Low power architecture for fall detection system

Thi Khanh Hong Nguyen

▶ **To cite this version:**

Thi Khanh Hong Nguyen. Low power architecture for fall detection system. Other. Université Nice Sophia Antipolis, 2015. English. NNT : 2015NICE4093 . tel-01288526

HAL Id: tel-01288526

https://theses.hal.science/tel-01288526v1

Submitted on 15 Mar 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

UNIVERSITE NICE-SOPHIA ANTIPOLIS

ECOLE DOCTORALE STIC

SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

T H E S E

pour obtenir le titre de

Docteur en science

De l'Université Nice-Sophia Antipolis

Mention:

Présentée et soutenue par

Thi Khanh Hong NGUYEN

Low power architecture for Fall Detection System

Thèse dirigée par Cecile BELLEUDY et Van Tuan PHAM

Soutenue le 18 Novembre 2015

Jury:

Mme. Nathalie Julien	Professeur à l'Université de Bretagne Sud	Rapporteur
M. Bertrand Granado	Professeur à l'Université de Pierre and Marie Curie	Rapporteur
Mme. Cecile BELLEUDY	Maitre de Conférences, HDR, Université Nice Sophia Antipolis	Directrice
M. Van Tuan PHAM	Associate Professeur, Université de Science et Technologie, Université de Da Nang.	Co-directeur
M. François Brémond	Directeur de recherche de INRIA	Examinateur

Contents

Figures

Tables

Abstract

Nowadays, fall detection is a major challenge in the public health care domain, especially for the elderly living alone. Falls are the leading cause of injury deaths among older adults, those aged 65 or older. Moreover, the lack of medical staff, who take care of rehabilitants in hospital, is an urgent problem in the 21st century. Therefore, the demand for surveillance systems, especially for fall detection, has considerably increased within the healthcare industry.

This thesis presents an exploration for a Fall Detection System based on camera under an algorithmic and architectural point of view. The studied Fall Detection System is suitable not only for the elderly living alone, but also for rehabilitants in hospitals. Our system is composed of four modules: Object Segmentation, Filter, Feature Extraction and Recognition and can give an urgent alarm for detecting different kinds of fall.

Firstly, different algorithms are proposed and studied for the modules which compose the Fall Detection System like the Background Subtraction-Neural Network (BGS-NN), the Background Subtraction-Template Matching (BGS-TM), the Background Subtraction-Hidden Markov Model (BGS-HMM), and the Gaussian Mixture Model (GMM-HMM). In order to evaluate the efficiency of these algorithms, a comparison is made on the accuracy (Acc), precision (PR) and recall (RC) performance. This comparison leads to select the BGS/TM which will be used for the remainder of this research work. This algorithm is simulated on Matlab with 91.67% (RC), 100% (PR) and 95.65% (Acc) and implemented on ZYNQ platform by using C++ and OpenCV. Furthermore, in order to evaluate the efficiency of our Fall Detection System, a DUT-HBU database which is classified with different actions: fall, non-fall (sitting, lying, creeping, etc.) in three camera directions (face, sides and cross) is created and used for simulation, evaluation and implementation purposes.

Secondly, the aim is to define a methodology to explore low cost architectures for this fall detection system. As we consider heterogeneous architecture, new power consumption and execution time models for processor core and FPGA are defined according to the different configurations of the target architecture (ZYNQ platform) and the features of the applications like: core frequency, number of processor cores, image resolution. To validate the accuracy of the proposed models, we also analyze the error rate of these models that show that they don't exceed 3.5%. The power consumption and execution time models are then extended to hardware/software architectures according to the assignment of the Fall Detection System tasks and have been coupled with an accuracy model to evaluate the performance of this system. With these extended models, our approach targets to explore low cost architecture

by defining a suitable Design Space Exploration methodology. We also apply two techniques for parallelization which are based on intra-task and inter-task static scheduling with the aim to enhance the accuracy and the power consumption of this system. As example, first execution on ARM Cortex A9 processor of ZYNQ platform achieves an accuracy of 62% with energy per frame of 43mJ/f. When the parallel techniques based on hardware/software architecture are applied, the frame rate of our system is considerably increased and the accuracy rate reaches 98.3% with energy per frame of 29.5mJ/f.

Résumé

De nos jours, la détection de chute est un défi majeur dans le domaine de la santé publique, en particulier pour les personnes âgées vivant seules. Les chutes sont la principale cause de décès, suite à des blessures, chez les personnes âgées de plus de 65 ans. Par ailleurs, le manque de personnel médical, par exemple pour les patients en rééducation, dans les hôpitaux ou les maisons de retraite, est un problème important mondial pour le 21ème siècle. Par conséquent, le développement de systèmes de surveillance, en particulier pour la détection de chute, devient une nécessité pour le secteur de la santé.

Le but de cette thèse est de concevoir un système de détection de chute basée sur une surveillance par caméra et d'étudier à la fois les aspects algorithmiques et architecturaux. Notre système de détection de chute est conçu non seulement pour les personnes âgées vivant seules, mais aussi pour les personnes en réadaptation à l'hôpital ou placée en maison de retraite. Notre système se compose de quatre modules: la segmentation d'objet, le filtrage, l'extraction de caractéristiques et la reconnaissance qui permettent en plus de la détection de chute d'identifier le type de ces chutes (avant, arrière, coté) dans le but de définir un niveau d'alarme.

Dans un premier temps, différents algorithmes ont été étudiés pour réaliser les traitements des modules qui composent notre système de détection de chute comme le Background Subtraction-Neural Network (BGS-NN) ; le Background Subtraction-Template Matching (BGS-TM) ; le Background Subtraction-Hidden Markov Model (BGS-HMM) ; et le Gaussian Mixture Model (MGM-HMM). Afin d'évaluer l'efficacité de ces algorithmes, une comparaison est effectuée sur les paramètres qui permettent d'évaluer la performance du système soit: Accuracy (Acc), Precision (PR), Recall (RC). Le résultat de cette comparaison nous a amené à sélectionner le BGS/TM qui sera utilisé dans la suite de ces travaux de recherche. La simulation de cet algorithme sous Matlab a permis d'évaluer le RC à 91,67%, le PR à 100% et l'ACC à 95,65%. Cet algorithme a aussi été testé après implémentation sur une plateforme ZynQ en utilisant le langague C++ et OPENCV. De plus, afin de mieux évaluer l'efficacité du système de détection de chute proposé, une base de donnée DTU-HBU a été construite et classifiées selon les différentes actions: chute, non-chute (assis, couché, rampant, etc.) selon trois angles de caméra (de face, de côtés et de biais).

Dans un second temps, l'objectif a été de définir une méthodologie permettant de sélectionner les architectures à faible coût qui présenteraient les meilleures performances. Comme nous considérons des architectures hétérogènes, un premier travail fut de définir des modèles de consommation et du temps d'exécution pour différentes cibles technologiques,

processeur et FPGA. A titre d'exemple, la plateforme ZYNQ a été considérée. Les différentes configurations de cette plateforme ont été caractérisées pour les algorithmes de détection de chute. En particulier, des paramètres comme la fréquence, le nombre de cœurs actifs, la résolution de l'image ont été pris en compte. Pour valider la précision des modèles proposés, des expérimentations ont été menées et l'erreur pour ces tests n'a pas excédé 3,5%. Ces modèles ont ensuite été étendus à des architectures hétérogènes et complétés par un modèle de l'Accuracy (ACC) qui permet d'évaluer la performance du système complet. Sur la base de ces modèles, notre approche vise à explorer les architectures à faible coût par la définition d'une méthodologie adaptée de DSE. Afin d'exploiter le parallélisme offert par la plateforme ZYNQ, deux techniques d'ordonnancement statique (Intra tâche et inter tâche) ont été utilisées dans le but de réduire la consommation d'énergie tout en possédant une performance adaptée. Les résultats obtenus montrent qu'une première implémentation fournit un ACC=62% avec une énergie par image de 43 mJ alors qu'après optimisation, l'ACC atteint 98,3% pour une énergie de 29,5 mJ.

Acknowledgement

I would like to express my deep and sincere gratitude to all those who have helped and supported me the possibility to complete this thesis.

I am deeply indebted to my supervisor and principal academic advisor, Mme Cecile BELLEUDY, who has always been amazingly thoughtful, enthusiasm and sharp. Her constructive advice, belief and support, from research strategy and general approach to detailed writing styles, have been invaluable. Mme Cecile BELLEUDY has given me trust, especially in difficult moments, and certain freedom to develop a research topic.

My deepest gratitude also goes to Associate Professor Van Tuan PHAM that I am truly passionate about, for his continual guidance on the first ideals for this research topic. I sincere thank him to give me an opportunity to participate the co-research project between University of Nice Sophia Antipolis and University of Danang under supporting of EMMA's program in ten months.

I would also like to thank my official referees, Professor N. Julien and Professor Bertrand Granado, for their thoughtful and detailed review, constructive criticism and excellent advice during the preparation of this thesis.

I wish to express my sincere thanks to Human Behaviour Understanding group in Da nang University of Technology (HBU-DUT) for working together in developing the algorithm of Fall Detection, the database called HBU-DUT and some achievements in simulation during the first PhD's year.

I warmly thank the members of the MCSOC group for their valuable advices on my research methodology and contributions. Special thanks also go to Professor VERDIER François, Mr PEGATOQUET Alain and Mr BILAVARN Sébastien.

During this time I have collaborated with many colleagues for whom I have great regard, and I wish to extend my warmest thanks to all those who have helped me with my work for all their help, support, interest and valuable hints. Special thanks also go to Professor DAUVIGNAC Jean Yves, Professor AUGUIN Michel, Mme PROSILLICO Marie Hélène and Mme GUYON Marie-France for all their administrative supports.

My deepest gratitude also goes to the board of principals at the College of Technology, University of Danang. Futhermore, I want to thank all my colleagues of the Electronic Department to share the teaching work for me during last 4 years. Thanks for all

Vietnamese friends support and share the difficult and unforgettable moments in research and in my life.

I owe my loving thanks to my family for all their love and encouragement. Without their encouragement and understanding it would have been impossible for me to begin, continue and finish this work. Thanks for my little daughter giving me the most motivation to finish this PhD. I love you all dearly.

This work was support by the Vietnam Ministry of Education and Training and the ten months of Erasmus Mundus Mobility with Asia - EMMA in Exchange Doctoral Research project.

Chapter 1. Introduction

Currently, accompanying with the development of society, e-health systems are playing increasingly important roles in our lives. Among them, medical care and falling accidents for the elderly seem to draw attention as important topics in healthcare and human behaviour recognition domains. In addition, to efficiently monitor the situation of patients, there is at least a medical staff (i.e. a doctor or a nurse) who presents next to patients for hours. The extremely increasing demands on healthcare services have been, thus, urging the development of new generation of surveillance systems. These novel systems can be benefit from new advances in sensors, digital video processing, and broadband access network infrastructures. Therefore, the health services today have been dominated by high-tech devices as well as automatic systems which have greatly facilitated the higher ability in convenience, fast response, and high reliability.

According to the statistics given in the international advisory conference on innovation, training of health forces in the 21st century took place on 28/04/2011 in Hanoi, Vietnam showed that there were only 0.5 doctors and/or 0.8 nurses per 1000 population. It means that according to the World Health Organization (WHO) Vietnam should add about 80 thousand health forces to meet the general requirements [1]. That was the general reality of the countries on over the world, including developing countries. In addition, another study of the elderly from Vietnam Association of Rheumatology, every year, every three people over 65 years old have at least one person fell. The accident was the sixth most common causes of death for the elderly, in which the incidence of falls was the majority [2]. In Vietnam, the proportion of people of 60 years and older increased from 6.7% in 1979 to 9.2% in 2006 [3]. Vietnamese life expectancy at birth increased from 66 years in 1990 to 72 years in 2006 and the average life expectancy of elderly Vietnamese is 73 years old [4].

The application of the intelligent video surveillance for taking care the elderly at home or the rehabilitants in Vietnam's hospital is proposed to find out the solutions for this reality in Vietnam. Before more details of this approach are discussed, many approaches applied in healthcare systems based intelligent surveillance (sensors, audio, video, communication networks) as following sections will be reviewed. For this intelligent video surveillance system, usually the performance is the main objective but today the power consumption is also a critical parameter and more especially for autonomous object. The challenge is to build some tools or some ways to evaluate the performance, the recognition rate, and the power consumption at early step of the design.

These ways help in reducing the time-to-market for a quality product. In this thesis, a new Fall Detection algorithm is proposed and a methodology is defined to design low cost architectures.

1.1 The healthcare systems

With the rapid development of science and technology, surveillance systems are developing quickly. Nowadays, these systems not only generally monitor but also enable to analyse and process the captured data to activate an alarm when there are abnormal actions. This feature is quite suitable for a healthcare system.

1.1.1 The healthcare system based on sensors

1.1.1.1 Sensors worn on human body

A sensor is an electronic device that easily detects events or changes from the external environment, and then the received signals are transformed into an electrical or optical signal to control other devices. There are many sensors such as micro switches, spirit levels, accelerometers, and gyroscopes that are embedded in garments, or walking sticks [5]. The sensor systems can be stuck at the area where needs to collect the surroundings and then transmit to the central processor and finally process it for a specific purpose. Sensor and its essential features have been implemented in industrial and in medical equipment in particular. By capturing signals from sensors, doctors or healthcare scientists have a chance to easily monitor or even remotely diagnosis while patients are in hospital, clinic or even at home. The model of a healthcare system based on sensors is depicted in Figure 1.1.

A sensor mounted on body generally divides into three main factors: wireless, wire, and integrated in patients' body [6]. There are specific sensors for each part of the body and for each surveillance. Wearable embedded systems use sensors that can detect changes in postures, activities or motions of the person wearing devices. A wearable motion detection device using tri-axial accelerometer, which can detect and predict events based on tri-axial acceleration of human upper trunk, was designed and realized, [7]. With this method, the elderly who suffer from chronic diseases can be not only monitored effectively, but the early symptoms of the disease are also found. Wearable devices are simple and cheap; however, they might disturb the user's normal life. In addition, the issue of surveillance is that the elderly often forget to wear this equipment, and it depends on the ability and willingness of the elderly.

Figure 1.1-A model of the healthcare system based on sensors stuck on human body [8].

1.1.1.2 Ambient sensors

Due to the adverse effects of the wearable devices, some ambience systems which are not stuck directly on the human body are researched and applied recently. Welfare Techno House in Japan is an excellent example for the ambience systems [9].

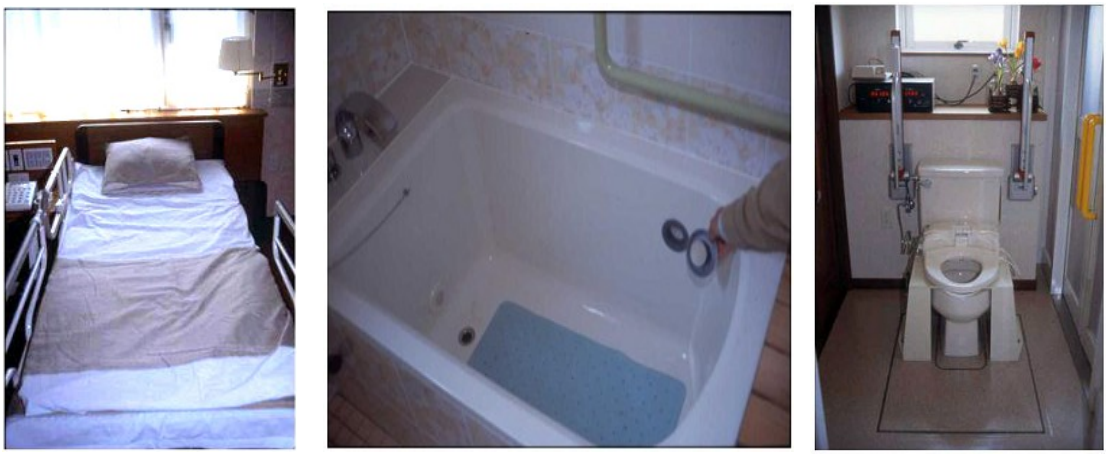


Figure 1.2-Welfare Techno House system [9]

In this system, automated electrocardiogram (ECG) measurements can be taken while the subjects are in bed, in the bathtub, and on the toilet, without their awareness and without using body surface electrodes. The sensors are installed in furniture and/or sanitary goods and the subject needs to attach the sensor. The heart rate and body weight can be obtained without any special measurement and the subject can receive daily physiological parameters without any awareness and discomfort. It is useful for understanding personal health status and daily activity information without the use of

invasive measurements. Some monitoring devices in the Welfare Techno House are shown in Figure 1.2.

1.1.2 The healthcare system based on audio

The healthcare system based on sound includes circular microphone array located in a room or relatively narrow and quiet space. When a sound is detected, the sound system will automatically amplify the signal, then identify and classify whether it is sound caused by the monitoring action or not [6]. However, this is the main limitation of this kind of the sound system, because it is usually placed in daily life environment, it is easily disturbed by surroundings noise, like falling objects or crying, etc. Furthermore, the sound also depends on each patient, type of actions or the position. Therefore, this system is not reliable enough to apply in the real life. A sound falling detection system consisting of the circular microphone array is depicted in Figure 1.3 [6].

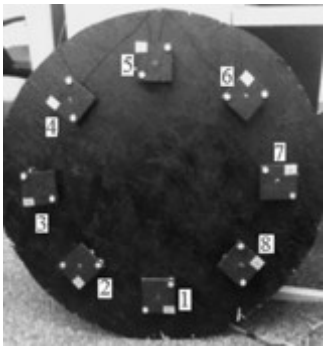


Figure 1.3-A circular microphone array used to automatically detects falling actions[6]

1.1.3 The healthcare system based on communication network

The care of patients now is involved in many different individuals, namely doctors, nurses, patients and their families. All of them need to share patient information and discuss their management. As a result, communication technologies are becoming a significant role in supporting health services. However, it is generally accepted that there is still a gap in applying communication technologies in healthcare systems, especially in rural areas where voice-mail or electronic mail is still not available.

A communication system involves people, the messages they wish to convey, the technologies that mediate conversations, and the organisational structures that define and constrain the conversations that are allowed to occur [10]. A possible communication pathway for a laboratory test, ordered by a general practitioner is shown in Figure 1.4.

There are significant organisational and communication challenges facing those delivering healthcare in the community. The model of shared care often adopted means that many different healthcare professionals may be involved in the management of an

individual patient. Even apparently simple activities such as ordering a laboratory test in general practice, and receiving the report, can involve many individuals, and many opportunities for inefficiency and errors.

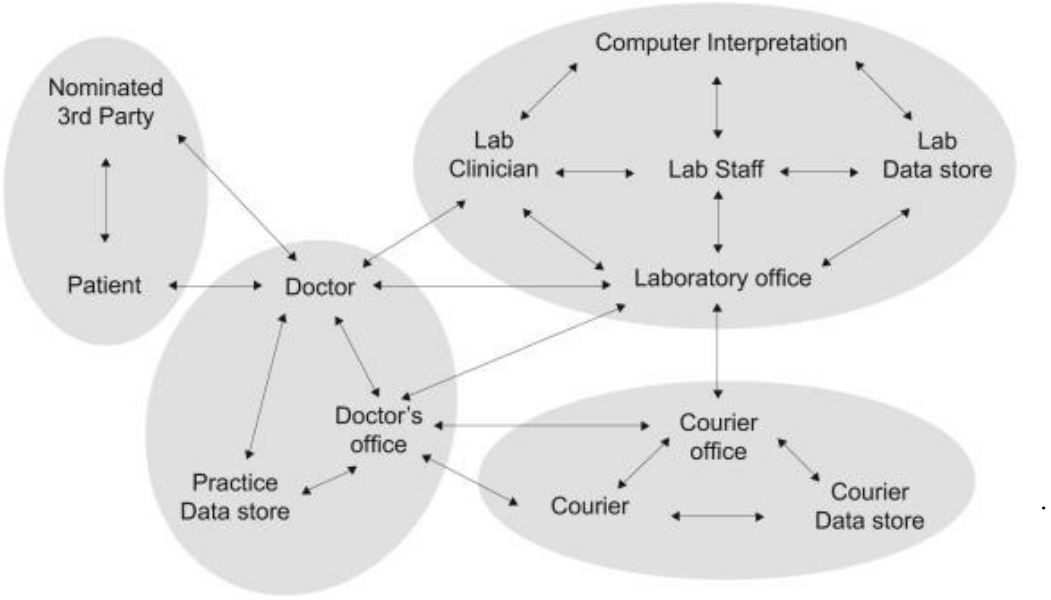


Figure 1.4-A possible communication pathways for a laboratory test [10]

1.1.4 The healthcare system based on intelligent video surveillance

Ambience systems use sensors installed in the living environments that certainly does not disturb the users. However, these kinds of sensor-based ambient systems normally result in high false alarm rate. Several studies have been proposed recently to use vision-based systems. Visual surveillance systems have been installed at many places in our lives, for instance offices, factories, schools or buildings. They recorded visual as well as any actions in the camera's area. At present, a variety of cameras are adopted to obtain the real-time situation for the elderly at home, and elderly abnormality is judged according to the above situations.

According to abnormality types and credibility, the systems analyse, process, and evaluate patients' situation to make some correction measurements in advanced in order to notify the guardian or those who concern. In today's modern life, intelligent video surveillance for elderly people living alone is an important application in the field of intelligent video surveillance. To reduce the workload of the remote monitoring staffs, the scene images are pre-processed by using techniques like image processing, or data mining, etc. After that, the captured scene images are transmitted by the internet or other communication methods to the distant guardian, and then these signals are carefully examined. By using intelligent video surveillance, people can effectively monitor their family as well as their business and issue warnings via computers or gadgets as soon as possible with many benefits. If there is any unusual events, user can take the warning to

others who are nearer to that area to prevent a bad outcome. An example of using intelligent video surveillance in hospital is illustrated in Figure 1.5. In addition, Ming-Liang Wang et al. [11] has proposed a video surveillance system using an omni-directional charge-coupled device (CCD) camera which is adopted to provide a 360° view angle of the indoor scene in a single image. The authors combined different algorithms for robust human motion tracking such as motion history image (MHI), Continuously Adaptive Mean Shift (CamShift) and optical flow in order to increase the robustness of the surveillance and tracking system. And for the human activity recognition, they use a calibrated one-to-one correspondence between the ground locations and the omni-directional vision sensor (ODVS) images.

Figure 1.5-A person is monitoring the healthcare camera in a hospital [12]

1.2 Fall Detection Approaches.

1.2.1 Classification Fall Detection Approach

The biggest advantage of video surveillance is the ability of real time execution by using standard computing platforms and low cost cameras. The methods have the capability to deal with robustness, however, still leave a widen horizon for further research and development. There are some different types of fall detection as follows[13]:

1.2.1.1 Spatiotemporal

Shape modelling using spatiotemporal features provides crucial information of human activities, which is used to detect different events. Image analysis requires efficient and accurate shape modelling methods [14]. Homa Foroughi et al. [15] proposed a novel approach for human fall detection based on combination of integrated time motion images and eigenspace technique. Integrated Time Motion Image (ITMI) is a type of spatio-temporal

database that includes motion and time of motion occurrence. Based on these observations, they extracted some motion information from the video sequences. Although, this motion information can be used directly in motion classification, they used eigenspace techniques for feature. Finally a multilayer perceptron (MLP) Neural Network is used for precise classification of motions and determination of a fall event.

1.2.1.2 Inactivity/change of shape

This algorithm bases on shape change analysis as well as inactivity detection. Vinay Vishwakarma et al. presented an approach for human fall detection consists of two parts: object detection and the use of a fall model. The authors used an adaptive background subtraction method to detect a moving object and mark it with its minimum-bounding box. The fall model uses a set of extracted features to analyze, detect and confirm a fall. Then they implemented a two-state finite state machine (FSM) to continuously monitor people and their activities [16].

1.2.1.3 Posture

The use of posture information contributes towards accurate fall detection. Different body positions are used to calculate postures. Specific types of postures are identified and localised in image sequences. Generally, model dependent methods obtain postures relatively easy and are robust to occlusion to an extent after labelling the body parts. Rita Cucchiaraour et al. proposed a human behaviour classification by the posture of the monitored person and, consequently, detected corresponding events and alarmed situations, like a fall. There are two phases in this project: firstly, posture classification performed frame-by-frame. This classification exploits simple visual features. Secondly, the obtained posture is further validated exploiting the information extracted by a tracking module in order to take into account the reliability of the classification of the first phase. This is motivated by the concept of "posture state" defined in a state-transition graph that takes into account for the classification the reliability of the track and acquired knowledge of the people's average behaviour in changing their posture[17].

1.2.1.4 3D head position analysis

Head position analysis is based on head tracking that determines the occurrence of large movement within the video sequences. Different state models are used to track the head based on the magnitude of the movement. In 3D head motion analysis methods, the principle of faster vertical motion than horizontal motion during a fall is applied. The head is initially located and then the 3D head position is estimated using fiHead. The idea of using appropriate thresholds to distinguish a fall from other actions is applied by computing vertical and horizontal velocities of the head [18][19].

In this thesis, we propose algorithms for the Fall Detection System including Object Segmentation, Object Enhancement, Feature Extraction and Recognition modules. To understand the behaviour of human object inside the video, the five features are calculated by the different positions of object performed frame by frame. The postures of object which are classified in our DUT-HBU dataset: fall and non-fall (bending, sitting, lying, creeping, etc.) are used for evaluation of this system. Our system is evaluated in terms of the accuracy, recall and precision performance. We also compare the performances among various algorithms such as BackGround Subtraction/Hidden Markov Model (BGS-HMM), Gaussian Mixture Model/Hidden Markov Model (GMM-HMM), BackGround Subtraction/Neural Network (BGS-NN), and BackGround Subtraction/Template Matching (BGS-TM).

1.2.2 Efficient Architecture for Fall Detection on heterogeneous platform

Currently, the advanced co-design step can lead to many solutions, especially in video processing: there are many ways of partitioning a system, and of writing the software with the chosen hardware. It is now well-known that the software has a very considerable impact on the final power consumption of a system. To find the best solution is a complex task.

New System on Chips (SOCs) that combine processor cores and field-programmable gate array (FPGA) architecture will help to build complex and performing systems. Some examples of video processing on heterogeneous architecture are shown as follows:

Eduardo Gudis, Pullan Lu et al. [20] have described an architecture framework using heterogeneous hardware accelerators for embedded vision applications. They presented a framework using an extensive library of pipelined real time vision hardware accelerators and service-based software architecture. Their framework allows the service-based software to take advantages of the hardware acceleration blocks available and perform the remainder of the processing in software. Three applications - Video stabilization (pre-processing), Moving Target Indication, and Contrast Normalization - were implemented on two Xilinx Zynq platforms: (1) the Xilinx ZC702 evaluation board with 7020-1part, and (2) a custom board with 7045-1 board.

Another video application applied OpenCV which is implemented on Zynq platform is illustrated in [21]. Road sign recognition is an autonomous application in driver assistance systems and road sign maintenance. This algorithm is presented using the Xilinx Zynq-7020 chip on a Zedboard to scan 1920×1080 images taken by an ON Semiconductor VITA-2000 sensor attached via the FPGA Mezzanine Card (FMC) slot. The Programmable Logic section of the Zynq is used to perform essential image pre-processing functions and colour based on filtering of the image. Software classifies the shapes in the filtered image, and they used OpenCV's template matching function to

identify the signs from a database of United Kingdom road signs. The system was designed in six weeks, and can process one frame in approximately 5 seconds. This is a promising start for a real-time System on Chip based approach to the problem of road sign recognition and also for using the Zynq platform for rapid deployment of these types of applications.

In addition, the jpeg decoder, image rectification, Semi Global Matching algorithm, and Stixel clustering are the applications which are accelerated on FPGA by using Xilinx Zynq 7045. In [22] Gilliland. S et al. evaluated the performance of an FPGA based embedded ARM processor system to implement signal processing for ultrasonic imaging and non-destructive testing applications. FPGA based on embedded processors possesses many advantages including a reduced overall development time, increased performance, and the ability to perform hardware-software (HW/SW) co-design. This study examined the execution performance of split spectrum processing, chirplet signal decomposition, Wigner-Ville distributions and short time Fourier transform implementations on two embedded processing platforms - a Xilinx Virtex-5 FPGA with embedded MicroBlaze processor and a Xilinx Zynq FPGA with embedded ARM processor. Overall, the Xilinx Zynq FPGA significantly outperforms the Virtex-5 based system in software applications.

Dobai and Sekanina [23] demonstrated evolutionary design of switching image filters on the platform. The investigated implementations included virtual reconfigurable circuits and the use of dynamic partial reconfiguration. The achieved results demonstrated the advantages and disadvantages of the Zynq platform. The observations intended to be useful for designers who would develop evolvable hardware on this new platform. They presented the time required to evaluate one filter candidate (individual), the time for a generation of filter candidates (4 individuals), the number of generation per second and the relative acceleration in comparison ARM processor (without the PL). First, the pure software-based approach was evaluated on the ARM processor of the available Zynq device. Second, they compared the processor of Zynq with a desktop processor (Intel i5). The code in language C was pre-ported to that processor. According to this experiment, the Intel i5 processor was 5 and 6 times faster than the ARM processor of the Zynq device. The third experiment was to determine the magnitude of the FPGA-based acceleration of the filter evolution. The implementation revealed that the operational frequency of the pure virtual reconfigurable circuits (VRC) and hybrid VRC-DPR (dynamic partial reconfiguration) approach was 203.6 MHz and 265.3 MHz, respectively. The hybrid approach was able to evaluate the candidate filters approximately by 30% faster. On the other hand, the VRC approach mutated the circuit in negligible time and the hybrid approach required more time. The hybrid approach changed the interconnections similarly to the pure VRC approach but the replacement of the processor elements by dynamic partial reconfiguration (PEs by DPR) takes longer.

Monson et al. developed their application in High-Level Synthesis (HLS) for FPGAs making it possible to “run” C code on FPGAs and thereby making modern programming environments available to FPGA developers. In their research, C code for a complex optical-flow algorithm was optimized for both a desktop PC and a FPGA-based system, the Xilinx Zynq-7000, which is a device containing both a programmable fabric and two ARM cores. They discussed how the code was optimized and restructured to execute effectively on the programmable fabric and the ARM cores. The resulting Zynq version of the C code was competitive with the desktop PC but only consumed 1/7 as much energy.

In the other co-design Digital Signal Processor (DSP)/FPGA, Giuseppe Baruffa et al [24] presented the architecture of a DSP/FPGA based hardware platform, which is conceived to leverage programmable logic processing power for high definition video processing. Their system was reconfigurable and scalable, since multiple boards may be parallelized to speed-up the most demanding tasks. The application frameworks, JPEG 2000 and H.264, both at high dimension (HD) and Super HD (SHD) resolutions have been simulated and performed on the embedded processing cores. The issues such as real-time, or near real-time encoding was viable, the modularity of the architecture allowed parallelization and performance scalability were proposed in this study. Cooperation of FPGA and DSP processing modules was required to fulfil the proposed objectives. Performance results showed that real-time encoding and decoding of HD and SHD video were possible by using a parallelized configuration.

In addition of co-design, Felix Büsching et al. [25] proposed an outdoor fall detection system. The system consisted of an Android smartphone and an INGA wireless sensor node. This node was equipped with an accelerometer, a gyroscope and a barometric pressure sensor. However, only the accelerometer for the fall detection was utilized. In the system, the smartphone was used as a counterpart. It was implemented for the three different applications to:

a) send a text message with a predefined text to a predefined phone number;

b) all raw data which is transmitted to the smart phone and processed a fall detection;

c) work as a standalone fall detection and alert system applying the same algorithms as the second application. Nevertheless, this application only utilized the acceleration sensor of the smartphone.

The research works described below show that this kind of video application needs heterogeneous architecture (Processor cores + FPGA) to meet a sufficient frame rate. So in this thesis, we consider this type of architecture (Zynq platform) and we firstly define the power and execution time models for different target circuit: processor core, FPGA with the aim to evaluate the performance (recognition rate and energy) of the fall detection system. Our models are determined based on the Functional Level Power

Analysis for the processor cores and related with hardware resources for FPGA. In order to find architectures and suitable configuration which allow an acceptable fall recognition rate, we propose a new methodology for exploring low cost architectures and by applying parallelism techniques such as intra-task and inter-task static scheduling based on hardware/software architecture for Fall Detection System.

1.3 Research questions

The problem landscape of implementing the video processing has many facets. Within the limited framework of this thesis, all aspects of this research domain cannot be addressed. We focus on the following questions:

1. *How can we avoid transmitting the image to other remote systems for privacy reason?*

When the Fall Detection System works, all activities of a person are recorded and processed. The video content will automatically be analysed and carried out by computer or hardware device. If a falling is occurred, the system will immediately send a warning of FALL or NON FALL to the remote monitoring center. It also provides the exact cause of human falling and all input video are processed inside closed system.

2. *How are the performances (accuracy, precision and recall) of this system?*

Fall detection is a challenge in the public healthcare system, especially for the elderly, and reliable surveillance is a necessity to mitigate the effects of falls. The technology and products related to fall detection have always been in high demand within the security and the health-care industries. An effective Fall Detection System is required to provide urgent support and to significantly reduce the medical care costs associated with falls. Therefore, we first should have a database with various kinds of human activities. In this system, the DUT-HBU dataset [26] is used and all video data are compressed in *.avi* format and captured by a single camera in a small room with the changeable conditions such as brightness, objects, direction of camera, etc. In this database, the fall direction is subdivided into three basic directions which are Direct fall, Cross fall, and Side fall. In terms of non-fall videos, usual activities which can be misrecognized with fall action such as lying, sitting, creeping, and bending are also classified into three mentioned directions. Moreover, to evaluate the efficient and accuracy of a system, we analyse the Precision (PR), Recall (RC) and Accuracy (Acc).

3. *How does a video system meet the real-time processing?*

Designing a real-time system requires a holistic approach that is considered many aspects such as algorithms, architectures, and implementation methods of applications in order to meet a specified deadline. The constraints on the considered video processing are that the system must be able to maintain an average processing rate higher than the required frame rate in order to get a sufficient accuracy for fall detection recognition rate. This requires a deterministic and bounded execution time. In our system, we apply parallelism techniques based on hardware/software architecture to improve the execution time and by this way the performance of our system.

4. *How is the "cost" of this system?*

Fall detection systems based on cameras have proven to offer a promising solution which is complementary to the wearable sensors. One advantage of visual-based fall detections is they can be installed in-door and not required to be worn by any users. The cameras can be wall- or ceiling-mounted, depending on the interests on orientation and field of view of the frames to be captured. Besides, the recorded video allows more efficient use of multiple events analysis and post verification. Moreover, cameras are increasingly becoming a strong candidate for the choice of fall detection sensor due to the rapid drop of camera costs. So far, different visual-based fall detection techniques have been identified. Nevertheless, the development of such systems has been implemented as software-based solutions on computers.

In addition, we want to design a standalone Fall Detection System corresponding with the less energy consumption and the higher performance. Therefore, in our system, we try to extract the power consumption and execution time models for processor cores and FPGA to explore the low cost architectures which offer sufficient frame rate, low power/energy consumption and an adequate accuracy rate based on Design Space Exploration methodology. Some parallelism techniques are also applied to improve the execution time and by the way the performance of Fall Detection System.

1.4 Thesis contributions

Most of video surveillance for fall detection researches includes two or three modules to detect the behaviour of human object. They recognise the movement of object by analysis of shape of object modeling in several frames or the different centroid of object or by comparing with the available template, etc. Therefore, it is also necessary to explore the algorithms which provide a high performance of recognition ability for a Fall Detection System. In addition, the algorithms have usually been compared together by simulating on Matlab and evaluated the reliable performance by a database. Besides, several video applications such as object segmentation, video compression format

(H.264 or MPEG4), filter (sobel, canny, etc.), and recognition (using Neural Network, Template Matching, etc.) have been studied on power/energy consumption characterization and modeling at different levels for embedded video system. Many methodologies have handled the low and high level models related to the processor, memory or FPGAs. In this thesis, the first main contribution is to propose efficient algorithms for the Fall Detection System which consists not only to signalize a fall but also the type of this fall in order to determine the urgency of the situation. Comparing to others fall detection, our system consists of 4 modules: Object Segmentation, Filter, Feature Extraction and Recognition with an automatic alarm whenever FALL is occurred. Then, in order to find out fall detection architectures which meet users and application constraints, power consumption and execution time models are defined taking into account architecture and application parameters. Thus, low cost architectures for our system are explored by using the parallelism techniques to find out the heterogeneous architectures which couple the energy consumption and fall detection accuracy rate. To achieve this goal, we intend to pursue the methodology scheduled as follows:

- (1) We determine recent algorithms which are applied on each module of Fall Detection System and our system also give an urgent alarm for detecting different kinds of fall. We make then various simulations to compare the recognition rate performances among algorithms such as BackGround Subtraction/Hidden Markov Model (BGS-HMM), Gaussian Mixture Model/Hidden Markov Model (GMM-HMM), BackGround Subtraction/Neural Network (BGS-NN) and BackGround Subtraction/Template Matching (BGS-TM). In this thesis, in order to evaluate the efficiency of the Fall Detection System, a DUT-HBU database which is classified with different actions: fall, non-fall (sitting, lying, creeping, etc.) in three camera directions (face, sides and cross) is created and used for simulation, evaluation and implementation purposes. The testing databases are then regrouped into three different scenarios types: well-matched (WM), medium-mismatched (MM) and highly-mismatched (HM). We select the BGS/TM algorithm, which is sufficient for recognition performance by using the well-matched test in this database, and is in order to implementation purpose. Some factors which affect to the quality of recognition in this system such as environment brightness, occlusion of object, many movement objects appearing in a frame at the same time are also analysed.
- (2) In order to find out a suitable architecture for the proposed algorithm, power consumption and execution time models are proposed for processor cores based on Functional Level Power Analysis (FLPA) and FPGA related with the hardware resources and then for heterogeneous architecture. Our video application was implemented on processor cores (ARM Cortex A9 processor) of ZYNQ Platform different configurations. TI USB Interface Adapter PMBus

associated with TI Fusion Digital Power Designer GUI are used to measure the power consumption on processor cores. Some tasks of the application are also synthesized, and characterized on FPGA by using Vivado 2012.4 tool. The curve fitting of regression law is used to get the mathematic models for power consumption and execution time which depend on algorithm and architecture parameters, such as operating frequencies, number of cores, image resolution. The error rate of these models is then evaluated and used for exploring the architecture for the Fall Detection System.

(3) By extending the power consumption and execution time models, we propose a Design Space Exploration methodology to define low cost architecture for Fall Detection System. In this methodology, in order to explore heterogeneous architectures for our system, two parallelism techniques *intra-task and inter-task* static scheduling are applied. The low cost architectures are selected with the compromising of energy consumption and accuracy rate performance of the Fall Detection System.

1.5 Publications

1.5.1 International journal publication

[1] Hong. Nguyen.T.K, Cecile. Belleudy and Tuan.V.Pham, "Performance and Evaluation Sobel Edge Detection on Various Methodologies", 2014 International Conference on Advances in Electronics Engineering, 19-20 February, 2014.

This paper has been extended and published in International Journal of Electronics and Electrical Engineering Vol. 2, No. 1, March, 2014.

[2] Hong Thi Khanh Nguyen, Cecile Belleudy and Pham Van Tuan "Fall Detection Application on an ARM and FPGA Heterogeneous Computing Platform" International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol.3, Issue 8, August, 2014.

1.5.2 International conference's publication

[3] Hong. Nguyen.T.K, Cecile. Belleudy and Tuan.V.Pham, "Power Evaluation of Sobel Filter on Xilinx Platform", IEEE FTFC, 4-6 May, 2014, Monaco.

[4] Hong Thi Khanh Nguyen, Hassoon Fahama, Cecile. Belleudy and Tuan.V.Pham "Low Power Architecture Exploration for Standalone Fall Detection System Based on Computer Vision", IEEE-EMS2014 European Modelling Symposium 2014, 21st – 23rd October, 2014, Pisa, Italy.

1.5.3 Other publications

[5] Hong. Nguyen. T. K, Cecile Belleudy and Tuan.V.Pham, “FPGA-based Object Segmentation of Fall detection”, RUNSUD 2013, 23-24 April, 2013.

[6] Thi Khanh Hong Nguyen, Cecile. Belleudy and Tuan.V.Pham, “Low Power Exploration Design Flow for Fall Detection System”, COLLOQUE NATIONAL of GDR SoC-SiP, 11-13 juin 2014 in Paris.

[7] Hong Thi Khanh Nguyen, Cecile. Belleudy and Tuan.V.Pham,“Low cost architecture for Fall Detection System”, e-PSP 2014, 27th November , 2014, Biot, France.

1.6 Thesis Organization

This thesis is divided into 5 chapters. After the introduction in Chapter 1, we present the Fall Detection approaches and analyse the comparison of the performance of these approaches by their simulation on Matlab in Chapter 2. In Chapter 3 the extraction of the execution time and power models based on Function Level Power Analysis for processor cores and based on hardware resources for FPGA is discussed. And then the Design Space Exploration methodology including two parallelism techniques (intra-task and inter-task static scheduling) which is applied to explore low cost architectures for this system based on heterogeneous architectures with different configurations is described in Chapter 4. Finally, Chapter 5 contains the conclusions and proposals for the future works of this work.

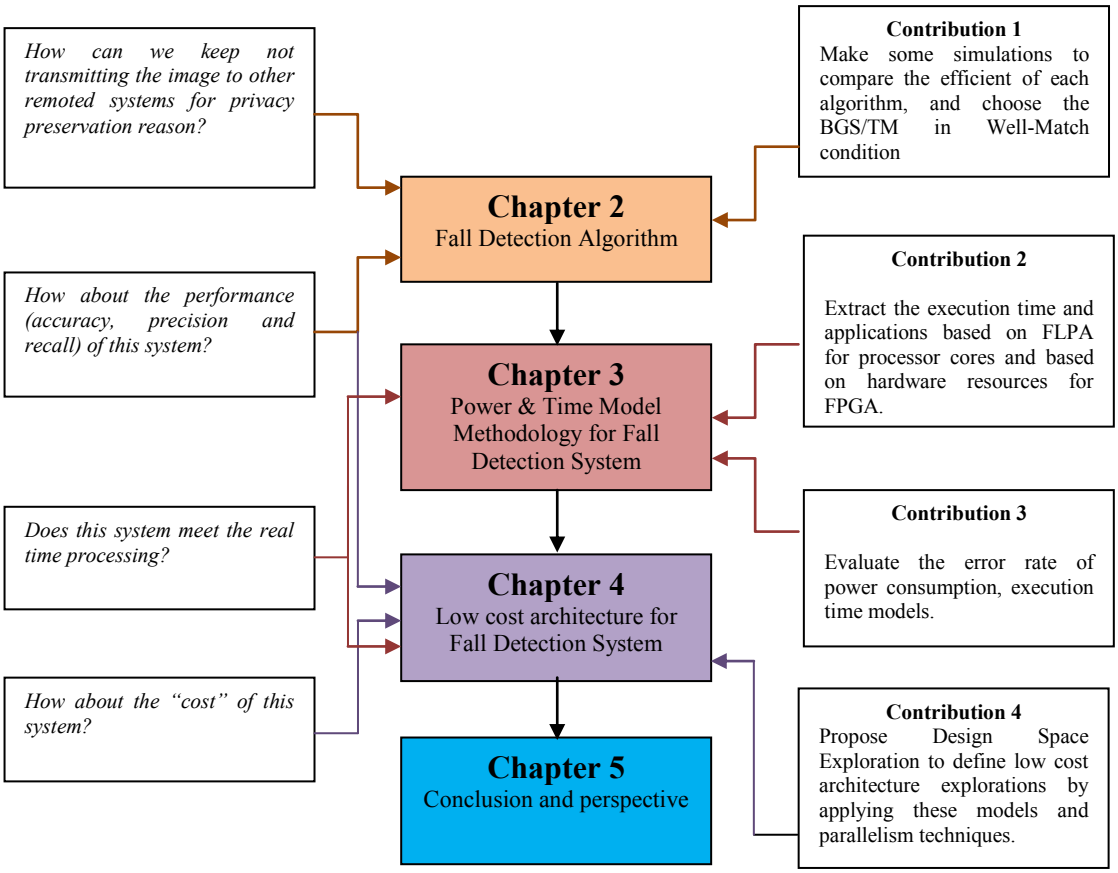


Figure 1.6-An outline of the different chapters, research questions and contributions in this thesis

Chapter 2. Fall Detection Algorithm

Nowadays, fall detection is a serious challenge in the public health care domain, especially for the elderly living alone. There are some health care systems based on sensors, audio, communication network and video processing. In our work, we develop a Fall Detection System based on video processing. The Fall Detection System is developed in two sides: algorithms and architectures. In this Chapter, we study and make the simulations of algorithms used in the Fall Detection System.

In Section 2.1 of this chapter, the overview of Fall Detection System describes the definition of a falling event and the Fall Detection approaches which are currently used in recognition the postures or action of human object. The most important part is in Section 2.2 where we propose studied algorithms for the Fall Detection System including four modules: Object Segmentation, Object Enhancement, Feature Extraction and Recognition. In order to understand the behaviour of human object in our system, some recognition models are studied and illustrated more detail in Section 2.3. Moreover, the DUT-HBU database which is classified with different actions: fall, non-fall (sitting, lying, creeping, etc.) in different directions of camera (face, sides and cross) is used for evaluation our system. The system is evaluated in terms of the accuracy, recall and precision performance. Finally, the results on Matlab's simulation in comparing the performances among these algorithms such as BackGround Subtraction/Hidden Markov Model (BGS-HMM), Gaussian Mixture Model/Hidden Markov Model (GMM-HMM), BackGround Subtraction/Neural Network (BGS-NN), and BackGround Subtraction/Template Matching (BGS-TM) are shown in Section 2.4. At the end of this Chapter, we make the general discussion about parameters which impact on the quality of recognition ability in the Fall Detection System.

