using modelling technique is used.
Segmentation				
Thresholding	N.A.	N.A.	Complete image require $W \times H$ logical operations.	Thresholding is used for simple cases.
Adaptive Thresholding	mem = $[(n-1) \cdot W \cdot b]$ bits where n is the number of rows for a neighbourhood.	N.A.	A technique based on a neighbourhood of $n \times n$ requires $n \cdot n$ additions, 1 division and 1 comparison is required.	Complete image requires $W \times H \times n \times n$ add., $W \times H$ divs., and comp. required.
Edge based*	mem = $[(n-1) \cdot W \cdot b]$ bits.	N.A.	Filter mask of $m \times n$ requires mn additions, mn multiplications.	Filtering an image $W \times H$, with a mask $m \times n$, requires $W \times H \times m \times n$ adds. and $W \times H \times m \times n$ mults.
Region based	Size depends on the selection of seed region.	Size depends on the selection of seed region.	Complexity depends on the selection of seed and order in which pixels and region are examined.	These techniques require much computational time.
Watershed*	Internal memory required depending on technique used.	mem = $W \times H \times b$ bits.	Depends on the technique such as immersion/flooding based and toboggan.	Having efficient techniques but still not suitable for real time.
Filtering				
Linear spatial	mem = $[(n-1) \cdot W \cdot b]$ bits	N.A.	The avg. filtering mask requires $m \times n$ adds. and 1 divs. and weighted avg. requires mults, adds., and div.	For complete image, multiplied image size with mask such as, for averaging $W \times H \times m \times n$ adds., $W \times H$ divs.
Order statistic	mem = $[(n-1) \cdot W \cdot b]$ bits	N.A.	The complexity of these filters depends on the selection of window size and the sorting algorithm.	The bit level implementation is usually better for hardware.

Table 1: Continued.

	Line memory	Frame memory	Arithmetic operations	Comments
Binary morphology	mem = $[(n-1)\cdot W\cdot b]$ bits	N.A.	Erosion and dilation with a mask $m \times n$ require $m \times n$ operations (AND/OR).	The run time complexity of dilation of two images is $O(WH)$.
Grey scale morphology	mem = $[(n-1)\cdot W\cdot b]$ bits	N.A.	Dilation and erosion algorithms have runtime complexity of $O(WHn)$ where n is roughly the number of points on the boundary of flat structuring elements.	Mathematical morphology is extended to grey scale images which is based on the notion of minimum and maximum.
Labelling	$\text{mem}_{\text{tot}} = \text{mem}_{\text{BUF}} + \text{mem}_{\text{LOOKUP}} + \text{mem}_{\text{DATA}}$ (see (2))	In classical approach. mem = $W \times H \times b$ bits.	Arithmetic complexity depends on the algorithm, image size, and number of objects in the image.	Single pass is preferred for real time component labelling [13].
Feature extraction	$\text{mem}_{\text{area}} = \log_2 N^2 \times$ no of objects. $\text{mem}_{\text{cog}} = (2\cdot(s\_\text{num}_{\text{bits}}) + S\_\text{din}_{\text{bits}}) \times$ no of objects. $\text{mem}_{\text{bbox}} = \log_2(\max(R_i, C_i)) \times$ no of objects.	N.A.	Complexity requirements involved in feature calculation depend on the number and size of objects.	Different types of features include position, width, height, bounding box, area, and centre of gravity [13, 24].
Classification	Application and machine learning algorithm dependent.	Application and machine learning algorithm dependent	This field is still at an immature stage so it is a challenging task to give absolute complexity.	SVM is suitable for on-node operation due to its low arithmetic complexity [2].
Recognition	Depends on design.	Depends on the number of objects, training samples, and algorithm.	The runtime complexity depends on the number of poses which can be reduced to gain the speed.	To avoid design complexities, it is better to use standard software packages.
Tracking	Application and learning algorithm and objects dependent.	Application and learning algorithm and objects dependent	Tracking algorithms can be simplified by imposing constraints on the motion and appearance of the objects.	Light weight algorithms are developed with certain constraints.
Intensity transforms	Histograms mem = $2^L \cdot b_w$ where $b_w = \log_2(W \cdot H)$ and b_w is bits for $W \cdot H$ pixels and L is intensity levels.	N.A.	Complexity depends on the technique such as pricewise linear transform or histogram processing being selected.	These spatial domain processings are computationally efficient and require less resources for implementation.
Spatial transforms	Memory buffers required depending on the technique.	mem = $W \times H \times b$ bits. In case of multiple rotation and image registration.	The arithmetic complexity depends on the operations performed.	This transform includes matrix transposes, image rotation, scaling, and registration.
Maths transforms				
Fast Fourier transform	Memory buffers required depending on the design.	mem = $W \times H \times b$ bits. For coefficients storage mem = $2((W \times H)/2) \times \text{bit}_{\text{coeff}}$	Direct implementation of DFT requires $(WH)^2$ operations while FFT reduces it to $WH\log WH$ operations.	Fast algorithms like radix-2^m, FPA, and FHT are good candidates in real time systems.

Table 1: Continued.