2.1 Overview of Fall Detection algorithm

2.1.1 Definition of falling event

According to the World Health Organization (WHO), there is no any specific definition of falls. In our work, falling accident is defined as "the loss of balance with involuntary causes the body suddenly fell to the ground". Accidental falls are always dangerous for children, adults, especially the elderly and cause serious consequences.

Sudden fall (or falls by accident) is often affected by the impact and the external factors and by many different reasons, such as: slip and fall, walk and fall, fall from

heights, etc. Unexpected fall mainly for the elderly (or falls not by accident) may occur by many reasons. When considering the characteristics of the falls, the following factors are generally considered:

- Direction falls: front, sides, behind.
- Position of the body before and after the fall: lying, sitting, standing, kneeling, and leaning.
- Speed of fall: rapidly or slowly falling down with wobbled knees.
- Active or react before falling: legs or arms raise high, head thrown back, head peers ahead.

Daily actions are mistaken as falling like running, sit down, laying and etc.

These above mentioned factors are used to classify and build scenarios, various situations of fall and then used to train and build the fall detection, evaluation and development in our system.

There are four stages in the process of falls include [10]: pre-fall, fall phase (main phase), after falling phase and recovery phase.

- In the pre-fall phase, people are doing the normal activities in daily (probably occur with sudden movements like sitting or lying down quickly). Fall detection system distinguishes this stage with the following phase.
- Fall phase includes sudden movement of the body to the ground, and ends with a crash to the ground. The period of this phase is usually very short, 300-500 ms [11], it is determined from 400-800 ms.
- In the after falling phase, the body is not a normal movement, still lying on the ground.
- Recovery phase: the falling elderly can stand up by themselves or by the other helps.

2.1.2 Fall Detection algorithms

Nowadays, fall detection is a major challenge in the public health care domain, especially for the elderly living alone. In 2013, the Center for Disease Control and Prevention[1] (CDC) reported that rate of fall injuries for adults from 85 years old and older was almost ten times than that for adults between 65 and 74 in the United State[2]. This statistic also shows that falls are the primary reason of injury related to death for seniors aged 65 and older. Along with the population explosion of the elderly in the world, the demand for surveillance systems, especially for fall detection, has considerably increased within the healthcare industry. Developing intelligent surveillance systems take an important role, especially vision-based systems, which can

[1] http://www.cdc.gov/

[2] http://www.cdc.gov/injury/wisqars/pdf/leading_cause_of_nonfatal_injury_2012-a.pdf

automatically monitor and detect falls. It has been proved that the medical consequences of a fall are highly contingent upon the response and rescue time. Thus, a highly-accurate automatic Fall Detection System is an important part of the living environment for the elderly to expedite and improve the medical care. In order to provide immediate medical attention, and contribute to solve the lack of manpower in the health sector, many Fall Detection Systems have been studied recently. Several studies have been proposed to use vision-based systems. By using a single camera, Rougier et al. [27] propose an algorithm based on a combination of motion history and human shape variation which provides promising results on video sequence of daily activities and simulated falls. In order to reduce the occlusion areas by cause of the irrelevant position of camera or movement of object, several research works have developed to use multiple cameras [28]. Rougier and his research group [29] present a new method to detect falls by analysing human shape deformation during video sequences captured from four cameras. The shape matching technique is used to track the person's silhouette along the video sequence. Auvinet et al. [30] reconstruct 3-D volume of a person from eight cameras using calibration information. If a big portion of the body volume is near the ground for a period of time they recognise it as a fall.

Figure 2.1-A typical fall while walking

Figure 2.2-Some typical types of non-fall action

In most of the developed camera-based systems, the algorithms mainly carried out in four phases as illustrated in Figure 2.3. Among them, Object Segmentation plays an important role in ensuring system robustness. From the practical aspects, the system should be able of dealing with lighting changes, movement through clutter areas, unexpected objects overlapping in the visual field and objects being introduced or

removed from the scene. First, moving object is segmented from each frame of the input video in the first module of the following diagram. An Object Enhancement is also used to improve the impact of noise in order to get the better object for the next modules. Afterwards, an ellipse model is built from the segmented object and used for extracting five features. These features are applied for training by recognition models. From the trained features, probability of each observed input data is calculated. By looking at the output results from sequential frames stored in a buffer, a final decision is made. Figure 2.4 shows the process of fall detection by video.

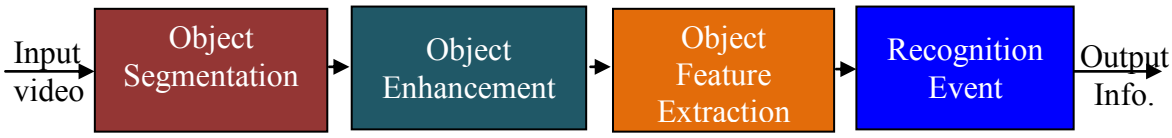


Figure 2.3-Block diagram of Fall Detection System.

Figure 2.4-Detecting the fall by video analysis

2.1.2.1 Object Segmentation

Video signal including sequence of frames captured from the camera is put into Object Segmentation module to extract objects from the background. The output of this module is a stain, shape of the moving object. This is the first module in the Fall Detection System, therefore, its accuracy makes a considerable impact to next processing modules.

Human objects are extracted from the background of an image by using the Object Segmentation algorithms. The object is segmented based on the difference between two consecutive frames in the time domain [31] or the type of removing background from an image [32][33]. In recent years, Background Subtraction method has become more popular and many thanks go to the development of optimization techniques in estimating dynamic background [34]. Stauffer and Grimson [35] have done

the Gaussian mixture model which is very flexible and suitable for objects moving slower than the background.

Because monitored objects continuously move up, so objects are tracked to set the connection of objects via consecutive frames in the sequence images. To achieve accurate detection, tracking objects have the ability to handle the causes which are happened by segmented objects are not perfect and handle the occlusion by camera angles. Cheng and Hwang present not only the partial adaptive sample called Kalman algorithm [36] but also the combination of statistical data in order to detect multi-objects [34] with high accuracy and reliability. Comaniciu et al. propose tracking method in which the moving average values changed based on the centre of the image [37]. In this method, human object is performed as the convolution of the object's properties with the central image are calculated the cost vie spatial. One of the biggest advantages of this method is less computation complexity compared to other methods.

2.1.2.2 Object Enhancement

The Object Segmentation methods are sensitive to the changes of background. But in reality, the background pattern is always affected by external factors such as intensity light, wind shake or reasons due to the colour of some object's part coincides with the background. All these factors make the object (foreground) which is extracted from the background not only is necessary moving object but also includes noises or the inner object are not filled. Therefore, to ensure the provided object is the better quality for the next modules of the system, the object needs to be purified by removing the silhouette, the part of noise is not the object and blobbing the object model. One of the methods used to filter object after object segmentation module is Mathematical Morphology (MM) [38]. The other methods are Sobel Filter, Canny, Prewitt, etc.[39][40][41].

2.1.2.3 Object Feature Extraction

In order to understand the behaviour of extracted object, its features are calculated and analysed. After having extracted object from the previous modules, the movement curves which are bounded the object based on 2D/3D model are used to calculate the features of object [42][43]. To estimate the posture of object based on video analysis faces to some obstacles such as the depth of image, lots of object postures are continuously moving from video and the free gradient of joints are rather high. There are a number of methods that can solve this problem [44]. We can divide them into two categories either based on the model (model-based) [45][46] or not dependent on model (model-free) [47][48].

The above methods reflect the change of human posture quite accurate. However, the number of application research to automatically track the 3D human posture is very limited. In [49], R. Holt et al. suggest a method for automatic estimation

of 3D human pose from the video signal in real time effectively. The method is of analysis-via-synthesis splitting human body into several different sections. The human body is segmented from the background, extracted the 2D features, tracked the 2D features and reconstructed 3D posture. This method performs the minimization of complexity computation and restores lost events or obscured part as shown in [50].

2.1.2.4 Recognition events

After obtaining the results of modeling the human body 2D/3D and motion curves around each human body, human actions are recognised and understood their behaviour from video sequences. Recently, there are some methods used in this module such as Template Matching (TM) or Threshold-based, Neural Network (NN), Hidden Markov Model (HMM), Dynamic Bayesian Network (DBN), etc. [51][52][53][54].

For Template Matching, the extracted features are compared directly to the stored features (templates). These stored templates are extracted from the process of the past experiences and learning [55]. The application of machine learning techniques in identification of human behaviour is still facing many obstacles and very complicated. This is due to the extreme diversity of the same action but made by different people or even an action, but with a different angle and different duration. Furthermore, the features are reliably coped with the changing spatial-temporal scales related to human activity. These features must be separated by encapsulating in the unique properties of the same action, but made by different people. After describing the feature of an action, the next important issue is how to develop a method for identifying properties in the space available. Recently, Artificial Neural Networks (ANN), Hidden Markov model (HMM) and Dynamic Bayesian Network (DBN) are the most common methods to modelise and classify human action sequences. However, the methods are ongoing to verify which the most effective one is.

2.2 Proposed Fall Detection Algorithm

In our work, we propose an algorithm for elderly fall detection including four modules: Object Segmentation, Object Enhancement, Feature Extraction and Recognition. Human activities are captured in a video that is further analysed using image processing and an embedded system to detect fall and generate an alarm of FALL or NON FALL. It also provides the exact causes of human fall. To get more efficient in recognition and classification of falls, the database with different kinds of falls is built, called HBU-database. Different scenarios are considered when identifying different kinds of falls and non-fall actions: falls from walking or standing, falls from sleeping or lying in the bed and falls from sitting on a chair, etc. or non-fall events like walking, lying, creeping, sitting and so on.

In following subsection, these algorithms in each module are described.

2.2.1 Object Segmentation

2.2.1.1 Background subtraction (BGS)

In our work, two main methods are used to deal with Object Segmentation. Object Segmentation module is responsible for detecting and distinguishing between moving objects and the rest of the frame which is also called background [56]. In this work, a Background Subtraction method is applied to distinguish background and moving objects. A pixel is marked as foreground if

$$\left|\mathrm{I}_i - \mathrm{B}_i\right| > \tau \tag{2.1}$$

Where, τ is a "predefined" threshold. And the updated background is estimated as:

$$\mathrm{B}_{i+1} = \alpha \mathrm{I}_i + (1-\alpha)\mathrm{B}_i \tag{2.2}$$

Where α is kept small to prevent artificial "tails" forming behind moving objects. Here, we use average of 3 consecutive frames instead of using the current frame I. And α is chosen 0.05 as in [57].

2.2.1.2 Gaussian mixture model (GMM)

In cases of movement through clutter areas, objects overlapping in the visual field, shadows, lighting changes, and effects of moving elements of the scene, Background Subtraction methods achieve less efficiency of output. An adaptive Gaussian mixture model [58] is one of the solution to handle variations in lighting, moving clutter scene, multiple moving objects and other arbitrary changes to the observed scene. In this work, the values of a particular pixel over time are considered as a "pixel process". The "pixel process" is a time series of pixel values that was shown in equation 2.3 as follows:

$$\left\{\mathrm{X}_1,...,\mathrm{X}_t\right\} = \left\{\mathrm{I}(x_0, y_0, k) : 1 \le k \le t\right\} \tag{2.3}$$

If each pixel results from a specific surface under particular lighting, a single Gaussian is sufficient to model the pixel value while accounting for acquisition noise. If only lighting changes over time, a single adaptive Gaussian per pixel will be sufficient. In practice, multiple surfaces often appear in the view of a particular pixel and the changeable lighting conditions. Thus, multiple adaptive Gaussians are necessary [35]. The recent history of each pixel, $\{\mathrm{X}_{1...}\ \mathrm{X}_t\}$, is modeled by a mixture of *K* Gaussian distributions. The probability of observing the current pixel value can be calculated by:

$$p(\mathrm{X}_t) = \sum \omega_{i,t} \eta(\mathrm{X}_t, \mu_{i,t}, \sum_{i,t}) \tag{2.4}$$

Where $\omega_{i,t}$ is the weight parameter of the i^{th} Gaussian distribution; η is the probability density function of i^{th} Gaussian distribution. At each frame, the value of pixel at $(x_{0,} y_{0)}$ is X_t. A Gaussian distribution is a match if X_t is within 2.5 times of its standard deviations. Each pixel can have at most one matching Gaussian from its mixture. From the labelling of each Gaussian we have two following cases:

a) **In the first case**: if pixels have a matching distribution, Gaussian is marked as matched that will be updated by following equation:

$$\omega_{i,t} = (1-\alpha)\omega_{k,t-1} + \alpha\left(M_{k,t}\right) \tag{2.5}$$

$$\mu_t = (1-\rho)\theta_{t-1} + \rho X_t \tag{2.6}$$

$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t - \mu_t)^T(X_t - \mu_t) \tag{2.7}$$

Where α is the learning rate parameter and $\rho = \alpha\eta(X_t \mid \mu_k, \sigma_k)$. For the other unmatched Gaussian, only the weight of distribution is updated with equation 2.4.

b) **In the second case**: none Gaussian is marked as matched. X_t is assigned as a foreground pixel and the least probable component is replaced by a distribution with the current value as its mean, an initially high variance, and a low weight parameter.

Distributions having a high weight and low variance are precisely the distributions that represented the background model. In order to find them, the K distributions are ordered based on the fitness value $\omega_{k,t} / \sigma_{k,t}$ and the first S distributions are used as a model of the background of the scene where S is estimated as

$$S = \arg\min_b \left(\sum_{k=1}^{b} \omega_{k,t} > T\right) \tag{2.8}$$

Where T is the minimum fraction of the background model.

So as to remove noise and improve image's quality from object's binary image, mathematical morphology methods such as dilation and erosion, opening and closing to improving quality of segmented objects are applied. Figure 2.5 shows the result of object segmentation.

As shown in Figure 2.5, the result derived from the GMM method is significantly better than the BGS method. This is due to adaptability of the Gaussian distributions and an automatic pixel-wise threshold (that is presented by two significant parameters - α, the learning rate and T, the minimum fraction of the background model). Meanwhile, the BGS method is a simple Object Segmentation method based on a predefine threshold.

2.2.2 Object Enhancement

There are some supporting methods to improve the quality of image from the object binary image such as Morphology Mathematic (MM), Edge Detection Filter (Sobel, Canny and Prewit Filter).

In the following subsection, we describe the two techniques used to improve the quality of segmented object such as Mathematical Morphology and Sobel Edge Detection.

2.2.2.1. Mathematical Morphology

The Object Enhancement module as Filter removes noise and improves image's quality from object's binary image [56]. Mathematical Morphology (MM) is a mathematical theory which is used to process, analyse the images and improve quality of segmented objects [59]. It provides an alternative algorithm to image processing based on shape concept stemmed from set theory [60], not on traditional mathematical modeling and analysis. In the MM's theory, images are treated as sets, and morphological transformations which is derived from Minkowski addition and subtraction are defined to extract features in images [61]. MM methods involving dilation and erosion, opening and closing are used in our work.

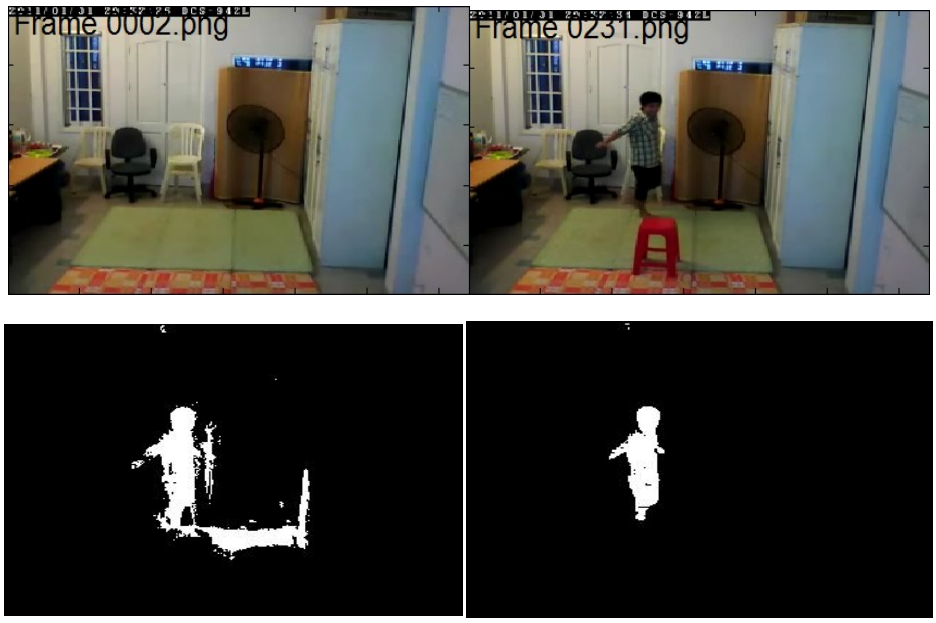


Figure 2.5 (a) Estimated background; (b) Frame input;
(c) Background Subtraction method; (d) Adaptive GMM method

The morphologic operations work with two images: the original data is processed and a structuring element. Each structuring element has a shape which can be thought as a parameter to the operation. Two most common structuring elements are

connected to sets 4 and 8 as shown in Figure 2.6. To apply mathematical morphology, digital images must be binary images in which pixels represent the object image encoded by the pixel "1" and background encoded by the pixel "0". Most fundamental morphological operations are morphological *dilation* and morphological *erosion*. Besides, we also have two compound operations named as *opening* and *closing* are defined.

When the case of binary image is considered, *A* is the set of points representing the binary one pixel of the original binary image and *B* is the set of points representing binary one pixels of structuring element.

Dilation is an operation that enlarges the objects presented in a binary image. Results of *dilation* are influenced both by the size and shape of a structuring element. *Dilation* of a binary image *A* by binary structuring element *B* is defined as:

$$A \oplus B = \left\{x | (\widehat{B})_x \cap A \neq \emptyset\right\} \tag{2.9}$$

Erosion is the opposite of *dilation*. This operation shrinks the objects in a binary image. That is erosion operation causes object to lose its size. *Erosion* of a binary image *A* by binary structuring element *B* is defined as:

$$A \ominus B = \{x | (B)_x \subseteq A\} \tag{2.10}$$

Figure 2.6-Structuring elements in Mathematical morphology

Opening is simply the erosion of *A* by a structuring element *B* followed by a *dilation* of the output by the same structuring element. *Opening* of a binary image *A* by a binary structuring element *B* is defined as:

$$A \circ B = (A \ominus B) \oplus B \tag{2.11}$$

Closing of an image is also a combinational operation of *erosion* and *dilation*. It differs from the *opening* operation in the sense of order of occurrence of *erosion* and *dilation* operation. *Closing* of an image *A* by a structuring element *B* is defined as:

$$A \bullet B = (A \oplus B) \ominus B \tag{2.12}$$

Examples of four basic morphological operations namely *dilation*, *erosion*, *opening* and *closing* are shown in Figure 2.7. These morphological operations are

changed relying on the structuring element. The objects are detected by the certain structuring element to determine whether the structuring element fit or not with the edge of the image. In each issue in this figure, the structuring elements are compared with each pixel of the objects and then use one of four methods above to change the edge to filter as the expected targets, like removing noise or cutting the unexpected edges.

Erosion is a transformation of shrinking, which decreases the grey-scale value of the image, while *dilation* is a transformation of expanding, which increases the grey-scale value of the image. But both of them are sensitive to the image edge whose grey-scale value changes obviously. *Erosion* filters the inner image while *dilation* filters the outer image. *Opening* is *erosion* followed by *dilation* and *closing* is *dilation* followed by *erosion*. *Opening* generally smooths the contour of an image, and breaks narrow gaps. Opposed to *opening*, *closing* tends to fuse narrow breaks, eliminates small holes, and fills gaps in the contours. Therefore, morphological operation is used to detect image edge, and at the same time, denoise the image.

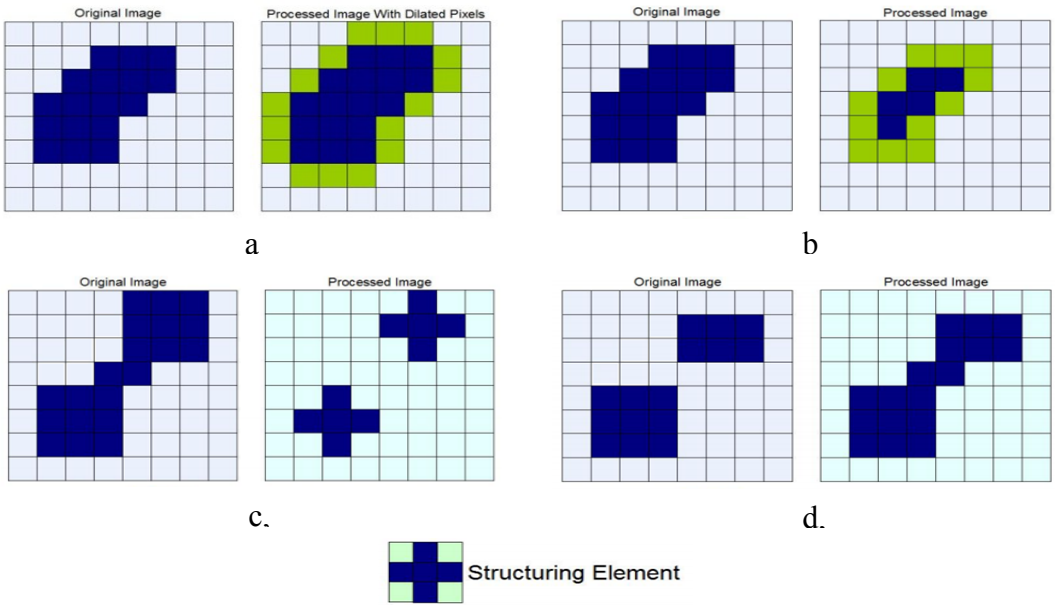


Figure 2.7-Example of Mathematical morphology operations[62].

a, Dilation (green blocks are the ones added to the original).

b, Erosion (the green blocks are the ones that will disappear from the image).

c, Opening. *d, Closing.*

2.2.2.2 Sobel Edge Detection

Sobel edge detection algorithm is the most commonly used techniques in image processing for edge detection [63]. Two types of Sobel operators, which are horizontal and vertical, are used. The operator calculates the gradient of the image intensity at each

point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The Sobel kernels are given by

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.13}$$

In this case, the kernel G_x is sensitive to changes in the x direction, i.e. edges that run vertically, or have a vertical component. Similarly, the kernel G_y is sensitive to changes in y direction, i.e. edges that run horizontally, or have a horizontal component. The two gradients computed at each pixel (G_x and G_y) by convolving with above two kernels can be regarded as the x and y components of gradient vector. This vector is oriented along the direction of change, normal to the direction in which the edge runs. Gradient magnitude and direction are given by:

$$G = \sqrt{G_x^2 + G_y^2} \tag{2.14}$$

An approximate magnitude is computed using:

$$|G| = |Gx| + |Gy| \tag{2.15}$$

The angle of orientation of the edge (relative to the pixel grid) giving rise to the spatial gradient is given by

$$\theta = a\,tan\left(\frac{G_y}{G_x}\right) \tag{2.16}$$

In our other work, we evaluate hardware resources and power consumption of Sobel Edge Detection which is implemented with two studies: Xilinx system generator (XSG) and Vivado_HLS tools. These tools both are very useful for developing computer vision algorithms. The comparison the hardware resources and power consumption among FPGA platforms (Zynq-7000 AP SoC, Spartan 3A DSP) are analysed.

2.2.3 Object Feature Extraction

Before extracting features of object to understand its behaviour, it is necessary to modelise the shape of object, called body modeling step.

2.2.3.1 Body modeling

Objects are generally represented by their shapes and appearances. In this section, we first describe the object's shape representations for tracking and then address the joint shape and appearance representations [64]:

- Points. The object is represented by a point, that is, the centroid (Figure 2.8(a)) [65] or by a set of points (Figure 2.8(b)) [66]. In general, the point representation is suitable for tracking objects that occupy small regions in an image.

- Primitive geometric shapes. Object shape is represented by a rectangle, ellipse (Figure 2.8(c), (d) [67], etc. Object motion for such representations is usually modelled by translation, affine, or projective (homography) transformation. Though primitive geometric shapes are more suitable for representing simple rigid objects, they are also used for tracking non rigid objects.
- Object silhouette and contour. Contour representation defines the boundary of an object (Figure 2.8(g), (h)). The region inside the contour is called the silhouette of the object (see Figure 2.8(i)). Silhouette and contour representations are suitable for tracking complex non rigid shapes [68].

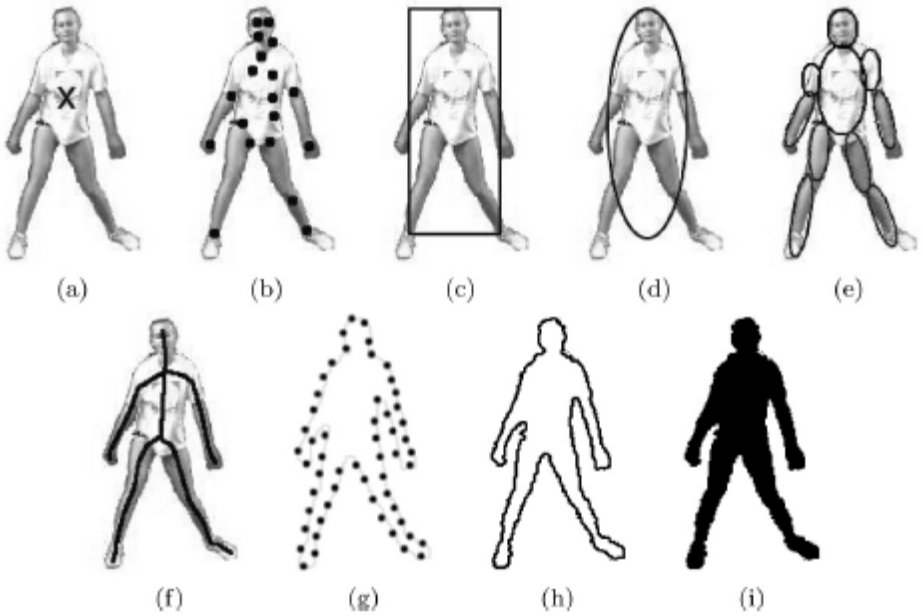


Figure 2.8-Object representations. (a) Centroid, (b) multiple points, (c) rectangular patch, [64]
(d) Elliptical patch, (e) part-based multiple patches, (f) object skeleton,
(g) Complete object contour, (h) control points on object contour, (i) object silhouette.

- *Articulated shape models.* Articulated objects are composed of body parts that are held together with joints. For example, the human body is an articulated object with torso, legs, hands, head, and feet connected by joints. The relationships among the parts are governed by kinematic motion models, for example, joint angle, etc. In order to represent an articulated object, one can model the constituent parts using cylinders or ellipses as shown in Figure 2.8(e).
- *Skeletal models.* Object skeleton is extracted by applying medial axis transform to the object silhouette. This model is commonly used as a shape representation for recognizing objects [69]. Skeleton representation is modelised both articulated and rigid objects (see Figure 2.8(f)).

There are a lot of ways to represent the appearance features of objects. It is noted that shape representations are also combined with the appearance representations [70] for

tracking. Some popular appearance representations in the context of object tracking are presented as follows:

- *Probability densities of object appearance*. The probability density estimates of the object appearance is either parametric, such as Gaussian and a mixture of Gaussians [71] or nonparametric. The probability densities of object appearance features (colour, texture) are computed from the image regions specified by the shape models (interior region of an ellipse or a contour).
- *Templates*: are formed using simple geometric shapes or silhouettes [72]. An advantage of a template is that it carries both spatial and appearance information. Templates, however, only encode the object appearance generated from a single view. Thus, they are only suitable for tracking objects whose poses do not vary considerably during the course of tracking.
- *Active appearance models* are generated by simultaneously modelling the object shape and appearance [73]. The object shape is generally defined by a set of landmarks. Similar to the contour-based representation, the landmarks reside on the object boundary or, alternatively, they can reside inside the object region. For each landmark, an appearance vector is stored which is in the form of colour, texture, or gradient magnitude. Active appearance models require a training phase where both the shape and its associated appearance are learned from a set of samples using, for instance, the principal component analysis.
- *Multi-view appearance models.* These models encode different views of an object. One approach to represent the different object views is to generate a subspace from the given views. Subspace approaches, for example, Principal Component Analysis (PCA) and Independent Component Analysis (ICA), have been used for both shape and appearance representation [74].

2.2.3.2 Body modeling based on ellipse model

Ellipse model is a very simple model describing the motion or the shape of the human body. In this model, a single object is surrounded by an ellipse. Ellipse model accompanying with fall action is shown in Figure 2.9. There are three important parameters of the ellipse model that are defined as follows:

- Centroid of ellipse: In each frame, the centroid coordinate of ellipse *O (Ox, Oy)* is an average of the all *x* coordinates and the all *y* coordinates of the white pixels.

$$x = \frac{\sum_i \sum_j [j.P(i, j)]}{Height.Width} \tag{2.17}$$

$$y = \frac{\sum_i \sum_j [i.P(i, j)]}{Height.Width} \tag{2.18}$$

Where, *Height, Width* are the height and width of the image frame.

i = 1: Height; j = 1: Width

$P(i,j)$ is the binary value of current image frame which position pixel is at the *(i,j)*;

In which $P(i,j)=0$ if *(i, j)* image pixel is black and $P(i,j)=1$ if *(i, j)* image pixel is white [26].

- Vertical angle of the object: The vertical angle of the object is the angle between the major axis of ellipse and horizontal line.

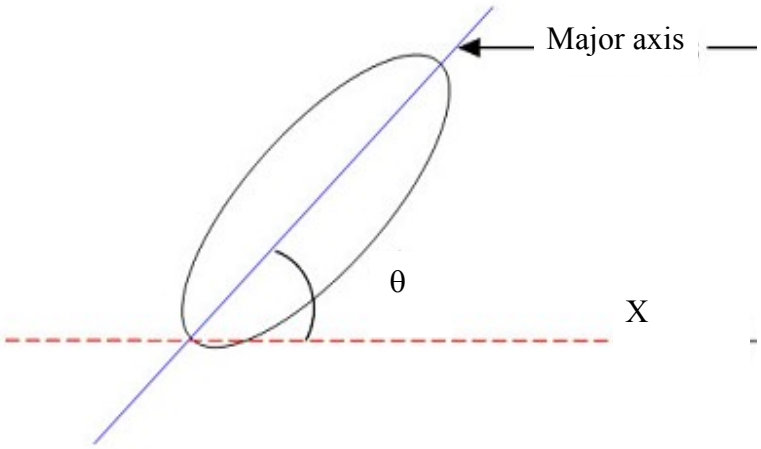


Figure 2.9-Vertical angle θ

- The value of θ is determined as interval of $\left[-\frac{\pi}{4},\frac{3\pi}{4}\right]$. The goal of getting the defined domain is to easier distinguish the fall of object being face fall or side fall. The vertical angle θ is calculated by

$$\tan 2\theta = \frac{2\tan\theta}{1-\tan^2\theta} \tag{2.19}$$

If we move the axis to the centroid of ellipse and calculate the mean of pixels, the vertical angle θ will be recalculated as following:

$$\theta = \frac{1}{2}.\arctan\left(\frac{2\sum_i\sum_j x.y.P(i,j)}{\sum_i\sum_j x^2.P(i,j)-\sum_i\sum_j y^2.P(i,j)}\right) \tag{2.20}$$

Where, $x = i - O_x$ and $y = j - O_y$ (O_x, O_y: centroid of ellipse coordinate).

Figure 2.10 shows some examples of ellipse models which describe the duration of fall action.

- Major and minor axis of the object: Major and minor axes are double distances from O to $O_1(x_1, y_1)$ and $O_2(x_2, y_2)$, respectively, where:

* O_1 is average of the all x coordinates and the all y coordinates of the white pixels *W(Wx, Wy)* located in the limited angle so that $\left|W\hat{O}h - \theta\right| < 60^0$

* O_2 is average of the all x coordinates and the all y coordinates of the white pixels *W(Wx, Wy)* located in the limited angle so that $\left|W\hat{O}h - (\theta + \pi/2)\right| < 60^0$

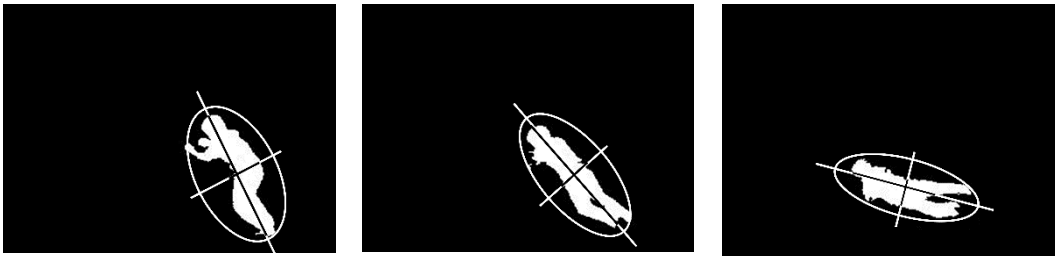


Figure 2.10-Ellipse model for fall action

2.2.3.3 Object Feature Extraction

After modeling the human body with ellipse model, features from model are extracted to identify the falls of object. There are many features that are used to recognize the fall action from human with 2D ellipse model. In the context of our work, 5 features are used to detect and classify falls comparing with the daily activities of object:

- *Vertical angle* θ (or Current Angle).
- Coefficient of motion (C_{motion}).
- Deviation of the angle (C_{Theta}).
- *Eccentricity*.
- Deviation of the centroid ($C_{Centroid}$).

a) Current angle is vertical angle of the object. At the same camera angles the object with various movement releases difference for the deviation angle of the object. Instantaneous angle is vertical angle of the object in the frame, is also elliptical angle θ calculated above [26]. Figure 2.11 shows that at the same camera angles, if the object is walking, the vertical angle of object closes to 90^O via horizontal, and if object is bending or falling, the current angle of object will far from 90^O.

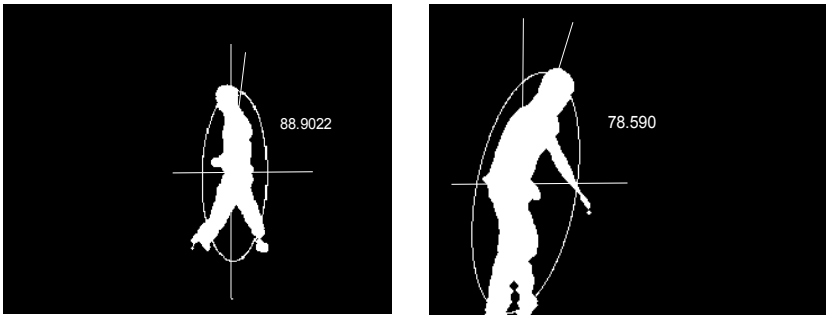
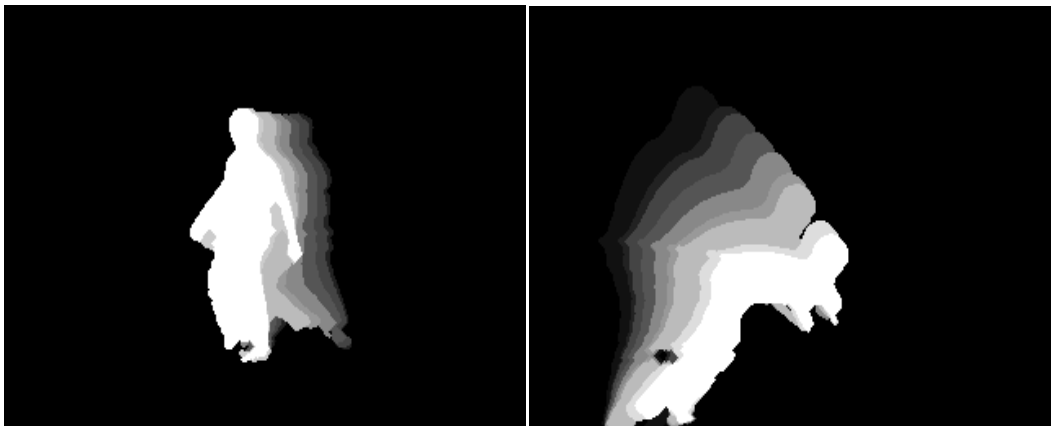


Figure 2.11-Current angle of object

b) Coefficient of motion (C_{motion})

The brightness of a gray image has value in the interval [0, 255], in which the value of 0 is blackest and 255 is whitest. The speed of the object motion is determined by a gray image in which the black pixels are background pixels, white pixels is the object extracted from the current frame. At the same position over time, the gray pixels are the white pixels of the previous image frame. Therefore, the gray frame is used to determine speed of the object motions as known Motion History Image (MHI)[75] as shown in Figure 2.12.



(a *(b)*

Figure 2.12-Motion History Image.
(a) MHI of slow action (b) MHI of fast action

The MHI or the C_{motion} of object is determined by following equation

$$C_{Motion} = \frac{Graypixel}{Graypixel + Whitepixel} \tag{2.21}$$

In each MHI, C_{motion} has value in a haft of interval [0, 1). C_{motion} = 0 when the object almost does not walk around. C_{motion} comes to 1 when the object moves as quickly as he/she does. Whenever the fall occurs, C_{motion} has high values because the motion speed

is rather high. As shown in Figure 2.13, during the period of fall in the red line C_{motion} is significantly higher value with 0.7 than the blue line of normal walking C_{motion} is within (0.5- 0.6). However with the different camera angle, C_{motion} has different value. For the face fall action C_{motion} decreases lower than normal walking. Thus, to distinguish two cases, other features are determined.

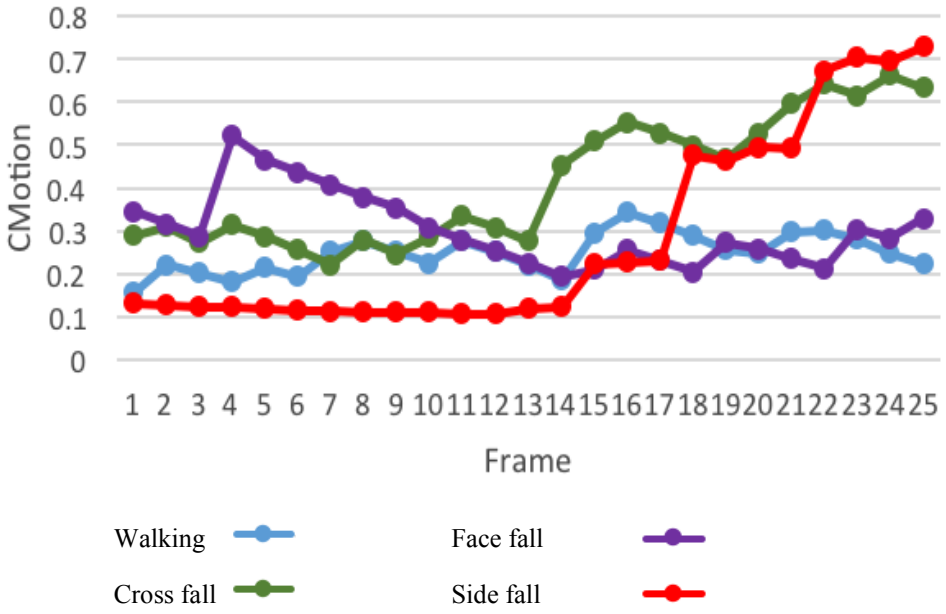


Figure 2.13-The variable of coefficient of motion

c) Deviation of the angle (C_{Theta})

Considering the frame at the moment t, C_{Theta} is calculated based on the values of the current frame of n frames from the frame $(t - n + 1)^{th}$ to the frame t^{th} (with $t \leq n$). Standard deviation is a value which performs the degree of convergence or the histogram of the database. If a database has small standard deviations, this means that the data elements have a high level of similarity. In contrast, the data elements scatter in the space of their value. In our work, C_{Theta} is standard deviation of angles θ from 15 successive frames. C_{Theta} is usually higher when a fall occurs [26].

The current vertical angle changes slowly and the level of moving vertically of the object is slow in the database. In this case, the standard deviation of the object changes at low standard deviations or vice versa. Therefore, when moving normally, the object is slightly changing in vertical angle, and thus the standard deviation value is small. On the other hand, when a fall action happens, the standard deviation value is suddenly high, as presented in Figure 2.14.

It is seen that in case of side falls and cross falls, the C_{Theta} is much higher than the non-fall actions and in light of direct falling actions. Figure 2.14 shows that C_{Theta} is slightly higher than walking, but smaller than side fall cases.

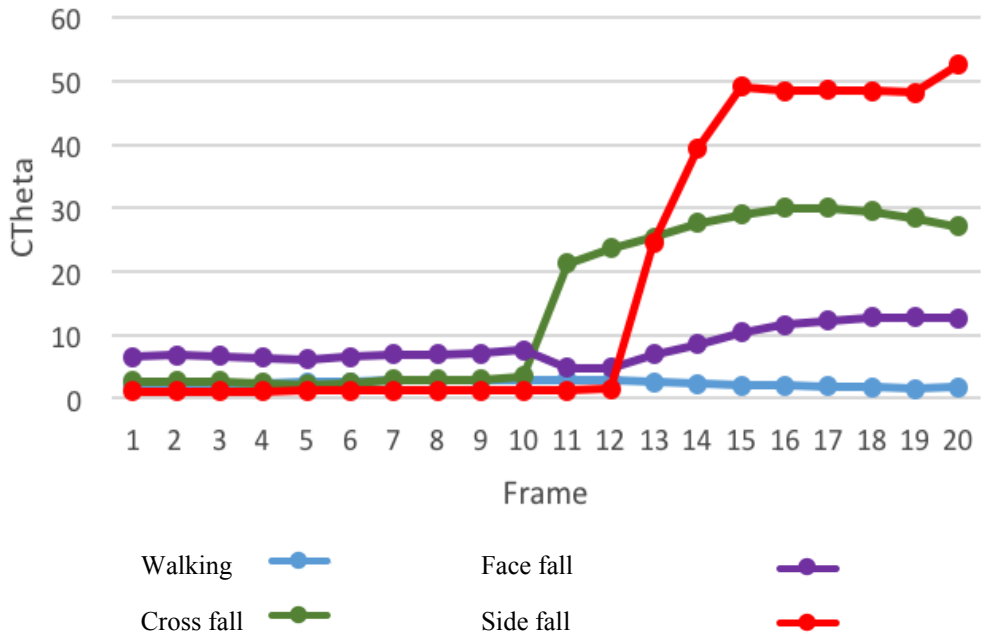


Figure 2.14-Deviation of the angle (C_{Theta})

d) Eccentricity

Eccentricity at current frame is computed as below:

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{2.22}$$

Where, e: eccentricity; a, b: semi-major and semi-minor axis of ellipse. e is smaller when direct fall happens [76]. This is clearly seen in Figure 2.15.

e) Deviation of the centroid ($C_{Centroid}$)

Considering the frame at the moment t, the system calculates $C_{Centroid}$ based on the values of y-coordinate of the centroid of n frames from the frame $(t - n + 1)^{th}$ to the frame t^{th} (with $t \leq n$). $C_{Centroid}$ is the standard deviation consisting of the y-coordinates of the certain n frames. $C_{Centroid}$ is standard deviation of centroid coordinates from 15 successive frames. $C_{Centroid}$ decreases rapidly when the fall occurs [26].

This feature distinguishes between fall actions and non-fall actions. When a fall action occurs (red line), the $C_{Centroid}$ is high as the vertical change of the eccentricity is so fast. In addition, when a non-fall action happens (blue line), the $C_{Centroid}$ is small as the slowly vertical changes.

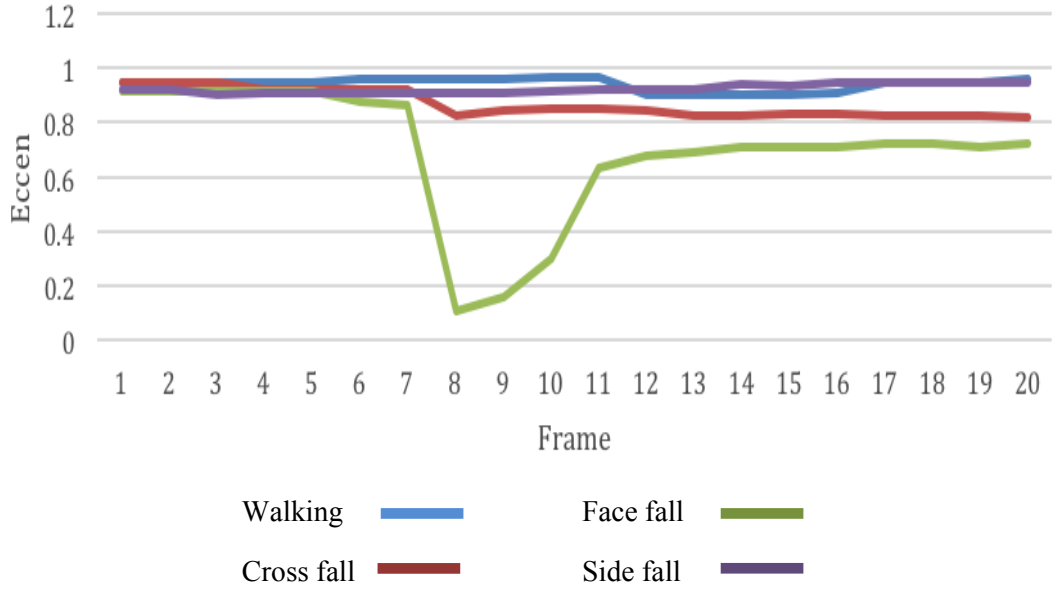


Figure 2.15-Eccentricity

2.3 Recognition event

In our work, three recognition algorithms are applied to realise the behaviour of object. Firstly, it is Threshold-based algorithm. However, this method is not robust with many different scenarios of our database. The second method is Neural Network algorithm. This is more effectively than Threshold-based one, but it is too slow to apply our system in case of online. Therefore, the third algorithm, Hidden Markov Model, is also used to train and test on this system.

2.3.1 Threshold-based algorithm

Threshold-based algorithm is to set some hard thresholds to distinct whether an input action is fall or non-fall. These thresholds are picked up from training process which is to choose the relevant values for 5 main parameters in our system, namely C_{Motion}, C_{Theta}, $C_{Centroid}$, *Theta* and *Eccentricity*.

Based on the direction of falls and the type of falls, three models are built to detect falling accidents [77]. The first model is direct fall. In this case C_{Motion}, C_{Theta}, $C_{Centroid}$ are high but *Theta* is low. The second model is cross fall, in which C_{Motion}, C_{Theta} is high, and *Theta*, $C_{Centroid}$ and *Eccentricity* get a medium value. In the last model, the victim falls in the both side directions of the camera. Consequently Theta is almost constant, C_{Motion} is in average, *Eccentricity* is low while $C_{Centroid}$ is quite high. The features are combined with each other depending on the fall models, the thresholds are selected from the survey of training videos.

Although this algorithm offers a significantly fast method to train and test the fall detection, there is still a drawback in system's recognition ability. The hard thresholds is not flexible, therefore it is difficult to have an exact decision in a complicated data.

2.3.2 Neural Network algorithm

Neural Network (NN) is the second method dealing with fall action's recognition in this study. The neural network is divided into three layers: the input layer, the hidden layer and the output layer. Each layer in this order gives input to the next one [78]. The threshold function of the units is modified to be a sigmoid function. The use of the sigmoid function gives the extra information necessary for the network to implement the back propagation training algorithm. Actually, the network utilised in this work is feed forward neural network of which neurons are only connected foreword. Each layer of the neural network contains connections to the next layer (for example, from the input to the hidden layer), but there are no back connections. Back propagation which is a form of supervised training describes how this type of neural network is trained. Back propagation works by finding the squared error of the entire network, and then calculating the error term for each of the output and hidden units by using the output from the previous neuron layer. The weights of the entire network are then adjusted with dependence on the error term and the given learning rate. Training continues on the training set until the error function reaches a certain minimum. If the minimum is set too high, the network might not be able to correctly classify a pattern. But if the minimum is set too low, the network will have difficulties in classifying noisy patterns.

In the computer-based part of our work, the neural network's initialization follows 4 major steps: configuration, training algorithm selection, training optimization and test [26].

2.3.2.1 Configuration

To detect person's falling, a two-layer feedforward NN is initialized with a five-dimension input vector which comprises five extracted features from the previous step and a two-dimension output vector which represents for fall and non-fall decision. The single hidden layer consists of *nhu* neurons (*nhu*-number of hidden unit) with weights and biases. The activation function of the neurons in the hidden layer is the hyberbolic tangent sigmoid function due to its desirable attributes. The number of hidden units *nhu* was chosen to be variable, as the optimal value depends mainly on the complexity of the problem. *nhu* was varied for each optimization step in a wide range to derive the best value for a specific configuration. The output layer consists of two output neurons whose activation transfer functions are again the hyperbolic tangent sigmoid function. The first output signalizes the falling, and the second output presents non fall. And the first target

output is labelled with 1 for falling and 0 for non-falling. Biases and weights of all units were initialized randomly.

2.3.2.2 Training algorithm selection

The optimization criterion is selected to be the minimization of the mean square error (MSE) derived over the whole training set [79]. The maximum number of epochs is set to quasi ∞, and the goal of training, which is asked to be a small MSE value, is set to quasi zero. The training set is first split up into a training subset and a validation subset. The training of the ANN only stops if the MSE derived from the validation subset could not be reduced within 5 consequent training epochs. This method is considered to retain generalization.

Two learning algorithms are considered: Scale Conjugate Gradient (SCG) [80] and Resilient Backpropagation (RP) [81]. Because SCG algorithm is investigated due to its common usage in pattern classification tasks and RP algorithm is fast convergence.

2.3.2.3 Training optimization

To optimize training process, there are three main steps involving selecting learning algorithm, size of validation subset and number of hidden units.

- Firstly, two learning algorithms are considered: Scale Conjugate Gradient (SCG) and Resilien Backpropagation (RP). The database is randomly split up into 20% validation and 80% training subsets. From training process, the SCG algorithm is selected as providing a better work, which means the MSE gets optimized within a significantly smaller number of training epochs, where $f - score = \frac{2.RC.PR}{(RC+PR)}$.
- Secondly, to avoid overfitting the validation course is performed during the training period. The database is divided into a training subset and a validation subset (VS) in such a way that VS is large enough to have similar characteristics to the training subset, otherwise the training algorithm will stop early. On the other hand, the size of the validation subset should be kept as small as possible to retain a large training subset. From the former step, the Scale Conjugate Gradient (SCG) algorithm is used to train the neural network with hidden layers is fixed at 10 with validation subsets of different relative size (5%, 10%, 15%, 20%, 25%, 30% and 35%).
- After the process, it is said that 20% validation set and 80% training set offer the largest f-score and the smallest MSE. So this case is chosen as the appropriate configuration. Eventually, the SCG algorithm and the 20%-validation-subset size are implemented. We only changes the number of hidden layer in a wide range of *nhu* = {10, 20, 30, 40, 50, 60} to the optimal configuration. At the end, the

configuration with *nhu* = 50 is optimal, due to f-score is the largest and MSE is the smallest.

2.3.2.4 Test

After being trained by neural network, the neural network is used to test DUT-HBU database described more detail in Appendix.

2.3.3 Hidden Markov Model algorithm.

The Hidden Markov Model (HMM) is a useful statistical tool for modelling generative sequences that can be characterized by a basically process generating observable sequences. HMM, which has recently been applied with particular success to speech recognition, is a kind of stochastic state transit model [82]. Besides, dealing with variable length feature vectors as fall and non-fall action is an advantage of HMM over other machine learning methods. In this work, fall or non-fall state transition is analysed through observation series $O = \{O_1, O_2,\ldots,O_n\}$ indirectly. The 5 state HMM expresses fall process is denoted as λ and it can be demonstrated by arrays: $\lambda = (\pi, A, B)$ for short.

For training process, λ is adjusted to get the conditional probability P(O|λ) maximum. The Baum-Welch algorithm for unsupervised training is used for training purpose [82]. This algorithm computes maximum likelihood estimates and posterior mode estimates for the parameters of the HMM. This is updated weights through recursion to get better model. In order to distinguish falling actions from the other non-fall activities, two HMMs were built. The observation data of HMMs is code-words in the codebook. The initial condition used for HMM training is 5 hidden states and random values for initial state distribution (π), state transition matrix (A) and emission matrix (B). Then the Baum-Welch algorithm is run until convergence condition is satisfied. In this study, experimental results showed that the 5-state left-to-right hidden Markov model provided the highest performance.

For testing process, this process in this proposed system is separated into two following steps:

- Clustering: The Euclidean distance from each feature vector to each codeword in the codebook is calculated. This feature vector is marked by codeword coefficients that have the shortest distance to it.
- Decoding and decision making: A vector containing 15 coefficients is taken into decoding process for both fall and non-fall models. After decoding, the system compares the results of two models to make the decision to label “1” (if fall model is more likelihood) or “0” (if non-fall model is more likelihood). Then the

label will be stored in a buffer length is 15. When the total number of "1" in this buffer is greater than "predefined" threshold, the falling incident is detected.

2.4 Evaluation

2.4.1 DUT-HBU database

2.4.1.1 Database description (see more in Appendix)

Database name: **DUT-HBU** database (**D**anang **U**niversity of **T**echnology- **H**uman **B**ehavior **U**nderstanding) of Electronic & Telecommunication Engineering Department, Danang University of Technology, Danang, Vietnam.

In our database, there are 216 videos which are divided into 106 falling videos and 110 non-fall videos as shown in Table 2.1. In our work, 113 videos are used for training and the rest are used for testing purpose. Scenario of creating the database of falling is based on direction of object with camera. Three falling directions (Figure 2.16) are defined in this database as:

- Direct: object falls the same orientation with the direction of the camera.
- Cross: objects created the 30^o-60^o angles with camera when the falling occurs.
- Side: object falls in a perpendicular direction to the camera.

In each direction of the falling action, these videos also include activities as follows:

- Slip: Objects are slipped and fallen backward.

- Stumble: Objects are fallen ahead, do not kneel but raise their hand.

- Faint: Objects are fallen ahead, kneel but do not raise their hand.

- Roll-fall: Objects are rolled down from high position when they are lying.

Side *Direct* *Cross*

Figure 2.16-The position of falling compared with angles of camera

Besides the fall clips, non-fall videos are also classified by three directions above. These videos consist of activities which are easily confused with falling action such as: lying, sitting, creeping, and bending [83].

- Bending: Doing exercise or putting your arm down.

- Creeping: Crawling to find something on the floor.

- Lying: Walking and lying on the floor.

- Sitting: Sitting on the chair or the floor.

Table 2.1-Classifier of videos according to activities

DATABASE			**Training**		**Testing**			**Sum**
			Pure data	Noisy data	Test1	Test2	Test	
Fall	*Cross*		4	18	4	4	8	34
	Direct		4	19	4	6	9	38
	Side		5	17	4	5	8	34
Non-fall	*Bending*	Cross	1	4	1	1	1	7
		Direct	3	5	1	1	1	8
		Side	1	3	1	2	2	8
	Creeping	Cross	1	3	1	2	1	7
		Direct	2	4	1	1	1	7
		Side	1	4	1	1	1	7
	Lying	Cross	1	3	1	1	2	7
		Direct	3	5	1	1	0	7
		Side	1	4	1	1	2	8
	Sitting	Cross	0	2	0	1	2	5
		Direct	3	6	1	1	1	9
		Side	1	4	1	1	1	7
	Others		0	12	0	0	11	23
Sum			31	113	23	29	51	216

From the classification in Table 4.1, we divide to analyse more detail the action cases as presented in Table 4.2.

2.4.1.2 Training databases

Two training scenarios are implemented in this study:

- Scenario 1: Training with pure data.

Pure data consists of videos which have stable background. These videos are captured in a small room under good brightness condition. The object is not obscured by furniture in the room. Furthermore, our subjects wear natural clothing (as opposed to motion capture suits that is often done for pure motion capture sessions). Training set in the Scenario 1 is named as Scenario1 set. It contains 31 video clips of clear data with 13 falling clips and 18 non-fall clips.

Table 2.2-Glossary of action classification

Action	Meaning
Fc (Fall Cross)	Fall creates a cross direction with camera angles
Fd(Fall Direct)	Fall creates the same orientation with the direction of the camera
Fs (Fall Side)	Object falls in a perpendicular direction to the camera
Ncb (Non-Fall Cross Bending)	Object bends in a cross direction with camera angles
Ndb (Non-Fall Direct Bending)	Object bends in the same orientation with the direction
Nsb (Non-Fall side Bending)	Object bends in a perpendicular direction to the camera
Ncc (Non-Fall Cross Creeping)	Object creeps in a cross direction with camera angles
Ndc (Non-Fall Direct Creeping)	Object creeps in the same orientation with the direction
Nsc (Non-Fall Side Creeping)	Object creeps in a perpendicular direction to the camera
Ncl (Non-Fall Cross Lying)	Object lies in bed or on a bench in a cross direction with camera angles
Ndl (Non-Fall Direct Lying)	Object lies in bed or on a bench in the same orientation with the direction
Nsl (Non-Fall Side Lying)	Object lies in bed or on a bench in a perpendicular direction to the camera
Ncs (Non-Fall Cross Sitting)	Object sits on a chair or on the floor in a cross direction with camera angles
Nds (Non-Fall Direct Sitting)	Object sits on a chair or on the floor in the same orientation with the direction
Nss (Non-Fall Side Sitting)	Object sits on a chair or on the floor in a perpendicular direction to the camera

- Scenario 2: Training with noisy data.

The noisy data consists of videos that have activities or situations similar to the ones of the Test2, and Test3 sets (which will be described later). Noise data is used for enriching the training set and provided others cases for a better training. The training set is named as Scenario2. It includes 31 clear data videos, 29 videos similar to the videos of Test2 set and 51 videos similar to the ones of Test3 set. They have 54 falling video clips and 59 non-fall video clips in the Scenario 2. The testing sets of scenario 1 are reused in this scenario.

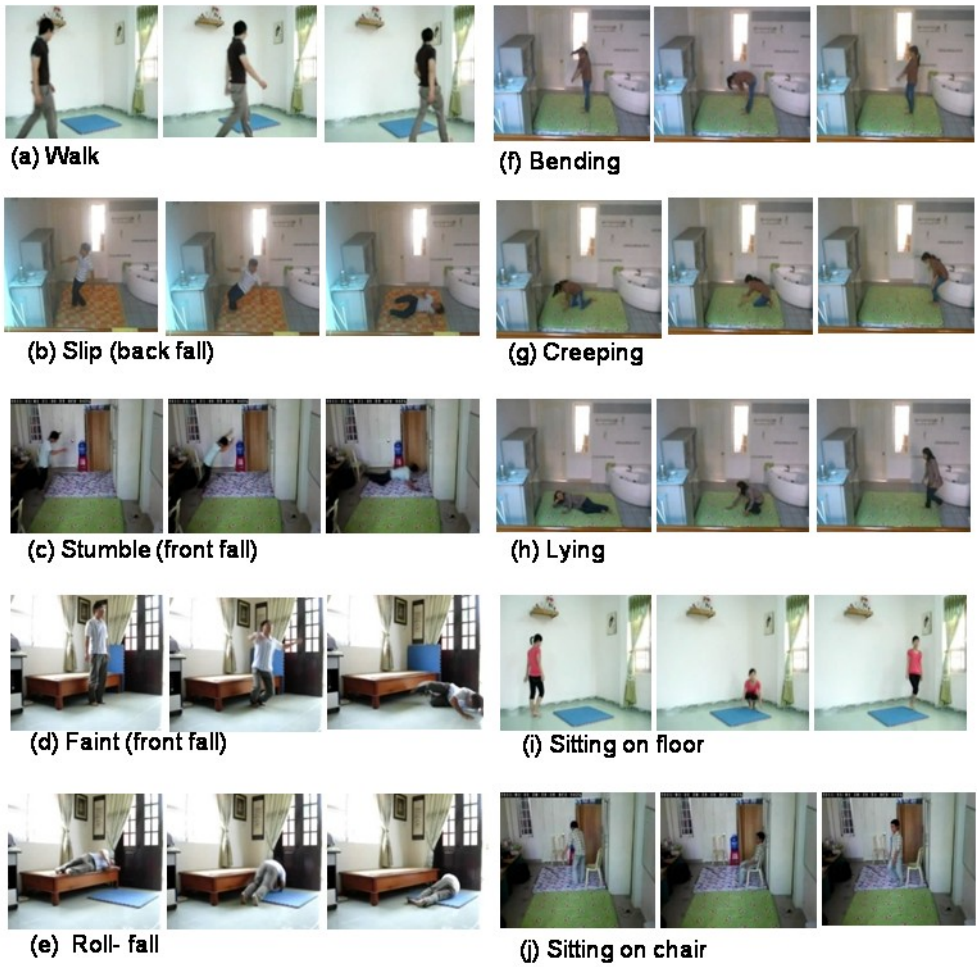


Figure 2.17-Daily activities look like falling

2.4.1.3 Testing databases

The training set consists of videos which have static backgrounds. They are captured in a small room under good brightness condition. The object is not obscured by furniture in the room. Furthermore, moving subjects wear natural clothing (as opposed to motion capture suits that is often done for pure motion capture sessions). In comparison to the training data in [26], this training database contains 31 videos structured from 13 falling videos and 18 non-fall videos.

The testing set is regrouped into three main types which are named as Test1, Test2 and Test3 corresponding to three different testing scenarios as well-matched (WM), medium-mismatched (MM) and highly-mismatched (HM) condition, respectively. This setup is designed to qualify robustness of the developed algorithms as illustrated below:

- WM test clips: Their contents and recoding conditions are very similar to the ones for training. In each clip, there is only one moving object with static background. This set has 12 falling videos and 11 non-fall videos.

- MM test clips: includes these actions which have similar characteristics of object in the training videos but the environment brightness and camera position are changed. This consists of 15 fall videos and 14 non-fall videos.
- HM test clips: There are many changes in activities and recording conditions compared to those of the training videos such as: part of the object is obscured, background is changed with extra static objects, or there are more than two moving objects in these video. This consists of 25 fall videos and 26 non-fall videos.

The detailed breakdown of different types of fall videos and non-fall videos is shown in Table 2.1.

2.4.2 Performance measurement

Receiver-operating characteristic (ROC) analysis was originally developed during World War II to analyse classification accuracy in differentiating signal from noise in radar detection [84]. ROC analysis is a useful tool for evaluating the performance of database tests and more generally for evaluating the accuracy, recall and precision. Performance of such systems is commonly evaluated using the data in the matrix called as contingency table or confusion matrix. In Figure 2.18 a confusion matrix gives information about actual (True class) and predicted (hypothesized class) classifications done by a classification system.

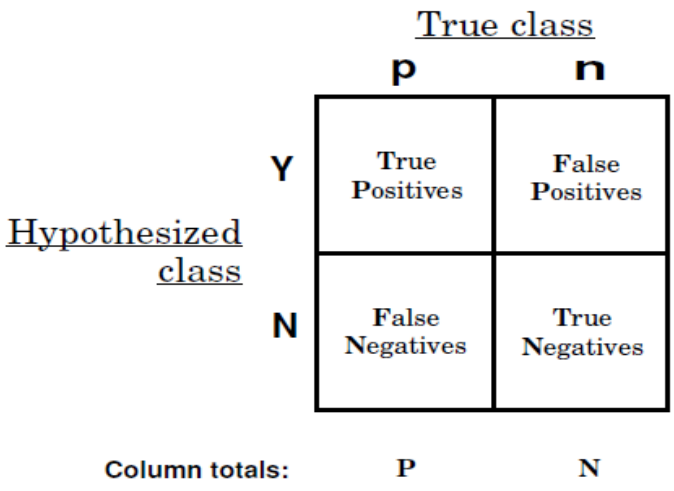


Figure 2.18-Confusion matrix

In our work, the following statistical measures are exploited to assess the examined algorithms: Recall (RC) [%], Precision (PR) [%] and Accuracy (Acc) [85]. They are defined as follows:

$$RC = \frac{TP}{TP + FN}, PR = \frac{TP}{TP + FP}, Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.23)$$

Whereas *TP*: true positives (fall detection with fall videos); *FP*: false positives (fall detection with non-fall videos); *FN*: false negatives (no fall detection with fall videos) and *TN*: true negative (no fall detection with non-fall videos).

The role of the *RC*, the *PR*, and the *Acc* is to evaluate overall performance of the system. The higher values of the *RC*, the *PR*, the *Acc* are, the more effective system is. Actually, two parameters *PR, RC* play a key role in a fall detection system due to the fact that when applying the detection system falls to the elderly, especially elderly or patients living alone, we need to figure out exactly fall actions in order to have a correct and prompt warning to ensure life safety for the monitored people.

2.4.2.1 Performance of the system based on Neural Network

Besides two training scenarios in the previous section, two different feature sets are extracted and examined:

- The first feature set (FS1) contains 5 features which are extracted in every frame as proposed in our previous work [12].
- The second feature set (FS2) is built by extracting five features above from 20 consecutive frames. In this feature set, 20 frame-sequence times is equal to fall action, so FS2 is dynamic feature set.

In this subsection, we evaluate performance of feature sets as well as roles of different training sets, there are four models as follows:

- Model 1: Feature Set 1, Scenario 1 (FS1-SN1).
- Model 2: Feature Set 1, Scenario 2 (FS1-SN2).
- Model 3: Feature Set 2, Scenario 1 (FS2-SN1).
- Model 4: Feature Set 2, Scenario 2 (FS2-SN2).

a) Performance of the Model 1 (FS1 – SN1)

Table 2.3 describes the detailed results of Test1, Test2 and Test3 for this model.

Table 2.3-Performance of the Model 1

Scenario1 FS1		Fall				Non-fall				Sum			
		Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL
Fall	Fc	4	3	6	13	0	1	2	3	4	4	8	16
	Fd	4	3	6	13	0	3	3	6	4	6	9	19
	Fs	4	5	7	16	0	0	1	1	4	5	8	17
Non-fall	Ncb	0	0	0	0	1	1	1	3	1	1	1	3
	Ndb	0	0	0	0	1	1	1	3	1	1	1	3
	Nsb	0	0	0	0	1	2	2	5	1	2	2	5
	Ncc	0	0	1	1	1	2	0	3	1	2	1	4
	Ndc	0	1	0	1	1	0	1	2	1	1	1	3
	Nsc	0	0	0	0	1	1	1	3	1	1	1	3
	Ncl	0	0	1	1	1	1	1	3	1	1	2	4
	Ndl	0	1	0	1	1	0	0	1	1	1	0	2
	Nsl	0	0	1	1	1	1	1	3	1	1	2	4
	Ncs	0	0	0	0	0	1	2	3	0	1	2	3
	Nds	0	0	0	0	1	1	1	3	1	1	1	3
	Nss	0	0	0	0	1	1	1	3	1	1	1	3
	No	0	0	3	3	0	0	8	8	0	0	11	11
Sum										23	29	51	103

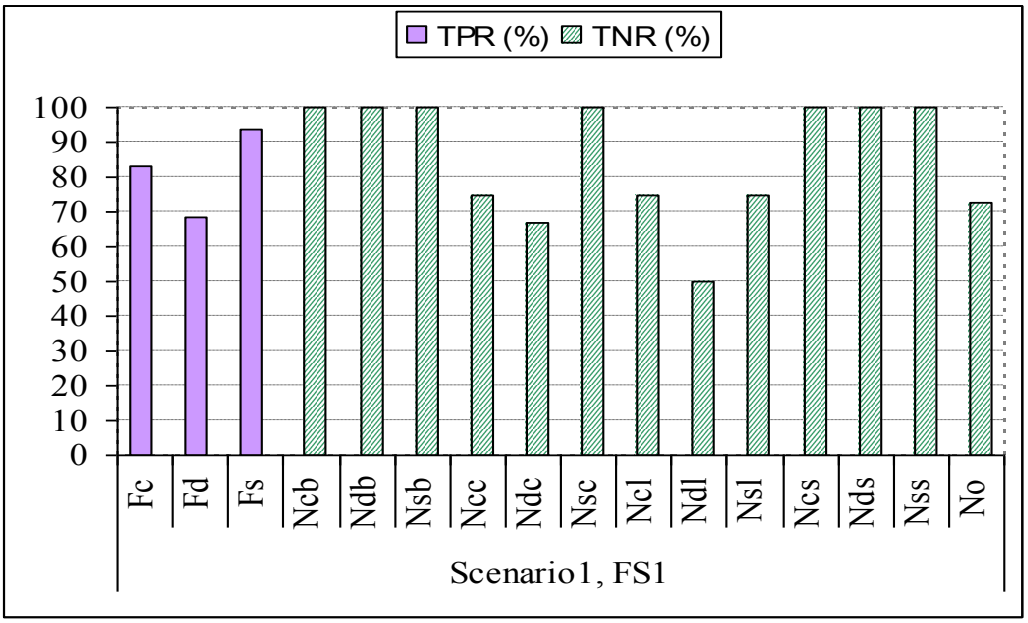


Figure 2.19-Evaluating TPR and TNR of ALL three tests (FS1, SN1)

Based on the results in Table 2.3 the statistical results namely True positive rate-TPR [%] and True negative rate-TNR [%] are calculated and presented as shown in Figure 2.19

- When the direction of the camera is considered, TPR decreases accompanying the percentage of seeing objects decline, such as TPR is much over 90% for side falls, is around 90% for cross falls and below 90% in direct falls.
- For non-fall actions: Figure 2.17 presents the bending or sitting on a chair which is not confused with falling actions, on the other hand actions such as sitting on the floor, lying or creeping are easily to be confused by falling actions.

b) Performance of the Model 2 (FS1 – SN2)

The detailed result of Test1, Test2 and Test3 for this model is shown in Table 2.4.

- In this model, the performance of the system is improved significantly; however, the one for direct falls is still not good. After analysing the result, we realise that there are some confusion between direct falls and sitting on the floor. These actions have the similar features but happen in different duration.
- The action falls down and not be mistaken to be considered in detail and we find often is confused between direct action and action fall sitting on the floor. Two actions have similar properties, but the duration of action is different.

To overcome this shortcoming, a features' vector consisting of 20 consecutive frames which is the same as the time a falling action happens is considered.

Table 2.4-Performance of the Model 2

Scenario2 FS1		Fall				Non-fall				Sum			
		Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL
Fall	Fc	4	4	7	15	0	0	1	1	4	4	8	16
	Fd	3	4	8	15	1	2	1	4	4	6	9	19
	Fs	4	5	6	15	0	0	2	2	4	5	8	17
Non-fall	Ncb	0	0	0	0	1	1	1	3	1	1	1	3
	Ndb	0	0	0	0	1	1	1	3	1	1	1	3
	Nsb	0	0	0	0	1	2	2	5	1	2	2	5
	Ncc	0	1	0	1	1	1	1	3	1	2	1	4
	Ndc	0	0	0	0	1	1	1	3	1	1	1	3
	Nsc	0	0	0	0	1	1	1	3	1	1	1	3
	Ncl	0	0	0	0	1	1	2	4	1	1	2	4
	Ndl	0	0	0	0	1	1	0	2	1	1	0	2
	Nsl	0	0	0	0	1	1	2	4	1	1	2	4
	Ncs	0	0	0	0	0	1	2	3	0	1	2	3
	Nds	0	0	0	0	1	1	1	3	1	1	1	3
	Nss	0	0	0	0	1	1	1	3	1	1	1	3
	No	0	0	2	2	0	0	9	9	0	0	11	11
Sum										23	29	51	103

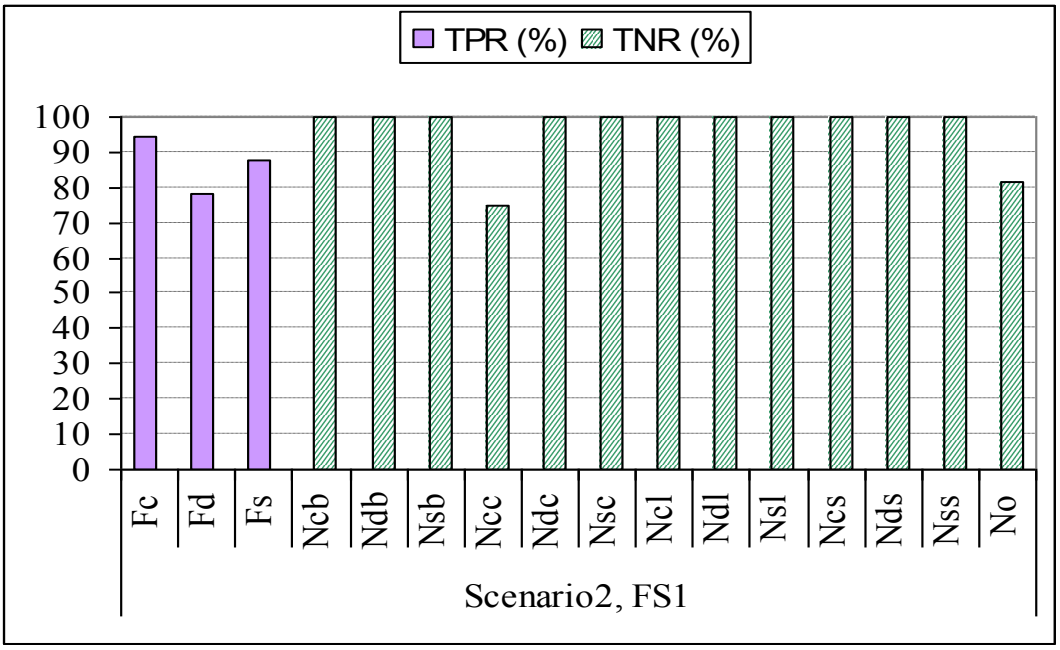


Figure 2.20-Evaluating TPR and TNR of ALL three tests (FS1, SN2)

c) Performance of the Model 3 (FS2 – SN1)

The detailed results of Test1, Test2 and Test3 for this model are shown in Table 2.5.

Table 2.5-Performance of the Model 3

Scenario1 FS2		Fall				Non-fall				Sum			
		Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL
Fall	Fc	4	3	7	14	0	1	1	2	4	4	8	16
	Fd	4	3	5	12	0	3	4	7	4	6	9	19
	Fs	4	4	7	15	0	1	1	2	4	5	8	17
Non-fall	Ncb	0	0	1	1	1	1	0	2	1	1	1	3
	Ndb	0	0	0	0	1	1	1	3	1	1	1	3
	Nsb	0	0	0	0	1	2	2	5	1	2	2	5
	Ncc	0	1	0	1	1	1	1	3	1	2	1	4
	Ndc	0	0	0	0	1	1	1	3	1	1	1	3
	Nsc	0	0	0	0	1	1	1	3	1	1	1	3
	Ncl	0	0	0	0	1	1	2	4	1	1	2	4
	Ndl	0	0	0	0	1	1	0	2	1	1	0	2
	Nsl	1	0	0	1	1	1	2	4	1	1	2	4
	Ncs	0	0	0	0	0	1	2	3	0	1	2	3
	Nds	0	0	0	0	1	1	1	3	1	1	1	3
	Nss	0	0	0	0	1	1	1	3	1	1	1	3
	No	0	0	2	2	0	0	9	9	0	0	11	11
Sum										23	29	51	103

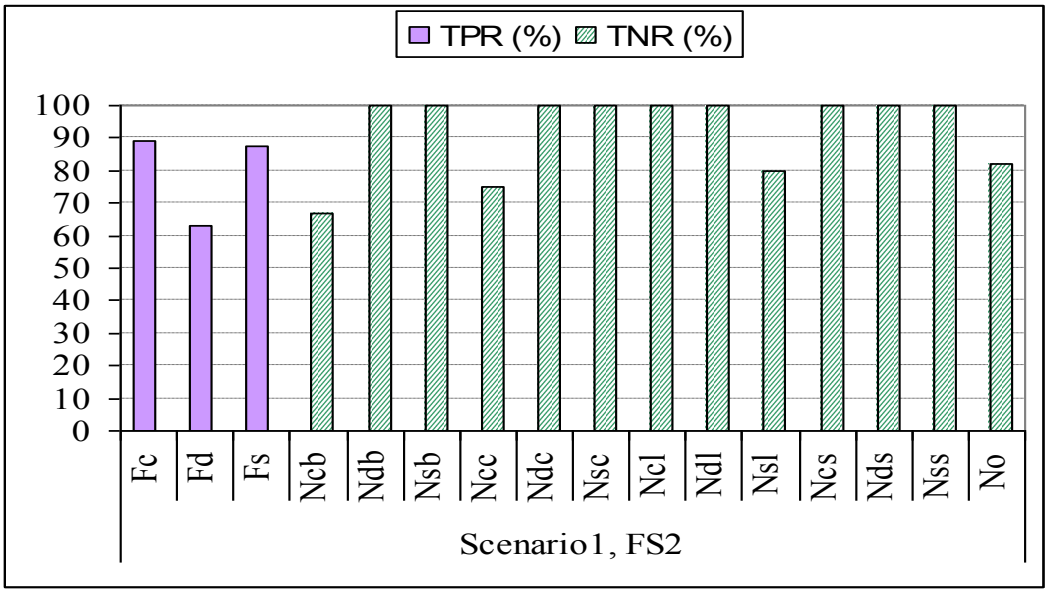


Figure 2.21-Evaluating TPR and TNR of ALL three tests (FS2, SN1)

As we can see in Figure 2.21 the performance of FS2 is better than the previous ones.

d) Performance of the Model 4 (FS2 – SN2)

Table 2.6 presents the detailed results of Test1, Test2 and Test3 for this model.

Table 2.6-Performance of the Model 4

Scenario2 FS2		Fall				Non-fall				Sum			
		Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL	Test1	Test2	Test3	ALL
Fall	Fc	4	4	8	16	0	0	0	0	4	4	8	16
	Fd	3	5	8	16	1	1	1	3	4	6	9	19
	Fs	4	5	8	17	0	0	0	0	4	5	8	17
Non-fall	Ncb	0	0	0	0	1	1	1	3	1	1	1	3
	Ndb	0	0	0	0	1	1	1	3	1	1	1	3
	Nsb	0	0	0	0	1	2	2	5	1	2	2	5
	Ncc	0	0	1	1	1	2	0	3	1	2	1	4
	Ndc	0	0	0	0	1	1	1	3	1	1	1	3
	Nsc	0	0	0	0	1	1	1	3	1	1	1	3
	Ncl	0	0	0	0	1	1	2	4	1	1	2	4
	Ndl	0	0	0	0	1	1	0	2	1	1	0	2
	Nsl	0	0	1	1	1	1	1	3	1	1	2	4
	Ncs	0	0	0	0	0	1	2	3	0	1	2	3
	Nds	0	0	0	0	1	1	1	3	1	1	1	3
	Nss	0	0	0	0	1	1	1	3	1	1	1	3
	No	0	0	1	1	0	0	10	10	0	0	11	11
Sum										23	29	51	103

In this model, the performance increases considerably. All side and cross falling actions are recognised correctly and the recognition of direct falling actions is over 80%. However, some fast non-fall actions are still confused with falling actions.

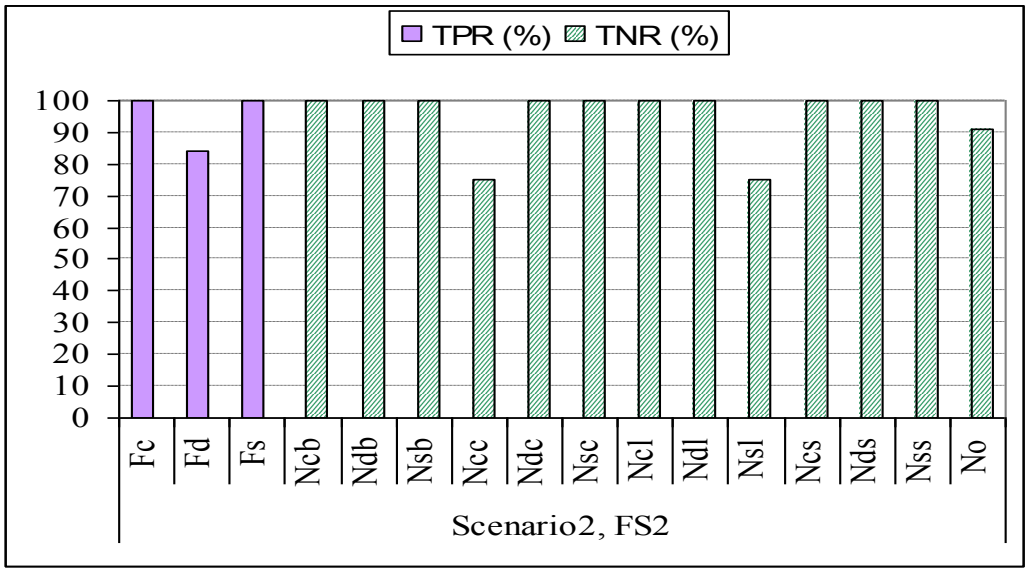


Figure 2.22-Evaluating TPR and TNR of ALL three tests (FS2, SN2)

e) Overall performance of the Neural Network

The overall performance for all four models is presented in Figure 2.23 This shows Recall (RC) [%], Precision (PR) [%] and Accuracy (Acc) [%].

The statistical results depicted in Figure 2.23 below provide information about classification performance of training methods.

- Statistical results in training Scenario2 are higher than in Scenario1. We observe that the accuracies of ALL set in scenario 2 with FS1 and FS2 are 90.38% and 94.23%, respectively. But in scenario 1, these factors are 82.69% and 84.76%.
- Training Scenario1: Its performance is acceptable in clean data. Statistical results decreased dramatically in the noise data. With FS1, results indicate that Acc is high for Test1 set up to 100%, but for Test2 set, this result is 79.31% and the lowest is in Test3 set, only 76.92%.
- Training Scenario2: This model obtains stable statistical results in almost any conditions. This proves that behaviour is fair when many data with different conditions are trained.
- NN which is trained with FS2 gives much better results than NN which is trained with FS1, in the same training scenario. With FS2, the time element is added. It increases the recognition ability.
- In four fall detection models above, the fourth model which used FS2 and training Scenario2 performs the best and behaves fair in almost any conditions.

We can see statistical results of scenario 2, FS2 for ALL set in Figure 2.23. They are 94.34% for RC, 94.34% for PR and 94.23% for Acc.

2.4.3 Performance of the system based on Hidden Markov Model

The performance of the algorithm GMM-HMM evaluated with three different test sets is depicted in Figure 2.24 and Table 2.7. The overall performance of the proposed algorithm is quite high in most scenarios: (i) the best result obtained under the WM test, due to similarities in action styles and environment conditions between training and testing; (ii) in the MM test, because there are differences between camera angle and brightness of the test environment, performance is reduced to about 86%; (iii) with many actions performed naturally in daily life, there are several practical challenging situations happened such as: object is obscured by other static objects in the room, new object is added into the room background, the light is suddenly changed, falling directions do not match with the classified directions assumed during the algorithm development, etc. Thus, this test provides lower recognition rates than the others, but still acceptable.

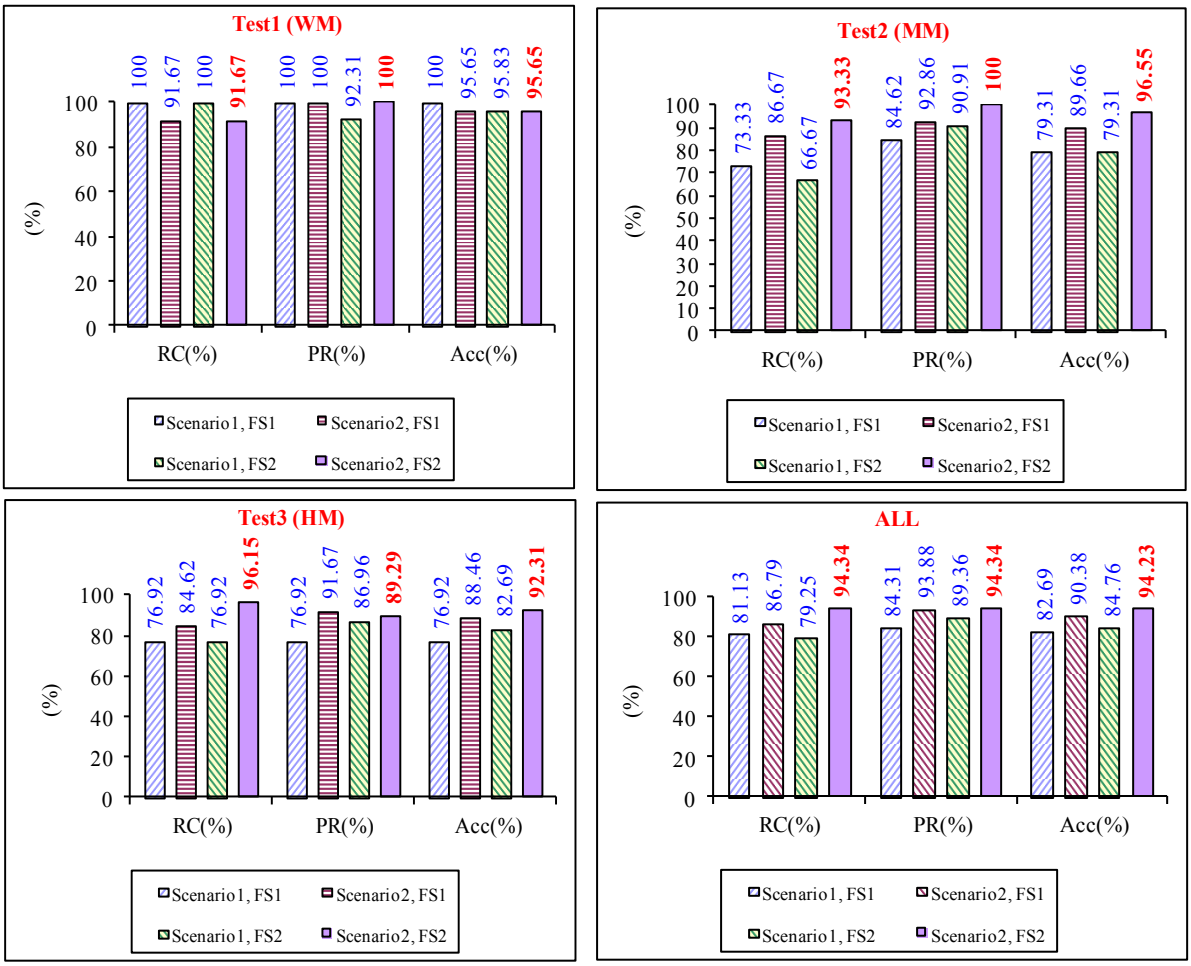
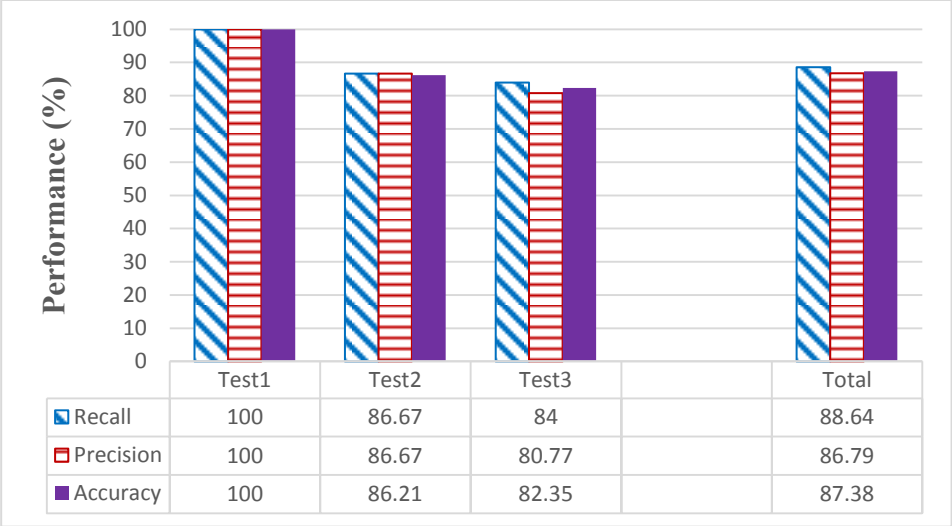


Figure 2.23-Evaluating three tests for four different models



	Test1	Test2	Test3		Total
Recall	100	86.67	84		88.64
Precision	100	86.67	80.77		86.79
Accuracy	100	86.21	82.35		87.38

Figure 2.24-Statistical results following to Recall (RC) [%], Precision (PR) [%] and Accuracy (Acc)

As detailed in Table 2.7, performance of the proposed algorithm varies correspondingly with object vision ability of camera. This system is good at cross or side falling actions. For non-fall activities, some actions such as bending or sitting on chair are usually recognized exactly, and other actions such as sitting on floor, lying or creeping are confused sometimes.

2.4.4 Comparison these methods

Beside the proposed algorithm GMM-HMM as described above, we also develop another algorithm using BGS for Object Segmentation and HMM for detection (named as BGS-HMM). Together with these two algorithms, our previous published work [26] based on BGS and Neural Network (BGS-NN) algorithm and the method using BGS and template matching (BGS-TM) reported in [77] are reprogrammed and comparatively evaluated on the same test database above. Their performances are depicted in Figure 2.25, we mention the following comments:

- While the BGS-TM method provides rather good results for the WM condition (95.65%), it performs worse under more difficult scenarios MM (55.17%) and HM (66.67%). Because of using the constant thresholds, this template matching method causes higher false alarm and lower recognition rate.
- The hard threshold problem has been solved by training NN and HMM in this study. This evident is shown in Figure 2.25b and Figure 2.25c. Here, the results derived from the BGS-NN are slightly higher than the ones obtained by the BGS-HMM. We assume this due to the algorithm used larger amount of data for training NN. Meanwhile, the HMM was trained with fewer training data [26].

Table 2.7-Classifier of videos according to activities

Database				**Fall**		**Non-fall**			**Sum**
			Test1	Test2	Test3	Test1	Test2	Test3	
Fall	*Direct*		4	4	7	0	2	2	19
	Cross		4	4	6	0	0	2	16
	Side		4	5	8	0	0	0	17
Non-fall	*Bending*	Direct	0	0	0	1	1	1	3
		Cross	0	0	0	1	1	1	3
		Side	0	0	0	1	2	2	5
	Creeping	Direct	0	0	1	1	1	0	3
		Cross	0	0	0	1	2	1	4
		Side	0	1	0	1	0	1	3
	Lying	Direct	0	0	0	1	1	0	2
		Cross	0	0	1	1	1	1	4
		Side	0	0	0	1	1	2	4
	Sitting	Direct	0	0	0	1	1	1	3
		Cross	0	1	0	0	0	2	3
		Side	0	0	0	1	1	1	3
	Others		0	0	2	0	0	9	11
Sum			12	15	26	11	14	25	103

- By comparing performance of the GMM-HMM algorithm with three other methods, we can see the positive affect of using adaptive background Gaussian model. This method deals with lighting change by slowly adapting the Gaussian parameters. By adapting the old background with new added static object, it helps to remove obstacle objects from the real segmented object. The slow moving objects are therefore also eliminated from the foreground estimation.
- While, only pure data (good brightness, static background, etc...) in training the HMM system is used in this study, the NN actually exploits much more noisy training data including many different environments for learning purpose. However, excepting the WM condition, its performance is less than the one of our proposed algorithm in both MM and HM scenarios. We assume this

improvement due to the contribution of adaptive GMM in object segmentation. With Gaussian Mixture Model, interest objects are well segmented which leads to better extracted features [26].