	Line memory	Frame memory	Arithmetic operations	Comments
Discrete cosine transform	$\text{mem}_{\text{coeff}} = \text{bits} \times N \times N$ where bits are coefficient bits and $N \times N$ is pixel block. $\text{mem}_{\text{pro}} = N \times W \times b$	mem = $W \times H \times b$ bits. In the case of parallel block processing.	Direct implementation of 2-D 8×8 DCT requires 1024 multiplications and 896 additions.	mem_{coef} is memory for precomputed coefficients and mem_{pro} is memory for processing.
Discrete wavelet transform	mem = $L \times N \times b$ where L is the number of rows and N is the number of pixels.	mem = $W \times H \times b$ bits.	The arithmetic complexity depends on the method being selected.	There are different implementation methods. Lifting based is a widely used method.
Hough transform	Memory buffers required for intermediate operations.	mem = $W \times H \times b$ bits. $W \times H \times r \times 2 \times \text{sqrt}(W^2 \times H^2) \times K$ where r is ratio of nonzero pixels in a binary image and K is number of angles [25].	For calculating 10% feature points by using (1), an image size of 256×256 requires 2.3 M multiplications and 1.1 M adders with an angle of 180° using a voting algorithm.	The Hough transform requires complex computations. Some efficient solutions include line based, multiplierless, incremental [25], and a Hough transform using convolution.
Postprocessing				
Lossy compression	Depends on the IP cores being used.	Depends on the IP cores being used.	Lossy compression is based on DCT and DWT which are discussed above.	IP core is preferred because of the availability.
Lossless compression	Depends on the IP cores being used.	Depends on the IP cores being used.	Depends on the algorithm and image contents as well as image size.	

*The memory requirements for edge based segmentation are given for techniques involving local processing. The memory requirements for edge based segmentation involving global processing are given in the column for Hough transforms. The memory requirements for watershed segmentation are given for techniques involving morphological watersheds.

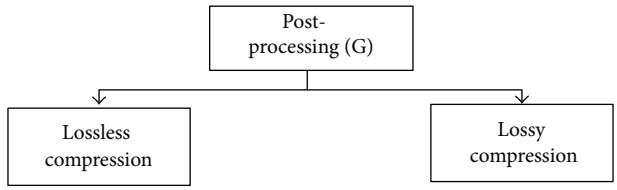


Figure 10: Postprocessing.

Table 2: Vision functions on software and hardware platform.

Vision functions	T_AVR32 (ms)	T_FPGA (ms)	P_FPGA (mW)	E_ AVR32 (mJ)	E_ FPGA (μJ)
Background subtraction	332.5	19.91	0.34	25.7	6.76
Segmentation	225	19.91	0.13	17.4	2.58
Morphology	2327	19.97	1.14	180.3	22.76
Low pass filter	202.5	19.91	0.18	15.6	3.58
Labelling and features	2610	19.91	2.7	202.3	53.76
ITU-T G4 compression	345	19.97	1.42	26.7	28.35

4.1. Complexity on Different Processing Platforms. Early vision processing tasks such as background subtraction, spatial filtering, have inherited parallelism. This parallelism can be exploited by using either a hardware platform or processors with multicores. With the advancements in technology, parallel machines with multicores have now spread from supercomputing to embedded computing. The recent evolutions of parallel machines have drawn the attention of researchers. There is a great potential for the multicore systems to be used for WVSN as the raw performance increases come from the increasing number of cores instead of the frequency. This approach will result in low power consumption [14]. However, there are different challenges in exploiting the parallelism in this emerging technology. The challenges include the parallel programming techniques, compilers for these architectures, and the management of memory hierarchy. The available vision libraries on uniprocessors are required to be tailored for multicore processors in order to exploit the parallel nature of image processing operations such as spatial filtering [5, 15].

The implementation of vision processing on reconfigurable platforms offers performances which are competitive with ASICs and, at the same time, providing flexibility in relation to design changes. In WVSN, the application requirement is often to capture the data for a particular time and then switch the node to a sleep mode so as to conserve the energy. This low duty cycling approach is suitable when the platform has a small sleep power. With the advent of FLASH based FPGAs with a small sleep power consumption [4] and development of techniques to use SRAM based FPGAs [16] effectively for duty cycle applications, the reconfigurable platforms are the choice of WVSN with regard to data intensive tasks. Uniprocessors such as embedded processors are commonly employed for vision processing because of the availability of ready-to-use libraries. In our previous work [17], a system has been implemented on both software and hardware platforms.

The functions of this system can be used in this case to provide a comparison of the processing complexity on the software and hardware platforms. By software, we mean microcontroller implementation and by hardware, we mean FPGA implementation. The vision functions used in the system include background subtraction, segmentation, morphology, filtering, labelling, feature extraction, and compression. For background subtraction, the background is stored in the initial stage in the FLASH memory and then subtracted from the current frame. After this, manual segmentation is applied in order to segment the objects from the background.

In morphology, a 3×3 erosion followed by dilation is applied and, in low pass spatial filtering, the previous binary frame is stored and then subtracted from the current frame to remove the unwanted objects. In labelling and feature extraction, objects are first assigned unique labels after which features information in the form of area and centre of gravity is calculated. The input sample image used in this experiment is shown in Figure 12(a). The input image used for this experiment has a size of 640×400 and contains real objects in the form of magnetic particles. These particles are used to predict failure in hydraulic machines. The processing time and energy consumed by each of the vision functions are given in Table 2. The processing time of these vision functions on the software and hardware platforms is represented by T_AVR32 and T_FPGA, respectively. The power consumption of each of the algorithms on the hardware platform is represented by P_FPGA, the energy of the software platform is represented by E_AVR32, and the energy consumption on the hardware platform is represented by E_AVR32. Table 2 shows that the vision functions require a small execution time on the hardware because of inherent parallelism as compared to the software platform. This results in a small energy consumption on the hardware platform. It must be noted that implementing more vision functions on the hardware will require both high design and development time.