- The last comment is good balance between RC and PR scores of the proposed GMM-HMM method while this cannot be achieved by other algorithms.

2.4.5 Analysis of error recognition

The fourth model is GMM-HMM as shown in Figure 2.25 gives the best results, but it is not the optimum. The causes of the error recognition come from many aspects. A part is from the algorithm and the others are from the objective conditions such as poor light conditions, more than one object moving at the same time, or bad extracted features chosen to distinguish the actions. We will review many different aspects to analyse advantages, shortcomings and orient optimization solutions in recognition.

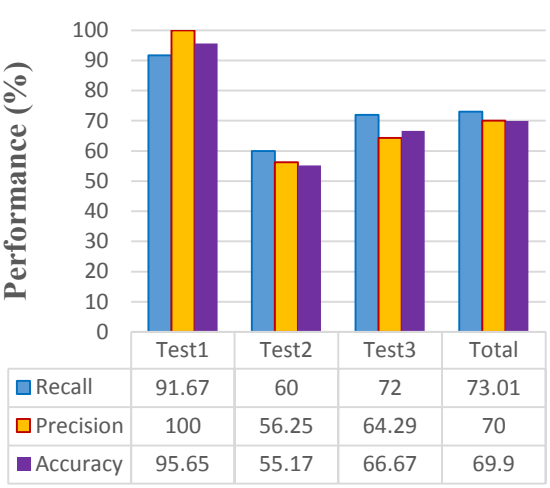
2.4.5.1 False extraction objects

a) Environment's brightness

To correct the interference caused by poor brightness conditions such as weak or sudden changeable light intensity, we need to use better methods at extracting objects under less influence of light. Other cases as clothes of object coincide with the background colour that will make extracted object lost some parts. So, this lets the characteristics of the activity caused false identification system as described in Figure 2.26.

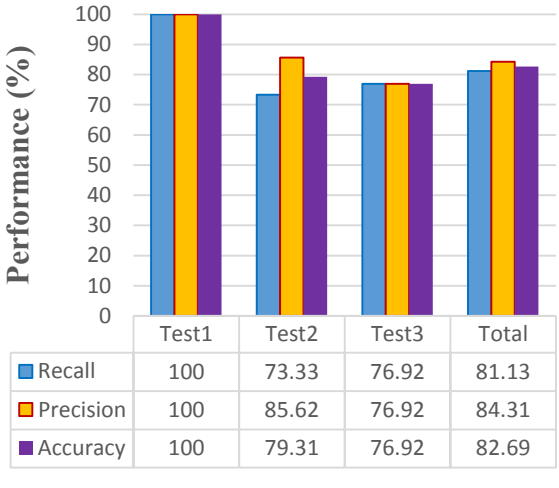
b) Object is obscured

One of the problems in the stage of extracting objects that are obscured some parts of theirs body by preventing between objects and camera in the room as illustrated in Figure 2.27 This is a very difficult problem to overcome if one camera is only used. Some researchers have suggested solutions like hanging camera on the ceiling to avoid occlusion. But it is difficult to distinguish the action falls to the sitting down action or lying down on floor with this solution.



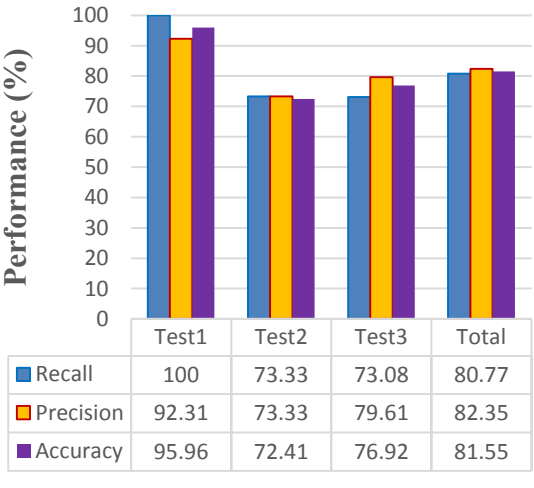
	Test1	Test2	Test3	Total
Recall	91.67	60	72	73.01
Precision	100	56.25	64.29	70
Accuracy	95.65	55.17	66.67	69.9

a)



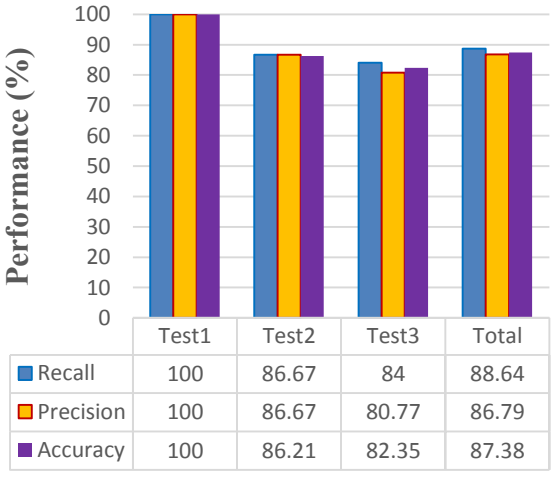
	Test1	Test2	Test3	Total
Recall	100	73.33	76.92	81.13
Precision	100	85.62	76.92	84.31
Accuracy	100	79.31	76.92	82.69

b)



	Test1	Test2	Test3	Total
Recall	100	73.33	73.08	80.77
Precision	92.31	73.33	79.61	82.35
Accuracy	95.96	72.41	76.92	81.55

c)



	Test1	Test2	Test3	Total
Recall	100	86.67	84	88.64
Precision	100	86.67	80.77	86.79
Accuracy	100	86.21	82.35	87.38

d)

Figure 2.25-Recognition performance derived from

(a) BGS-TM (b) BGS-NN (c) BGS-HMM (d) GMM-HMM

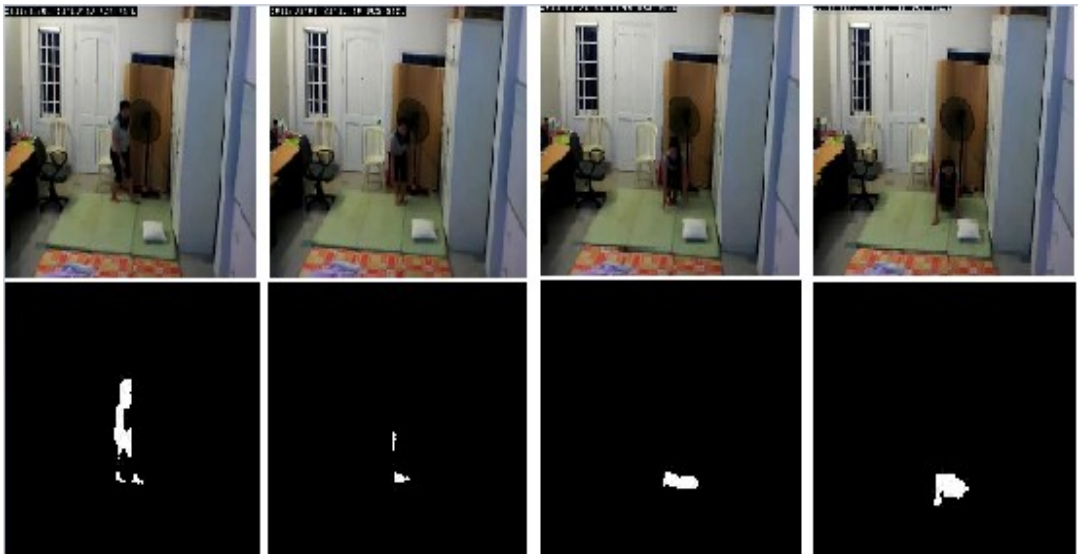


Figure 2.26-Extracted object under not good environment's brightness

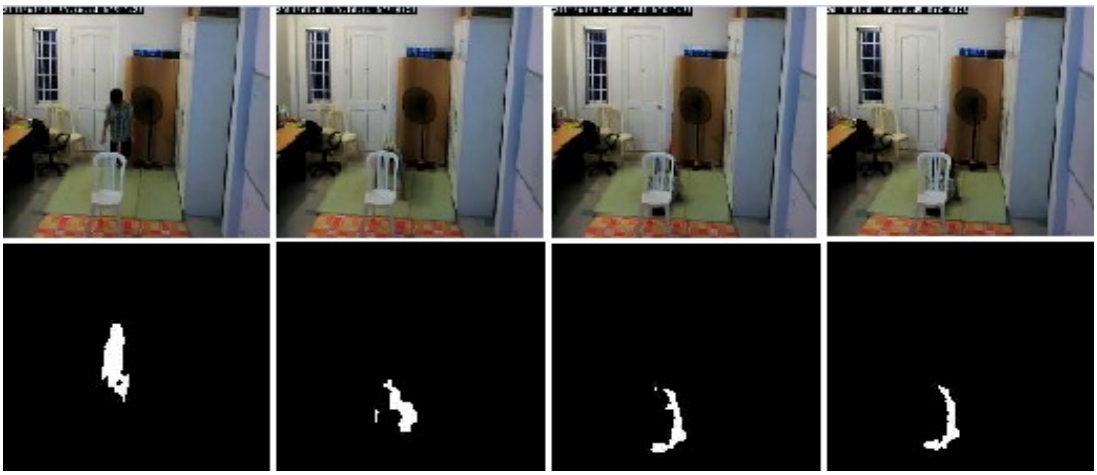


Figure 2.27-Object is obscured some parts of body

c) Many movement objects appear in the frame at the same time

Extracted Object method based on background subtraction only works well in the case of a static background with one moving object as suggest at the beginning of this Chapter. For the video has other objects moving with complex routine, the quality of the output of background subtraction stage is very bad and hard to track the object as shown in Figure 2.28. To distinguish the supervision subject with other objects which move at the same time, we should use the more complex methods as follows tracking multi-points on the body or Stick-figures technique instead of silhouette.

Figure 2.28-Many objects are moving at the same time

2.4.5.2 Extracted features

The 2D model estimated from a camera in this thesis will not show the depth of the object; hence, the five selected features are used to classify the actions. Figure 2.29 shows the values of extracted features of face fall action and sitting down on the floor and continuously stand up to go.

Obviously with two different actions, face fall and sitting actions, the five features are very similar, thus the system cannot distinguish the two actions and it causes misclassification. It means that the five extracted attributes are not sufficient to distinguish the fall actions given in this thesis.

From the above analysis, we found that this solution has to be further processed. However, it is basically solved the problem of fall recognition. The algorithms achieve high effective due to the large of recognition sample set. The purity video gives right results with higher rates. It is necessary to consider the other problems such as the obscured object, silhouette of object, etc. in identifying the real situation.

2.5 General discussion

The experimental results show that the performance of this proposed algorithm is quite high and robust even in conditions such as lighting changes, added background, varied camera vision, obscured objects, long-term scene changes, etc. Although the final result of the computer-based Fall Detection System is quite good, its design faces some major challenges discussed in this section.

2.5.1 Performance under real-life conditions

In Fall Detection System, accuracy and reliability are two major criteria which should be improved as much as possible. This is easier to achieve in experimental environments under controlled conditions. However, the detection rate perhaps decreases when applied to a real situation [86]. Especially, our database use data recorded from falls of young people simulated at the discretion of each impersonator in the videos. So, our database lacks of a standardized procedure or compares with a public database. Meanwhile, the real fall detection aim to older people or patients who have some distinctions with young people in the database. There are only few studies incorporated data from older people [87][88], but their participation is limited to perform a set of simulated activities of daily living for a few minutes or hours. That is not enough to assess the system performance in a real situation.

2.5.2 Usability

The fall detection based on camera is particularly implemented at certain areas, namely small hospitals or nursery homes where a small surveillance area and not many people moving simultaneously have. Moreover, this system still limits to become an online system which can capture a moving object, train itself and test at the same time. Furthermore, the most important restriction here is that this system is not wearable, which is more convenient and applicable at the contemporary time.

2.6 Conclusion

In this chapter, the algorithms used in Object Segmentation, Filter, Feature Extraction and Recognition module of the Fall Detection System is described. Moreover, we introduce the speciality of DUT-HBU database which is used for evaluation our proposed algorithms in this Chapter and also in other Chapter. We assess the describe Fall Detection System by using the accuracy, recall, precision performances. The comparison is then shown simulation results between these algorithms such as BackGround Subtraction/Hidden Markov Model (BGS-HMM); Gaussian Mixture Model/Hidden Markov Model (GMM-HMM); BackGround Subtraction/Neural Network (BGS-NN); BackGround Subtraction/Template Matching (BGS-TM). Therefore, we select the BGS-TM algorithms for implementing our system on processor cores and FPGA. Finally, we analysis some shortcomings of these techniques to give the false recognition output related environment brightness, occlusion of object, multi objects, etc.

In the next Chapter, we characterise the algorithm and architecture parameters to model the execution time and power consumption of video processing applications which support for exploring the low cost architecture of Fall Detection System on both processor cores and FPGA. For the implementation aim, each modules of Fall Detection

System is considered as a task: Object Segmentation, Filter, Feature Extraction and Recognition.

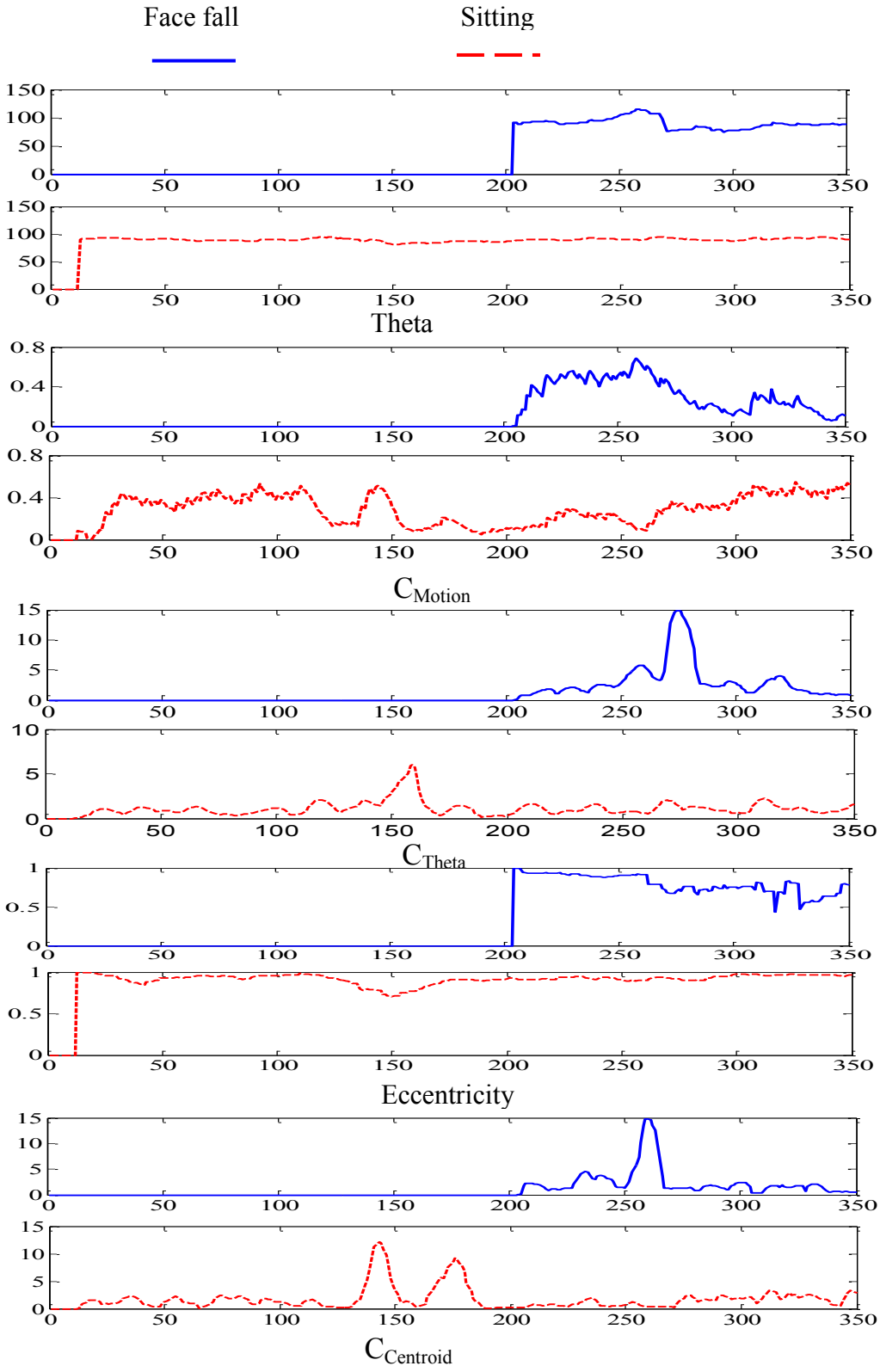


Figure 2.29-Comparison the five features of face fall and sitting action

Chapter 3. Power and Time Model Methodology for Fall Detection System

Various design methods have been proposed to estimate power from low level to high levels that requires design information in language such as Verilog/VHDL/SystemC/C. However, the more detailed the model is the slower the simulation and thus the slower the estimation. Therefore, to improve this process, it is desirable to define a power estimation methodology.

Consequently, the target of this Chapter is on defining an efficient power model to estimate the power consumption of video applications in the Fall Detection System. We also extract the execution time model by this way. In general, the aim of power estimation methodology mentions about the speed and accuracy. In our work, we target accuracy based modeling style and analysis information collected from measurement on real board to obtain sufficiently accurate power estimation for the Fall Detection System on heterogeneous platform. Therefore, we experiment and verify the model's accuracy on Zynq-7000 AP SoC platform, to show the applicability of our model.

The following of this chapter is structured as follows: Section 3.2 and 3.3 describe the power consumption modeling approaches and execution time estimation approaches. The discussion for selecting the suitable heterogeneous platform such as Zynq 7000 AP SoC to implement the Fall Detection System is clarified in Section 3.4. Section 3.5 presents the method to measure the power/execution time on processor cores and FPGA for our system. Then, the proposed power model of the Fall Detection System on heterogeneous platform is illustrated in Section 3.6. The next Section 3.7 is dedicated to the definition of execution time models for heterogeneous platform.

3.1 Power and energy consumption characterization and estimation in MPSoC

In today's high-performance system designs, power consideration is becoming increasingly important leading to develop the power consumption in MPSoC at early step of the design. Understanding source of power consumption allows designers to configure on MPSoC environment to minimize power consumption, successfully meet a given power budget to maintain a reliable product and also sufficient the recognition rate.

The power consumption of CMOS circuit consists mainly of dynamic and static power.

$$P = P_{static} + P_{dynamic} \tag{3.1}$$

In a CMOS device including n transistors, the static power consumption, P_{static}, is calculated as a function of the number of transistors, the leakage current I_{lkg_i} of each transistor and the supply voltage V_{dd}. It is represented by equation 3.2

$$P_{static} = \sum_{i=1}^{n} I_{lkg\_i} . V_{dd} \tag{3.2}$$

Actually, when no switching activity occurs, transistors in CMOS circuits lose an amount of current that can be negligible or not according to the architecture and the technology of the circuit. The static power, *Pstatic,* in our work is the additional continuous power dissipation when the device is configured and there is no activity.

The dynamic power dissipation, $P_{dynamic}$, of a CMOS circuit is depicted by an approximate relation given by equation 3.3 which includes the operating frequency *f*, the supply voltage V_{dd} and the total load capacitance of all gates C_L. Where α is the average switching activity of the component.

$$P_{dynamic} = \propto . f . C_L . V_{dd}^2 \tag{3.3}$$

Dynamic power can be an important source of power dissipation and is considerably dependent on the application and the architecture of processor cores or FPGA.

In addition, the energy consumed by a system is the number of power dissipated during a certain period of time. For instance, if a task T is running on a MPSoC during an execution interval of T: [a, b] then the energy consumed by the MPSoC during this time interval is given by equation 3.4:

$$\mathrm{E_T} = \int_{\mathrm{a}}^{\mathrm{b}} \mathrm{P(t)}\ \mathrm{dt} \tag{3.4}$$

To define a methodology of power/energy consumption estimation, it is necessary to characterize the variation of power/energy consumption on hardware (FPGA) and software (processor) parts separately or the combination of heterogeneous solution. The power/energy estimation is a process to evaluate of the power consumption of a design. The aim is to check whether power and reliability constraints are met or not. After defining the power consumption estimation, it is necessary to explore architectures

with the low cost power/energy consumption for the Fall Detection System. Generally, the estimation methods are based on the followings [89]:

- Power estimation is based on the simulations of the embedded system and its results depend on the activities/toggles of the design.
- Power estimation using mathematical models shows the dependence of power consumption of the embedded system on certain parameters. These parameters can be static such as the processor frequency, the number of cores, the memory size, etc. or dynamic such as cache miss rate, pipeline stall, instruction per cycle, etc.

In the following Section, we review the state of the art of power consumption modeling approaches on embedded systems at different abstraction levels.

3.2 Power consumption modeling approaches

In this Section, we present an overview of recent approaches to model the power consumption. The power consumption models can be distinguished into two main categories [90]:

- Low-Level models.
- High-Level models.

Low-level models calculate the power from detailed electrical descriptions: circuit levels, gate level, register transfer level (RTL) and architecture level. The current low level tools are such as SPICE [91] at the transistor level, Diesel [92] at the gate level and Petrol [93] at the RTL level, which deal with fine-grained activities. The simulation time of these tools depends on circuit size and circuit complexity. Thus, it is rather difficult in application for complex MPSoC. Notwithstanding, the tools supply a good accuracy, but it is not always practical to implement in the early design flow as they require generally a deeper knowledge of the circuit. While, high-level models deal with instructions and functional units of the programs with less architectural and technology knowledge [90].

3.2.1 Low-level power consumption estimation techniques

Both accuracy of the estimation and speed of the simulator compromise the efficiency of the power simulator. There are many power simulators available in the industry and also in academic research centers. In this subsection, we survey the estimation techniques of the power consumption frequently used at lower levels. The low-level power consumption estimation techniques cover a wide range of abstractions levels such as the:

- Circuit/Transistor level estimations;
- Logic gate level estimations;
- RTL estimations;

- Architectural level estimations;

Circuit-level estimations

At this level, the processor cores are presented in terms of transistors and nets which also require undergoing all the steps in the design flow. Furthermore, the circuit-level of the system uses element models which are based on linear differential equations [94] and works in continuous time domain. This implies that a simple simulation for a small set of transistors requires a large number of time which is not always available in this fast moving industry and not practical for complex SoC [95]. PowerMil [96] is an early attempt to build a low-level power consumption simulator. This tool is used for simulating current and the power characteristic in VLSI circuits. It is also accomplished to simulate detailed current behavior in modern deep sub-micron CMOS circuits, including sophisticated circuits such as sense-amplifiers, with speed and capacity approaching conventional gate level simulators [97]

Gate-level estimations

The description about the gate-level method to estimate the power consumption as example of processor cores is presented in this subsection. The main advantage of these methods with respect to circuit-level simulation method is that the simulation is event-driven and takes place in a discrete time domain which considerably reduces the computational complexity, without any significant loss of accuracy [95].

In [97], Subodh Gupta and Farid N. Najm propose an automatically generation of 3-dimensional table to estimate the power consumed in circuit for a given statistics. Their power model is constructed with three variables: average input signal probability, average input transition density and average output zero-delay transition density. A novel and significant aspect of this approach is that they extend the same model for all types of combinational gate-level circuits and requires no user intervention. Another important fact is that their model works gives very good accuracy, with a Root Mean Square error (RMS) of under about 6%.

Besides, Ding Chih-Shun et al. [98] present an accurate and efficient gate-level power estimation technique called tagged probabilistic simulation (TPS). TPS is based on the notion of tagged (probability) waveforms which divide the logic waveform space into a small number of disjoint parts and then represents all the logic waveforms in a part by a probability waveform. The advantage of this simulation strategy is that the correlations among circuit internal nodes (referred as logic gates) can be effectively accounted for. In [99], S.T. Oskuii et al. extend this technique by using a novel waveform set method. Previous method has local glitch filtering approaches that fail to model this phenomenon correctly. Glitches originated from a node may be filtered in some, but not necessarily all, of its successor nodes. Their technique allows modeling the removal of glitches in more detail by using a global glitch filtering.

RTL estimations

Most of the RTL designs are illustrated as a collection of blocks and a network of interconnections. The blocks, sometimes referred to as adders, registers, multiplexers and macros (a complex functionality block including among of register, adders, etc.), while the interconnections are simply nets or group of nets. Most of the tools presented in the literature follow similar pattern like the power properties of the block can be derived from an analysis of the block isolated from a design, under defined conditions. The main factor influencing the power consumption model of a macro is the input statistics [95].

Most of the research in RTL power estimation is based on empirical methods that measure the power consumption of existing implementations and produce models from those measurements. There is another approach which is widely used by RTL designers is based on measurement for estimating the power consumption of data-path functional units. Liu et al. [100] develop a method and a tool for power modeling to estimate the power consumption of different parts of a chip such as logic gate, local and intermediate interconnection, memory size. However, this tool is not as accurate as gate level simulators, it gives a fast estimation before circuit and layout design. In the other sides, the method power consumption estimation is based on predictable input signal statistics proposed by Landman and Rabaey [101] with a quite accurate (10% to 15% error rate). The feasibility of this method depends on correct input statistics or the ability to correctly model.

A methodology for creating power macro models bases on linear regressions but their flow is specific to the structural RTL macros and power estimation performed at the gate-level is proposed by Bogliolo et al.[102]. They analyze the application of linear and nonparametric regression for the automatic construction of RTL power macro models for registers and combinational logic blocks (called macro). Their approaches are focused on: off-line and online characterization. For off-line characterization, the power of RTL macro is based on tests. And they adaptively do online characterization for error minimization. Continuously, Qing Wu et al. [103] propose macro-model predicted not only the cycle-by-cycle power consumption of a module but also the moving average of power consumption and the power profile of the module over time for RTL. The authors introduce a power function and approximation steps to generate the power macro-model. Potlapally et al. [104] present another technique related RTL circuit that are the cycle-accurate power macro modeling. Their techniques are based on RTL components demonstrate different power behavior for input vectors at different cycle. They create power macro model for each of these behaviors also known as power modes. Their design flow selects an appropriate power mode given from the input vector in each cycle and then applies power macro model techniques.

Architectural-level estimations

In the previous abstraction of low level estimation, the literature discuss about the power consumption estimation on processor cores and FPGA. This subsection describes the state of art for power consumption estimation based on architectural-level estimation. The simulators derive power estimates from the analysis of circuit activity induced by the application programs during each cycle and from detailed capacitive models for the components activated. A major difference between these simulators is the estimation accuracy and estimation speed. SimplePower tool [105], works with a transition-sensitive power model for the data-path functional unit. The SimplePower core accesses a table containing the switch capacitance for each input transition of the functional unit exercised. The use of a transition-sensitive approach has both design challenges as well as performance concerns during simulation [94]. The first concern is that the construction of these tables is time consuming. Unfortunately, the size of this table grows exponentially with the size of the inputs. The table construction problem can be addressed by clustering algorithm [106] and partitioning mechanisms [105]. Further, not all tables grow exponentially with the number of inputs. The second concern is the performance cost of the lookup table for each component access in a cycle. In order to overcome this concern, simulators like SoftWatt [107] and Wattch [108] use a simple fixed-activity model for the functional units. These simulators only track the number of accesses to a specific component and utilize an average capacity value to estimate the power consumed. In contrast to the datapath components that use a transition-sensitive approach, the models estimate the power consumed per access and do not accommodate the power differences found in sequences of accesses.

One of the most widely used another tool in architectural domain is Wattch [108]. Wattch tool is used for superscalar processor. The base infrastructure is offered by SimpleScaler [34] for this tool. SimpleScaler carries out fast, flexible and accurate simulation of modern processors that implement a derivative of MIPS architecture. In addition, it also supports detailed cycle accurate information for all models, including datapath elements, memory and Content Addressable Memory (CAM) arrays, control logic, and clock distribution network. Wattch uses activity-driven, parametrisable power models, and it displayed accuracy better than 10% when tested on three different architectures. Energy measuring tools can be either transition-sensitive or based on analytical formulas. Since transition-sensitive simulators estimate the energy consumption based on bit-switching activities, they take a significant amount of time to generate energy estimates (for example, SimplePower). In [109], the other approaches to evaluate energy estimates at the architectural-level for memory system called Virtual Energy Counters (vEC) tool by the following formula:

$$Energy = E_{bus} + E_{cell} + E_{pad} + E_{main} \tag{3.5}$$

Where, E_{bus} represents data and address bus energy between processor and cache; E_{cell} represents cache energy; E_{pad} represents data and address pad energy between cache and main memory; and finally E_{main} represents the main memory energy.

vEC provides a user interface to estimate the energy consumption for memory system. The energy estimates are provided for those consumed in the data, instruction and extended caches, main memory, address bus, data bus, address pads and data pads. Energy estimations for instruction number and clock cycles are also supported by vEC.

System for Early Analysis of SoCs (SEAS) [110] proposes a methodology for analysis SoCs in early design stage. SEAS provides integrated algorithms which use for testing the performance, floorplan, timing and power. Power evaluation in this system works at a granularity of processor cores, where pre-characterized data for power is used based on the power state of the design. Power states of the cores are based on active, idle or sleep states of cores. By extracting the power values for the states of cores, users can estimate the average power for the whole system in early stage with high accuracy.

3.2.2 High-level power consumption estimation techniques

The accurate power estimation at high level takes a significant role in any successful design methodology. Many researchers are interested in extending this area because of increasing the complexity of the MPSoC's architecture. In this section, we present some high level power estimation approaches which consist of spreadsheet, Instruction Level Power Analysis (ILPA) and Functional Level Power Analysis (FLPA).

3.2.2.1 Spreadsheet based approaches

In the early stage of design process, spreadsheets are determined for the initial planning to take some important decision [111]. The users are not necessary to learn any complex/sophisticated tool in order to get design decisions based on spreadsheet approach. One of the basic applications of spreadsheet is area estimation. We can get estimation values on area by using data sheets from intellectual property (IP) providers, library cell estimate, etc. Spreadsheet supplies an ability to capture such information that can be used for quick area estimation. In [112], the designer can find out some decisions to control power based on spreadsheet approach. Power budgeting approaches using spreadsheets are very essential for printed circuit board (PCB), power supplies, voltage regulators, heat sink, and cooling systems.

A spreadsheet tool is as example for Xilinx Power Estimator (XPE) [113]. In industry, the power estimation for programmable devices with a complex process and architecture like FPGAs needs to be done very efficiently. To produce accurate estimates, the power estimation process requires reliable input values, such as resource utilization (e.g. flip-flops, look-up tables, I/Os, block RAMS, DCMs, etc.), clock rates and toggle rates.

In fact, XPE uses your design and environmental input, and then combines this information with the device data model to compute and present an estimated distribution of the power in the targeted device. XPE presents the following power types:

- Power by voltage supplies is useful information to select and size power supply components such as regulators, etc. Supply power includes both off-chip and on-chip dissipated power.
- Power by User logic resources allows users to experiment with architecture, resources, and implementation trade-off choices in order to remain within the allotted power budget.
- Thermal power is the expected thermal properties of the device. It helps the users to evaluate the necessary for passive or active cooling for a design.

In addition, Power Estimation Tool (PET)[3] provides users the ability to gain insight in to the power consumption of select Texas Instrument (TI) processors such as OMAP35x, AM35x and AM335x Processors. The tool includes the ability for the user to choose multiple application scenarios and understand the power consumption as well as how advanced power saving techniques can be applied to further reduce overall power consumption.

Spreadsheets are fast, flexible, and generally well understood. Unfortunately, the disadvantages are also applicable-error prone nature, wide accuracy variance, and manual interface. Nonetheless, spreadsheets such as Microsoft's Excel are used to model entire systems. System components and sub-blocks are modelled with customizes equations using parameters such as supply voltage, operating frequency, and effective switched capacitances. Technology data may or may not be explicitly parameterized, but it is typically derived from data-book information published by the technology vendors. Spreadsheets are most often used for project planning but may not be able to provide accurate guidance for block-level hardware power estimation and reduction. This motivates a need to provide a power model, which can perform accurate yet efficient power analysis at early stage of the design. Thus, it is necessary to define another estimation method of power consumption on processor cores, hardware. The next subsection provides an overview the power model based on approaches for power estimation purposes such as Instruction Level Power Analysis (ILPA) and Functional Level Power Analysis (FLPA).

[3] http://www.ti.com/tool/powerest

3.2.2.2 Instruction Level Power Analysis (ILPA)

An instruction level power model for individual processors was first proposed by Tiwari et al. [114][115]. The total energy consumption of a program E_p is expressed with the following equation:

$$E_p = \sum_i (B_i \, x \, N_i) + \sum_{i,j} \left(O_{i,j} \, x \, N_{i,j} \right) + \sum_k E_k \tag{3.6}$$

The following factors contribute towards the energy cost of a program:

B_i - Base cost for each instruction *i*.

N_i - Number of execution times for each instruction *i*.

$O_{i,j}$ - Circuit state change overhead for each instruction pair *(i,j)*.

$N_{i,j}$ - Number of execution times for each instruction pair *(i,j)*.

E_k - Energy cost for other inter-instruction effects (stalls, cache misses etc.).

In[114], Tiwari et al. measure the current drawn by the processor as it repeatedly executes distinct instructions or distinct instruction sequences, it is possible to obtain most of the information that is required to evaluate the power consumption of a program for the processor under test. The authors model the power consumption of the Intel DX486 and Fujitsu SPARClite 934 processor. Power is modeled as a base cost for each instruction plus the inter-instruction overheads that depend on neighboring instructions. The base cost of an instruction can be considered as the cost associated with the basic processing needed to execute the instruction. However, when sequences of instructions are considered, certain inter-instruction effects come into play, which are not reflected in the cost computed solely from base cost. This effect and others can be summarized as the following:

a) **Circuit state**: switching activity depends on the current inputs and previous circuit state.

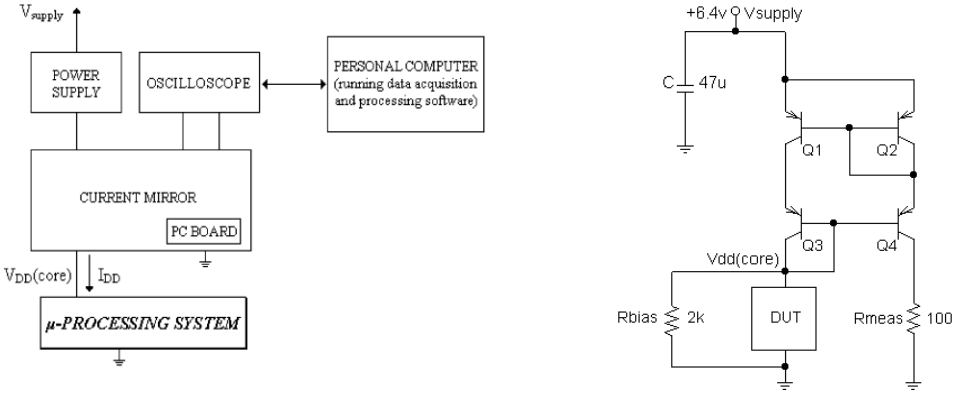
b) **Resource constraints**: resource constraints in the CPU can lead to stalls, for instance, pipeline stalls and write buffer stalls.

c) **Cache misses**: the instruction timings listed in manuals provide the cycle count assuming a cache hit. For a cache miss, a certain cycle penalty has to be added to the instruction execution time.

During executing, certain instruction sequences which these effects occur may provide a way to isolate the power cost of these effects. Thus, the total of the power costs in each instruction that is executed in a program enhanced by the power cost of the inter-instruction and other effects can be an estimate for the power cost of the program.

Much more accurate measuring environments have been proposed to precisely monitor the instantaneous current drawn by the processor instead of the average current. One of these approaches has used current mirror, based on bipolar junction transistors as

current sensing circuit as shown Figure 3.1. The Instruction level power models in the work of Nikolaidis et al. [116] are derived by measuring the instantaneous current drawn by the ARM7 TDMI processor at each clock cycle. Their model is traced by executed assembly instructions, generated by an appropriate processor simulator and estimated the base and inter-instruction energy cost of the executed program taking into account the energy. Niloladies et al. improve their power model by using the energy sensitive factors as well as the effect of pipeline stalls and flushes in [117]. This method is developed for pipelined processors like the ARM7 (three-stage pipeline: instruction fetch, instruction decode and instruction execute). Another approach, to reduce the spatial complexity of instruction-level power models, is also presented in their work in relation to a reference instruction as No Operation (NOP). The main drawback of this method is the complexity in measurement the current. More researchers attempted to enhance the original Tiwari ILPA power consumption modeling technique as in[118][119]



a) b)

Figure 3.1-(a) Experimental Setup for current measurement, (b) The simple current mirror. DUT is the Device Under Test [116]

The ILPA-based methods have some disadvantages, one of these disadvantages is that the number of current measurements is directly related to the number of instructions in the Instruction Set Architecture (ISA) and also the number of parallel instructions composing the very long instruction in the VLIW processor. The problem complexity of instruction level power characterization of *K*-issue VLIW processor is $O(N^{2K})$ where N is the number of instructions in the ISA and K is number of parallel instructions composing the VLIW [120].

This technique helps to evaluate the power cost of embedded software and verify specified power constraints if a design meet. Moreover, it is also use to search the design space in software power optimization.

3.2.2.3 Functional Level Power Analysis (FLPA)

In order to overcome the shortcoming of ILPA, J. Laurent, N. Julien et al., first introduce Functional Level Power Analysis (FLPA) method in [121]. The functional level power modeling approach is applicable to all types of processor architectures. Furthermore, FLPA modeling can be applied to a processor with moderate effort, and no detailed knowledge of the processors circuitry is needed. This approach is based on a functional analysis of the core of processor to determine a set of consumption rules. The way of which interactions between functional blocks induce power consumption depends on several identified parameters which called architecture parameters are configuration and algorithmic parameters as shown in Figure 3.2. Their functional analysis presents a very efficient and straightforward method for energy optimization. The error rate between estimation and measurement is not higher than 7.4% for their considered application and architecture. The result of this method is applied to a FIR 16 filter on a TMS320C6201 DSP and has extended to other processors.

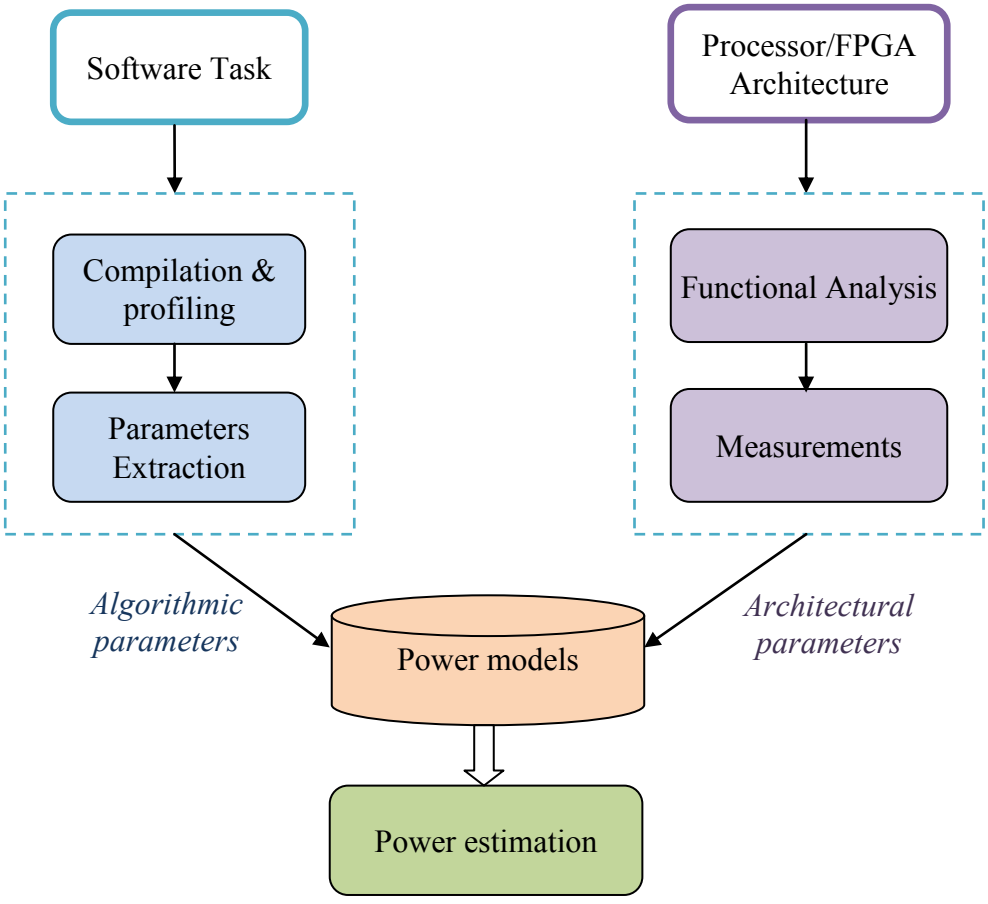


Figure 3.2-The Functional Methodology[121]

In the previous work [121], the estimation are already validated at the assembly level by direct comparison with measurements. Then Eric Senn, N. Julien et al. [122] apply the same process for the C-level. In this method, the identification of a set of

functional blocks will influence on the power consumption of these components such as Processing Unit, Instruction Management Unit (IMU), internal memory and others. First, a functional analysis of these blocks is performed to specify and then discard the non consuming blocks (those with negligible impact on the power-consumption). The second step is to figure out the parameters that affect the power consumption of each of the power consuming blocks. For instance, the IMU is affected by the instructions dispatching rate which in turn is related to the degree of parallelism.

In addition, the relevant consumption parameters are chosen as the significant links between the blocks. There are two types of parameter: algorithmic parameter values depend on the executed algorithm (such as the cache miss rate, parallelism rate, processing unit rate, external memory access rate and Direct Memory Access rate) and architectural parameter values depend on the processor configuration resolved by the designer (typically the clock frequency, word length of input data, memory mode) [123]. The model is shown by a set of analytical function or a table of consumption values that accord to functional and architectural parameters. As the model is established, the estimation process includes extracting the appropriate parameter values from the design, which will be injected into the model to compute the power consumption.

The SoftExplorer tool is developed based on this method [124]. This tool realizes the suitable trade-off between the estimation accuracy and time in order to ensure a rapid and reliable feedback to the designer. The SoftExplorer tool is allowed to estimate the power consumption of algorithm which is developed on C code[125]. This tool also is used to optimize the power consumption of an application. Eric Senn et al. show a functional level power analysis to extract the different power models and illustrate how to perform the best data mapping for an application. This methodology is implemented on various processor such as ARM7, the low-power (C55) and Very Large Instruction Word-VLIW (C62) processor [126]. Moreover, these crucial phenomena like pipeline stalls, caches misses, and memory accesses are applied. The recently work of M. E. A. Ibrahim et al. [94] present a precise high-level power estimation methodology for the software loaded on a VLIW processor based on a FLPA. Their targeted processor is the TMS320C6416T DSP from Texas Instrument.

In the work of S. Rethinagiri et al. [127], they extend the FLPA to create generic power models for the different target processors such as ARM processors (ARM9, ARM Cortex-A8 and ARM Cortex-A9 processors), DSP processor, Heterogeneous multiprocessor (OMAP5912 and OMAP3530) under test. Their estimation of power and energy results provides a maximum error of 5% for mono-processor and 9% for heterogeneous multiprocessor based system when compared against the real board measurements.

For these reasons, in context of our work, we apply this methodology to generate power models at high level estimation. Then, the FLPA methodology is used to establish the power model for different components of video processing in the Fall Detection

System on processor cores. This approach is extended in extracting the power models for software (processor cores), hardware (FPGA) and heterogeneous architecture. Moreover, we consider several important issues in our model. The main contributions are as follows:

First, precise models are defined to estimate the power consumption of the targeted processor cores for the Fall Detection System related these basic parameters such as number of cores, instruction per cycle, frequency of cores, caches miss rate, etc. and for FPGA. We, then, prove the validation and accuracy of our model for two technology target: processor cores and FPGA. After extracting the power consumption models for the Fall Detection System, we continuously to discuss some approaches in estimation of execution time for calculating the energy of these exploration architectures as following Section.

3.3 Execution time estimation approaches

Execution time estimation is important for designing processor cores. It provides the basic suggestion for selections of core of processors and other hardware components for the systems. It is also necessary to manage resource unit when scheduling program execution to meet the design constraints (such as efficient of energy, real-time) and to optimize the system performance and any other optimization goal. Estimating a program execution time is particularly critical in design of real-time systems [128]. Real-time systems require more than delivering accurately produced computational results. They also require tasks to meet their deadlines because, for applications such as Fall Detection System, process control, flight control, avionics, defense systems, vision and robotics, pervasive and ubiquitous computing, etc.