On the contrary, the software platform has both a small design and development time because of the availability of

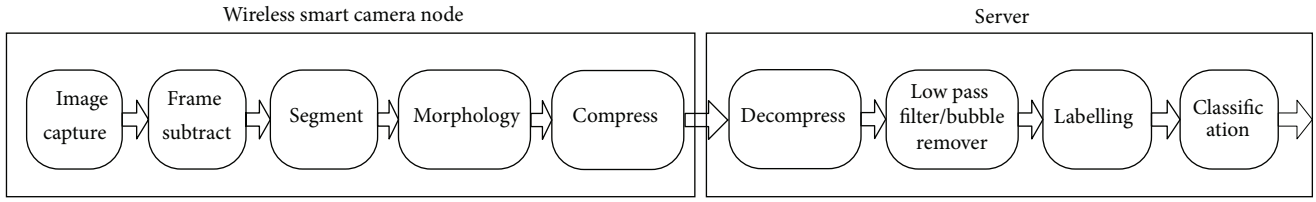


Figure 11: Algorithm flow for particle detection.

ready-to-use libraries. Depending on the requirements and constraints of the application, it is possible to select any of the platforms.

4.2. Energy Consumption. For battery operated wireless smart camera systems, the lifetime is an important consideration [3, 18, 19]. Battery lifetime can be extended by reducing energy consumption. The energy reduction can be achieved by reducing the processing time and/or the average power consumption. Processing time can be reduced by efficient implementation techniques and by introducing high performance embedded platforms. The power consumption in embedded platforms can be categorised into dynamic and static power. Dynamic power depends on the design and is related to switching signals from 0 to 1 or vice versa. Static power is related to power consumption when there is no switching on the signals and it is a function of the physical implementation. The dynamic power consumption is given by

$$P_{\text{dynamic}} = CfV_{\text{dd}}^2, \tag{3}$$

where C is capacitance, f is frequency, and V_{dd} is voltage.

In some applications, the peak performance is not always required so the operating frequency can be reduced for the time during which the node is in low performance node [20]. This will linearly decrease the power consumption as is evident in (3) shown by symbol f. However, in some real time applications, the timing constraints of the system may be violated by lowering the frequency. When the frequency is reduced for a design, requiring the same performance all the time, the frequency reduction would not affect the energy consumption because the design would take a longer execution time with a small frequency. The two factors, namely, the design complexity and voltage scaling, can offer a reduction in energy consumption. The design complexity is related to the capacitance C and voltage scaling is related to V_{dd} as shown in (3).

On system level, the voltage and frequency parameters for VSN's architecture are fixed [2, 4, 19, 21] because the components such as the interconnection between the devices, lighting, and memory require a fixed voltage. However, other alternatives such as better duty cycling approach, in node processing and suitable devices with low active and sleep power consumption can be investigated for extending the battery lifetime. For in node processing, complexity information of vision processing algorithms can be investigated with the help of proposed complexity model. This complexity information can be used as input for preimplementation evaluation tools such as Xilinx Xpower estimator [22] and Altera Early Power estimators [23] for power measurement and resource utilization before actual implementation.

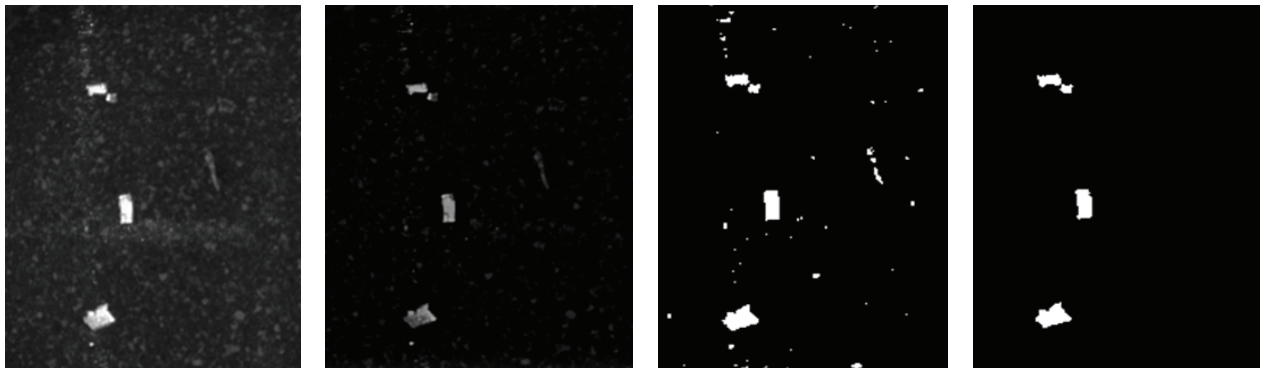
Following this, we will investigate a number of systems as a case study in order to provide a comparison of the vision systems. For this purpose, we will use the system taxonomy in order to identify a common class of systems with respect to the experimental system. After classification, a complexity model is used for resource estimation and then the vision systems are compared for implementation on a single generic architecture. The description of the experimental system is now provided.