Estimation of the execution time is an important part to estimate the efficiency of our system such as energy consumption and the frame rate. The aim of this section is to highlight two types of techniques in estimating execution time: static and dynamic. Static techniques apply a structural analysis of a piece of software and analytical model of the underlying hardware to define execution time without executing the software. Besides, dynamic techniques require executing the program of interest in order to estimate the execution time of a program. Furthermore, some dynamic estimation techniques also use a (static) structural analysis of a program when estimating its execution time. Both types of techniques are described in the following subsection, and we also discuss about the tools based on these techniques.

3.3.1 Static timing estimation

Static timing analysis techniques estimate the execution time of a program without actually executing any code. They are mainly used to determine the Worst-Case

Execution time (WCET) of a program, meaning a conservative estimate or upper bound for the execution time of a program. Most existing solutions are based on static program analysis techniques to model the execution of a piece of software on a given target processor [129]. In general, they are roughly concluded three steps [130]:

a) **Control flow analysis** decomposes the structure of the program into atomic units for the subsequent analysis steps. The result of program representation is consequently the control flow graph (CFG), which is composed of basic blocks. A basic block is a maximal sequence of program statements with only one point of entry and exit.

b) **Micro-architectural analysis** determines the execution time for the atomic units of a program using the result of the control flow analysis. In most cases this analysis is performed using an abstract model of the target processor. This model can be based on abstract interpretation[131] or symbolic execution [132]. The both cases focus on the execution time for sequences of machine instructions and neglect details of the computations these instructions perform on the real hardware.

c) **Global bound calculation** uses the results of the two previous steps to obtain an estimate for the total execution time of a program. The prevalent technique for doing this is implicit path enumeration [132]. This approach translates the structural constraints and local execution time estimates into an integer linear programming (ILP) problem, which is then solved using standard ILP solvers. If a program contains loops, the maximal number of times a loop may execute must be determined by a previous analysis or provided by the user. This is necessary for the ILP solver to find a worst case path.

In addition, Theiling et al.[131] propose an approach which employs abstract interpretation for micro-architectural modeling and integer linear problem (ILP) for path analysis. In their study, they show how the micro architecture analysis is separated from the path analysis in order to make the overall analysis fast.

3.3.2 Dynamic timing estimation

The dynamic timing technique measures execution time directly on the hardware, for some set of inputs, and measuring the execution time of the task or its parts. This means that target hardware must be available. The level of granularity at which these measurements can be performed varies for different processor architectures. The counters only provide a limited level of accuracy must be taken to obtain accurate measurements, while most current processor architectures support hardware performance counters [133]. In addition, it is necessary to modify the observed program by adding instrumentation code for manipulating the hardware performance counters of the processor. The measurements are impacted on the modification of program code. To perform measurements with an increased level of accuracy, for example, up to the level

of individual instruction, additional tools like logic analyzers or processors with dedicated tracing hardware support are required.

Probabilistic and statistical timing analyses [134] are variants of dynamic approaches for execution time estimation. Probabilistic timing analysis tries to capture the variance of execution times by providing a probability distribution for the possible execution times of a program. This distribution is calculated by analyzing the execution time of individual program parts from a large set of measurements. The complete measurement process can take several days of observing the system in operation and produce gigabytes of data. Using this data, the execution time distributions of smaller program parts are incrementally combined to get the distribution for the complete program. The limitation of this combination step is that the execution time of individual program parts is assumed to be independent, which is often not the case in practice. More recent approaches for dynamic timing analysis apply statistical methods, like extreme value theory, to reason about the worst-case execution of a program without ever observing it [135]. However, this is still an area of active research without a generally accepted solution.

3.3.3 Timing estimation tools

The tool providers and researchers participating in this survey have received the following list of questions:

- What is the functionality of your tool?
- What methods are employed in your tool?
- What are the limitations of your tool?
- Which hardware platforms does your tool support?

In the following Section, we try to reply these questions for different tools.

3.3.3.1 The aiT Tool

The AbsInt Timing analyzer [136] is a timing analysis tool developed and commercialized by AbsInt Angewandte Informatik, a German company. The purpose of AbsInt's timing-analysis tool aiT is compute automatically upper bounds for the the worst-case execution time (WCET) of code design in executables. These codes may be tasks called by a scheduler in some real-time application, where each task has a specified deadline. aiT works on executables because the source code does not contain information on register usage and on instruction and data addresses. Such addresses are important for cache analysis and the timing of memory accesses in case there are several memory areas with different timing behavior. In aiT's case, value analysis and cache/pipeline analysis

are realized by abstract interpretation, a semantics-based method for static program analysis.

The aiT tool contains some limitations. This tool includes automatic analysis to determine the target of indirect calls and branches and to determine upper bounds of the iterations of loops. These analyses do not work in all cases. If they fail, the user has to provide annotations. aiT relies on the standard calling convention. If some code doesn't adhere to the calling convention, the user might need to supply additional annotations describing control flow properties of the task.

This tool supports to the following hardware platforms: Versions of aiT exist for the Motorola PowerPC MPC 555, 565, and 755, Motorola ColdFire MCF 5307, ARM7 TDMI, HCS12/STAR12, TMS320C33, C166/ST10, Renesas M32C/85 (prototype), and Infineon TriCore 1.3.

3.3.3.2 The Heptane tool of IRISA, Rennes

The Heptane [4] is an open-source static WCET analysis tool [137]. The purpose of Heptane is to obtain upper bounds for the execution times of C programs by a static analysis of their code (source code and binary code). The tool analyses the source and/or binary format depending on the calculation method the tool is parameterized to work with.

Heptane integrates mechanisms to take into account the effect of instruction caches, pipelines and branch prediction.

- Pipelines are tackled by an off-line simulation of the flow of instructions through the pipelines.
- An extension of Frank Mueller's so-called static cache simulation [138], based on data flow analysis is implemented in the tool. It classifies every instruction according to its worst-case behavior with respect to the instruction cache. Instruction categories take into account loop nesting levels.
- An approach derived from static cache simulation is used to integrate the effect of branch predictors based on a cache of recently taken branches. The modeling of the instruction cache, branch predictor and pipeline produce results expressed in a micro-architecture-independent formalism, thus allowing Heptane to be easily modified or retargeted to a new architecture.

Limitations of this tool: there are no automatic flow analysis, no detection of mutually exclusive or infeasible paths and resulting in pessimistic upper bounds for some tasks. The bound-calculation method based on timing schemata currently does not

[4] https://team.inria.fr/alf/software/heptane/

support compiler optimizations that cause a mismatch between the task's syntax tree and control flow graph. This tool doesn't support for data cache analysis and limits the number and types of target processors excepting the gcc compiler.

The hardware platforms supported for Heptane are designed to produce timing information for in order mono-processor architectures such as Pentium1 - accounting for one integer pipeline only, StrongARM 1110, Hitachi H8/300, and MIPS as a virtual processor with an overly simplified timing model.

AiT tool does not consider cache and pipeline in core of processors, thus these parameters also effect on the execution time. In addition, The Heptane tool is a static analysis of their code (source code and binary code) to predict the off-line execution time of a system. In our work, we have to measure total cycles, stall cycles, and level cache miss profile related the execution time of processor. Therefore, the Performance Monitor Unit is part of the ARM processor supports to our purpose in modeling the execution time for core of processors.

3.4 Heterogeneous platform: Zynq7000 AP SoC platform

3.4.1 Motivation

The Fall Detection System based on Computer vision systems, which are elaborated in chapter 2, can act upon still images or video and are able to extract meaningful information from the content of images. In which image processing (as a whole, i.e. to include video processing and computer vision) can be segmented into three levels of abstraction: pixels, features and objects, description, which are characterised by the amount of image data being processed and the amount of knowledge available regarding the content of the image [139].

Considering the implementation of image processing systems in general, it is significant that different types of processing are required to operate on different types and volumes of data. This process requires a very large amount of pixel data on two first tasks (as Object Segmentation and Filter) of Fall Detection System which repetitive operations are performed, while a lot of data using to process more complex algorithms in calculating the five features (*current angle*θ, C_{motion} (MHI), C_{theta}, *Eccentricity* and $C_{centroid}$). Continuously, analysis or classification of these features is necessary to understand the object's behaviors in context by using the different model in recognition such as.

For this reason, Zynq is a highly optimised platform for image processing. The Programmable Logic (PL) is well suited to fast, parallel operations like those required for pixel-level image processing. Computer vision functionality can be implemented in software for execution on the Zynq Processing System (PS) and integrated with higher-level software applications as required. The transition between the two, via the detection of features and objects within the image, might be accomplished using the PL with

appropriate interfacing to the PS, or by leveraging the SIMD facilities of the NEON processor. Extensive support for NEON is available in third party image and video processing products [140].

In addition to the device architecture, the role of Xilinx and third party development tools in enabling the design of image processing systems for Zynq is considered. The following are worthy of particular note [139]:

- ***Xilinx IP blocks***: a number of IP blocks are available in IP Integrator for image and video processing applications, including video memory, image enhancement, and colour adjustment functionality.
- ***OpenCV,*** Open Computer Vision in [141] is an open source project providing a set of C/C++ libraries for image and video processing. The facilities of OpenCV can be used to develop software algorithms for running on the PS.
- ***Vivado HLS Video Libraries*** include specific support for image and video processing, via a library of functions synthesisable to HDL. These can replace selected OpenCV functions and therefore functionality can be partitioned into hardware if desired [142].
- ***MATLAB/Simulink*** are available extensive facilities for image and video processing and computer vision [143]. In addition, to providing relevant functions and a development environment, developed algorithms can be converted to C/C^{++} code for implementation on Zynq.

3.4.2 Description of Zynq-7000 AP SoC

The Xilinx Zynq-7000 family is a System on Chip architecture that integrates a dual-core ARM Cortex-A9 MPCore based Processing System (PS) and Xilinx Programmable Logic (FPGA) in single device, built on 28nm process technology. The ARM Cortex -A9 MPCore CPUs are the heart of the PS which also includes On-Chip Memory (OCM), external memory interfaces and a set of I/O peripherals. The Zynq offers the flexibility and scalability of an FPGA, while providing performance, power, and ease of use typically associated with ASIC and ASSPs.

The Zynq platform is different from the older approaches. The PS is considered to be an essential part of the the chip and so it is possible to see Zynq as just a kind of an ARM SoC with an optional FPGA fabric. Figure 3.3 illustrates the functional blocks of the Zynq-7000 AP SoC. The PS and the PL are on separate power domains, enabling the user of these devices to power down the PL for power management if required.

The Zynq-7000 AP SoC is composed of the following major functional blocks:

- Processing System (PS):
  - Application processor unit (APU)

- Memory interfaces
- I/O peripherals (IOP)
- Interconnect

- Programmable Logic (PL).

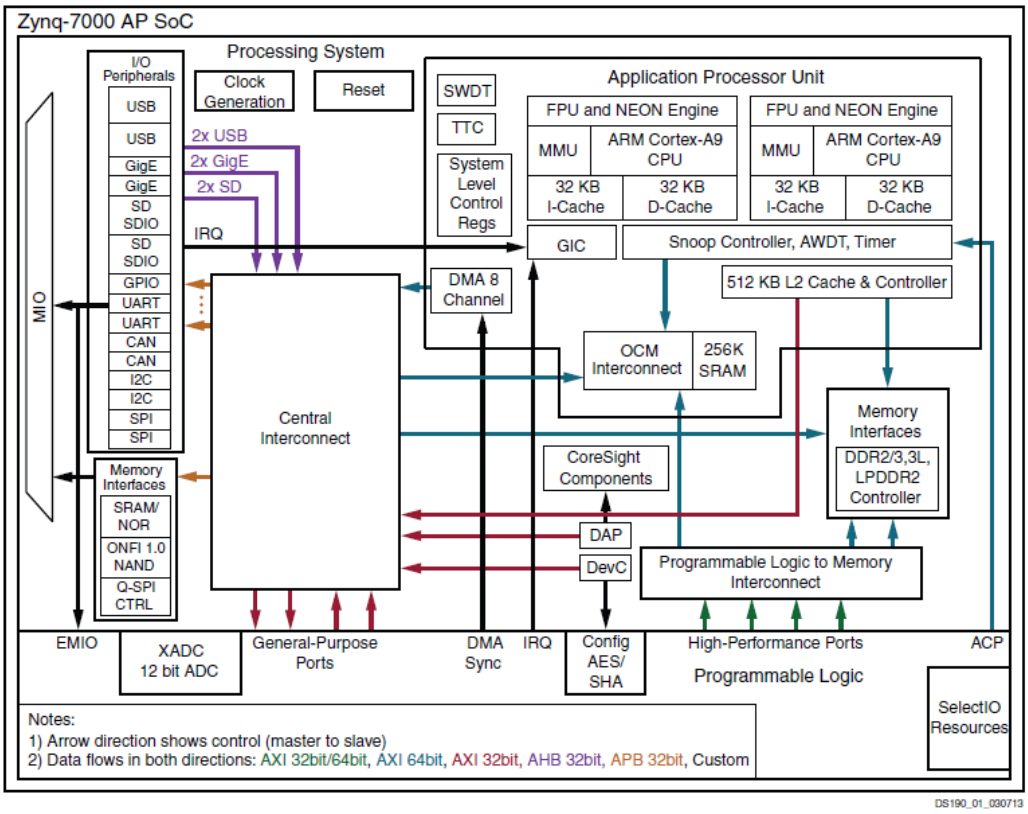


Figure 3.3-Zynq-7000 All Programmable SoC Overview[144]

3.4.3 The Performance Monitor Unit (PMU)

From the description of Zynq platform which consists of dual core of ARM Cortex A9 and FPGA is presented in previous subsection. The ARM Cortex A9 processors are the latest and highest performance ARM processors implementing the full richness of the widely supported ARMv7 architecture [145]. The Performance Monitors are part of the ARM Debug architecture and is an optional feature of an implementation used to define execution time for the Fall Detection System [146]. The basic form of the Performance Monitors is presented as follows:

- A cycle counter is the ability to count every cycle or every 64th cycle.
- A number of event counters, ARMv7 provides space for up to 31 counters. The actual number of counters is set as IMPLEMENTATION DEFINED, and the specification includes an identification mechanism.
- Controls for: enabling and resetting counters; flagging overflows; enabling interrupts on overflow.

Monitoring software can enable the cycle counter independently of the event counters. The events that can be monitored split into:

- Architectural and micro-architectural events which are likely to be consistent across many micro-architectures.
- Implementation-specific events.
- The PMU architecture defining for common events, for use across many architectures and micro-architectures by using event numbers.
- Reserves a large event number space for IMPLEMENTATION DEFINED events. When the full set of events for an implementation is IMPLEMENTATION DEFINED. ARM recommends that processors implement as many of the events as are appropriate to the architecture profile and micro-architecture of the implementation.

The accuracy of The Performance Monitors provides approximately accurate count information. To keep the implementation and validation cost low, a reasonable degree of inaccuracy in the counts is acceptable. ARM does not define a reasonable degree of inaccuracy but recommends the following guidelines:

- Under normal operating conditions, the counters must present an accurate value of the count.
- In exceptional circumstances, such as a change in security state or other boundary condition, it is acceptable for the count to be inaccurate.
- Under very unusual non repeating pathological cases then counts can be inaccurate. These cases are likely to occur as a result of asynchronous exceptions, such as interrupts, where the chance of a systematic error in the count is very unlikely.

Limitation of PMU is permitted inaccuracy. In particular, the architecture does not define the point in a pipeline where the event counter is incremented, relative to the point where a read of the event counters is made. This means that pipelining effects can cause some imprecision. Entry to and exit from Debug state can also disturb the normal running of the processor, causing additional inaccuracy in the Performance Monitors. It disables the counters while in Debug state limits the extent of this inaccuracy. An implementation can limit this inaccuracy to a greater extent, for example by disabling the counters as soon as possible during the Debug state entry sequence.

In spite of this limitation, we have applied PMU to extract the parameters used to estimate the power consumption for processors and then create the power models for processor cores.

3.5 Power/execution time models for video applications

Currently, there are generally three types of implementations: processor based solutions (software solutions), FPGA based solutions (hardware solutions) and a combination of both. One of FPGA designers' problems is that the open-source community for IP cores is not very developed, at least in comparison to open-source software such as OpenCV. Therefore, we develop and valid the power/execution time models for Fall Detection System which is run on ARM processors, while offering the possibility to accelerate through FPGA.

For modeling the power consumption and execution time on processor cores, we have applied the FLPA (Functional Level Power Analysis) methodology, which are developed by Laurent et al [123] and allows to extract the processor power consumption model with a set of high level parameters in the research of N. Julien et al [147].

3.5.1 Power estimation methodology for Fall Detection System

In our work, as explain before we select the Zynq 7000 AP SoC platform which has both processor cores and FPGA. The aim is to estimate the power consumption and execution time in order to evaluate the performance of different implementation.

For processor cores, we first need to realize the power/time characterization of the target. This methodology is based on physical measurements in order to guarantee realistic values with good accuracy. The FLPA methodology, as shown in Figure 3.4, has four main parts, which are given below:

- Firstly, a primary functional analysis helps the designer to determine which relevant parameters have an impact on the power consumption. There are two types of parameter: algorithmic parameter values depend on the specificity of the application and architectural parameter values depend on the processor configuration settled by the designer.
- Then, they characterized the power consumption behaviour and execution time (obtained by measurements) in varying independently parameters.
- Next, a mathematical model is determined by regression law.
- Finally, the accuracy of the determined model is validated against a new measurements set.

In our system, we consider to extract the characteristics and to determine the power consumption model for the Object Segmentation, Filter (Mathematical Morphology), Feature Extraction and Recognition tasks. Therefore, the number of experiments for exploring the best architecture of Fall Detection System is reduced. Two types of parameters are considered in this approach:

- Algorithmic parameters depend on the executed algorithm (typically the cache miss rate for the processor cores).
- The component configuration set by the designer (i.e., Clock frequency) is the dependent of architectural parameters.

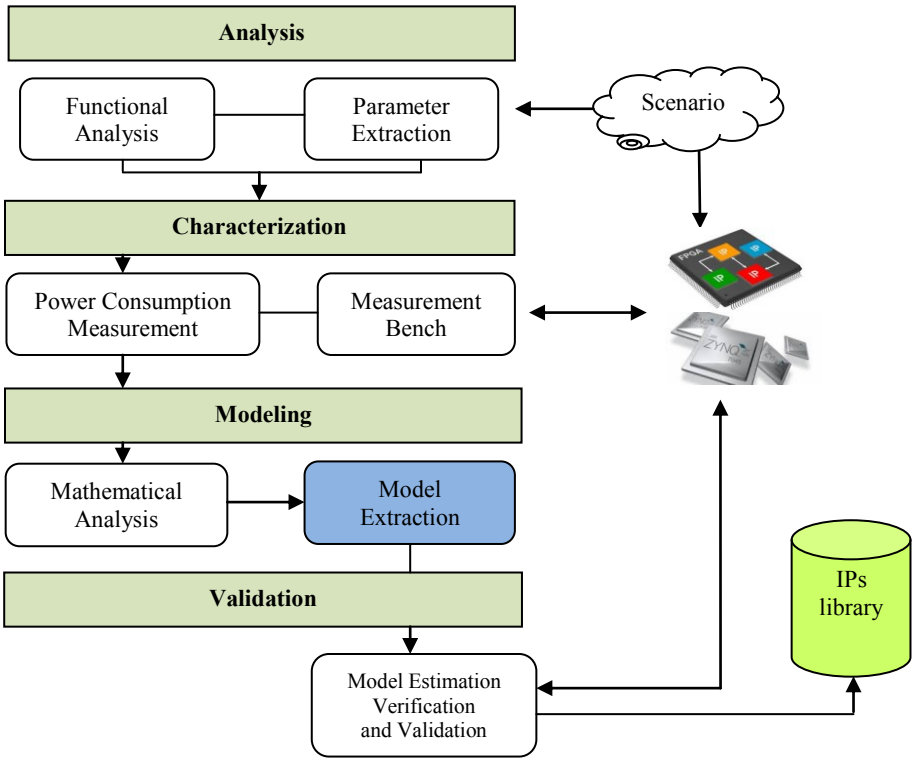


Figure 3.4-Functional Level Power Analysis Methodology [147]

For FPGA side, the estimated power is given by the sum of static power and dynamic power. The static power depends on the specific FPGA family. The dynamic power is the sum of logic power, signal power and clock power. Deng. L.et al. in [148] derive the power models which the power components are proportional to the area of a design, including hardware resource power, signal power and clock power. Their power models for the components that are proportional to the area of the design are derived by performing nonlinear regression analysis on the area and power data of applications.

The power consumption model on FPGA of our system is extended from their approach. Our model which is extracted based on the hardware resources including the BRAM, DSP, LUT and FF for two first tasks (Object Segmentation and Filter using Mathematical Morphology) in Fall Detection System is presented in Section 3.6.2.

3.5.2 Power measurement

A power measurement bench is developed in order to reduce the time of each scenario measurements. Voltage and current monitoring and control are available for selected power rails through Texas Instruments' Fusion Digital Power graphical user interface. The three onboard Texas Instrument (TI) power controllers in Figure 3.5 (U32 at address 52, U33 at address 53, and U34 at address 54) are wired to the same Power Management Bus (PMBus). The PMBus connector, J59 (as shown in Figure 3.76), is provided an interface of the TI USB Interface Adapter PMBus pod (TI Evaluation Module USB-TO-GPIO) [149] and associated TI Fusion Digital Power Designer GUI [150]. This is the most convenient way to monitor the voltage and current values for the power rail.

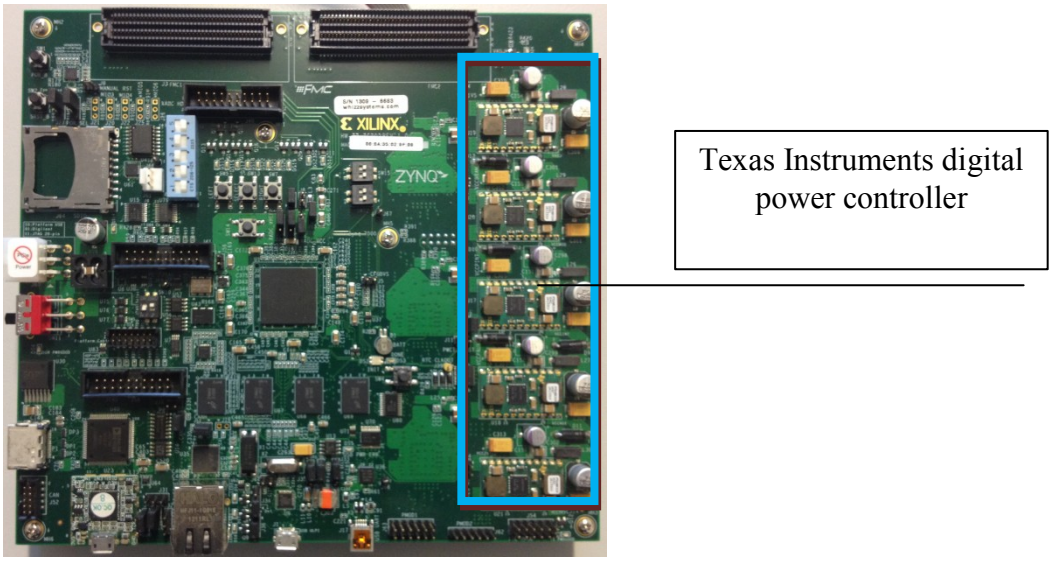


Figure 3.5-Integrated Texas Instruments digital power controller on Zynq-7000 Ap SoC

Voltage and current levels of the power supply are measured with Texas Instruments' UCD9248 Digital PWM System Controller, integrated on the Zynq board. This multi-rail and multiphase PWM controller for power converters supports the Power Management Bus (PMBus) communication protocol. Its PWM signal drives a UCD7242 integrated circuit that regulates V_{ccint} supply voltage. A set of PMBus commands is used to configure IC functions. The UCD7242 possesses on-chip voltage and current sensing circuitry and communicates with the UCD9248.

Fusion Digital Power Designer is a Graphical User Interface (GUI) used to configure and monitor a Texas Instruments digital power controller as shown in Figure 3.5 (UCD 91XX, UCD 92XX), they typically embedded on an EVM. The application uses the PMBus protocol to communicate with the controller over serial bus by way of a TI USB adapter. PMBus uses the System Management Bus (SMBus) to communicate

with a controller over serial bus. The PMBus specification defines the application layer while the SMBus standard defines the transport layer.

Figure 3.6 shows the measurement environment for Zynq-7000 AP SoC platform composed of a power measurement instrument. The EVM has four separate power rails:

Rail 1 Vccint: 1.0V nominal supply of Zynq-7000 AP SoC platform that powers all of the PL internal logic circuit.

Rail 2 Vccpint: 1.0V nominal supply that powers all of the PS internal logic circuits.

Rail 3 Vccaux: 1.8V nominal supply that powers all of the PL auxiliary circuits.

Rail 4 Vccpaux: 1.8V nominal supply that powers all of the PS auxiliary circuits.

The following steps describe how to make the measurements on the Zynq-7000 AP SoC platform:

a) Plug in the USB cable to both the PC and the USB interface adapter with Zynq platform across jumper J59 (Figure 3.7) and wait for the green LED to illuminate.

b) Monitor real-time data such as input voltage, output voltage, output current, temperature, and warnings/faults are continuously monitored and displayed by the GUI.

c) Monitor power in each rail of PS or PL.

d) Calculate the average power from step 3 for different part of Zynq -7000 Ap SoC platform.

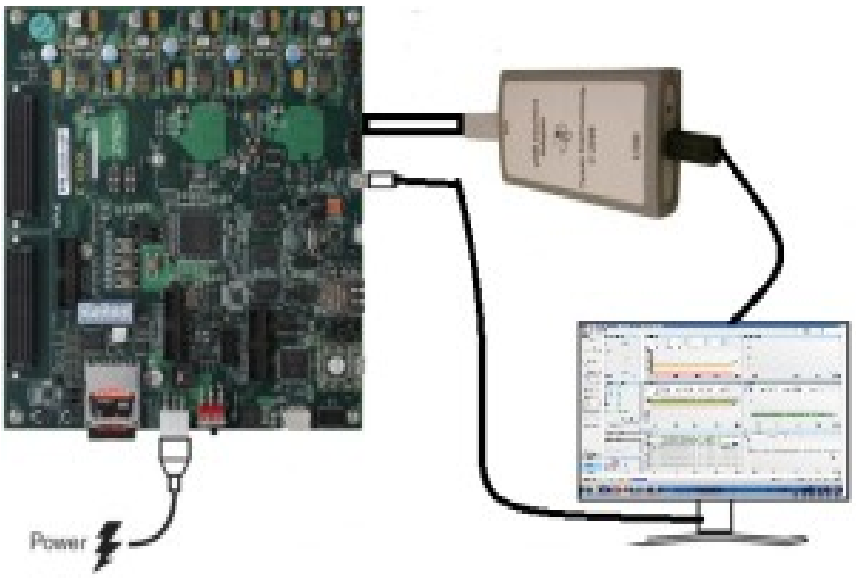


Figure 3.6-Measurement environment for Zynq-7000 AP SoC platform

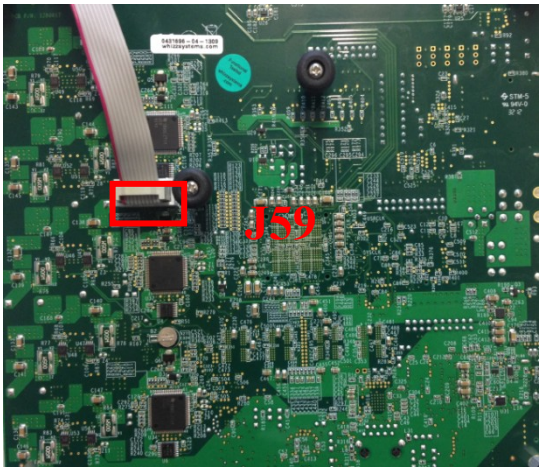


Figure 3.7-Power Measurement probes across jumper for Zynq-7000 AP SoC

3.5.3 Execution time measurement

For performance measurement code, there are too many variations of timing mechanisms, operating system behaviors and run-time environment to have one single, simple solution [151]. In our system, the measurement time is performed by interval counting. The operating system also uses the timer to record the cumulative time used by each process. This information provides a somewhat imprecise measure of program execution time. The operating system maintains counts of the amount of user time and the amount of system time used by each process. When a timer interrupt occurs, the operating system determines which process is active and increments one of the counts for that process by the timer interval. It increases the system time if the system is executing in kernel mode, and the user time otherwise.

We can also read the process timers by calling the library function *times*, declared as follows:

```
t_start = get_time ();
process task ();
t_stop = get_time ();
Execution_time = t_stop – t_start;
```

As a return value of execution time, times is the difference of number of t_stop and t_start since each process task start. Therefore, computation the total time between two different points in a program execution is calculated by making two calls to times and computing the difference of the return values.

For measure the execution time on FPGA, we have to define the latency time and the clock period while synthesising a video application by using Vivado_HLS. The execution is multiple by the average latency and clock period during process.

3.6 Proposed power model of the Fall Detection System on heterogeneous platform

Increased demand and reduced time for getting low power architecture of Fall Detection System based HW/SW co-design. Design methodologies, where decrease of power consumption in such system can be done at the first stage of design. It knows that any optimization requires good analysis of design and so is the case with managing the power consumption of the design. To reduce the time when extracting the low power consumption in the final Fall Detection System version, it is necessary to have early power consumption estimation. Thus, we develop power models as achieved in this section.

Figure 3.8 illustrates the methodology for extracting power models applied on Fall Detection System. Firstly, we choose both traditional implementation ways for video applications are on Software (ARM Cortex A9 processor) and Hardware (FPGA). On the software (SW) side, Object Segmentation, Filter tasks are implemented on one core and both two cores of processor. The basic idea is to expose data parallelism for these two tasks, the original image can be split into slices (2 slices or 3 slices) that can be segmented and filtered in parallel. Therefore, the object in each slice is merged in a complete binary image. Secondly, we can extract the parameters such as power consumption and execution time. The analysis of these tasks which take the longest processing will be considered for hardware acceleration. Filter tasks is selected in this case. Notwithstanding, Feature Extraction and Object Segmentation tasks take not too much difference in execution time together. Thus, other ones, Object Segmentation task, will be candidate implementation based FPGA. All tasks are processed on software with two different resolutions of input images, and various operating frequencies (667 MHz, 333 MHz and 222MHz).

In context of our work, some parameters are derived to extract power/time model for heterogeneous platform and find out more complex architectures based platforms. Table 3.1 summarizes the model parameters

3.6.1 Power models for processor

To extract the performance of processors which include a set of metrics such as Data cache access, data cache refill, total instruction, total cycle and data memory access, etc, we enable the optional non-invasive debug component, Performance Monitors

Extension. In ARMv7, the Performance Monitors Extension is an optional feature which helps to derive the specification of the earlier ARM implementations [146].

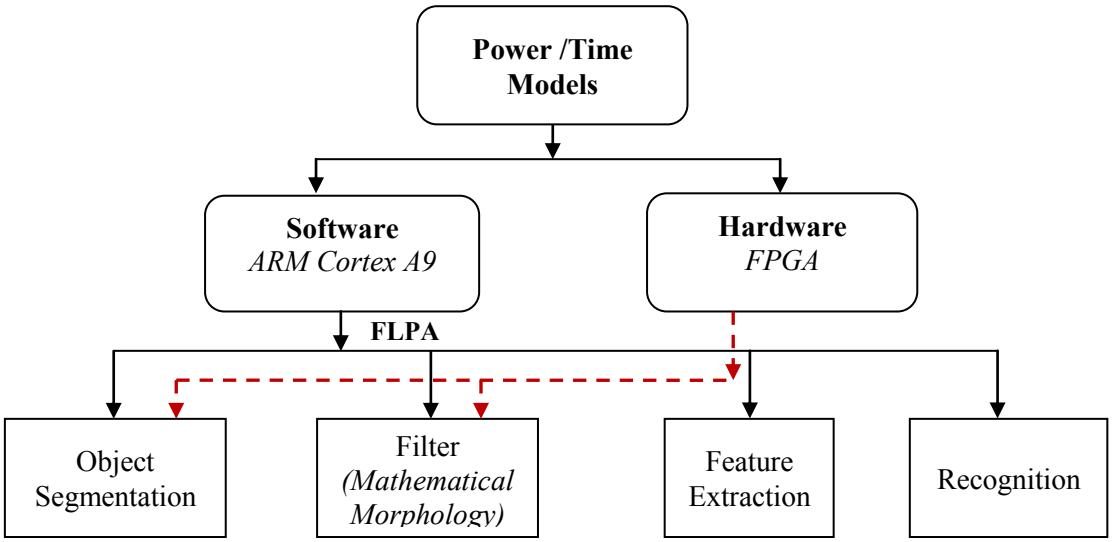


Figure 3.8-Framework for extracting power models

Table 3.1-Model parameters

Symbol	Description
Power Model	
λ	Caches miss rate for processor
IPC	Instructions per cycle
s	Resolution Images
F_{core}	Frequency of the core
N	Number of cores
Time Model	
CPI	Cycles per Instruction
I	Total Instructions
M_{stall}	Memory stall
R_{stall}	Read Stall cycles
W_{stall}	Write Stall cycles
$R_{stall}IP$	Read Stall cycle per Instruction
$W_{stall}IP$	Write Stall cycles per Instruction
T	Execution time of a program

3.6.1.1 Scenario implementations

In our work, the four tasks of Object Segmentation, Filter, Feature Extraction and Recognition are independently executing, with negligible interference from other tasks. The system is composed of low power processor with N cores, operating at clock frequency F, where F $\in$ $\{F_{min}, F_{max}\}$. As we discussed the concepts of Fall Detection System in Chapter 2 and the FLPA methodology is explained in subsection 3.2.2.3. Firstly, different functional blocks are divided into such as the memory unit, clock system unit as shown in Figure 3.9. These parameters are indicated for each functional block of the processor and they are λ_1 and λ_2 respectively for L_1 and L_2 cache miss rates, Instruction per cycle (*IPC*) for all the activated cores and *F* for clock unit. The second step is the characterization of the power model by varying the parameters. The scenario of our test is also the two separate modules of Fall Detection System with different resolution images and number of cores. In our work, characterization is accomplished by measurement on Zynq 700 AP SoC platform.

Figure 3.10 and Figure 3.11 present the relationship between frequencies and power consumption on various cores of processor of two above applications. The clock is operated to run on the platform at different available frequencies such as 222MHz, 333MHz and 667 MHz. The various estimation of power consumption for Object Segmentation and Filter task (using Mathematical Morphology) is distinguished with application on one core or two cores and with no application. Power modeling methodology is proposed for different frequency scaling and the number of cores.

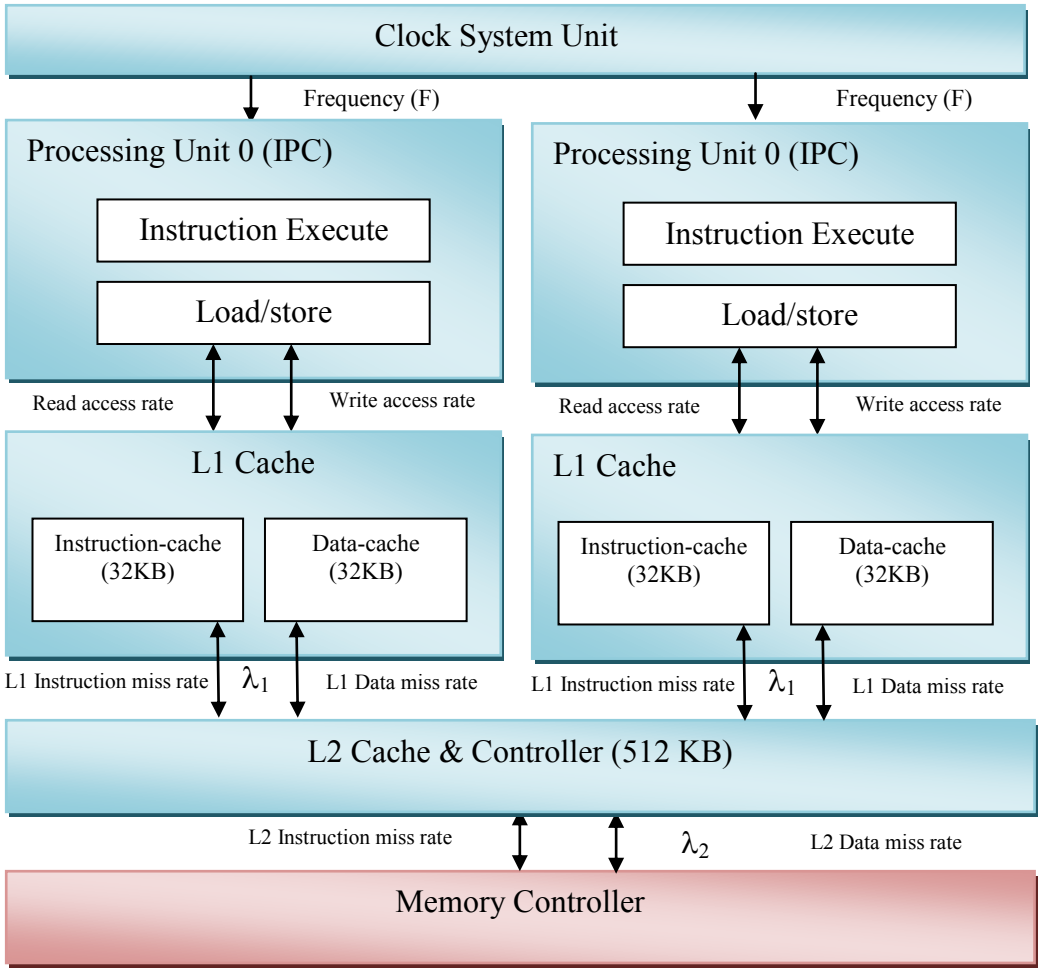


Figure 3.9-Functional Blocks of Dual Core ARM Cortex A9 processor[5]

Table 3.2-Power model of Object Segmentation and Filter task

Tasks	Parameters	Power models
Object Segmentation	Frequency of cores (F)	$P_{Obj_{1core}} = 0.3284.F + 180.91$ $P_{Obj_{2cores}} = 0.4276.F + 174.17$ $P_{no\_application} = 0.1317.F + 193.36$
Mathematic Morphology		$P_{MM_{1core}} = 0.3249.F + 190.97$ $P_{MM_{2cores}} = 0.4169.F + 189.19$ $P_{no\_application} = 0.1317.F + 193.36$

[5] http://www.design-reuse.com/articles/16875/the-arm-cortex-a9-processors.html

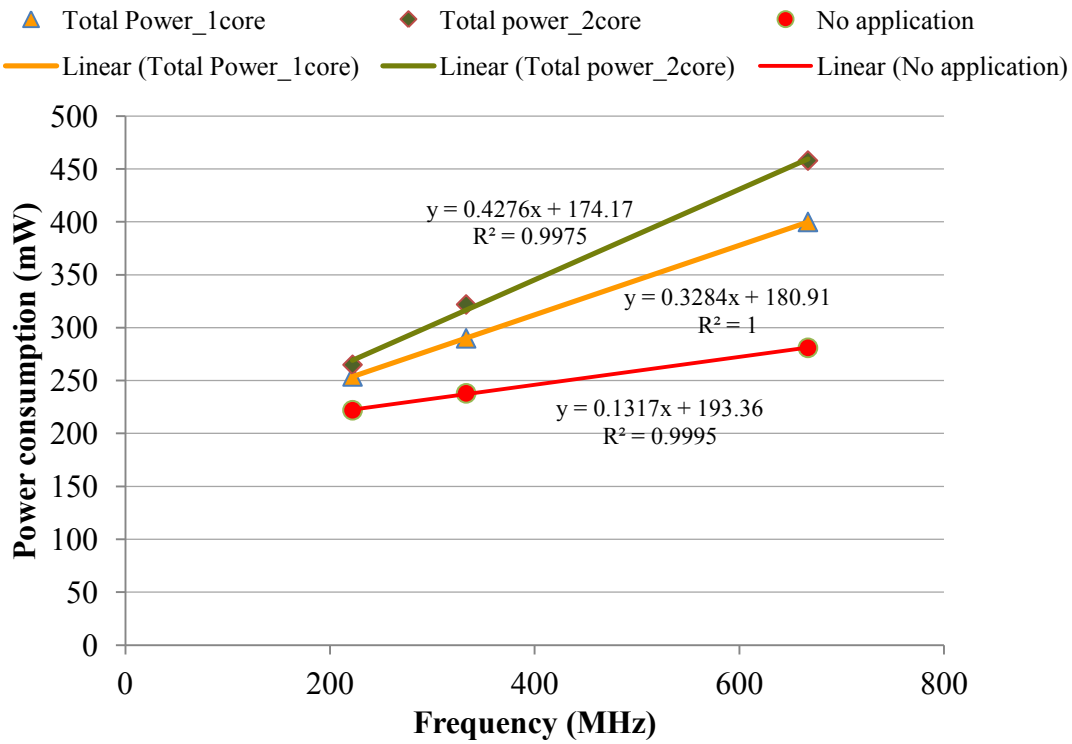


Figure 3.10-Power models of Object Segmentation with 320x240 input images on SW

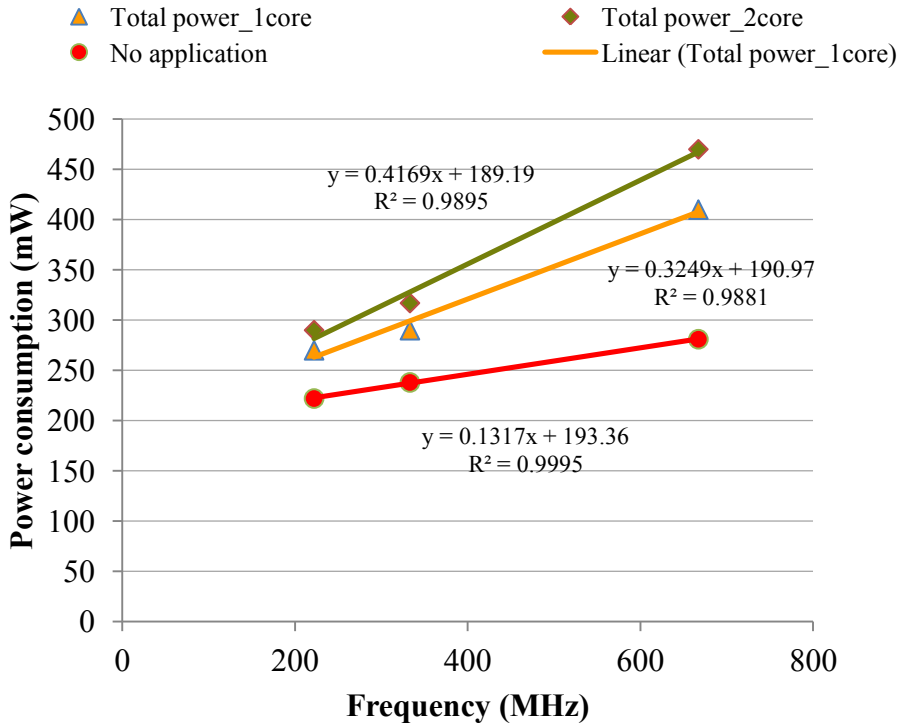


Figure 3.11-Power models of Mathematical Morphology with 640x480 input image on SW

In addition, power depends on not only frequencies and number of cores but also the cache miss rate (λ) of two levels caches on processor (for instances, ARM Cortex A9) and Instructions per Cycle. Figure 3.12 and Figure 3.13 present the variation of the power consumption according to the IPC and cache miss rate (λ) parameter in different resolution images. In the others, λ and IPC are not unremarkable affected by the resolutions and the number of cores. In fact, we also find that λ and IPC are independent with frequencies from experiments.