5. Case Study: Particle Detection

To demonstrate the use of the system taxonomy and complexity model, a vision system [4], which we have developed for failure prediction of industrial machines, is selected as a reference system. The main focus in this system is to develop image processing/analysis methods to automatically detect magnetic particles, which are detached from machines and then transmit the information of these particles over a wireless link to the user. The vision function flow is shown in Figure 11 and the images at different stages of the algorithm are shown in Figure 12. By employing the approach of partitioning the vision functions between the VSN and the server [4], the vision functions such as image capturing, background subtraction, segmentation, filtering, and post-processing function, that is, ITU-T G4 compression, are performed on the VSN.

The compressed data is transmitted to the server in order to process the remainder of the vision functions. This system is classified by using the system taxonomy, depicted by means of a dashed line in Figure 3. The extended functions are shown with labels such as A for storage and B for segmentation. After the system classification, the complexity of vision functions is analyzed with the assistance of the complexity model of Table 1 and the outcome of this analysis is concluded in Table 3. After the classification and complexity analysis of the reference system, the architecture for this system is presented. In Section 6, we will identify systems with similar requirements with the assistance of taxonomy and will evaluate the target architecture for their implementations. In this manner, a single generic architecture can be identified/proposed for systems with similar requirements.

5.1. Target Architecture. The target architecture is presented in Figure 13 which includes a CMOS Micron Imaging



Original image (a) | Subtracted from background (b) | Segmented image (c) | Morphological operation (d)

Figure 12: Images at each algorithm step.

Table 3: Arithmetic complexity and memory estimation of reference VSN.

Functions	Line memory	Frame memory	Arithmetic operations	Comments
Data triggering	N.A.	N.A.	N.A.	Time driven.
Data sources	$H = 640$ $W = 400$	$H = 640$ $W = 400$	N.A.	Area scan, 640×400.
Data depth	$b = 8$ bits.	$b = 8$ bits.	$b = 8$ bits.	Grey scale, 8 bits.
Frame storage	N.A.	mem = $640 \times 400 \times 8$ = 2048000 bits.	N.A.	N.A.
Frame subtraction	N.A.	N.A.	$640 \times 400 = 256000$ subtractions/additions.	Background image is stored in FLASH memory.
Segmentation	N.A.	N.A.	$640 \times 400 = 256000$ comparison operations.	Manual thresholding is used.
Filtering Binary Morphology	mem = [(3–1)·640·1] = 1280 bits. mem = [(3–1) ·640·1] = 1280 bits.	N.A.	$640 \times 400 \times 9 = 2304000$ AND operations. $640 \times 400 \times 9 = 2304000$ OR operations.	For a mask of 3×3 being used, 2 line buffers are required for dilation and 2 for erosion.
Postprocessing lossless compression	mem = [3·640·1] = 1920 bits 3 line buffers for coding and Internal memory are required for storing 2 (27) + 2 (64) + 13 Huffman codes.	N.A.	The ITU-T G4 is used which includes arithmetic operations for run length coding and entropy.	Objects are in a white colour and the background is black so bilevel ITU-T G4 compression scheme is used, which is a lossless compression method.

sensor, MT9V032, and can be programmed through an I2C interface for different resolutions in real time. The camera has a maximum clock frequency of 26.6 MHz and is able to produce 60 frames per second. For vision processing, the Xilinx Spartan6 XC6SLX9L FPGA [28] is selected, which has 5720 logics, having 32, 18 Kbits block rams and 90 Kbits distributed rams. The vision functions include capturing, background subtraction from a frame stored at the initial stage in the FLASH memory, segmentation, filtering, and ITU-T G4 compression.

A serial FLASH memory [29] of 64 Mbits is used for background storage. For handling transmission, a software platform SENTIO32 [4] is used which has a 32 bit AVR microcontroller, AT32UC3B0256 [30] and an IEEE 802.15.4 compliant transceiver (CC2520) [31]. In the proposed target architecture, the FPGA has 12.5 mW static power and a dynamic power of 16.92 mW for the design, which includes algorithms such as background subtraction, filtering to remove noise, segmentation, and compression, the AVR32 microcontroller has 77.5 mW active power, the camera has 160 mW, and the radio transceiver has 132 mW. By evaluating this architecture with a small static power consumption such as 5 μW, which is claimed in FLASH based ACTEL AGL600V5 FPGA [32], a greater lifetime can be achieved for the VSN. It is concluded [18] that a VSN with this architecture results in a lifetime of approximately 5 years for a sample period of 1.5 minutes.