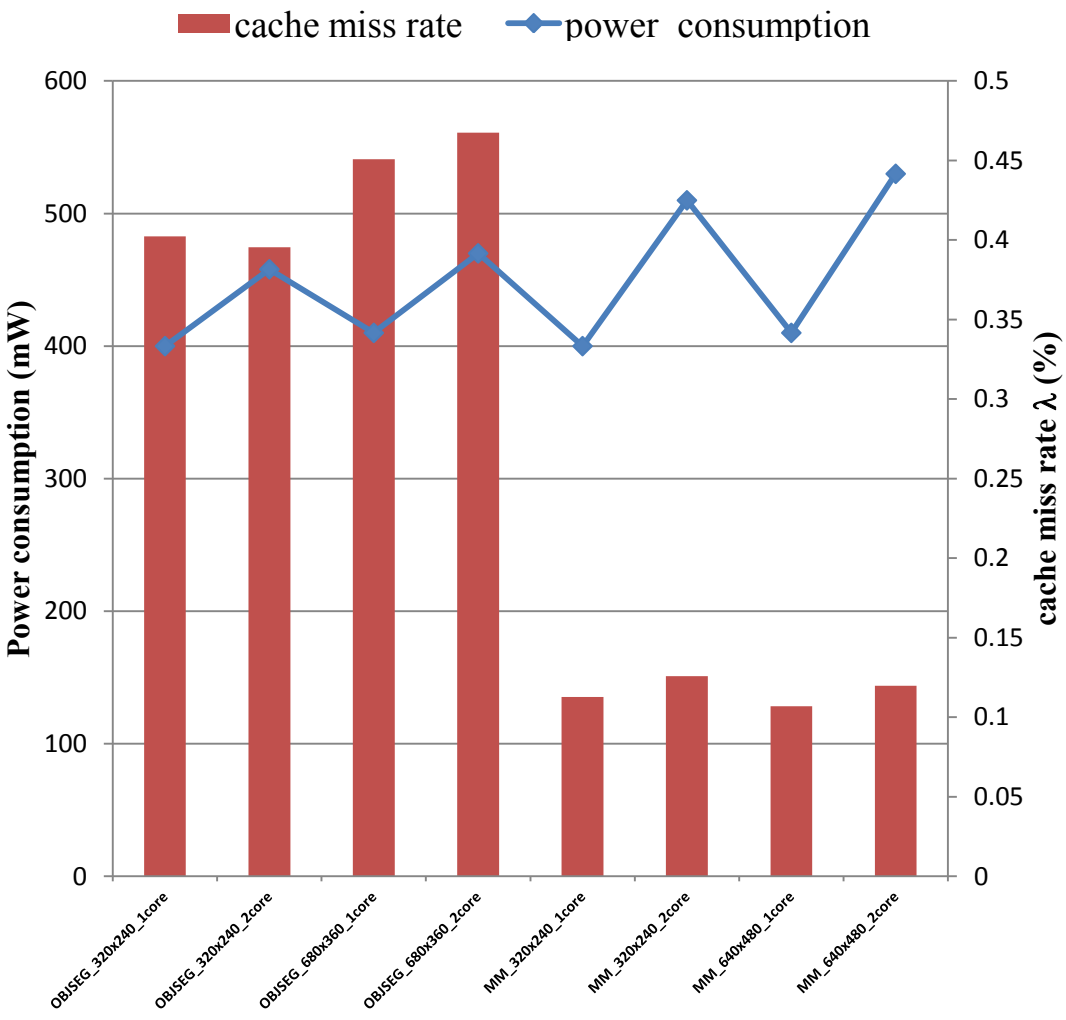


Figure 3.12-The power consumption and cache miss rate of Object Segmentation and Mathematic Morphology with various resolutions

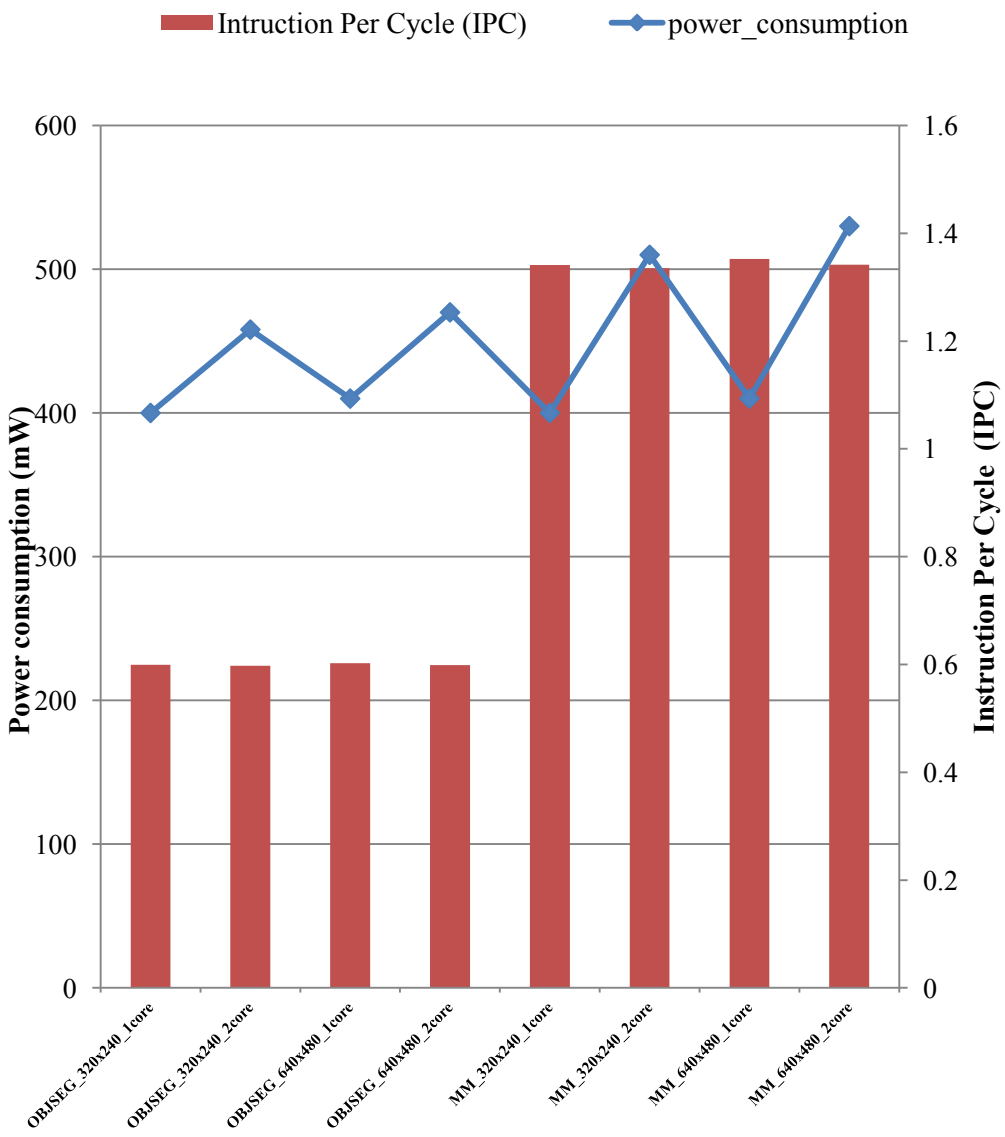


Figure 3.13-The power consumption and Instruction per Cycle (IPC) of Object Segmentation and Mathematic Morphology with various resolutions

In these applications, for instance, Object Segmentation, these images are parallelized by splitting each image (frame) into two slices running on dual core of processor. In this case, the multithreading technique is implemented on the processing system of Zynq 7000 AP SoC platform. Each thread will be scheduled by Linux to running on each separate processor. Object Segmentation task is processed in parallel on two cores as follows:

```
Function parallel_object_segmentation is
   Input: pixel array of origin image A
   Output: pixel array of processed image P

   //------------------------
   Declare an pixel array A1
   Declare an pixel array A2
   Declare an pixel array P1
   Declare an pixel array P2

   Set A1 to first slice of A
   Set A2 to the second slice of A
   //------------------------

   Create 2 threads to process each slice of A
   //Each thread will be scheduled by Linux to running on each separate processor if possible

      Thread 1: call object_segmentation function
         P1 = object_segmentation(A1)

      Thread 2: call object_segmentation function
         P2 = object_segmentation(A2)

   Waiting for finished threads

   Merge P1, P2 to P
      Copy P1 to first slice of P
      Copy P2 to the second slice of P

   Return P
```

3.6.1.2 The general models of power consumption

The power consumption models are determined from the all experiments by using regression analysis. Regression analysis is a statistical process for estimating the relationships among variables in statistics. It includes many techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. Regression analysis is widely used for prediction and forecasting. In restricted circumstances, regression analysis is used to infer causal relationships between the independent and dependent variables [152].

The models in our work are defined in which are related the parameters are indicated such as core frequency, number of cores, Instruction per Cycle, Cache miss rate, resolution images (as shown in Table 3.1). Therefore, the power model for the ARM Cortex A9 processor is created by (see more in Equation 3.6):

$$P_{PS}\,(mW) = 31.7 + 0.42*F + 52.9*N + 7.7*(\lambda_1+\lambda_2) + 68.3*IPC \quad (3.6)$$

Where,

P_{PS}: Power consumption on processor cores.

F: frequency of processor cores.

N: Number of core.

λ_1, λ_2: Caches miss rate of L1 and L2 caches on processor.

IPC: Instruction per Cycle.

The different power models are validated by the real board measurement in order to find the efficiency of FLPA modeling for processor cores applied in this thesis. The video applications are compiled for FPGA and processor cores on Zynq 7000 AP SoC platform. While these applications are running, the power consumption is measured online. Finally the measurement of experiment from the platform is compared with the estimation from the power consumption model which is extracted the useful activities of the power model.

Table 3.3 shows the maximum and average errors obtained with our approach modeling against measurements on ARM Cortex A9. The results obtained for the twelve experiments (for Object Segmentation and Filter tasks) and six experiments (for Feature Extraction and Recognition tasks) validate our approach. Furthermore, Table 3.3 describes the parameter numbers including image resolution, number of processor cores, frequency of processor cores and execution time for each model. With each models the power estimation is obtained. Our power modeling approach has a negligible maximum error equal to 3.5 %.

Table 3.3-Maximum and average errors for power consumption model on processors

Applications	**Processors**	**Maximum error %**	**Average error %**	**Measurement numbers**	**Parameter numbers**
Object Segmentation	ARM Cortex A9	5.5	2.2	12	4
Filter (Mathematical Morphology)	ARM Cortex A9	5.3	2.4	12	4
Feature Extraction	ARM Cortex A9	6.2	3.5	6	4
Recognition	ARM Cortex A9	5.8	2.9	6	4

3.6.1.3 The power model for Fall Detection System on processor cores

We estimate the power consumption of processor cores for Fall Detection System based on the power model as illustrated in equation 3.6. The new modeling related not only frequency of core, number of cores but also image resolution parameters is considered in this subsection. The image resolution is one of a factor impacting on the accuracy of our system. Therefore, the power consumption model for each task of Fall Detection System extended from equation 3.6 is determined as follows:

$$P(\tau_i) = P_{core}(N, F_{cores}, s) = 152 + 0.000416 * s + 0.39 * F_{cores} + 30.5 * N \quad (3.7)$$

Where, P (τ_i) is the power consumption on task τ_i. In which τ_i is i^{th} of task and *i= (1:4)*; *N* is the number of processor cores; F_{cores} is the frequency of processor cores; the image size or the image resolution is assigned by *s*.

The evaluation of the general model is analysed by the real measurement on processor cores with the maximum error 3.5%. This error rate is not too high, therefore it is a good adequacy in extending power consumption models for the Fall Detection System.

3.6.2 Power models for hardware

3.6.2.1 The mathematic models of power consumption on hardware

For modeling the power consumption on FPGA, two selected tasks in Fall Detection System, the Object Segmentation and Filter tasks, are implemented on HW (FPGA). By this way, the power of these tasks is estimated. In addition, in order to extract more architecture for Fall Detection System then power consumption and hardware resources are evaluated.

Table 3.4-The power consumption on FPGA

Application	**Resolution**	**Power on Hardware (W)**
Object Segmentation	320x240	0.124
	640x480	0.124
Mathematical Morphology	320x240	0.184
	640x480	0.184
	1920x1080	0.184

While running the video applications on the Zynq 7000 AP SoC platform, the total on- chip power and its details are estimated and are shown in Table 3.3. The total power is calculated as follows:

$$\boldsymbol{P_{hardware} = P_{no\_application} + P_{FPGA}} \quad (3.8)$$

Where, $P_{no\_application}$ indicates the power of processor cores without any implementation. P_{FPGA} includes P_{FPGA_static} and $P_{FPGA\_dynamic}$.

Table 3.4 illustrates the relationship of hardware resources and power consumption of these tasks on Zynq platform. Although, the image resolutions of Object Segmentation and Filter task (using Mathematical Morphology) processed on FPGA is adjusted, there are not many differences of hardware resources between them. Therefore, the power consumption on FPGA almost doesn't change while varying the image resolutions.

As we discuss in Section 3.5.1, the model of power consumption on FPGA is extracted as following equation 3.9:

$$P_{FPGA}(mW) = 123.5 + 8.07P_{BRAM} + 0.02P_{LUT} + 0.00432P_{FF} \quad (3.9)$$

Where, P_{BRAM} , P_{LUT} and P_{FF} are power consumption on BRAM, LUT and Flip Flop.

Table 3.5-Hardware resources and power consumption on different input image resolutions

Tasks	Resolutions	BRAM		DSP		LUT		FF		Power
		Usage	%	Usage	%	Usage	%	Usage	%	(mW)
Object Segmentation	320x240	0	0	0	0	60	0.34	370	0.34	124
	640x480	0	0	0	0	65	1.2	458	0.6	124
Filter (Mathematical Morphology)	320x240	21	15	11	5	7329	6.88	9282	17.44	184
	640x480	21	15	11	5	7384	6.93	9341	17.55	184
	1920x1080	21	15	11	5	7473	7.02	9727	18.28	184

3.6.2.2 Validation of the power models on FPGA

Table 3.6 illustrates the validation results of estimated and measurement power consumption for two video applications on FPGA. After extracting the power consumption models, we can estimate the power consumption of two first tasks such as Object Segmentation, Filter (using Mathematical Morphology) at different of image resolutions based on equation 3.9. The error rate of our model for FPGA is minimum value of 0.08% and the maximum of 1.36%. The accuracy of this model is all lower than 2%.

Table 3.6-The validation of power consumption model on FPGA

Tasks	Resolutions	Pestimation (mW)	Pmeasure (mW)	Error rate (%)
Object Segmentation	320x240	123.9	124	0.08
	640x480	124.2	124	0.16
Filter (Mathematical Morphology)	320x240	186.5	184	1.36
	640x480	185.6	184	0.87
	1920x1080	185.5	184	0.82

The aim of our work is to define the power consumption models for both processor cores and FPGA, as called heterogeneous architecture. It is necessary for exploration the low cost architecture for Fall Detection System. We present the power consumption model for heterogeneous architecture in next subsection.

3.6.3 Power consumption models for heterogeneous architecture

Recently, the FPGAs incorporate processor cores, arithmetic elements and memory blocks, in addition to the usual logic elements. They allow the realization of complex SoC (System on Chip) by combining hardware and software design. So, in the two previous sections, both processor cores and FPGA are selected to validate our power modeling approach. From Equation 3.7, the power modeling is deduced as follows:

$$P_{Total} = P_{PS} + P_{FPGA} \tag{3.10}$$

Where, P_{PS} is the total power on processor cores; P_{FPGA} includes the static and dynamic of power on FPGA, in which dynamic power consumes on various logic blocks, DSP, BRAM, or others.

3.7 Execution time models for heterogeneous platform

Estimating the execution time of computer programs is an important but challenging problem in the computer systems. Existing methods require experts to perform detailed analysis of program code in order to construct estimators or select important features. We recently developed a new model to automatically extract a large number of features from program execution on various video application inputs, on which estimation time models can be constructed without expert knowledge.

The execution time for these applications in parallel is modelised when they execute in multiprocessors. The execution time models are extracted for both processor and FPGA which are presented as follows:

3.7.1 Execution time models for processor

The total cycles, stall cycles, and level cache miss profile are estimated to create the execution time model of processor with factors such as number of processor cores *N,* frequency *F,* etc. The execution time of processor cores is defined as follows:

Execution time of processor cores T = (Processor execution clock cycle + memory stall cycle)* clock cycle time.

Where,

Memory stall cycle = (Read stalls per Instruction + Write stalls per Instruction)* Total instruction

$$M_{stall} = (RPI_{stall} + WPI_{stall}) * I \quad (3.11)$$

Processor cores execution clock cycle = Cycle per Instruction * Total instruction

Processor cores execution clock cycle = CPI* I (3.12)

Therefore, the execution time of processor cores is defined in the following equation:

$$\boldsymbol{T(\tau_i) = \left(\frac{CPI + RPI_{stall} + WPI_{stall}}{N * F}\right) * I} \quad (3.13)$$

Where,

τ_i is the i^{th} task, with i=(1:4);

CPI is Cycle per Instruction;

RPI_{stall} , WPI_{stall} are Read and Write stalls per Instruction;

I is total instructions;

N, F are number and frequency of processor cores.

For an application on processor cores, we can evaluate the execution time by:

- First and most importantly, the models are applied to estimate the execution time in different frequencies, number of cores, Cycle per Instruction, etc. based on equation 3.13.
- Second, the measurement of the implementation programs with reasonable complex functionality is obtained. An inexperienced observer is not trivially identified the important features by this way.

Therefore, the error rate between estimation time and measurement is analysed. In our system, the assessment is applied for two first tasks: Object Segmentation and Filter (using Mathematical Morphology). The error rates take around 0.05 % for 1 core and

0.053% for 2 cores of Object Segmentation task. The same comparison of Filter (Mathematical Morphology) task, its rates are determined at 0.02% and 0.07% (as shown in Figure 3.15).

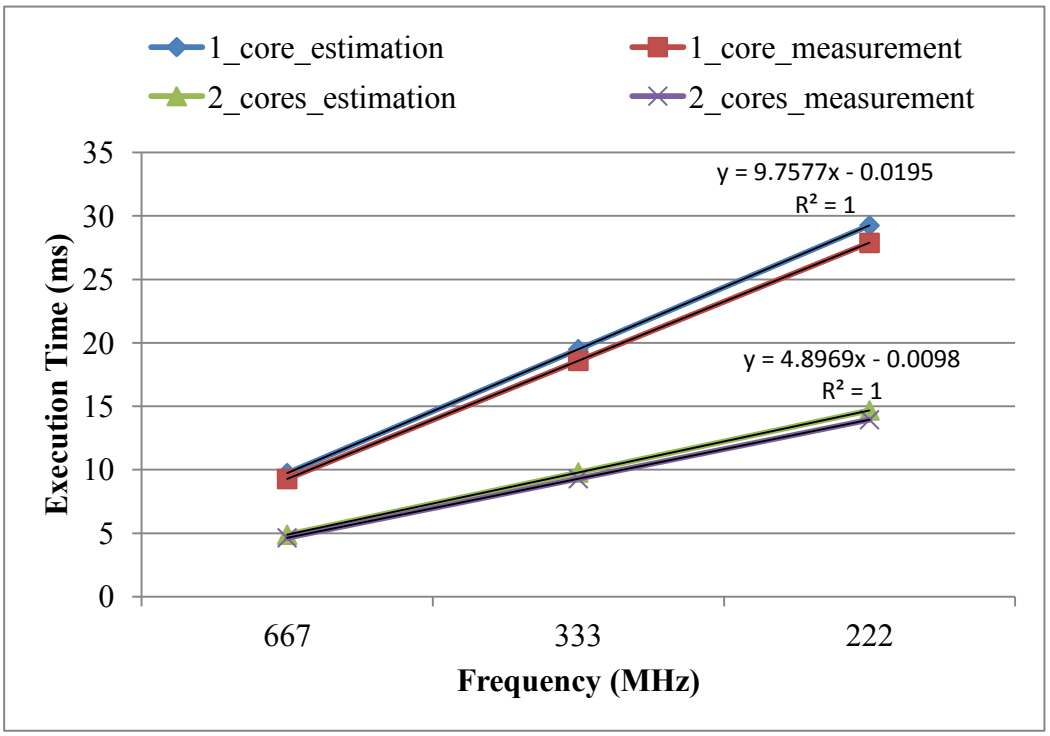


Figure 3.14-Execution time validation of Object Segmentation task

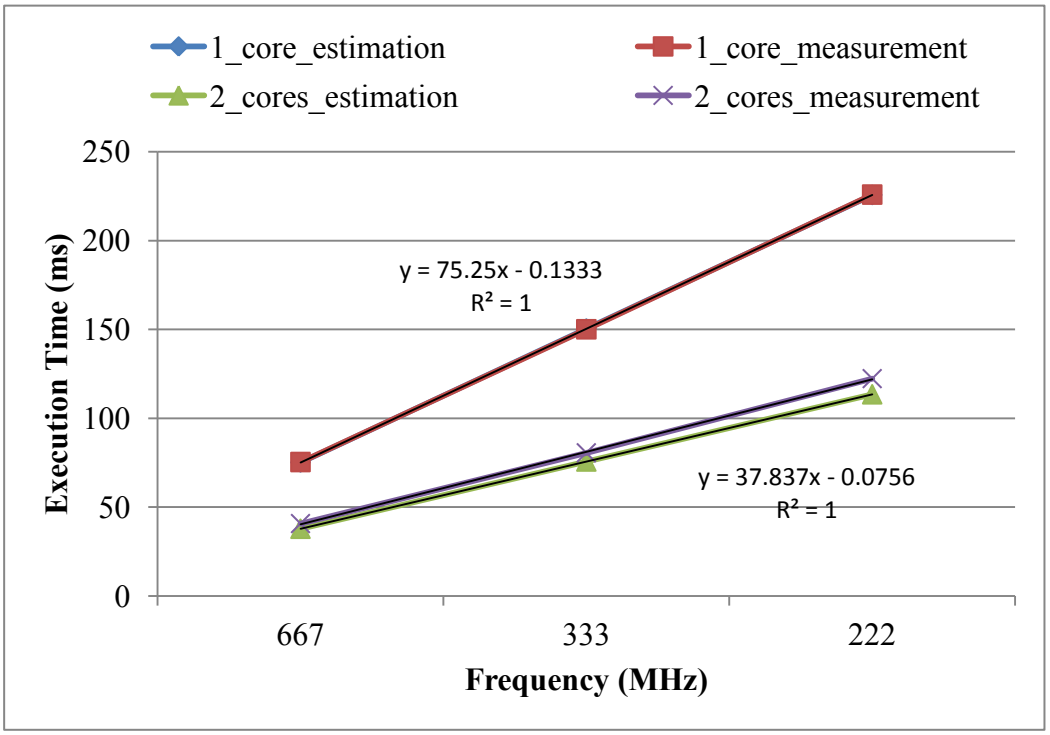


Figure 3.15-Execution time validation of Filter task (Mathematic Morphology)

However, the extraction of these previous parameters is not sufficient for exploring the heterogeneous architecture of the Fall Detection System. We create the execution time model which relates not only the basic factors (based on equation 3.13) but also the image resolutions. In addition, the image resolution is a parameter which influences the accuracy of our system (see more detail in Section 4.5.3). Thus, we extend to extract the time model $T(\tau_i)$ in ms for each task of our system as follows:

$$T_{core}(\tau_1, {}^{s}/_{N}, F_{core}, N) = 56.8 + 0.00019 * s - 0.064 * F_{cores} - 20.6 * N \quad (3.14)$$

$$T_{core}(\tau_2, {}^{s}/_{N}, F_{core}, N) = 520 + 0.0132 * s - 0.58 * F_{cores} - 182.7 * N \quad (3.15)$$

$$T_{core}(\tau_3, {}^{s}/_{N}, F_{core}, N) = 30.3 + 0.0032 * s - 0.096 * F_{cores} \quad (3.16)$$

$$T_{core}(\tau_4, {}^{s}/_{N}, F_{core}, N) = 6.55 + 0.0000318 * s - 0.0145 * F_{cores} \quad (3.17)$$

Where, *s* is the image resolution; *N, F_{core}* are number and frequency of processor cores.

In order to explore the suitable architectures for our system, we extract the time models of tasks which are executed both on processor cores and on FPGA. In the next subsection, we continuously propose the execution time models for hardware of two tasks Object Segmentation and Filter tasks (using Mathematical Morphology techniques).

3.7.2 Times models for hardware acceleration (FPGA)

The execution time of two tasks such as Object Segmentation and Mathematical Morphology is defined with various image resolutions, at frequency of 50MHz and targeting with 20ns of clock default. The Vivado_HLS tool supports for this high level estimation. The execution time of these tasks depends on the image resolution inputs as presented in Table 3.7.

Table 3.7-Estimation of Execution time on hardware for video applications

Application	Resolution	Execution Time (ms)
Object Segmentation	320x240	0.33
	680x480	1.32
Mathematical Morphology	320x240	4.35
	680x480	17.72
	1920x1080	218

Execution time model for FPGA is extracted as follows:

$$T_{FPGA}(\tau_1) = 0.025 + 4.10^{-6} * s \quad (3.18)$$

$$T_{FPGA}(\tau_2) = 0.0002 + 0.00011 * s \quad (3.19)$$

Where, $T_{FPGA}(\tau_i)$ is the execution time on FPGA of i^{th} task with *i= (1:2)*; τ_1, τ_2 are execution time on Object Segmentation and Filter task; *s* is the image resolutions of input image.

3.8 Conclusion

In this Chapter, we specific the separate video tasks which are used in the Fall Detection System to extract the general power consumption and execution time models. The modeling methodology is defined by analysing processor cores (based on FLPA) and FPGA (related to hardware resources) with the aim to combine then for heterogeneous architecture. To defined power consumption and time models for processor cores, different scenery of experiments are implemented according to the different configurations offered by the ARM Cortex A9 processor of Zynq platform. On the basis of the FPLA techniques, power consumption and execution time models have been extracted for the different tasks of the Fall Detection System**.**

Moreover, these models are extended for the Fall Detection System regarding the features of the target architectures and the considered application such as image resolutions, core frequency and number of activated cores. The analysis of the error rate shows a maximum of 3.5% for the power consumption and 0.07% for the execution time. The error rates offer a good quality models on processor cores.

In addition, the video applications are synthesized on FPGA part with various image resolutions by supporting of Vivado_HLS tool in order to estimate and model the power consumption and execution time. These models take into account hardware resource requirements and features of the application. The error rate is inferior to 2%. Our models also allow to assess the power consumption and execution time for different configurations of heterogeneous architecture and assignments of tasks of the Fall Detection System.

The next Chapter introduces a new exploration methodology for low cost architectures of the Fall Detection System. At first, the execution time and power consumption models has been completed by the evaluation of the accuracy, precision and recall performances of the Fall Detection System for different configurations of the architecture and the application. An accuracy model for this system needs to be determined. Then I define a Design Space Exploration (DSE) Methodology for the Fall Detection System by applying the parallelism techniques such as intra-task and inter-task static scheduling.

Chapter 4. Low Cost Architecture for Fall Detection System

Currently, designing low-power complex embedded systems is a main challenge for corporations in a large number of electronic domains. There are multiple motivations which lead designers to consider low-power design such as increasing lifetime, improving battery longevity, limited battery capacity, and temperature constraints. Unfortunately, there is a lack of efficient methodology and accurate tool to obtain power/energy estimation of a complete system. From functional estimation based on real board's measurements, our methodology helps designers to develop new power models and to explore new architectures for the Fall Detection System. We apply parallelism techniques at task level in order to reduce energy, power consumption and execution time for our system with sufficient performance of accuracy.

This chapter is organized as follows: Section 4.1 presents the literature of high level synthesis tools based on C/C++ specification. Section 4.2 describes some techniques in reducing low power at low level and high level of abstracts. The recent implementations on video applications and especially on the Fall Detection are illustrated in Section 4.3. Section 4.4 depicts an overview of low cost architecture exploration methodology. At first, the Fall Detection System is implemented on software with the different configurations and the comparison between image resolutions in execution time, power/energy consumption is made. The evaluation of accuracy, precision and recall performances are all given in Section 4.5 and shown insufficient frame rate. We then assess the execution time, power/energy consumption on two tasks of the Fall Detection System implemented on hardware in Section 4.6. In order to explore the different architectures, assignment and scheduling of tasks, the parallelism techniques such as intra-task and inter-task for the Fall Detection System are applied and elaborated in Section 4.7. Section 4.8 describes the Design Space Exploration (DSE) Methodology for the Fall Detection System to define the low cost architectures.

In the next Section, the literature of high level synthesis tools based on C/C++ languages for heterogeneous architectures such as CatapultC[6], Gaut[7], Spark[8], PICO and Vivado HLS [9]is presented below.

---

[6] http://calypto.com/en/products/catapult/overview/

[7] http://hls-labsticc.univ-ubs.fr/

[8] http://mesl.ucsd.edu/spark/

4.1 High level synthesis tools based on C/C++ specification

For industry and academic research, the recent High-level synthesis (HLS) tools use C/C++/SystemC code targeting processor cores and FPGA implementation. However, in this Section, we only introduce the state-of-art of high level synthesis tools based on C/C++ code.

4.1.1 CATAPULT

Different manual methods are automated to reduce the time for hardware production in the Catapult synthesis flow [153]. The flow is centered around Catapult HLS tool, where the selection of microarchitecture is based on the constraints (provided by the designer/user). This tool creates the RTL architecture based on these constraints. Moreover, the target technology used, clock period or clock frequency are also specified as constraints. Some important points of Catapult based design methodology are discussed as follows:

- **Verifying generated RTL against original C code**, the function is one of the most important stages of the design flow. RTL architecture is wrapped around a SystemC transactor. By performing the wrapping, original C++ testbenches are compiled with the SystemC top module instantiating generated RTL module and finally comparator is used to compare the outputs. The wrapper code along with the makefiles is auto-generated to complete the verification flow.
- **Synthesis Constraints for Catapult Flow**: there are two types of constraints in the Catapult flow. The former is related to target technology and clock-frequency, etc. The latter is used to control the architecture. These constraints can be inserted using GUI or using directives. The directives facilitate loop unrolling, loop pipelining and hardware interface synthesis, etc. In addition, these constraints are not encoded in the source code, hence appropriate micro-architectures are created during synthesis stage.
- **C++ and optimization support**: It is necessary to underline that most of the C++ constructs are supported by this tool except the code, which requires dynamic memory allocation/deallocation such as use of malloc, free, new and delete. In other words, code should be statically deterministic, therefore all the properties, memory allocation can be performed during compile time. Catapult also supports pointer synthesis, classes & templates and bit-accurate data-types, etc. Catapult C provides loop pipelining, loop merging, loop unrolling, technology driven resource allocation and scheduling, etc., for the optimization side.

---

[9] http://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html

4.1.2 Program In Chip Out (PICO) tool's

PICO tool's [154] is mainly targeting SoC platforms. Before introducing the tool, it is crucial to characterize different IPs in the context of this tool and their approach. The IPs are categorised in the following groups:

- **Star IPs** are CPUs/DSP blocks. These blocks are generally fixed for many generations of SoCs. The IPs are manually created, therefore their instruction level characteristics are well-defined. In SoC, various models of such IPs are used at different granularities, including instruction level simulation model, RTL/gate-level model, etc.
- **Complex Application IPs** such as video-codec, wireless modem, etc. are different factor for the end product especially for embedded systems.
- **Connectivity and Control IPs**, such as DMA, USB, etc. are generally utilized in communication. It can be considered as system-level glue. Notwithstanding, their functionality is not needed and requires very minimal tailoring.
- **Memory** is generally the biggest contributor to silicon area. Memory models are compiled and built from the bottom-up.

These discussions of IPs above are necessary for SoC development. However, in complex application engine, it generally requires the bulk of effort for design and verification purposes.

PICO accepts the sequential C specification and tries to extract the parallelism from the sequential C code such as in the specific domain like signal processing applications. In such domains, a lot of parallelism is available during application processed by the hardware. A programming model is useful where a part of function has no dependency between the different tasks. If one task works on a block of data and other works on another block of data, then a lot of parallelism can be extracted in pipelined manner. The execution model of PICO is based on Kahn Process Network (KPN), where a set of sequential processes communicates via streams with block-on-read and unbounded buffering. These processes are the hardware blocks, which communicate with each other through streams. The restriction on unbounded buffering, however, is big, which is basically solved by imposing additional constraints on the execution model.

4.1.3 GAUT, SPARK tools

GAUT [155] is an academic high-level synthesis tool based on C as design language and applicable for digital signal processing applications. GAUT starts from bit-accurate specification written in C/C++ and extracts the possible parallelism before going through the conventional stages such as binding, allocation and scheduling tasks. GAUT has

mandatory synthesis constraints, which are throughput and clock-period. The compiler of GAUT derives gcc/g++ to extract a data flow graph (DFG) representation of the application. The synthesis process is not completely technology independent but can be useful for virtual platform development for micro-architectural analysis.

SPARK [156] presents a high-level design methodology based on the high-level synthesis tool, which takes Behavioral C as input design language and capable of generating RTL VHDL. The main contributions of the methodology based on SPARK are:

- Inclusion of code motion and code transformation techniques at compiler level to include maximum parallelism for high-level synthesis.
- Proposal of high-level synthesis starting with behavioral C input.

The approach presents in SPARK helps designer in understanding how and by what amount of the quality results can get affected by the language level transformations such as loop unrolling, loop-invariant code motion, etc. on generated circuit from HLS. The approach suggests that no single code transformation approach is universally applicable. However, such techniques with heuristics applied for particular application domain leads to better quality or results. The transformations and techniques applied in this methodology include exploitation of instruction-level parallelism, such as speculative code motions, percolation scheduling and trail blazing.

4.1.4 Vivado HLS tool

The Xilinx® Vivado® High-Level Synthesis (HLS) [157] compiler provides a programming environment similar to those available for application development on both standard and specialized processors. The HLS shares key technology with processor compilers for the interpretation, analysis, and optimization of C/C++ programs. Their main difference is in the execution target of the application.

By targeting an FPGA, the HLS enables a software engineer to optimize code for throughout, power, and latency without the need to address the performance bottleneck of a single memory space and limited computational resources. This allows the implementation of computationally intensive software algorithms into actual products, not just functionality demonstrators. Therefore, our work bases on the HLS. This subsection introduces how the HLS compiler works and how it differs from a traditional software compiler.

Application code targeting the HLS compiler uses the same categories as any processor compiler. HLS analyzes all programs in terms of:

- **Operations** refer to both the arithmetic and logical components of an application that are involved in computing a result value. When working with operations, the

main difference between HLS and other compilers is in the restrictions placed on the designer. With a processor compiler, the fixed processing architecture means that the user can only affect performance by limiting operation dependency and manipulating memory layout to maximize cache performance. In contrast, HLS is not constrained by a fixed processing platform. An algorithm-specific platform is built based on user input. This allows an HLS designer to affect the application performance in terms of throughput, latency and power;

- **Conditional statements** are program control flow statements that are typically implemented as if, if-else, or case statements. These coding structures are an integral part of most algorithms and are fully supported by all compilers, including HLS. The only difference between compilers is how these types of statements are implemented. With a processor compiler, conditional statements are translated into branch operations that might or might not result in a context switch. In an FPGA, a conditional statement does not have the same potential impact on performance as in a processor. HLS creates all the circuits described by each branch of the conditional statement. Therefore, the runtime execution of a conditional software statement involves the selection between two possible results rather than a context switch;

- **Loops** are a common programming construct for expressing iterative computation. Although this might be true with early versions of compilers for FPGAs, HLS fully supports loops and can even do transformations that are beyond the capabilities of a standard processor compiler. HLS can parallelize or pipeline the iterations of a loop to reduce computation latency and increase the input data rate. The user controls the level of iteration pipelining by setting the loop initialization interval (II). The II of a loop specifies the number of clock cycles between the start times of consecutive loop iterations;

- **Functions** are a programming hierarchy that can contain operators, loops and other functions. The treatment of functions in both HLS and processor compilers is similar to that of loops. In HLS, the main difference between loops and functions is related to terminology. HLS can parallelize the execution of both loops and functions. With loops, this transformation is typically referred to as pipelining, because there is a clear hierarchy difference between operators and loop iterations. With functions, operations outside of a loop body and within loops are in the same hierarchical context, which might lead to confusion if the term pipelining is used. To avoid potential confusion when working with HLS, the parallelization of function call execution is referred to as dataflow optimization. The dataflow optimization instructs HLS to create independent hardware modules for all functions at a given level of program hierarchy. These independent hardware modules are capable of concurrent execution and self-synchronize during data transfer.

In our work, the Fall Detection System is built based on video processing. Compare with the other tools, only Vivado HLS supports for performing real-time approach for our system by the followings reasons:

- OpenCV is a useful framework for developing computer vision designs. OpenCV applications can be also used in embedded systems by recompiling them for the ARM architecture and executing them in Zynq devices. In this case, the video processing is still implemented using OpenCV functions calls executing on a processor (such as the Cortex™-A9 processor cores in Zynq Processor System).
- Alternatively, the OpenCV function calls can be replaced by corresponding synthesizable functions from the Xilinx Vivado HLS video library. HLS Video Library is a C/C++ library provided with Vivado HLS to help accelerate computer vision/image processing applications on FPGA. It includes commonly used data structures, OpenCV interfaces, AXI4-Stream I/O, and video processing functions. HLS Video Library uses OpenCV libraries as reference model, most video processing functions has the similar interface and equivalent behavior with corresponding OpenCV functions. The pre-built OpenCV libraries (with FFmpeg support) are also shipped with Vivado HLS on different platforms.

Besides, we review about the high level synthesis based on C/C++ language. The low power techniques such as Clock gating, operand isolation, dynamic voltage and frequency scaling, etc. are discussed as following Section:

4.2 Low power techniques

Dynamic power is one of the most crucial terms of power consumption of a design, thus it is is necessary to reduce the dynamic power targeted for the power-aware processes. Some research has been worked in the area of power reduction at the RTL and high-levels of abstractions. Approaches focusing on RTL or higher level depend on the knowledge designers. In the following, we briefly introduce the approaches are used in low power design such as clock gating, operand isolation, dynamic voltage and frequency scaling (DVFS) and others:

- **Clock-gating** is one of the most frequently used techniques at RTL to reduce dynamic power dissipation without affecting the functionality of the design. Clock gating works by taking the enable conditions attached to registers and uses them to gate the clocks. Therefore, it is imperative that a design must contain these enable conditions in order to use and benefit from clock gating. Since it removes large numbers of muxes and replaces them with clock gating logic, this clock gating process can also save significant die area as well as power. Nikhil Tripathi et al. [158] refer to ultilise combinational clock gating to reduce the

switching activity on the clock network, thereby reducing dynamic power consumption in the design but this task does not alter the behavior of the register being gated. To evaluate FPGA clock network architectures with built-in clock gating capability, the authors in [159] describe a flexible placement algorithm to operate various gating granularities. Their results show that the dependent of the clock gating architecture and the fraction of time clock signals are enabled, clock power can be reduced by over 50%, and a fine granularity gating architecture yields significant power benefits. However, the research shows the advantages of clock-gating for reducing power consumption. Their works require RTL simulation and generation of Value Change Dump (VCD) and subsequent analysis, making the entire process extremely time consuming. It will require a great deal of effort to make the entire process few orders of magnitude faster, while power savings should be better or equal at least. Moreover, Sumit Ahuja [112] proposes approaches to enable clock-gating from the C description itself for various granularities of clock-gating such as fine grain at variable level and coarse grain at function or scope level. They also present their extension this approach for sequential clock-gating and propose how to use power models to guide power reduction process at high-level. The advantage of the approaches is facilitation of power reduction features at the high-level design with faster than other. [160] also proposes an approach based on clock-gating in the HLS flow. Nevertheless, their approach lacks a simulation driven realistic power reduction feature. It requires RTL synthesis to insert the gating logic while the necessary logic is inserted into the source code before generating the RTL.

- **Operand isolation** is a technique, which helps in reducing the redundant activities around datapath unit. It is considered as a complementary technique to clock-gating. Although clock-gating does not help in controlling the datapath activity, it just controls the clock toggles of registers. Munch et al. present an opportunity to reduce power at the RTL using operand isolation based technique to reduce the dynamic power at the RTL [161].
- **Dynamic Voltage and Frequency Scaling** (DVFS): The dynamic power consumption P_d of a CMOS circuit as introduced in Section 3.1 is determined by:

$$P_d = \alpha . C . V_{dd}^2 . f \tag{4.1}$$

with V_{dd} the supply voltage, f the clock frequency, α the switching activity level and C the capacitance of the circuit. Dynamic power consumption in a processor can be decreased by reducing two of its key contributors, supply voltage and clock frequency. In fact, since the power dissipated in a CMOS circuit is proportional to the square of the supply voltage, the most effective way to reduce power is to scale down the supply voltage [162]. However, reducing the supply voltage also increases the device delay, so frequency also needs to be reduced.

DVFS is a highly significant method to minimize the power dissipation and thus maximize the battery service time in battery-powered portable computing and communication devices. The key ideal behind DVFS techniques is to vary the voltage supply and the clock frequency of the system so as to provide "just enough" circuit speed to process the workload while meeting the total computation time and/or throughput constraints and thereby reduce the energy dissipation. Several strategies have been proposed to exploit certain aspects of DVFS and over a particular method to build pseudo intermediate frequencies for use in conjunction with the techniques of Dynamic Voltage Scaling (DVS) [163], [164].

Dynamic voltage scaling (DVS) [165], [166] refers to runtime change in the supply voltage levels supplied to various components in a system so as to reduce the overall system power dissipation while maintaining a total computation time and/or throughput requirement.

- Besides the previous techniques used for reducing power consumption design, in the other researchs, high-level synthesis for C-like HDLs includes stages such as scheduling, allocation and binding. Therefore, various techniques are proposed for different stages to affect the power consumption of the design once the RTL is created from HLS. Scheduling of various operations in a design is exploited for generating power-efficient designs. The problem of resource-constrained scheduling for low-power has been addressed in [167]. These approaches use Control Data Flow Graphs **(**CDFGs) to first determine the mobility of various operations based on the ASAP (As Soon As Possible) and ALAP (As Late As Possible) schedules. Using the computed nobilities and other relevant factors, priorities are assigned to various operations. Based on the assigned priorities, various operations of the design are then scheduled in each clock cycle such that the power consumption of the design is reduced.

The authors of [168] propose a methodology to explore different hardware configurations and also to achieve accurate design matrices for each configuration. They utilise C2R high level synthesis tool to directly generate RTL description of the hardware. They present case studies to develop or modify behavioral IP descriptions and use standard FPGA boards to profile the IP in very short time. The differences in the measured and actual IP design matrices are not significant as one is more concerned with relative difference among various configurations. A variety of compute-intense benchmarks like AES is used to demonstrate how platform specific optimizations as well as higher level micro architectural optimizations can be done using a commercial HLS tool, Xilinx Spartan/Virtex boards and Xilinx EDK design suite. Their results show how various architectures in hardware/software co-design flow are chosen while keeping energy efficiency in mind and they reduce design cycle time to reach the optimal results.

In the next Section, we discuss about the implementation of video applications and Fall Detection System on different processor cores, FPGA, DSP, etc. or heterogeneous solutions.

4.3 Video applications and Fall Detection System implemented on various platforms

Firstly, the overview of video processing tasks, for example pre-processing, 3D shapes reconstruction, data compression, etc. implemented on hardware acceleration (FPGA), processors or on combination of HW/SW are presented as follows:

J. Ayoub, O. Romain, B. Granado et al. [169] research an active vision technique implemented in an embedded system for 3D shapes reconstruction. The major aim of their work is to have a balance in the accuracy of all components in the system where the size and autonomy of such an embedded sensor are hard constraints. They improve the pre-processing algorithms by reducing the time needed to compute the spots centers. In addition, lens distortion of the camera is included in the model to increase accuracy when reconstructing objects. The distortion correction method is implemented on Xilinx Virtex II Pro FPGA (xc2vp30). The evaluation of experiments presents that the size and the time are reduced, precision increased, when the resources spend on processing are relatively acceptable in comparison to the benefits.

The work of Floris Driessen [170] proposes the combination of embedded processors and customized accelerators on heterogeneous computation platform, the Zynq-7000 all programmable SoC. This combination offers a high-end embedded processor combined with field programmable gate array (FPGA) based on reconfigurable logic. Peng Shen Ong et al. [171] propose the fall detection system which is implemented on Terasic's DE2- 115 development board including Altera Cyclone IV (EP4CE115) FPGA device, a 5 megapixels CMOS camera sensor and a LCD touch panel. This system is also designed with highly exploitation of the parallel and pipeline architecture of the FPGA.

The authors of [172] present the system built by Shimmer technology and applied the orthogonal matching pursuit (OMP) algorithm for advanced data compression. This system is simulated and implemented on the Virtex-5 and Zynq7 (FPGA) using Vivado high level synthesis tool. It is used to estimate the area, power and computation time for the fall detection with different scenarios. Benaoumeur Senouci et al [173] propose another heterogeneous implementation is based on Xilinx's SoC named Zynq methodology for a embedded fall detection system using a smart camera. They propose a HW/SW implementation to detect falls in a home environment using a single camera and an optimized descriptor adapted to real-time tasks. The main contributions of this work are the proposal of a co-design methodology. In their methodology, the HW/SW is partitioned by using high-level algorithmic description and high-level

synthesis tools. They give the fast prototyping which allows fast architecture exploration and optimisation to be performed. They design a hardware accelerator to efficient algorithm used in image analysis.

Frederik R. Grüll in [174] discusses biomedical image processing that is accelerated and reconstruction on FPGA in his thesis. The implementation is carried out with the MaxCompiler library from Maxeler Technologies and Xilinx. For acceleration, every processing pipeline must be re-designed. The background measurement is changed to exponential smoothing for every pixel over time. The spot finder is modified to operate after the background subtraction. The former least-square fit is simplified to a Gaussian estimator for feature extraction. The resulting pipeline system consists of two statically scheduled pipelines connected by a FIFO. The first pipeline operates on entire frames. The second extracts the features of every detected spot and operates on the Region of Interest (ROIs) only. The latter reconstructs the density distribution in a 3D volume from 2D images obtained with an electron microscope from multiple angles. The method belongs to the class of computed tomography, which is widely used in medicine and biology.

Secondly, we also review some implementations for Fall Detection System are combined various methods. Besides, Michal Kepski and Bogdan Kwolek deploy the Kinect and accelerate-meter in fall detection system [175]. They implement this system on PandaBoard ES, which is a low-power and low-cost single board computer development platform based on Texas Instruments OMAP4 line of processors. In addition, a method for detecting falls at homes of elderly using a two-stage fall detection system is presented by Erik E. Stone et al. [176]. The first stage of the detection system characterizes a person's vertical state in individual depth image frames. The segmentation on ground events from the vertical state time series is then obtained by tracking the person according time. The second stage uses an ensemble of decision trees to compute a confidence that a fall precede on a ground event. Their database consists of 454 falls where 445 falls are performed by trained stunt actors and 9 naturally occurring resident falls. The database is collected in nine years at the actual homes of older adults living at 13 apartments. This means that the data collection allows for characterization of system performance under real-world condition, which is not shown in other studies. Cross validation results are included for standing, sitting and lying down positions, within 4 m versus far fall locations and occluded versus not occluded fallers.