Tipping Points of Failure. It is important to know the conditions under which the architecture fails to offer the desired functionality. This happens because, at some point, the resources available on the architecture could not accommodate the expected design. By looking at a number of factors

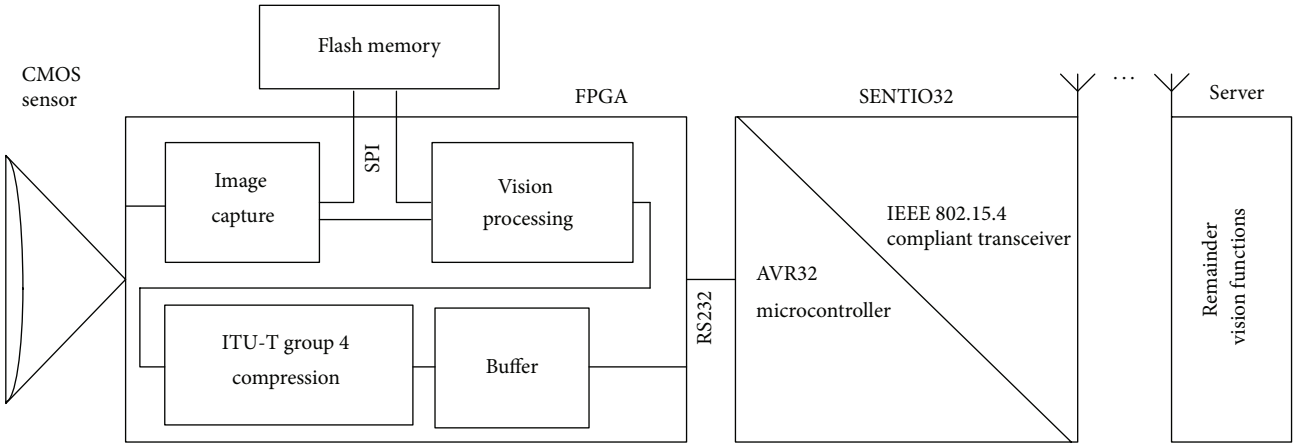


Figure 13: Target architecture of VSN.

such as clock frequency, memory, logics, communication among devices, and field of view, it is apparent that the clock frequency could not be a failure point in the architecture when there are no real time constraints. Depending on the speed, the clock frequency may increase or decrease the frame rate. For example, by lowering the clock frequency from 26.6 MHz to 13.3 MHz for the proposed target architecture in Figure 13, the frame rate is reduced from 60 to 30. It is important to note that in real time systems, where the timing constraints are important, it could result in a failure of the system. The architecture will fail to offer the required functionality and performance when the available resources, that is, internal memory, external memory, and logics are limited. Suppose, in the experimental system, the image sensor is changed to a size of 2000 × 3000 and the vision function, low pass filtering of Figure 10, is moved from the server to the VSN. This function requires the storage of a complete binary image in the internal memory in order to perform the operation. The complete binary image for the specific resolution would require 5859.37 Kbits in the internal memory, while the total internal memory available is 576 Kbits (32, 18 Kbits block rams). Similarly, when the RGB image is required to be stored for background, it would require 137.32 Mbits and the total available FLASH is 64 Mbits.

The communication among different devices, that is, hardware, software, and transceiver is important for a stable architecture because different devices are running at different speeds. There is a requirement to select a suitable size buffer in order to handle device communication, failing which could cause overflow/underflow for the data in a pair device. In real time systems, this could cause system failures in the systems. One of the critical factors of a VSN is the coverage area of the image sensor. A VSN is able to monitor a limited number of objects within the field of view and the missing of some of the objects may lead to a failure of the system, that is, in surveillance applications.

6. Comparison of Vision Systems

A comparison of different vision solutions is essential for the improvement of the current research work. In a traditional method of comparison, the systems under consideration must be implemented on each other's architecture. Suppose we have selected six sample systems (V1, V2, V3, V4, V5, and V6). The system V1, selected as the reference system, must be compared with the other five systems. In the traditional approach, depicted by Figure 14, five systems (V2, V3, V4, V5, and V6) must be implemented on the reference system's architecture. Similarly, the reference vision system is required to be implemented on the corresponding architectures of the five vision systems. This requires a significant amount of design effort and time. When the reference system is changed, the aforementioned process is repeated once more. It means that, for comparing N systems, $(N-1)^2$ implementations are required. By employing the system taxonomy and complexity model for comparison, the need for actual implementation can be circumvented. This approach is depicted in Figure 15. The six systems (V1, V2, V3, V4, V5, and V6) are firstly classified by using the system taxonomy to identify systems with similar functionality with regard to reference system V1.

The systems such as system V5 and V6, having different functionality, are dropped from any further investigation. In this way, larger problem space is reduced to a smaller one. After classification, the complexity model is used to generate complexity parameters, that is, arithmetic complexity, memory resources, and device selection, that is, processing platform, radio transceiver, and microcontroller for control functions. After the comparison, a single generic architecture can be proposed or an existing architecture can be employed for real implementation.

6.1. Example for Comparison of Systems. In this example, an actual vision system V1 [4] is compared with the other five systems (V2 [2], V3 [26], V4 [27], V5 [21], and V6 [19]) by using our proposed approach of Figure 15. We need to identify the systems which have similar functionality with respect to system V1. After this, the complexity model is used to estimate the resources and then a single generic architecture is evaluated for these systems.

Systems Classification. The systems are firstly classified by using the system taxonomy, and based on those systems

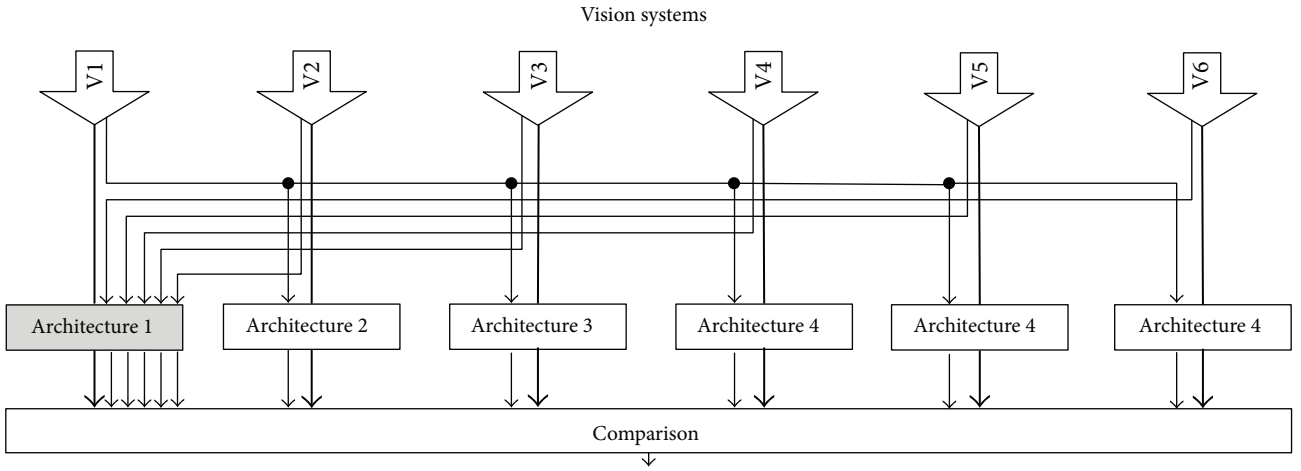


Figure 14: Comparison of vision systems using traditional approach.

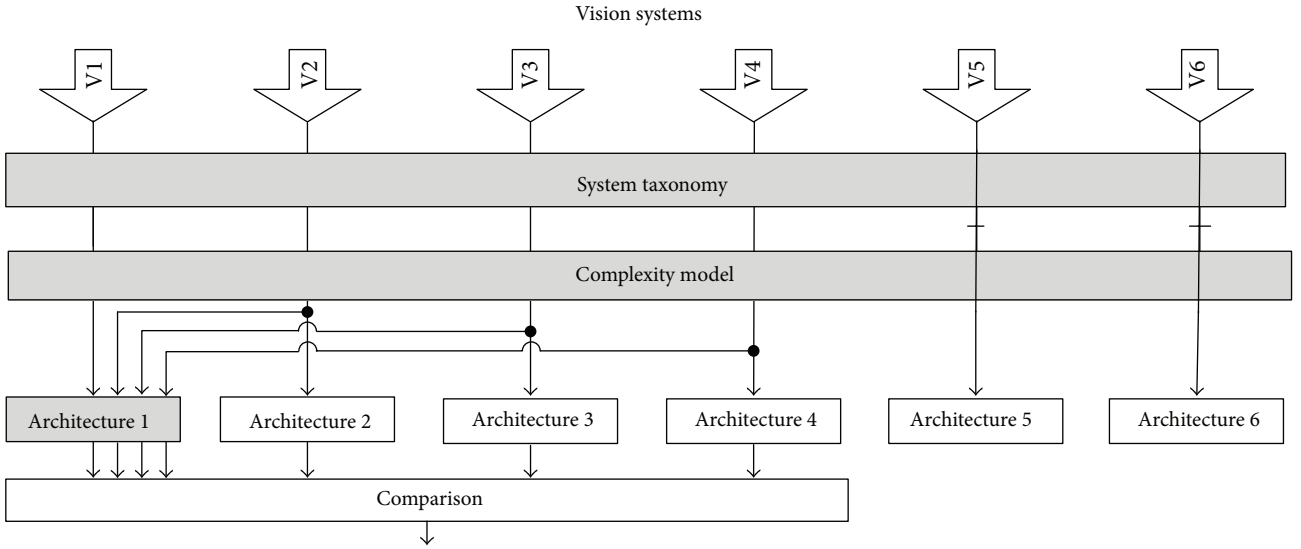


Figure 15: Comparison of vision systems using proposed model.

having a common functionality, are grouped into one class. By looking at all six systems, it is concluded that systems V2 [2], V3 [26], and V4 [27] have similar functionality with respect to V1 [4]. The common functionality is classification on binary data because vision functions include background subtraction, segmentation, spatial filtering, and segmentation. The system V3 [26] is placed in this group because its vision functions are similar to those of the reference system V1, but background storage is not used by the authors. This is, however, required for real time implementation. The vision systems V5 [21] and V6 [19] are dropped from further investigation because in V5 [21] the background is updated periodically and in V6 [19], and the system does not use segmentation function which converts data into binary format. The taxonomy showed that these systems do not involve classification on binary data. Therefore, these systems cannot utilize the architecture, proposed in Section 5.1.

Resource Estimation. Following the system classification, the resource requirements for the systems are estimated by using the complexity model. After this, the Xilinx synthesis tool [28] is used to generate the resource information, that is, logic and memory. The arithmetic operations are converted into logics per sample; for example, for subtraction with 8 bits pixel, 9 logics are required and the total logics of the design include logics for arithmetic, synchronization, and compression. The logics used for the synchronization of data are considered to be similar for systems when the pixel depth and line size are similar; otherwise, they were obtained separately from the synthesis tool. The line memory (L_MEM), frame memory (F_MEM), and logics are in close proximity with respect to the reference system V1 resources. Therefore, the systems are evaluated for implementation on a single architecture.