Martin Humenberger et al. in [177] present a bio-inspired, purely passive and embedded fall detection system by the combination of FPGA and DSP. Bio-inspired means that the use of two optical detector chips with event-driven pixels that are sensitive to relative light intensity changes only. The chips are used as stereo configuration which enables a 3D representation of the observed area with a stereo matching technique. In contrast to conventional digital cameras, this image sensor delivers asynchronous events instead of synchronous intensity or color images. Thus, the

privacy issue is systematically solved. Moreover, the stationary installed the fall detection system has a better acceptance for independent living compared to permanently worn devices. The fall detection is performed by a trained neural network. First, a meaningful feature vector is calculated from the point clouds. Then the neural network classifies the actual event as fall or non-fall. All processing is done on an embedded device consisting of an FPGA for stereo matching and a DSP for neural network calculation achieving several fall evaluations per second. The results of evaluation indicate that the fall detection system achieves a fall detection rate of more than 96% with false positives below 5% for the prerecorded database consisting of 679 fall scenarios.

Recently, with systems and software engineers programming in C/C++ and their hardware counterparts working in hardware description languages such as VHDL and Verilog, problems arising from the use of different design languages, incompatible tools and fragmented tool flows are becoming common. The SystemC[10] language and modeling platform, based on C++, are developed as the solution for representing functionality, communication, software and hardware. The reason is clear: increasing design complexity demands very fast executable specifications to validate system concepts, and only C/C++ delivers adequate levels of abstraction, hardware/software integration and performance. System design today also demands a single common language and modeling foundation in order to make a market for interoperable system-level design tools, services and IP a reality [178].

Apart from the modeling benefits available in C++ such as data abstraction, modularity, and object orientation, the advantages of SystemC include the establishment of a common design environment consisting of C++ libraries, models and tools, thereby setting up a foundation for hardware/software co-design; the ability to exchange IP easily and efficiently; and the ability to reuse test benches across different levels of modeling abstraction.

Despite, SystemC is built based on C/C++ language and all system specifications can be refined to mixed software and hardware implementations, but hardware implementations can be accurately modeled all the way to the RTL. Especially, SystemC isn't support for OpenCV integration in C/C++ language. Therefore, we select the tool which supports for not only using video libraries such as OpenCV but also combination HW/SW implementation.

From the state-of-art, we introduce about synthesis tools at the high level based on C/C++ such as Catapult, Pico, Gaut, Spark and Vivado_HLS; the low power techniques from low level to high level of abstract. The Vivado_HLS is selected for synthesising our work, the Fall Detection System with the advantage of including the OpenCV libraries and also supporting for heterogeneous platform. In addition, these

[10] http://accellera.org/downloads/standards/systemc

researches propose the implementation the Fall Detection on embedded system with the combination of FPGA (used for stereo matching) and DSP (used for neural network). Moreover, the authors present a HW/SW co-design with using wearable sensor based on Virtex 5 and Zynq platform. Some works implement combination of the Kinect and accelerometer for the Fall Detection System on PandaBoard ES, Texas Instruments OMAP4 platform. There are not in existence of the design exploration based on HW/SW co-design with low cost architectures for the Fall Detection System. Therefore, our research concentrates on exploring the architectures of the Fall Detection System which is applied power/time model and evaluated the recognition rate. The four tasks of Fall Detection System are implemented on processor cores and we explore the low cost architectures based on HW/SW co-design. In the following Section, we elaborate the description of our low cost architectures methodology for the Fall Detection System.

4.4 Overview of low cost architecture methodology

As mentioned in the Chapter 1, one contribution of this thesis is to define low cost architectures for the Fall Detection System which operates on heterogeneous platform. To explore the low cost architecture for our system: the experimental results of the Fall Detection System on processor cores are adjusted on different frequency scaling and image resolutions. The parameters such as execution time, power/energy and recognition rate are determined. Especially, the recognition rate such as accuracy, precision and recall performance of this system (see more on Section 2.4.2) are also given in the comparison picture of image resolutions and frame rates. In addition, to create the accuracy rate model for the extracted architectures of this system, thus the accuracy model is as a function of image resolutions and frame rates. The reason of accuracy model is only selected to create in exploration the low cost of architecture. The accuracy rate, which is a parameter of the test, is the proportion of true results (both true positives and true negatives) among the total number of cases examined in this system. Therefore, it is necessary for using this model to estimate the accuracy of our architectures. After that, execution time and power models are applied in combination of HW/SW. The relation between energy and accuracy rate of architecture is significant information in order to find out the best architecture for the Fall Detection System. The low cost architectures based on this methodology which compromises all parameters such as execution time, power/energy consumption, frame rate and accuracy rate are characterised (as depicted in Figure 4.1).

Our low cost architecture methodology starts with applying the power/time models which are presented in Chapter 3. The extracted power and execution time models are for separated tasks of the Fall Detection System based on processor cores and FPGA. In this case, we can address some situations for our methodology for heterogeneous architecture as follows:

- Software (processor cores): the implementation of this system with integration of operating systems is focused on. Furthermore, any algorithms which exhibit significant parallelism can be identified and are strong candidates for implementation on processor cores. This corresponds to a model where computationally intensive but parallel tasks can be off-loaded from the processor cores into hardware to achieve an overall performance increase.

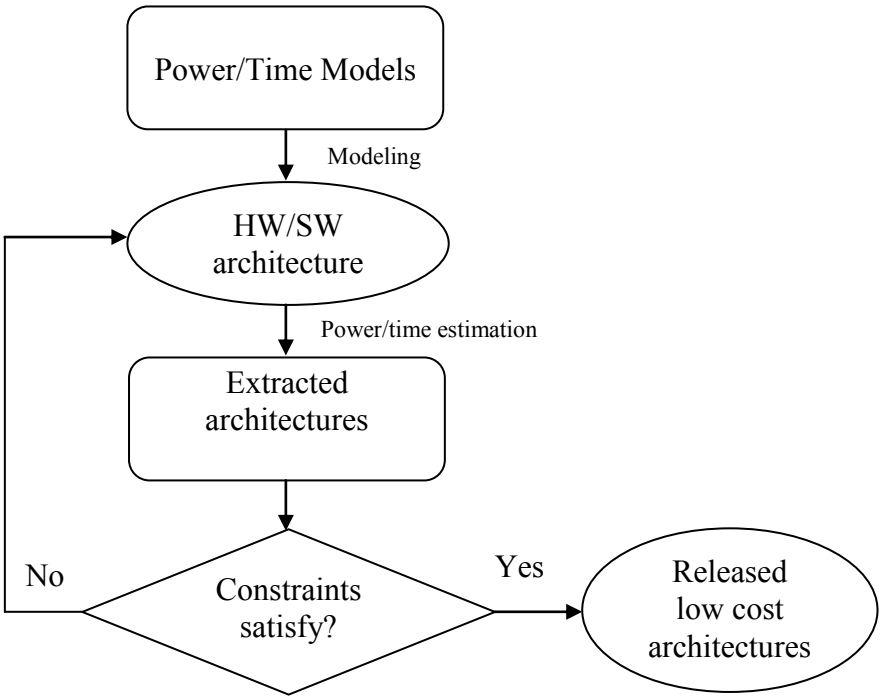


Figure 4.1-Our low cost architecture design methodology for Fall Detection System

- Hardware (FPGA) is selected as one of candidates for tasks in the Fall Detection System, which need to accelerate (execution time). The tasks which meet the time constraints are selected for this hardware purpose.
- Hardware/Software (HW/SW) co-design: This combination is currently trend, especially for the system to take full advantage of partitioning the system on software and hardware sides to improve execution time, reduce energy and apply the parallelism techniques. However, it can also deliver a sufficient recognition rate.

Continuously, HW/SW exploration architecture for all tasks of the Fall Detection System is estimated the execution time and power consumption. The results from this step help us to have earlier evaluations.

After, the first evaluations performed in previous step, some solutions for improving the performance for the Fall Detection System including intra-task and inter-

task techniques for HW/SW architectures are proposed. The parallelism in each core of processor and the accelerated modules on FPGA are combined.

It is a challenge to select suitable values for all parameters such as frame rates, power/energy and recognition rate which satisfy the constraints for the extracted architectures. Thus, suitable architectures for this system are compromised these parameters. One of important constraint is accuracy rate for HW/SW architectures, it influences on other constraint such as energy, frame rates. From the experiment, the frame rate of an architecture which accuracy rate satisfies at least 80% is defined. Therefore, the frame rate constraint is greater and equal 30 fps for our methodology.

Finally, after all extracted architectures are considered by the constraints. The low cost architectures, which compromising between the accuracy rate and energy, are released. The selected architecture depends on the aim of the designer for this system.

4.5 Software development and testing

In the software design, three entries include: the Board Support Package (BSP), the Operating System to communicate with the hardware and Software Applications run on top of the Operating System. Firstly, the Operating System such as Linux, Android, an embedded OS are selected; a Real-Time Operating System (RTOS) for deterministic, time-critical applications; or Standalone, a 'light' OS including only the most basic functions. Especially, for two available processor cores and two different types of OS on each core are deployed. In our system, the Linux operating System is selected to develop on ARM Cortex A9 processor which includes in Zynq 7000 AP SoC platform of Xilinx.

In this Section, the Fall Detection System in High Level Languages specified in C/C++ integrated OpenCV, cross-compiled along with libraries which implement the communication Application Programming Interfaces (APIs) and runtime layer using gcc/g++ toolchains are designed. The toolchains generate an .elf file downloaded to the processor ARM Cortex A9 on Zynq platform supported by SDK tools. Our system is executed by the configuration of image resolutions, frequencies of processor cores. The recognition rate is then evaluated. Moreover, the extracted accuracy model is based on the experiments of the Fall Detection System and use to apply in the exploration low cost architecture.

4.5.1 Case study

For exploring the various architectures for the Fall Detection, the case study is presented as follows:

- Input video is recorded by the Camera Web Cam-Philips SPC 900NC [11] that is mounted on the wall at the distance of 3m from the floor.
- Resolution of input video : 320x240 pixels, 680x360 pixels, 680x480 pixels and 704x576 pixels.
- Core frequency: 222 MHz, 333MHz and 667 MHz.
- Apply and extend the power and execution time models which are presented in Chapter 3 to estimate these values.
- Moreover, this system is explored the low cost architecture based on power/execution time model and accuracy rate model.

4.5.2 Primary implementation and experiment results for the Fall Detection System on software

The implementations are varied on different frequencies which are available on Cortex A9 processor with 667 MHz, 333MHz and 222MHz. An example of two first resolutions is 320x240 pixels and 680x360 pixels, as shown in Figure 4.2.

In addition, the measurement of power is taken by the Fusion Digital Power Designer GUI. The TI USB Adapter includes Power Management Bus (PMBus) which is already described in Section 3.4.3. PMBus is an open standard power-management protocol. This flexible and highly versatile standard allows for communication between Zynq platform and PC based on both analog and digital technologies and provides true interoperability, which will reduce design complexity and shorten time to market for power system designers. Therefore, the energy per frame is multiplied by the power consumption (P) and total execution time (T) as following equation:

$$Epf = P * T \tag{4.2}$$

Besides, the frame rate of this system is calculated by:

$$Frame\ rate = \frac{1}{T} \tag{4.3}$$

After defining the execution time, power/energy consumption and the frame rate of these video are calculated by using the equation 4.2 and 4.3.

[11] http://www.p4c.philips.com/cgi-bin/dcbint/cpindex.pl?ctn=SPC900NC/00&scy=gb&slg=en

Table 4.1-Fall Detection System implements on different frequencies

Image resolution	**Frequency MHz**	**Average execution time(ms)**						**Frame rate (fps)**
		Read data	*Object Seg.*	*Frame filter*	*Feature Extraction*	*Recog-nition*	*Total*	
320x240	667	10.9	8.6	75.3	10.9	1.65	107.4	9.3
	333	21.3	14.5	150	14.5	3.015	214.3	4.7
	222	36.3	25.6	225.8	21.4	4.6	313.6	3.2
680x360	667	24.7	17.7	234.9	24.6	4.4	306.8	3.3
	333	50.1	34.1	470.1	42.2	8.8	606.3	1.6
	222	71.7	52.9	705.3	72.9	13.3	917.6	1.1
640x480	667	35.2	40	295.4	54.8	5.5	431.6	2.3
	333	67.7	67.5	590	93	11	830.2	1.2
	222	101.3	101.3	726.4	139.2	16.6	1246.4	0.8
704x576	667	45.5	33.3	389.4	52.6	7.3	528.7	1.9
	333	73.7	66	778.7	104.5	14.5	1038.4	1
	222	108.9	312.7	1167.7	156.2	21.7	1554.8	0.6

Table 4.1 depicts the metrics of frame rates, execution time in different of the image resolutions and the frequencies. The mean of total execution time of the Fall Detection System is approximately 0.107s/frame. The Frame Filter task based on Morphology Filter takes around 2/3 times of total execution time. The similar observation has been obtained when using higher resolution of 680x360 pixels. In which the execution time is 0.234s/frame for Frame Filter and 0.3s/frame for total execution time. Frame Filter takes the most time, so that this evidence would be also considered for accelerating on hardware.

In addition, Table 4.2 illustrates the relation among the power/energy consumption and the different image resolutions and frequencies of cores. The higher frequency is scaled, the lower energy is taken. In contrast, the image resolutions and the energy consumption are proportional relationship.

Table 4.2-The Power/Energy of Fall Detection System on SW

Image resolution	Frequency (MHz)	Power mW	Energy mJ
320x240	667	420	45.11
	333	304.55	65.26
	222	254.55	79.83
680x360	667	420.91	129.13
	333	310	187.95
	222	264.55	242.75
640x480	667	437.27	188.73
	333	323.64	268.68
	222	269.09	335.39
704x576	667	446.36	235.99
	333	324.55	337.01
	222	281.82	438.17

Figure 4.2 illustrates the comparison execution time at two image resolutions, 320x240 and 680x360, processing on one processor of Zynq 7000 AP SoC platform. In each image resolution, the Frame Filter task, using Mathematic Morphology technique, executes the most value than the other ones. In this case, the measured power consumption of whole Fall Detection System is closed to 0.403W.

Therefore, the energy per frame is multiplied by the power consumption (P) and total execution time (T) presented as follows:

$$Epf = P*T= 0.403*0.107= 0.043 \ (J/frame) \quad (4.4)$$

As the result of this experiment, the frame rate of this system is calculated by:

$$Frame\ rate = 1/0.107 = 9.3 \ (fps) \quad (4.5)$$

It is found out that the over all of this system does not keep on operation at 30 frames per second. Thus, this parameter could have an effect on the recognition ability of this system. It is also a challenge in video design to get the reasonable precision, accuracy, recall performance.

From both Table 4.1 and Table 4.2, the processing speed on one core of ARM Cortex A9 processor is significant less than 10 fps compares with the 30 fps input.

Therefore, it is necessary to propose hardware accelerator or combine both of them to improve the execution time to satisfy the real time challenge for the Fall Detection System. We provide some solutions for this challenge in the next Section.

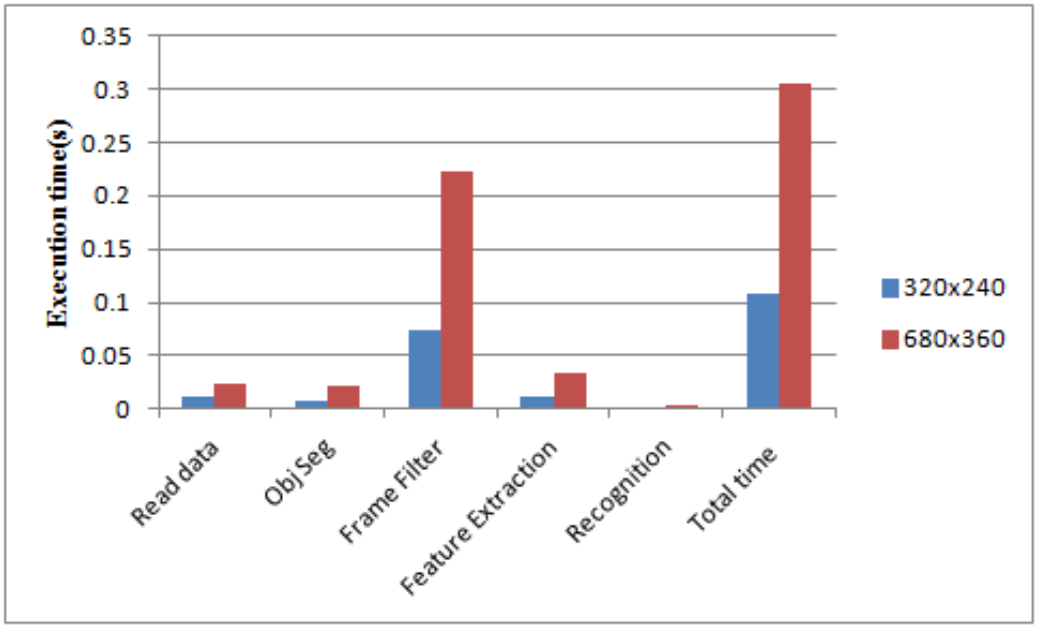


Figure 4.2-Comparison execution times at two image resolutions on one core

4.5.3 Performance evaluation for the Fall Detection System

4.5.3.1 The database

As discussing in Chapter 2, The DUT-HBU database [26] is used to evaluate the performance of this system. All video data are compressed in *.avi* format and captured by a single camera in a small room with the changeable conditions such as brightness, objects, direction of camera, etc. The fall direction is subdivided into three basic directions in this database: Direct fall, cross fall, side fall. In terms of non-fall videos, usual activities which can be misrecognised with fall action such as lying, sitting, creeping, bending are also classified into three directions above. In this study, we create two databases (as shown in Table 4.3):

Train set: Clear data consists of videos which have stable background. These videos are captured in a small room under good brightness condition. The object is not obscured by furniture in the room. Train set contains 21 videos of fall and 26 videos of daily activities.

Test set: Contents and activities in the video clips for testing are basically performed similar to the ones for training, just a small difference of environment

condition. In each clip, there is only an object with stable background and include 21 fall videos and the rest is 33 videos.

Table 4.3-Classification of videos

Action	**Video**	**Database**			
		Train	Test	Sum	Sum
Fall	Side -Fall (F1)	7	7	14	42
	Direct-Fall(F2)	8	6	14	
	Cross-Fall(F3)	6	8	14	
Non Fall	Bending (N1)	6	8	14	59
	Lying(N2)	5	8	13	
	Creeping(N3)	9	8	17	
	Sitting(N4)	6	9	15	
Sum		47	54	101	101

4.5.3.2 The classifying evaluation

The evaluation of recognition rate such as the Precision (PR), Recall (RC) and Accuracy (Acc) are given in Section 2.4.2 and shown in the Equation 4.3.

$$RC = \frac{TP}{TP + FN}, PR = \frac{TP}{TP + FP}, Acc = \frac{TP + TN}{TP + TN + FP + FN} \quad (4.6)$$

Where TP, TN, FN, and FP are defined as follows:

True positives (TP): amount of fall actions which are correctly classified as fall.

False positives (FP): amount of non-fall actions which are wrongly considered to be fall.

False negatives (FN): amount of fall actions which are wrongly rejected and classified as non-fall actions.

True negative (TN): amount of non-fall actions which are correctly classified as non-fall.

4.5.3.3 The confusion matrix

A confusion matrix presents classification system which includes actual and predicted classifications. Performance of such systems is commonly evaluated using the data in the matrix. Table 4.4 shows the confusion matrix for two classes which are categorised FALL or NON FALL for both database of *Train* and *Test* implemented on ARM Cortex

A9 of Zynq-7000 AP SoC platform. The evaluation is experimented on 101 videos, in which 47 videos are in *Train* set and 54 ones are in *Test* set. For instance, 7 videos are categorised in Side-Fall (F1) of *Train* set and the system can recognise all events in these videos are FALL. In case of Sitting (N4), we, however, have 6 videos, there are 5 videos which are detected as NONFALL and a video is misrecognised as FALL.

Table 4.4-Confusion matrix

	System					
Database	*Action*	*Video*	*Train*		*Test*	
			Fall	NonFall	Fall	NonFall
	Fall	F1	7	0	6	1
		F2	7	1	5	1
		F3	5	1	5	3
	Non Fall	N1	1	5	1	7
		N2	1	4	2	6
		N3	2	7	1	7
		N4	1	5	2	7
Sum			47		54	

From the confusion matrix, the Recall, Precision and Accuracy are calculated and depicted in Figure 4.3. The result of pure data in Train set is higher than Test set in all Recall, Precision, and Accuracy. The reason is that Template Matching uses "hard threshold" and the combination of features is quite simple to detect a fall event. Four models of the fall are not enough to describe all falls may occur in this system.

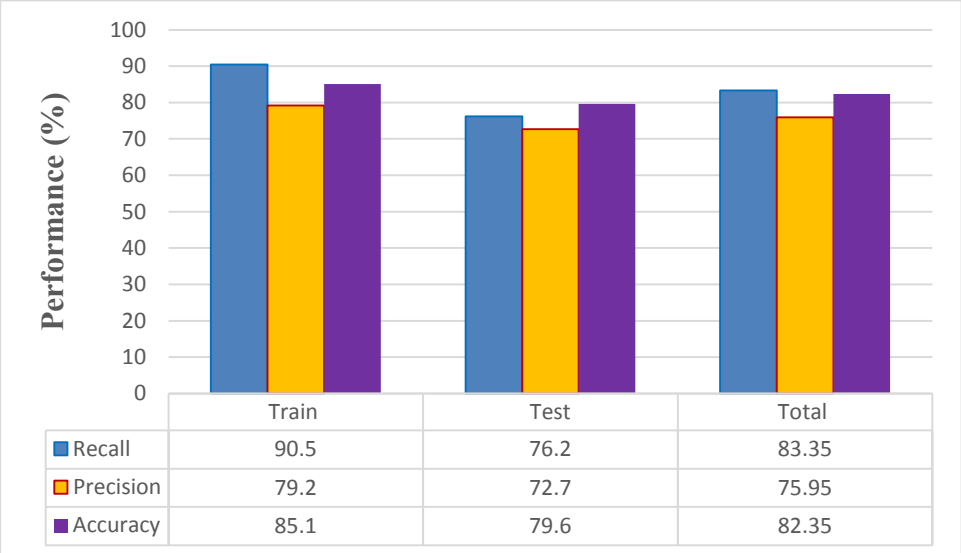


Figure 4.3-The results of Template Matching Algorithm with resolution_320x240

Figure 4.4 is shown the performance comparison of two image sizes: 320x240 and 640x480 with the frame rate is 30fps for the offline video processing of Train set. The lower performance of 640x480 resolutions of input image is calculated with 66.7 % of Recall, 57.1 % of Precision and 78.3% of Accuracy in the same conditions such as classification of Train set, the threshold and frame rate comparing with 90.5 % of Recall, 79.2% of Precision and 85.1 % of Accuracy in the 320x240 resolution.

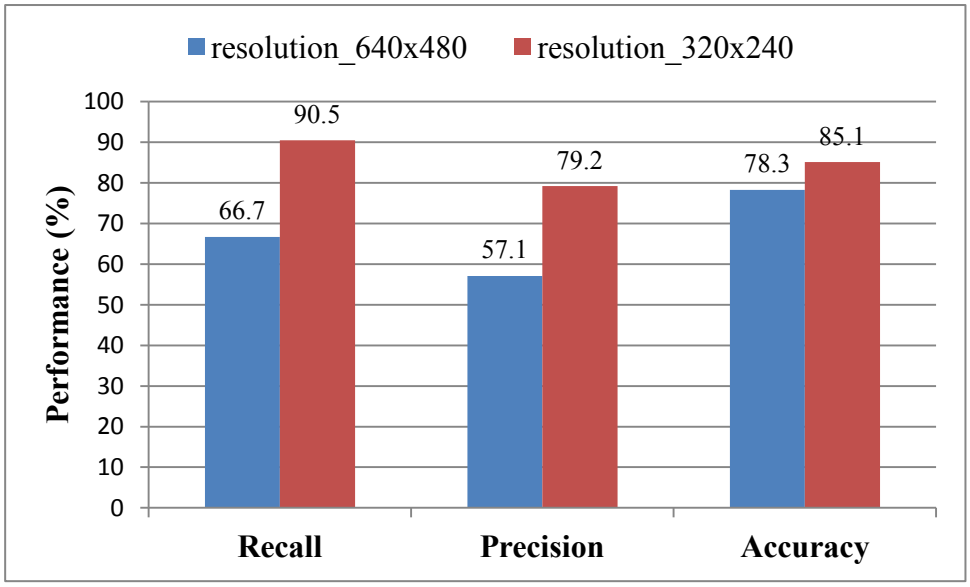


Figure 4.4-The performance comparison of two resolutions

4.5.3.4 The accuracy model

It is crucial to define the relationship between accuracy performances in the system with the other parameters such as frame rate, resolution, etc. Our aim is to extract an accuracy model for the heterogeneous architectures. In our work, some experiment on two different resolutions (320x240 and 680x480) and various frame rates is presented in Table 4.5.

Table 4.5-The relationship between Accuracy performance with resolution and frame rate of input video

Resolution	**fps**	**Accuracy (%)**	**Accuracy estimation (%)**	**Error rate (%)**
320x240	10	66.7	65.7	1.5
320x240	20	72.2	74	2.5
320x240	30	85.1	84.3	1
640x480	30	78.3	78.3	0

The performance of accuracy is extracted by using the regression law, the following equation presents the dependence of accuracy with image resolution and frame rate

$$Accuracy = 55.3+0.000026*s +1.034*fps \quad (4.7)$$

Where, s is the image size of the input video with two resolutions (320x240) and (640x480); *Fps* is the frame rate which is listed in Table 4.5.

To estimate reliable of the accuracy model, the accuracy of this system is validated and applied the above model as shown in Table 4.5. The error rate is achieved less than 3%. The accuracy model for our system is defined from 10 to 50 fps. This model is used to define the accuracy rate of our system with heterogeneous architectures in exploring low cost architectures as presented in the later Section.

4.6 Hardware development and testing

As discuss in the subsection 4.4.2, the Frame Filter using the Morphology Mathematics is selected for implementing on Hardware. However, to extract more architectures for this system, we also choose the Object Segmentation task for hardware purpose. By this way, the power and execution time of these tasks are estimated by the Vivado_HLS tool. Table 4.6 illustrates the summary the extracted power and execution time model presented in Chapter 3. Whenever, the resolution images increases the execution time and the energy is higher.

Table 4.6-Summary the results on hardware

Application	Resolution	Execution Time (ms)	Power (W)	Energy (mJ)
Object Segmentation	320x240	0.33	0.124	0.043
	640x480	1.32	0.124	0.183
Mathematical Morphology	320x240	4.35	0.184	6.851
	640x480	17.72	0.184	27.944
	1920x1080	218	0.184	40.111

4.7 Application of parallelism techniques

The aim of this Section determines optimal architectures which has the best compromise between execution time, power/energy consumption and accuracy rate. Besides, extending the extracted power and execution time models derived from Chapter 3, we then propose to use the parallelism techniques for this system on the Zynq platform. Parallelism techniques for the video application tasks may exist among several frames

(inter-task parallelism) as well as within a single frame (intra-task parallelism). These techniques are described as follows:

- Intra-task parallelism is handled several tasks corresponding with four tasks (Object Segmentation, Filter, Feature Extraction and Recognition) in Fall Detection System within a frame. The two first tasks perform in parallel by separating an image (frame) in two slices. It means that each slice is exploited on each core of processor at the same time.
- Inter-task parallelism: there are also four tasks (Object Segmentation, Filter, Feature Extraction and Recognition) in Fall Detection System. Meanwhile, one or two tasks are assigned to exactly each core of processor and/or in hardware. For instance, first task is run on first core of processor, the second task is exploited on FPGA, and the last tasks are executed on the second core of processor. Tasks are performed in parallel by consecutive frames.

4.7.1 Intra-task parallelism technique

In this subsection, the *intra-task technique* is performed for the Fall Detection System which has four tasks assigned to processor cores and/or hardware as follows:

- **Task 1:** Object segmentation.
- **Task 2:** Frame filter.
- **Task 3:** Feature extraction.
- **Task 4:** Recognition

We have many ways to schedule these tasks of our system based on the *intra-task technique.* In our context, we suppose that **Task 1** and **Task 2** have three solutions of execution: 1 core, 2 cores and FPGA (hardware). The **Task 3** and **Task 4** are just exploited on 1 core of processors. To understand more detail of the explored architecture cases (as presented in Table 4.12), two proposed cases, *A2* and *A5*, are described below:

Architecture 2: the **Task 1, Task 3** and **Task 4** are executed in one core of processor. **Task 2** is run on the FPGA. In the various frequencies the execution time (ms), power consumption (mW) and energy per frame (mJ) are shown in Table 4.7 and Table 4.8.

Table 4.7-Task regroups Architecture 2 with 320x240 resolutions

Frequency (MHz)	**Task 1**	**Task 2**	**Task 3**	**Task 4**
	Object segmentation	Frame filter	Feature extraction	Recognition
667	9.736	4.35	10.9	1.65
333	19.5	4.35	14.5	3.015
222	29.25	4.35	21.4	4.6

Table 4.8-The relationship of power and energy per frame at different frequencies

Frequency (MHz)	**T(ms)**	**P(mW)**	**Epf (mJ)**	**Fps**
667	26.6	1266	33.7	37.6
333	41.4	933.3	38.6	24.2
222	59.6	793.8	47.3	16.8

Our experiment is implemented on Zynq 7000 AP SoC which has three configurations of frequency such as 222 MHz, 333 MHz and 667MHz. As shown in Table 4.8, by corresponding with the maximum frequency, 667MHz, we can deduce the maximum of frame rate with 37.6fps.

As this frame rate is very high without significant increase on the accuracy, a desirable frame rate for the output of our system can be specifed and predetermined. We can adjust the frequency of cores that helps the designer decreasing the power. Thus, the frequencies of cores are recalculated by the equation 4.8 and Table 4.9 presents the value of frequencies when having given frame rate.

$$Fcores = Fmax * \frac{fps}{fpsmax} \tag{4.8}$$

Table 4.9-Example of frequency for different frame rate

Fps	**F_{core} (MHz)**
25	443.5
30	532.2
35	620.9

We continuously discuss about the other Architecture, A4, which is scheduled not only on processor cores but also on FPGA as shown below:

Architecture 4: the **Task 1** and **Task 2** are processed on the FPGA. **Task 3** and **Task 4** are executed on one core of processor. With the different frequencies, the execution time (ms), power consumption (mW) and energy per frame (mJ) are illustrated in Table 4.10 and Table 4.11.

Table 4.10-Task regroups case 4

Frequency (MHz)	**Task 1**	**Task 2**	**Task 3**	**Task 4**
	Object segmentation	Frame filter	Feature extraction	Recognition
667	0.330	4.350	10.9	1.65
333	0.330	4.350	14.5	3.015
222	0.330	4.350	21.4	4.6

Table 4.11-The estimation power/energy per frame at different frequencies (A4)

Frequency (Mhz)	**T(ms)**	**P(mW)**	**Epf (mJ)**	**Fps**
667	17.2	857.3	14.7	58.1
333	22.2	649.6	14.4	45
222	30.7	554.5	17	32.6

For architecture 4, the Object Segmentation task in Fall Detection System is accelerated on FPGA. So, the execution time is considerably improved. However, as shown in Table 4.11 the maximum of frame rate reaches at 58.1 for the maximum frequency, 667MHz. This maximum frame rate is higher than the limited frame rate of the accuracy model. Therefore, we can recalculate the frequency of processor cores by using the equation 4.8 with the boundary of input frame rate is lower than 50fps.

Table 4.12-Intra-task parallelism technique

Architectures		Task 1	Task 2	Task 3	Task 4
A1	Core 1	ObjSeg	Filter	FeatureEx	Recog
A2	Core 1	ObjSeg		FeatureEx	Recog
	FPGA		Filter		
A3	Core 1		Filter	FeatureEx	Recog
	FPGA	ObjSeg			
A4	Core 1			FeatureEx	Recog
	FPGA	ObjSeg	Filter		
A5	Core 1	ObjSeg	Filter	FeatureEx	Recog
	Core 2	ObjSeg	Filter		
A 6	Core 1	ObjSeg		FeatureEx	Recog
	Core 2	ObjSeg			
	FPGA		Filter		
A7	Core 1		Filter	FeatureEx	Recog
	Core 2		Filter		
	FPGA	ObjSeg			

Table 4.8 and Table 4.12 show that the *intra-task technique* supports the system in significant improvement of execution time. However, a trade-off between execution time which impacts the accuracy of fall detection and power/energy consumption is also considered. The power consumption model is extracted in Chapter 3 for processor cores, FPGA and the model when no application is running. These models are presented as follows:

$$P_{core}(N, F_{cores}, s) = 152 + 0.000416 * \mathrm{s} + 0.39 * \mathrm{F_{cores}} + 30.5 * \mathrm{N} \quad (4.9)$$

$$\mathrm{P_{FPGA}}(\tau_i) = 123.5 + 8.07\mathrm{P_{BRAM}} + 0.02\mathrm{P_{LUT}} + 0.00432\mathrm{P_{FF}} \quad (4.10)$$

$$P_{no\_application}(F_{cores}) = 0.1317.F_{cores} + 193.36 \quad (4.11)$$

In the *intra-task technique*, the power consumption is estimated based on two situations as follows:

- Situation 1: we consider this situation that all tasks are implemented on software (1 core and/or 2 cores)

$$P(\tau_i) = P_{core}(N, F_{cores}, s) \quad (4.12)$$

$$Epf = \sum_{i=1}^{4} P(\tau_i) x\, T(\tau_i) \quad (4.13)$$

Where, P(τ_i) is the power consumption on task τ_i. In which τ_i is i^{th} of task and *i= (1:4)*; *N* is the number of processor cores; F_{cores} is the frequency of processor cores; the image size or the image resolution is assigned as a half of input image size for parallel execution.

- Situation 2: one task is implemented on software (1 core and/or 2 cores) or hardware (FPGA)

If τ_i is run on hardware, the power consumption is calculated by:

$$P_{HW}(\tau_i) = P_{FPGA}(\tau_i) + P_{no\_application}(F) \quad (4.14)$$

If τ_i is run on software, the power consumption is determined by:

$$P_{SW}(\tau_i) = P_{FPGAstatic}(\tau_i) + P_{core}(N, F_{cores}, s) \quad (4.15)$$

For this case, the power consumption model based on *intra-task technique* is presented as follows:

$$P_{intra\_task}(\tau_i) = a_i P_{SW}(\tau_i) + b_i P_{HW}(\tau_i) \quad (4.16)$$

$$with \ \forall i \in (1:4);\ a_i, b_i = \{0,1\};$$

Where, *i* is the running task number; a_i =1 and b_i= 0 if τ_i is executed on processor cores; a_i =0 and b_i= 1 if τ_i is running on FPGA. For example, in architecture A6 (as shown in Table 4.12), if we want to calculate the power consumption on Task 1 which is run in parallel on core 1 and core 2, we have a_1 = 1 and b_1 = 0. In addition, the power consumption on Task 2 is implemented on FPGA, a_2=0 and b_2= 1.

The *intra-task technique* is applied for the Fall Detection System with both image resolutions presented: 320x240 and 680x360; three frequencies: 667MHz, 333MHz and 222MHz. The architecture cases are shown in Table 4.12.

We continuously consider the next *inter-task technique* for our system to explore more architecture cases with the estimation of the energy and the impact of accuracy. The main scheduling based on this technique is discussed briefly as following subsection.

4.7.2. Inter-task parallelism technique

The *intra-task technique* schedules the execution time of four tasks within a single frame. Each task is assigned on 1 core, 2 cores and FPGA. Notwithstanding, in the *inter-task technique*, the scheduling of the four tasks is on the core 1, the core 2 and FPGA in consecutive frames. For instance, **Task 1** is executed on the core 1, **Task 2** is implemented on the FPGA and the **Tasks 3** and **Task 4** are assigned to the core 2. The time slot (TS) corresponds to the execution time of the slowest task on processor cores. Therefore, the parallel and pipeline scheduling for this system with the assigned time slot are built and presented as follows:

Table 4.13-The inter-task technique with scheduling of five consecutive frames

	TS1	TS2	TS3	TS4	TS5	TS6	TS7	TS8
Core 1	(τ_1, I_1)	(τ_1, I_2)	(τ_1, I_3)	(τ_1, I_4)	(τ_1 , I_5)	(τ_1, I_6)	(τ_1, I_7)	(τ_1, I_8)
FPGA		(τ_2, I_1)	(τ_2, I_2)	(τ_2, I_3)	(τ_2, I_4)	(τ_2 , I_5)	(τ_2, I_6)	(τ_2, I_7)
Core 2			(τ_3, I_1)	(τ_3, I_2)	(τ_3, I_3)	(τ_3, I_4)	(τ_3 , I_5)	(τ_3, I_6)

Where,

T_{core} (τ_1, I_1) is the execution time of Task 1 (of I_1 frame) on core 1;

T_{FPGA} (τ_2, I_1) is the execution time of Task 2 (of I_1 frame) on FPGA;

T_{core} (τ_3, I_1) is the execution time of both Task 3 and Task 4 (of I_1 frame) on core 2.

In this context, we have only three tasks and the execution time of each task equals to a time slot (TS). The power consumption of our system based on *inter-task technique* is first considered at TS3 with the complete pipeline scheduling all tasks on SW and HW. We can extract the power model for this case:

$$P_{inter\_task}(F_{core}, s) = P_{core}(N, F_{core}, s) + P_{FPGA}(\tau_2);\ with\ N = 2 \qquad (4.17)$$

Where, $P_{core}(N, F_{cores}, s)$ is the power consumption of three tasks (task1, task3 and task4) which is implemented on processor cores. $P_{FPGA}(\tau_2)$ is the power of task 2 on FPGA. N is the number of processor cores and for inter_task technique N=2; F_{core} is the frequency of processor cores; s is the image size or the image resolution.

For example, Table 4.1 shows that Feature Extraction task takes the maximum execution time, 10.9 ms in case of 320x240 input image resolution at maximum frequency of core 667MHz. When the TS = 10.9 ms, we have the corresponding frame rate= 91.7 fps. This frame rate value (91.7 fps) is the maximum frame rate (fpsmax) for TS, but is higher than the boundary of maximum frame rate 50fps for accuracy model of

our system. So, we have to reconfigurate the frequency of core to get the suitable frame rate. Besides, from the equation 4.17, the power consumption on processor cores is impacted by different parameters such as frequency of cores, image resolutions and number of cores. Therefore, by adjusting the frame rate, we can recalculate the frequency of core, which help to reduce the power/energy consumption of our system and is determined by the following formula:

$$F_{cores} = F_{max} * \frac{fps}{fpsmax} \tag{4.18}$$

Then, energy consumption per frame is deduced:

$$Epf = P_{inter\_task}(F_{core}, s) * \frac{fpsmax}{fps} * TS \tag{4.19}$$

After using the parallelism techniques to schedule the execution time for Fall Detection System, we explore the low cost architectures which is compromised the parameters such as execution time, power/energy consumption, the accuracy rate and the frame rate by using the Design Space Exploration Architecture methodology. This methodology is illustrated as following Section:

4.8 Design Space Exploration Architecture for Fall Detection System

Design space exploration (DSE) is an analyzing process of functional implementation alternatives. It is used to define an optimal solution. The designer traditionally starts with an informal specification and develops a reference executable in kinds of high-level language. A methodology for the low cost architectures of the Fall Detection System with (execution time, power consumption, accuracy rate) constraints is determined DSE at the early stages of the development. The methodology is then validated for functional correctness as follows the system specification. It is used to get harsh estimations of its performance requirements. The initial step is followed by manual or semi-automatic generation of several alternative designs which are subjected by image resolutions, number of cores, frequency of cores, etc. Finally, the most suitable designs are chosen based on various performance metrics such as accuracy rate, frame rate, power/energy consumption.

On the other side, the aim of DSE finds an efficient design configuration. The design leads to an efficient HW/SW architecture, therefore, the requirement energy of system is minimized and the performance requirements are satisfy the constraints. Thus, it is worth noting that both performance of the system and a minimization of energy are based on architecture selections. The design exploration process is illustrated in Figure 4.5. The process consists of two entry points: architecture templates and application characteristics.

- The architecture templates are defined by architecture cases with different frequencies, number of cores, and hardware/software combination. The

architectures are scheduled by using inter-task and intra-task parallelism techniques.

- The application characteristics are different image resolutions and the frame rate. From the matching of architecture templates and application characteristics, numbers of low power architectures are defined. Then, the accuracy rates are estimated by performance model which is extracted in Section 4.4.3.4.

Moreover, the design exploration process is an initial platform-independent program which is subsequently enriched and integrated with information coming from the definition of the target architecture model of the implementations on that platform.

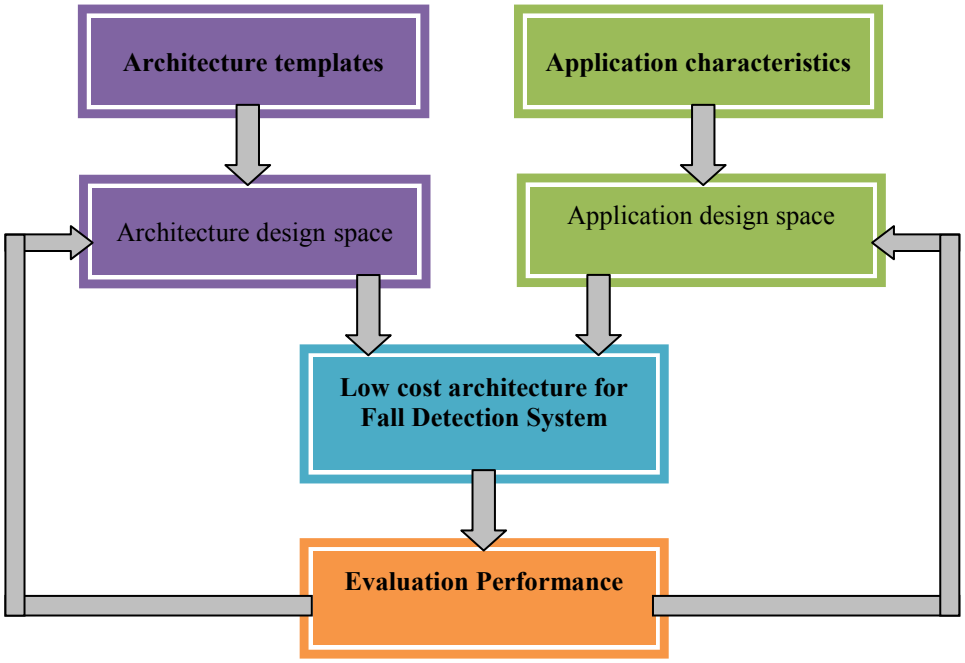


Figure 4.5-Design Space Exploration for Fall Detection System

4.8.1 Methodology

Thanks for Matlab which supports us to get the visual performance of this system. The low cost architecture exploration can be defined as:

- Firstly, two resolutions of images: 320x240 or 640x480 is selected in this simulation.
- Then, three frequencies such as 222MHz, 333MHz and 667MHz are configured.
- Architecture cases based on inter-task and intra-task parallelism techniques.
- From the selected input, number of architectures for Fall Detection System using the execution time/power models and accuracy rate are explored.
- The requirement of frame rate for processing in the system is greater than 10fps and less than 50fps. These best architectures not only consume the lowest energy

but also satisfy the highest accuracy rate performance among the explored architectures. In fact, it is necessary to compromise a best architecture among these power/energy consumption, frame rate and accuracy. The best architecture depends on the purpose of the users' demand.

Our methodology is to determine the low cost architectures for the Fall Detection System by following descriptions:

Initial: - Frequency of cores F_{core}= [667MHz, 333 MHz, 222 MHz]

- Maximum frame rate fps_{max} = 50;
- Image resolution s= [320x240, 640x480];
- Number_of_core N = [1, 2];
- Time slot (TS) is the slowest execution time of task on processor cores
- Architecture_case C = [1 1 1 1;
  1 0 1 1;
  0 1 1 1;
  0 0 1 1];

// select the architectures: C[i,j] = 1 mean task j^{th} executes on processor cores and C[i,j] = 0 mean task j^{th} runs on FPGA; i is the i^{th} test case

Computation:

For f =1 to 3 // select the frequency of cores: 667MHz, 333 MHz, 222 MHz

If select= **Intra-task technique** then

for s=1 to 2 // select image resolutions: 320x240 or 640x480

for N=1 to 2 // select the number of processor cores; N is extended to 3 or 4, it depends on the platforms.