Table 4: Cost and performance parameters of different systems.

Systems/ references	Image size	L_MEM	F_MEM	Arithmetic operations	Logic used	% age L_Mem	% age F_Mem	% age logic	Max. fps	Max. Pixel freq (MHz)	Implement on proposed architecture?
V1 [4]	640 × 400	4480	2048000	5120000	1873	4.9	3.2	32	59.5	15.2	√
V2 [2]	640 × 480	1920	7372800	614400	1221	2.1	11.5	21	25	7.6	√
V3 [26]	320 × 240	960	0	384000	1824	1.0	0.0	31	N.A.	N.A.	√
V4 [27]	640 × 480	1920	7372800	614400	1221	2.1	11.5	21	1.3	0.4	√
V5 [21]	128 × 128	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	4.1	0.06	×
V6 [19]	640 × 480	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	N.A.	5.7	1.7	×

Evaluation on Single Generic Architecture. Following the resource estimation, the investigated systems are evaluated for implementation on a single generic architecture. We can develop a new architecture or can use the existing architecture of Section 5.1 which is suitable for all of these systems. In these systems, the approach used by the system V1 can be employed in which vision functions such as background subtraction, segmentation, spatial filtering, and binary compression are implemented on VSN.

The compressed data is transmitted to the server for further processing. In Table 4, the percentage of resources used by each system is given with respect to the resources of architecture, presented in Section 5.1. The proposed architecture has 5720 logics, 32, 18 Kbits block rams, and 90 Kbits distributed rams. The systems V1 [4], V2 [2], V3 [26], and V4 [27] have resource requirements within the range of the architecture's resources and can therefore be implemented on the target architecture.

The performance parameters of different systems, given in Table 4, show that the system V1 has a greater sampling pixel frequency of 15.2 MHz. The pixels frequencies for other systems are given for their respective older implementations, which can be scaled to 15.2 MHz when implemented on the architecture of Section 5.1. This demonstrates that, with the assistance of our proposed approach, the system complexity can be easily estimated and a number of systems can be compared without the need for actual implementation. After this, a single generic architecture can be proposed for the same class of systems. This concludes that taxonomy together with complexity model can be effectively used for classification and comparison of solutions which in turn helps in proposing generic solutions.

7. Future Research

With the advancement in technology, the multicore processors are expected to grow in the future within the field of embedded vision systems. There are a number of challenges in employing these parallel machines directly for WVSN applications. The vision functions available for software implementation were developed for single processor platforms. These algorithms must be redesigned to exploit the parallelism. The early vision functions such as background subtraction and temporal filtering require data storage in an external memory. These tasks require the processing architecture with faster storage, high bandwidth memory interfaces, and data buses. To reduce the memory latency, the embedded multicore systems are required to have Direct Memory Access (DMA) based data transfer so that the cores are executing the tasks and data transfers between various modules such as cores, internal memory, and external memories are handled by DMA [15]. This will create new design challenges in relation to creating parallel programs for multicore systems. To extend the battery lifetime, suitable dynamic frequency scaling and voltage scaling techniques must be considered for the VSN architecture.

8. Conclusion

In this paper, we have presented an abstract model for the comparison and generalization of vision solutions for wireless smart cameras, with the assistance of system taxonomy. To develop this model, we have performed a detailed analysis of the vision functions, used in the system taxonomy, in order to predict the arithmetic complexity and memory requirements. To illustrate the use of the proposed model, we have analyzed a number of published systems as a case study and classified them by using the system taxonomy. Following this, the resources have been estimated with the assistance of the complexity model. It has been demonstrated that the proposed model helps in comparison of vision systems without the requirement for any actual implementation. After comparison, a single generic architecture can be proposed for the same class of vision systems. This work may not be exhaustive; however, it will provide an abstract reference model for benchmarking and development of efficient generic solutions.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] C. Leistner, P. M. Roth, H. Grabner, H. Bischof, A. Starzacher, and B. Rinner, "Visual on-line learning in distributed camera networks," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '08)*, pp. 1–10, Stanford, Calif, USA, September 2008.

[2] A. Orfy, A. El-Sayed, and M. ElHelw, "WASP: wireless autonomous sensor prototype for visual sensor networks," in *Proceedings of the IFIP Wireless Days (WD '10)*, pp. 1–5, Venice, Italy, October 2010.

[3] S. Hengstler and H. Aghajan, "Application-oriented design of smart camera networks," in *Proceedings of the 1st ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '07)*, pp. 12–19, Vienna, Austria, September 2007.

[4] M. Imran, K. Khursheed, M. O'Nils, and N. Lawal, "Exploration of target architecture for a wireless camera based sensor node," in *Proceedings of the 28th Norchip Conference (NORCHIP '10)*, pp. 1–4, Tampere, Finland, November 2010.

[5] N. K. Ratha and A. K. Jain, "Computer vision algorithms on reconfigurable logic arrays," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 1, pp. 29–43, 1999.

[6] M. Imran, K. Khursheed, N. Ahmad, M. A. Waheed, M. O'Nils, and N. Lawal, "Complexity analysis of vision functions for implementation of wireless smart cameras using system taxonomy," in *Real-Time Image and Video Processing*, vol. 8437 of *Proceedings of SPIE*, Brussels, Belgium, 2012.

[7] B. Dieber, C. Micheloni, and B. Rinner, "Resource-aware coverage and task assignment in visual sensor networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 10, pp. 1424–1437, 2011.

[8] C. H. Lin, W. Wolf, A. Dixon, X. Koutsoukos, and J. Sztipanovits, "Design and implementation of ubiquitous smart cameras," in *Proceedings of the IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing*, vol. 1, pp. 32–39, Taichung, Taiwan, June 2006.

[9] B. Rinner, T. Winkler, W. Schriebl, M. Quaritsch, and W. Wolf, "The evolution from single to pervasive smart cameras," in *Proceedings of the 2nd ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC '08)*, pp. 1–10, Stanford, Calif, USA, September 2008.

[10] S. Tilak, N. Abu-Ghazaleh, and W. Heinzelman, "A taxonomy of wireless micro-sensor network models," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 6, no. 2, pp. 28–36, 2002.

[11] S. Hengstler, "Stereo vision in smart camera networks," in *Stereo Vision*, A. Bhatti, Ed., pp. 73–90, InTech, Rijeka, Croatia, 2008.

[12] M. Imran, K. Benkrid, K. Khursheed, N. Ahmad, M. O'Nils, and N. Lawal, "Analysis and characterization of embedded vision systems for taxonomy formulation," in *Real-Time Image and Video Processing*, vol. 8656 of *Proceedings of SPIE*, Burlingame, Calif, USA, 2013.

[13] R. Walczyk, A. Armitage, and T. D. Binnie, "Comparative study on connected component labeling algorithms for embedded video processing systems," in *Proceedings of the International Conference on Image Processing, Computer Vision and Pattern Recognition (IPCV '10)*, p. 7, Las Vegas, Nev, USA, 2010.

[14] G. Blake, R. G. Dreslinski, and T. Mudge, "A survey of multicore processors: a review of their common attributes," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 26–37, 2009.

[15] S. Apewokin, *Efficiently mapping high-performance early vision algorithms onto multicore embedded platforms [Ph.D. thesis]*, Georgia Institute of Technology, Savannah, Ga, USA, 2009.

[16] J. Valverde, A. Otero, M. Lopez, J. Portilla, E. de la Torre, and T. Riesgo, "Using SRAM based FPGAs for power-aware high performance wireless sensor networks," *Sensors*, vol. 12, no. 3, pp. 2667–2692, 2012.

[17] K. Khursheed, M. Imran, A. W. Malik, M. O'Nils, N. Lawal, and B. Thrnberg, "Exploration of tasks partitioning between hardware software and locality for a wireless camera based vision sensor node," in *Proceedings of the 6th International Symposium on Parallel Computing in Electrical Engineering (PARELEC '11)*, pp. 127–132, Luton, UK, April 2011.

[18] M. Imran, K. Khursheed, N. Lawal, M. O'Nils, and N. Ahmad, "Implementation of wireless vision sensor node for characterization of particles in fluids," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 11, pp. 1634–1643, 2012.

[19] A. Kerhet, M. Magno, F. Leonardi, A. Boni, and L. Benini, "A low-power wireless video sensor node for distributed object detection," *Journal of Real-Time Image Processing*, vol. 2, no. 4, pp. 331–342, 2007.

[20] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low power embedded operating systems," in *Proceedings of the 18th Symposium on Operating System Principles (SOSP '01)*, vol. 35, pp. 89–102, New York, NY, USA, 2001.

[21] M. Rahimi, R. Baer, O. I. Iroezi et al., "Cyclops: in situ image sensing and interpretation in wireless sensor networks," in *Proceedings of the 3rd international Conference on Embedded Networked Sensor Systems (SynSes '05)*, San Diego, Calif, USA, 2005.

[22] Xilinx, Xpower estimator user guide, 2012, Xilinx power tools tutorial, 2010, http://www.xilinx.com.

[23] Altera, Powerplay early power estimators, 2012, http://www.altera.com/support/devices/estimator/pow-powerplay.jsp.

[24] A. Hornberg, *Handbook of Machine Vision*, John Wiley & Sons, Weinheim, Germany, 2006.

[25] Z.-H. Chen, A. W. Y. Su, and M.-T. Sun, "Resource-efficient FPGA architecture and implementation of Hough transform," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 8, pp. 1419–1428, 2011.

[26] A. C. Bianchi and A. H. Reali-Costa, "Implementing computer vision algorithms in hardware: an FPGA/VHDL-based vision system for a mobile rob," in *RoboCup 2001*, Lecture Notes in Computer Science, pp. 281–286, 2002.

[27] C. B. Margi, R. Manduchi, and K. Obraczka, "Energy consumption tradeoffs in visual sensor networks," in *Proceedings of the 24th Brazilian Symposium on Computer Networks (SBRC '06)*, Curitiba, Brazil, 2006.

[28] Spartan-6 family overview, 2010, http://www.xilinx.com.

[29] Micron, Numonyx Serial Flash Memory, 2007, http://www.micron.com.

[30] AT32UC3B0256, AVR32, http://www.atmel.com/.

[31] CC2520 transceiver, http://www.ti.com/.

[32] IGLOO video kit, 2009, http://www.actel.com/.