For j =1 to 4

If i = 1 then // select the architecture_ case 1 [1 1 1 1]

$$P_{intra\_task}(\tau_j) = P_{core}(N, F_{cores}, s)$$

$$T(\tau_j) = T_{core}(\tau_j, {}^{S}/_{N}, F_{core}, N)$$ // as shown in Section 3.7.1

Else // select the three rest of architecture_cases

for i = 2 to 4

$$P_{intra\_task}(\tau_j) = C_{i,j}P_{SW}(\tau_j) + (1 - C_{i,j})P_{HW}(\tau_j)$$

$$T(\tau_j) = C_{i,j}T_{core}(\tau_j, {}^{S}/_{N}, F_{core}, N) + (1 - C_{i,j})T_{FPGA}$$

//as shown in Section 3.7.1

end for j

$$Epf = \sum_{j=1}^{4} P_{intra\_task}(\tau_j) * T(\tau_j)$$

//Frame rate output:

$$fps = 1/\sum_{j=1}^{4} T(\tau_j)$$

// only accepting the architectures if they sastify with 10 ≤fps ≤50

// Accuracy of architectures:

$Acc = -0.000026 * s + 1.034 * fps + 55.3$

//Output:

$Architecture = [Power, Execution\ time, Energy, fps, Acc]$

End for i

End if

End for N

End for s

Else if select= **Inter-task technique:**

for s=1 to 2 // select image resolutions: 320x240 or 640x480

N=2;

i=2;

$P_{inter\_task}(F_{core}, s) = P_{core}(N, F_{cores}, s) + P_{FPGA}(\tau_2)$

$Epf = P_{inter\_task}(F_{core}, s) * \frac{fpsmax}{fps} * TS$

//Frame rate output: $fps = \frac{1}{TS}$

// only accepting the architectures if they sastify with 10 ≤fps ≤50

// Accuracy of architectures:

$Acc = -0.000026 * s + 1.034 * fps + 55.3$

//Output:

$Architecture = [Power, Execution\ time, Energy, fps, Acc]$

end for s

end If

end.

4.8.2 Model results

Based on the previous methodology, the exploration low cost architectures for the Fall Detection based on *intra-task technique* is related power/energy, accuracy rate, and frame rate are defined as shown in Table 4.14.

Figure 4.6 and Table 4.14 depict the simulation results of architecture exploration. It is the trade-off power/energy and the accuracy rate performance of the Fall Detection System. Three architectures which have compromised between energy and accuracy performance belong to A3, A4 and A7. A4 takes lowest energy 20mJ with 80.9 % accuracy, A7 spends 24.1mJ for energy with 97.6% accuracy and A3 consumes 29.5 mJ with the highest accuracy around 98%.

Table 4.14-The relationship between energy and accuracy in different architectures

Architecture (intra-task)	Energy (mJ)	Accuracy (%)
Architecture 4 (A4) *1 core (333MHz) + HW_ 640x480*	20	80.9
Architecture 2 (A2) *1 core (667MHz) + HW_640x480*	35.6	85.2
Architecture 6 (A6) *2 core (667MHz) + HW_640x480*	31.2	91.6
Architecture 5 (A5) *2core (333MHz)_320x240*	34.6	93.5
Architecture 1 (A1) *1 core(667MHz)_320x240*	42.6	95.7
Architecture 7 (A7) *2core (333MHz) +HW_320x240*	24.1	97.6
Architecture 3 (A3) *1 core (667MHz) + HW_320x240*	29.5	98.3

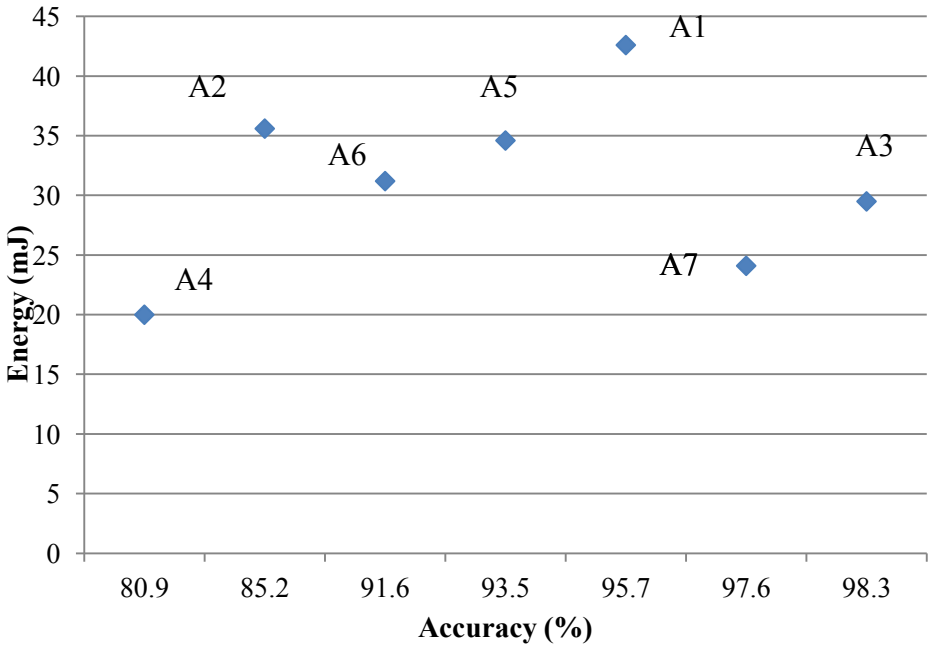


Figure 4.6-Architecture exploration for Fall Detection System

As discuss in Section 4.7.2 for the inter-task technique, the time slot is assigned to the slowest task on processor cores at the different frequencies. The task1, task3, task4 are just executed in parallel core 1, core 2 and task 2 is run on FPGA. We also consider the two configurations: image resolution (680x480 and 320x240); frequency of cores (667MHz, 333MHz and 222MHz).

Table 4.15 presents the energy per frame and accuracy performance of four architectures for Fall Detection System. The energy per frame of architecture A'1 takes 11.6 mJ with rather high accuracy 93.7 % and 34.8 mJ with only accuracy 66.2% for architecture A'2. The two suitable architectures, A'1 and A'2, are selected for inter_task technique corresponding with two input image resolutions.

Table 4.15-The power/energy consumption and architectures based on inter-task technique

Architecture (inter-task)	**Time slot ms**	**P_{inter_task} (mW)**	**Epf (mJ)**	**Accuracy (%)**
A'1 (320x240_222MHz)	25.6	453.8	11.6	93.7
A'2 (680x480_667MHz)	60.3	634.8	34.8	66.2
A'3 (680x480_333MHz)	104	501.2	46.6	58.4
A'4 (680x480_222MHz)	155.8	456.8	63.6	54.7

The Design Space Exploration methodology which applies the parallelism techniques: *intra-task* and *inter-task*, help the designer to extract the different architectures for Fall Detection System. In addition, the DSE can extend the various configurations of the image resolutions, frequencies of processor cores and the number of cores (for example three cores or four cores).

4.9. Conclusion

This chapter defines the low cost architectures to overcome the constraints such as execution time, frame rate, power/ energy consumption and accuracy rate. The power consumption and execution time models are extended in comparison in Chapter 3 to estimate these parameters for the complete Fall Detection System. We also propose a model for the accuracy rate performance which is function of image resolutions and

frame rate. This accuracy model then applied for evaluating features of the different candidate the architecture.

We defined a Design Space Exploration methodology in order to explore heterogeneous architectures with hardware/software combination for our system by applying two parallelism techniques: *intra-task and inter-task static scheduling*. As example, for the *intra-task technique some exploration of low cost architecture are*: if we are interested in the energy consumption, the architecture A4 would be selected with lowest energy per frame 20mJ and accuracy rate with 80.9 %. In contrast, if the main parameter is the accuracy performance, the architecture A3 would be selected and presents the highest accuracy about 98 % and the energy with 29.5 mJ. However, the architecture which gets the best compromise between energy and accuracy performance is architecture A7 which consumes 24.1mJ for energy with 97.6% accuracy. For the *inter-task* approach, if we consider A'1 and A'2 architectures for two image resolution 320x240 and 680x480, the energy per frame takes 11.6 mJ with rather high accuracy 93.7 % (A'1) and 34.8 mJ with only accuracy 66.2% (A'2). We notice that when the image resolutions increase, the fps significantly diminishes that induces a decrease in the accuracy rate. The proposed *inter_task* static scheduling must be enhanced to get better accuracy performance.

In addition, we can select one of the two optimal architectures (A7 and A'1) to develop on a certain heterogeneous platform without spending a lot of time on experiments. Furthermore, the DSE can be extended for the processors with more cores such as 3 cores, 4 cores, 8cores, etc., various image resolutions and frequencies.

Chapter 5. Conclusions and perspectives

This thesis has presented an exploration of the Fall Detection under algorithmic and architectural point of view. The aim is to find out low cost architectures based on power consumption, execution time model and accuracy rate performance. The study of Fall Detection System, which is established on a video processing and was investigated, in which:

(1) The human object inside the image has been segmented from the background; the technical construction of geometric modeling for the human body and extract features to recognise the fall actions;

(2) Solutions of different recognition model for fall action are training with high precision and reliability;

(3) Creating scenarios and building databases video are classified with different actions: fall, non-fall (sitting, lying, creeping, etc.) in three camera directions (face, sides and cross); and applying the evaluation criteria is for testing the model to detect falls.

(4) Moreover, we make the comparison (the recall, precision, and accuracy performance) among the suitable algorithms using in Fall Detection System such as Background Subtraction-Neural Network (BGS-NN); Background Subtraction-Template Matching (BGS-TM); Background Subtraction-Hidden Markov Model (BGS-HMM); Gaussian Mixture Model (GMM-HMM);

(5) In the architectural point, then, execution time and power consumption models have been extracted based on not only algorithm parameters (cache miss rate, instruction per cycle, and image size) but also architectural parameters (number of cores and operating frequency). Functional Level Power Analysis (FLPA) is applied for creating the power models on processor cores and power models on FPGA are based on the hardware resources;

(6) From extracted models, the power consumption and execution time models are extended with hardware/software architectures for the Fall Detection System;

(7) Our approach targets to explore low cost architecture for this system by using the parallelism techniques with the aim to find out the architectures which offer the best compromise between energy and fall detection accuracy rate.

5.1 Conclusion

In this thesis, at first, the efforts have been devoted to improve the Fall Detection algorithms. Heterogeneous architectures for video applications in Fall Detection System are studied to extract the models for execution time and power consumption. Then, we proposed a new Design Space Exploration methodology to define the low cost architecture for Fall Detection System which presents a sufficient accuracy rate. Our methodology also applies parallelism techniques.

In Chapter 2, the application which is divided in four modules, Object Segmentation, Filter, Feature Extraction and Recognition, is elaborated with different algorithms. Moreover, we described the speciality of DUT-HBU database, including data information, camera, environment, actor/actress and scripts of classification, which are used for simulation and evaluation purpose and implementation. The Fall Detection System was assessed by using the accuracy, recall, and precision performances. Notwithstanding, our database used data recorded from falls of young people simulated at the discretion of each impersonator in the videos. Hence, our database lacked of a standardized procedure or needed to compare with a public database. Meanwhile, the real fall detection aims to older people or patients who have some distinctions with young people in the database. The simulation results were then used to compare the performances among the algorithms such as BackGround Subtraction/Hidden Markov Model (BGS-HMM), Gaussian Mixture Model/Hidden Markov Model (GMM-HMM), BackGround Subtraction/Neural Network (BGS-NN) and BackGround Subtraction/Template Matching (BGS-TM). One of the most important advantages is that our system is well-accepted due to the fact that we have a local processing, it is convenient and more applicable at the contemporary indoor for elderly living alone or rehabilitants in hospital. We analysed the shortcomings which give the false recognition output such as environment brightness, occlusion of object, many movement objects appearing in a frame at the same time. Especially, the extracted features that are not strong enough to distinguish between face fall and sitting actions were also considered. These evidences are not enough to assess the system performance in a real situation. Moreover, these algorithms were evaluated with the off-line videos, making the execution time and estimation backgrounds are not too complex.

In Chapter 3, the video tasks which are defined in the Fall Detection System were at first analysed separately and power consumption and execution time models for them have been proposed. A power and execution time modeling methodology was proposed for processor cores based on FLPA and FPGA related with the hardware resources and then for heterogeneous architecture. All the implementation for extracting these models were executed on Zynq7000 AP SoC including processor cores (ARM Cortex A9) and FPGA with supporting of Vivado_HLS. The scenery of processor core experiments was implemented by considering different configurations. TI USB Interface Adapter PMBus associated with TI Fusion Digital Power Designer GUI was used to

measure the power consumption on processor cores to evaluate the error rate of these power consumption models. Moreover, the power consumption and execution time models for processor cores were extended for the Fall Detection System related to different parameters such as image resolutions, frequencies and number of cores. In addition, the power consumption and execution time models of FPGA were only extracted for two tasks (Object Segmentation and Filter) for which the execution time can be improve significantly. The defined models allowed to evaluate the power consumption and execution time with different configurations of heterogeneous architectures for Fall Detection System. The more architecture for the Fall Detection System will be explored, if the more tasks like Feature Extraction and Recognition are selected to implement on FPGA.

In Chapter 4, the Design Space Exploration methodology which is used to define the low cost architectures the Fall Detection System was introduced. The low cost architectures meet the constraints such as execution time, frame rate, and accuracy rate. The extracted power consumption and execution time models were extended to explore different hardware/software architectures for the Fall Detection System. In addition, these estimations are useful to explore low cost architectures based on two parallelism techniques: *intra-task and inter-task* static scheduling on heterogeneous architecture for the Fall Detection System. The low cost architectures were selected with the compromising of energy and accuracy rate performance of the Fall Detection System. However, the accuracy rate must be extracted with more parameters recognition features, filter method, etc. In our system, the BGS-TM algorithm with well-matched for was implemented on processor cores with accuracy of 62% and energy per frame of 43mJ/f. When the parallel techniques based on hardware/software architecture are applied, the frame rate of our system is considerably increased and the accuracy rate reaches 98.3% with energy per frame of 29.5mJ/f. Based on this methodology, the optimal architectures were selected to develop on a certain heterogeneous platform without carrying on time-consuming experiments. Moreover, the DSE can be extended for the processors with more cores such as 3 cores, 4 cores, 8 cores, etc., various image resolutions and frequencies.

5.2 Perspectives

Although we have presented the modeling approach of execution time and power consumption for processor cores and hardware by using HLS tools, there are still many aspects of our approach must be improved on algorithms, application domains, evaluation tools, database, and architecture definition, etc. Several perspectives that our work has created and how these opportunities may be addressed will be outlined below.

5.2.1 Improve the Fall Detection Algorithm

Most of recognition errors are due to impact of environment factors, so the human object is extracted from usual complex background stage. In addition, as the simulation results in Chapter 2, if we develop the algorithms for implementation on hardware/software architecture such as Background Subtraction/Hidden Markov Model (BGS-HMM), Gaussian Mixture Model/Hidden Markov Model (GMM-HMM), Background Subtraction/Neural Network (BGS-NN), it will give higher performance compare to our system using Background subtraction/Template Matching.

This system need to be tested on other databases to get more the evaluation of accuracy, recall, and precision performance. Therefore, it is necessary to develop and collect addition database from various sources to have a stronger database which uses for training our system in optimal way. It is one of the reliability of these algorithms before we decide to design a stand-alone system for the practical application in the hospital or at home.

The focus should also be on the stage of feature extraction to extract more new features to distinguish similar actions from the object or improve the effects of noise such as removing the silhouette of the object or the changeable brightness of environment. The objects obscured by other objects in the room have also to be handled. Modeling 3D human body is a prevalent method to create depth to object to enhance effective recognition of the actions.

In addition, the developments of not only the accuracy rate model but also the recall and precision models are needed in order to achieve more performances of the low cost architecture for Fall Detection System.

Furthermore, in our Fall Detection System, we have already detected the fall of object and it would be more interesting to analyse many kinds of the human motions after the fall in order to send different degree of alarm.

5.2.2 Power/execution time optimization

Most of video processing applications require real-time solutions. A usual approach to achieve this performance goal is to exploit the heterogeneous architecture consists of different types: GPUs (Graphical Processing Units) or FPGAs (Field Programmable Gate Arrays) or processor cores. The GPUs are very efficient at manipulating computer vision, video and image processing, and their highly parallel structure makes them more effective than general-purpose processor cores for algorithms where processing of large blocks of data is done in parallel. It means that the combination with GPUs is considerably improved the execution time for a system. Hence, there are some solutions for exploring more heterogeneous architectures for our system such as GPUs/ processor cores or even combination three types: processor cores, GPUs and FPGAs.

In addition, to design a stand-alone Fall Detection System, the autonomy must be powered on battery. And how long the battery power allows to the autonomous operation for this system must be considered.

5.2.3 Camera network

In this thesis, our system has concentrated on a single camera. While video surveillance system needs to work in a network environment, it is necessary to analyse the moving object patterns that evolve over long periods of time and large space. To understand the moving patterns are observed by a multi-camera network, the first step is to infer the spatial organization of the environment under surveillance, which is achieved by camera node localization, camera calibration, or camera network topology inference for different purposes.

In addition, video surveillance may interface with other wireless technologies, such as body area networks (BANs), and radio frequency identification (RFID) technology. In this case, more autonomous and intelligent E-healthcare applications can be generated to improve people's quality of life. For example, with a patient's personal information stored in RFID tag, and physiological data retrieved by a BAN worn by the patient, the doctor or other care-givers can remotely diagnose a problem by relying on video surveillance system.

5.2.4 Solutions for combination of many equipments for the Fall Detection System

After exploring the low cost architecture based on the relation of accuracy rate and energy, we are going to design the real system which is applied and developed in Vietnam. In addition, we ongoing to connect with the other equipments such as mobile phone, e-health bracelets, wireless sensors, smart watch, etc. which will make more flexible for users in case of getting out the room. Besides, this system needs to not only detect the fall of elderly but also diagnose the diseases, for example Alzheimer, absent-mindedness and heart disease and so on.

Appendix

Database description

Database name: **DUT-HBU** database

File format: *.avi

Authors:

- Hieu V.Nguyen
- Tuan V.Pham
- Hong T.K Nguyen
- Duy H.Le
- Hoan V.Tran
- Khue Tra
- Phung T.K Lai
- Viet Q.Truong

Electronic & Telecommunication Engineering Department - Danang University of Technology, Danang, Vietnam

Camera : Philips Webcam SPC 900NC [179]

- Sensor: CCD
- Resolution : 320x240
- Interpolated snapshot resolution: 1.3 MP
- Max. frame rate: 90 fps
- Colour depth: 24 bit

Environment: Lab room of HBU group

- Size : 3x5 m^2
- Brightness : Good (natural light)
- Background : quite stable
- Moving object : 1
- Camera position: see in Figure A

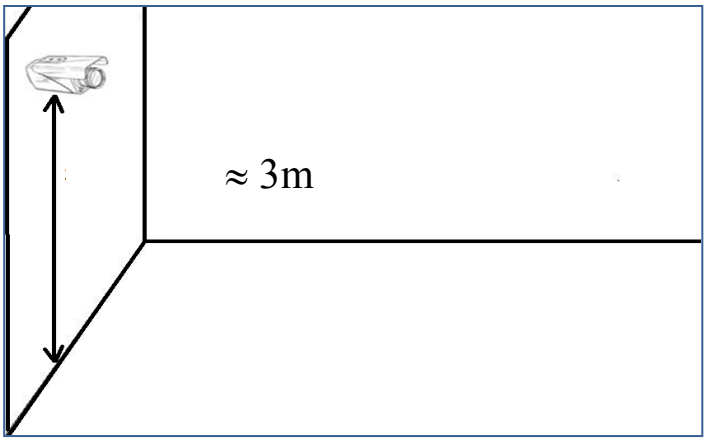


Figure A-Camera position

Actor/Actress: 4 persons

Table A-Descriptions of the actor /actress

	Duy	**Hoan**	**Phung**	**Hong**
Sex	Male	Male	Female	Female
Age	22	22	22	35
Height	1m65	1m75	1m56	1m53
Weight	56Kg	60Kg	40Kg	48Kg

Bibliography

[1] Community Health, "Vietnam still lacks of 78.700 health forces," *www.congan.com.vn*, 2011. [Online]. Available: http://www.congan.com.vn/?mod=detnews&catid=402&id=285056. [Accessed: 09-Jan-2015].

[2] "The Old Alone : Beware of Falling." [Online]. Available: http://dansotn.com/tin-dan-so/dansoxahoi/7477-nguoi-gia-o-mot-minh-de-phong-te-nga.html. [Accessed: 09-Jan-2015].

[3] GSO, "The population change and family planning survey 2006." 2007.

[4] L. V Hoi, H. D. Phuc, T. V Dung, N. T. K. Chuc, and L. Lindholm, "Remaining life expectancy among older people in a rural area of Vietnam: trends and socioeconomic inequalities during a period of multiple transitions," *BMC Public Health*, vol. 9, p. 471, Jan. 2009.

[5] B. Mirmahboub, S. Samavi, N. Karimi, and S. Shirani, "Automatic monocular system for human fall detection based on variations in silhouette area.," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 2, pp. 427–36, Feb. 2013.

[6] Y. Li, K. C. Ho, and M. Popescu, "A microphone array system for automatic fall detection," *IEEE Trans. Biomed. Eng.*, vol. 59, no. 5, pp. 1291–301, May 2012.

[7] L. Tong, Q. Song, Y. Ge, M. Liu, and M. Ieee, "HMM-Based Human Fall Detection and Prediction Method Using Tri-Axial Accelerometer," *IEEE Sens. J.*, vol. 13, no. 5, pp. 1849–1856, May 2013.

[8] G. S. Quirino, A. R. L. Ribeiro, and E. D. Moreno, "Asymmetric Encryption in Wireless Sensor Networks," 2012. [Online]. Available: http://www.intechopen.com/books/wireless-sensor-networks-technology-and-protocols/asymmetric-encryption-in-wireless-sensor-networks.

[9] T. Tamura, A. Kawarada, M. Nambu, A. Tsukada, K. Sasaki, and K. Yamakoshi, "E-Healthcare at an Experimental Welfare Techno House in Japan," *Open Med. Inform. J.*, pp. 1–7, 2007.

[10] E. Coiera, "Communication Systems in Healthcare," *Clin. Biochem. Rev.*, vol. 27, no. May, pp. 89–98, 2006.

[11] M. Wang, C. Huang, and H. Lin, "An Intelligent Surveillance System Based on an Omnidirectional Vision Sensor," *2006 IEEE Conf. Cybern. Intell. Syst.*, pp. 1–6, Jun. 2006.

[12] "Clinical Skills | AV Installation London." [Online]. Available: http://avinstallationlondon.co.uk/services/clinical-skills/. [Accessed: 21-Oct-2014].

[13] M. Mubashir, L. Shao, and L. Seed, “A survey on fall detection: Principles and approaches,” *Neurocomputing*, vol. 100, pp. 144–152, Jan. 2013.

[14] H. Yang, J. Tian, Y. Chu, Q. Tang, and J. Liu, “Spatiotemporal Smooth Models for Moving Object Detection,” *IEEE Signal Process. Lett.*, vol. 15, pp. 497–500, 2008.

[15] H. Foroughi, A. Naseri, A. Saberi, and H. Sadoghi Yazdi, “An eigenspace-based approach for human fall detection using Integrated Time Motion Image and Neural Network,” in *the 9th International Conference on Signal Processing*, 2008, pp. 1499–1503.

[16] V. Vishwakarma, C. Mandal, and S. Sural, “Automatic Detection of Human Fall in Video,” *Springer-Verlag Berlin Heidelb.*, 2007.

[17] R. Cucchiara, C. Grana, A. Prati, A. Member, and R. Vezzani, “Probabilistic Posture Classification for Human-Behavior Analysis,” *IEEE Trans. Syst. Man, Cybern. A Syst. Humans*, vol. 35, no. 1, pp. 42–54, 2005.

[18] L. Hazelhoff, J. Han, and P. H. N. De With, “Video-Based Fall Detection in the Home Using Principal Component Analysis,” in *Advanced Concepts for Intelligent Vision Systems*, 2008.

[19] X. Yu, “Approaches and Principles of Fall Detection for Elderly and Patient,” in *the 10th International Conference on e-health Networking, Applications and Services, 2008. HealthCom*, 2008, pp. 42–47.

[20] E. Gudis, P. Lu, D. Berends, K. Kaighn, G. van der Wal, G. Buchanan, S. Chai, and M. Piacentino, “An Embedded Vision Services Framework for Heterogeneous Accelerators,” *2013 IEEE Conf. Comput. Vis. Pattern Recognit. Work.*, pp. 598–603, Jun. 2013.

[21] M. Russell and S. Fischaber, “OpenCV based road sign recognition on Zynq,” in *IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 596–601.

[22] S. Gilliland, P. Govindan, T. Gonnot, and J. Saniie, “Performance evaluation of FPGA based embedded ARM processor for ultrasonic imaging,” in *IEEE International Ultrasonics Symposium, IUS*, 2013, pp. 519–522.

[23] R. Dobai and L. Sekanina, “Image filter evolution on the Xilinx Zynq Platform,” *2013 NASA/ESA Conf. Adapt. Hardw. Syst.*, pp. 164–171, Jun. 2013.

[24] G. Baruffa, F. Fiorucci, F. Frescura, P. Micanti, L. Verducci, and B. Villarini, “A reprogrammable computing platform for JPEG 2000 and H.264 SHD video coding,” in *the 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia*, 2010, pp. 107–113.

[25] F. Büsching, H. Post, M. Gietzelt, and L. Wolf, “Fall Detection on the Road,” in *the 15th International Conference on e-Health Networking, Applications & Services (Healthcom)*, 2013, pp. 439–443.

[26] Y. T. Ngo, H. V. Nguyen, and T. V. Pham, “Study on fall detection based on intelligent video analysis,” *2012 Int. Conf. Adv. Technol. Commun.*, pp. 114–117, Oct. 2012.

[27] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, “Fall Detection from Human Shape and Motion History Using Video Surveillance,” in *the 21st International Conference on Advanced Information Networking and Applications Workshops, AINAW*, 2007, vol. 2, pp. 875–880.

[28] A. M. Tabar, A. Keshavarz, and H. Aghajan, “Smart home care network using sensor fusion and distributed vision-based reasoning,” in *Proceedings of the 4th ACM international workshop on Video surveillance and sensor networks - VSSN ’06*, pp. 145–154.

[29] C. Rougier, J. Meunier, A. St-Arnaud, and J. Rousseau, “Robust Video Surveillance for Fall Detection Based on Human Shape Deformation,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 21, no. 5, pp. 611–622, May 2011.

[30] E. Auvinet, F. Multon, A. Saint-arnaud, J. Rousseau, and J. Meunier, “Fall Detection With Multiple Cameras : An Occlusion-Resistant Method Based on 3-D Silhouette Vertical Distribution,” *IEEE Trans. Inf. Technol. Biomed.*, vol. 15, no. 2, pp. 290–300, 2011.

[31] C. Kim and J. Hwang, “Fast and automatic video object segmentation and tracking for content-based applications,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 2, pp. 122–129, 2002.

[32] J.-S. Hu and T.-M. Su, “Robust Background Subtraction with Shadow and Highlight Removal for Indoor Surveillance,” *EURASIP J. Adv. Signal Process.*, vol. 2007, no. 1, pp. 1–14, 2007.

[33] R. Cucchiara, C. Grana, M. Piccardi, and a. Prati, “Detecting objects, shadows and ghosts in video streams by exploiting color and motion information,” *Proc. 11th Int. Conf. Image Anal. Process.*, pp. 360–365.

[34] S. Wang, M. Wu, and Y. Xie, “An effective segmentation cue for moving object segmentation from a moving camera,” in *7th International Symposium on Advanced Optical Manufacturing and Testing Technologies: Optoelectronics Materials and Devices for Sensing and Imaging*, 2014, vol. 9284.

[35] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2, 1999.

[36] H.-Y. Cheng and J.-N. Hwang, “Adaptive particle sampling and adaptive appearance for multiple video object tracking,” *Signal Processing*, vol. 89, no. 9, pp. 1844–1849, Sep. 2009.

[37] D. Comaniciu and V. Ramesh, “Real-Time Tracking of Non-Rigid Objects using Mean Shift,” in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, 2000, no. 7, pp. 1–8.

[38] L. He, “Generation Of Human Body Models,” Master thesis in Computer Science, The University of Auckland, 2005.

[39] E. Aybar, “Sobel Edge Detection Method For Matlab.” [Online]. Available: http://www.figes.com.tr/matlab/teknik-makaleler/eaybar_tam_metin.pdf.

[40] Z. Othman, M. Rafiq, and A. Kadir, “Comparison of Canny and Sobel Edge Detection in MRI Images,” in *Computer Science, Biomechanics & Tissue Engineering Group, and Information System*, 2009, pp. 133–136.

[41] C. S. Panda, “Filtering Corrupted Image and Edge Detection in Restored Grayscale Image Using Derivative Filters,” *Int. J. Image Process.*, vol. 3, no. 3, pp. 105–119, 2009.

[42] J. Wang, Z. Liu, and Y. Wu, “Learning Actionlet Ensemble for 3D Human Action Recognition,” in *Human Action Recognition with Depth Cameras*, SpringerBriefs in Computer Science, 2014, pp. 11–40.

[43] J. K. Aggarwal and L. Xia, “Human activity recognition from 3D data: A review,” *Pattern Recognit. Lett.*, vol. 48, pp. 70–80, Oct. 2014.

[44] T. B. Moeslund, A. Hilton, and V. Krüger, “A survey of advances in vision-based human motion capture and analysis,” *Comput. Vis. Image Underst.*, vol. 104, no. 2–3, pp. 90–126, Nov. 2006.

[45] M. W. Lee and I. Cohen, “A model-based approach for estimating human 3D poses in static images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 6, pp. 905–916, 2006.

[46] C. Sminchisescu and B. Triggs, “Kinematic Jump Processes For Monocular 3D Human Tracking,” in *Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

[47] A. Agarwal and B. Triggs, “Recovering 3D human pose from monocular images,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 1, pp. 44–58, Jan. 2006.

[48] G. Shakhnarovich, P. Viola, and T. Darrell, “Fast pose estimation with parameter-sensitive hashing,” in *Proceedings 9th IEEE International Conference on Computer Vision*, 2003, pp. 750–757.

[49] R. J. Holt, A. N. Netravali, T. S. Huang, and R. J. Qian, “Determining articulated motion from perspective views: a decomposition approach,” in *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*, 1994, pp. 126–137.

[50] S.-R. Ke, L. Zhu, J.-N. Hwang, H.-I. Pai, K.-M. Lan, and C.-P. Liao, “Real-Time 3D Human Pose Estimation from Monocular View with Applications to Event Detection and Video Gaming,” in *the 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2010, pp. 489–496.

[51] J. K. Aggarwal and Q. Cai, “Human Motion Analysis: A Review,” *Comput. Vis. Image Underst.*, vol. 73, no. 3, pp. 428–440, Mar. 1999.

[52] L. Wang, W. Hu, and T. Tan, “Recent developments in human motion analysis,” *Pattern Recognit.*, vol. 36, no. 3, pp. 585–601, Mar. 2003.

[53] M.-C. Roh and S.-W. Lee, “Human gesture recognition using a simplified dynamic Bayesian network,” *Multimed. Syst.*, Oct. 2014.

[54] S. Hongeng, R. Nevatia, and F. Bremond, “Video-based event recognition: Activity representation and probabilistic recognition methods,” *Comput. Vis. Image Underst.*, vol. 96, no. 2, pp. 129–162, 2004.

[55] M. Yang, D. J. Kriegman, S. Member, and N. Ahuja, “Detecting Faces in Images : A Survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 1, pp. 34–58, 2002.

[56] D. Anderson, J. M. Keller, M. Skubic, X. Chen, and Z. He, “Recognizing falls from silhouettes.,” *Conf. Proc. IEEE Eng. Med. Biol. Soc.*, vol. 1, pp. 6388–6391, 2006.

[57] M. Piccardi, “Background subtraction techniques: a review,” *IEEE Int. Conf. Syst. Man Cybern.*, vol. 4, pp. 3099–3104, 2004.

[58] Z. Zivkovic, “Improved adaptive Gaussian mixture model for background subtraction,” *Proc. 17th Int. Conf. Pattern Recognition, ICPR*, vol. 2, 2004.

[59] S. Mukhopadhyay and B. Chanda, “An edge preserving noise smoothing technique using multiscale morphology,” *Signal Processing*, vol. 82, no. 4, pp. 527–544, Apr. 2002.

[60] B. Naegel, “Using mathematical morphology for the anatomical labeling of vertebrae from 3D CT-scan images,” *Comput. Med. Imaging Graph.*, vol. 31, no. 3, pp. 141–156, Apr. 2007.

[61] R. M. Haralick and S. R. Sternberg, “Image Analysis Using Mathematical Morphology,” *IEEE Trans. Pattern Anal. Mach. Intell.*, no. 4, pp. 532–550, 1987.

[62] D. F. Pava and C. Science, *Object Detection In Low Resolution Video Sequences*. ProQuest, 2009.

[63] S. Gupta and S. G. Mazumdar, “Sobel Edge Detection Algorithm,” *Int. J. Comput. Sci. Manag. Res.*, vol. 2, no. 2, pp. 1578–1583, 2013.

[64] A. Yilmaz, O. Javed, and M. Shah, “Object Tracking: A Survey,” *ACM Comput. Surv.*, vol. 38, no. 4, pp. 1–45, Dec. .

[65] C. J. Veenman, M. J. T. Reinders, and E. Backer, “Resolving Motion Correspondence for Densely Moving Points,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 1, pp. 54–72.

[66] D. Serby, E. K. Meier, and L. Van Gool, “Probabilistic object tracking using multiple features,” *Proc. 17th Int. Conf. Pattern Recognition, ICPR*, vol. 2, pp. 184–187, 2004.

[67] D. Comaniciu, S. Member, and V. Ramesh, “Kernel-Based Object Tracking,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 5, pp. 564–577.

[68] A. Yilmaz, X. Li, and M. Shah, “Contour-based object tracking with occlusion handling in video acquired using mobile cameras,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1531–6, Nov. 2004.

[69] A. Ali and J. K. Aggarwal, “Segmentation and Recognition of Continuos Human Activity,” in *IEEE Workshop on Detection and Recognition of Events in Video*, 2001, pp. 28–35.

[70] R. Cutler and L. S. Davis, “Robust real-time periodic motion detection, analysis, and applications,” *IEEE Trans. Pattern Anal. Mach. Intell.*, pp. 781–796, 2000.

[71] N. Paragios and R. Deriche, “Geodesic Active Regions and Level Set Methods for Supervised Texture Segmentation,” *Int. J. Comput. Vis.*, vol. 46, no. 3, pp. 223–247, 2002.

[72] P. Fieguth and D. Terzopoulos, “Color-based tracking of heads and other mobile objects at video frame rates,” in *IEEE Conference on Computer Vision and Pattern Recognition*, 1997, pp. 21–27.

[73] G. J. Edwards, C. J. Taylor, T. F. Cootes, and M. Manchester, “Interpreting Face Images using Active Appearance Models,” in *International Conference on Face and Gesture Recognition*, 1998, pp. 300–305.

[74] B. Moghaddam and a. Pentland, “Probabilistic visual learning for object representation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 7, pp. 696–710, Jul. 1997.

[75] J. W. Davis, “Hierarchical motion history images for recognizing human motion,” in *Proceedings. IEEE Workshop on Detection and Recognition of Events in Video*, 2001.

[76] M. Valera and S. a. Velastin, “Intelligent distributed surveillance systems: a review,” *IEE Proceedings-Vision, Image, Signal Process.*, vol. 152, no. 2, p. 192, 2005.

[77] K. Tra, V. Q. Truong, and T. V Pham, “Threshold-based Versus Artificial Neural Network For Fall Detection,” in *the 5th Science Conference of the University of Da Nang*, 2012.

[78] D. Svozil, V. Kvasnieka, and J. Pospichal, “Introduction to multi-layer feed-forward neural networks,” *Chemom. Intell. Lab. Syst.*, vol. 39, no. 1, pp. 43–62, 1997.

[79] W. Roush, W. Dozier, and S. Branton, “Comparison of Gompertz and Neural Network Models of Broiler Growth,” *Poult. Sci.*, vol. 85, no. 4, pp. 794–797, 2006.

[80] M. F. Moller, “A scaled conjugate gradient algorithm for fast supervised learning,” *Neural networks*, 1993.

[81] F. M. Al-Naima and A. H. Al-Timemy, “Resilient Back Propagation Algorithm for Breast Biopsy Classification Based on Artificial Neural Networks,” *Comput. Intell. Mod. Heuristics*, 2010.

[82] B. H. Juang, “An Introduction to Hidden Markov Models,” *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, 1986.

[83] S. Abbate, M. Avvenuti, P. Corsini, A. Vecchio, and J. Light, *Monitoring of human movements for fall detection and activities recognition in elderly care using wireless sensor network : a survey*. InTech, 2010.

[84] L. B. Lusted, “Detectability Decision-Making,” *Science*, vol. 171, no. 3977, pp. 1217–1219, 1971.

[85] T. Fawcett, “ROC Graphs : Notes and Practical Considerations for Data Mining Researchers,” *Mach. Learn.*, 2004.

[86] N. Noury, A. Fleury, P. Rumeau, a K. Bourke, G. O. Laighin, V. Rialle, and J. E. Lundy, “Fall detection-principles and methods,” in *the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS*, 2007, pp. 1663–1666.

[87] F. Bagalà, C. Becker, A. Cappello, L. Chiari, K. Aminian, J. M. Hausdorff, W. Zijlstra, and J. Klenk, “Evaluation of accelerometer-based fall detection algorithms on real-world falls,” *PLoS One*, vol. 7, no. 5, p. e37062, Jan. 2012.

[88] A. K. Bourke, J. V O’Brien, and G. M. Lyons, “Evaluation of a threshold-based tri-axial accelerometer fall detection algorithm,” *Gait Posture*, vol. 26, no. 2, pp. 194–199, Jul. 2007.

[89] S. Ahuja, D. Mathaikutty, and S. K. Shukla, “Applying verification collaterals for accurate power estimation,” in *9th International workshop on Microprocessor test and Verifcation (MTV), Austin, Texas, USA.*, p.p 61–66.

[90] M. Casas-Sanchez, C. J. Bleakley, and J. Rizo-Morente, “Software Level Power Consumption Models and Power Saving Techniques for Embedded DSP Processors,” *J. Low Power Electron.*, vol. 2, pp. 281–290.

[91] “The Spice Home Page.” [Online]. Available: http://bwrcs.eecs.berkeley.edu/Classes/IcBook/SPICE/. [Accessed: 22-Jun-2015].

[92] Philips Electronic Design and Tools Group, “DIESEL User Manual,” 2001.

[93] R. P. Llopis and K. Goossens, “The petrol approach to high-level power estimation,” in *Proceedings of the 1998 international symposium on Low power electronics and design, ISLPED*, 1998, pp. 130–132.

[94] M. E. a. Ibrahim, M. Rupp, and H. a. H. Fahmy, “A Precise High-Level Power Consumption Model for Embedded Systems Software,” *EURASIP J. Embed. Syst.*, vol. 2011, pp. 1–14, 2011.

[95] C. Brandolese˘c, “A Codesign Approach to Software Power Estimation for Embedded Systems,” PhD disseration, Politecnico diMilano, Institute of Electronics Information, 2000.

[96] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski, “The design and implementation of PowerMill,” in *Proceedings of the 1995 international symposium on Low power design, ISLPED*, 1995, pp. 105–110.

[97] S. Gupta and F. N. Najm, “Power Macromodeling For High Level Power Estimation,” in *Proceedings of the 34th Design Automation Conference*, 1997, pp. 365–370.

[98] C. Ding, C.-Y. Tsui, and M. Pedram, “Gate-level Power Estimation Using Tagged Probabilistic Simulation,” *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 17, no. 11, pp. 1099–1107, 1998.

[99] S. Oskuii, P. Kjeldsberg, and E. Aas, “Probabilistic gate-level power estimation using a novel waveform set method,” in *Proceedings of the 17th ACM Great Lakes symposium on VLSI*, 2007, pp. 37–42.

[100] D. Liu and C. Svensson, “Power consumption estimation in CMOS VLSI chips,” *IEEE J. Solid-State Circuits*, vol. 29, no. 6, pp. 663–670, 1994.

[101] P. E. Landman and J. M. Rabaey, “Activity-sensitive architectural power analysis for the control path,” in *Proceedings of the 1995 international symposium on Low power design, ISLPED*, 1995, pp. 93–98.

[102] A. Bogliolo, L. Benini, and G. De Micheli, “Regression-based RTL power modeling,” *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 3, pp. 337–372, Jul. 2000.

[103] Q. Wu, Q. Qiu, M. Pedram, and C.-S. Ding, “Cycle-accurate macro-models for RT-level power analysis,” *Proc. 1997 Int. Symp. Low Power Electron. Des.*, pp. 125–130.

[104] N. R. Potlapally, A. Raghunathan, G. Lakshminarayana, M. S. Hsiao, and S. T. Chakradhar, “Accurate power macro-modeling techniques for complex RTL circuits,” in *the 14th International Conference on VLSI Design*, 2001, pp. 235–241.

[105] W. Ye, N. Vijaykrishnan, M. Kandemir, and M. J. Irwin, “The Design and Use of SimplePower : A Cycle-Accurate Energy Estimation Tool,” in *Proceedings of the 37th Annual Design Automation Conference*, 2000, pp. 340–345.

[106] H. Mehta, R. M. Owens, and M. J. Irwin, “Energy characterization based on clustering,” in *Proceedings of the 33rd annual Design Automation Conference, Dac ’96*, 1996, pp. 702–707.

[107] A. Sivasubramaniam, M. Jane, and I. N. Vijaykrishnan, “Using Complete Machine Simulation for Software Power Estimation : The SoftWatt Approach,” in *Proceedings of the Eighth International Symposium on High-Performance Computer Architecture*, 2002, pp. 141–150.

[108] D. Brooks, V. Tiwari, and A. Martonosi, “Wattch: a Frame work for Architecture- level Power Analysis and Optimization,” *ACM*, vol. 28, no. 2, pp. 83–94, 2000.

[109] I. Kadayif, T. Chinoda, M. Kandemir, N. Vijaykrishnan, M. J. Irwin, and A. Sivasubramaniam, “vEC : Virtual Energy Counters,” in *Proceedings of the 2001 ACM SIGPLAN-SIGSOFT workshop on Program analysis for software tools and engineering*, 2001, pp. 28–31.

[110] R. a. Bergamaschi, N. Dhanwada, S. Bhattacharya, W. E. Dougherty, I. Nair, J. Darringer, and R. Paliwal, “SEAS: a system for early analysis of SoCs,” in *Proceedings of the 1st IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2003, pp. 150–155.

[111] “Xilinx Power Estimator (XPE).” [Online]. Available: http://www.xilinx.com/products/technology/power/xpe.html. [Accessed: 22-Jul-2015].

[112] S. Ahuja, “High Level Power Estimation and Reduction Techniques for Power Aware Hardware Design,” 2010.

[113] “Xilinx Power Estimator User Guide,” 2014. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_1/ug440-xilinx-power-estimator.pdf.

[114] V. Tiwari, S. Malik, and A. Wolfe, “Power Analysis of Embedded Software : A First Step Towards Software Power Minimization,” *IEEE Trans. Very Large Scale Integr. Syst.*, 1994.

[115] V. Tiwari, S. Malik, a. Wolfe, and M. T.-C. Lee, “Instruction level power analysis and optimization of software,” *Proc. 9th Int. Conf. VLSI Des.*, 1996.

[116] S. Nikolaidis, N. Kavvadias, P. Neofotistos, and K. Kosmatopoulos, “Instrumentation Set-up for Instruction Level Power,” in *PATMOS2002, Springer-Verlag Berlin Heidelberg*, 2002, pp. 71–80.

[117] S. Nikolaidis, N. Kavvadias, T. Laopoulos, L. Bisdounis, and S. Blionas, “Instruction level energy modeling for pipelined processors,” *J. Embed. Comput.*, vol. 1, no. 3, pp. 317–324, 2005.

[118] S. Kerrison, U. Liqat, K. Georgiou, and A. Serrano, “Energy Consumption Analysis of Programs based on XMOS ISA-Level Models,” in *Logic-Based*

Program Synthesis and Transformation, Springer International Publishing, 2013, pp. 72–90.

[119] S. Hao, D. Li, W. G. J. Halfond, and R. Govindan, "Estimating mobile application energy consumption using program analysis," in *the 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 92–101.

[120] M. Sami, D. Sciuto, C. Silvano, and V. Zaccaria, "An instruction-level energy model for embedded VLIW architectures," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 21, no. 9, pp. 998–1010, 2002.

[121] J. Laurent, E. Senn, N. Julien, and E. Martin, "High-Level Energy Estimation for DSP Systems," in *PATMOS. IEEE*, 2001.

[122] E. Senn, N. Julien, J. Laurent, and E. Martin, "Power Consumption Estimation of a C Program for Data-Intensive Applications," in *Integrated Circuit Design. Power and Timing Modeling, Optimization and Simulation*, Springer Berlin Heidelberg, 2002, pp. 332–341.

[123] J. Laurent, E. Senn, N. Julien, E. Martin, F. Level, and P. Analysis, "Functional Level Power Analysis : An Efficient Approach for Modeling the Power Consumption of Complex Processors Complex Processors," in *Proceedings of the conference on Design, automation and test in Europe -Volume 1. IEEE Computer Society*, 2004.

[124] M. Schneider, H. Blume, and T. G. Noll, "Power estimation on functional level for programmable processors," *Adv. Radio Sci.*, vol. 2, no. 8, pp. 215–219, 2004.

[125] E. Senn, J. Laurent, N. Julien, and E. Martin, "SoftExplorer: estimation, characterization, and optimization of the power and energy consumption at the algorithmic level," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*, Springer Berlin Heidelberg, 2004, pp. 342–351.

[126] T. Ducroux, G. Haugou, V. Risson, and P. Vivet, "Fast and accurate power annotated simulation: Application to a many-core architecture," in *the 23rd International Workshop on Power and Timing Modeling, Optimization and Simulation, PATMOS*, 2013, pp. 191–198.

[127] S. K. Rethinagiri, O. Palomar, J. A. Moreno, O. Unsal, and A. Cristal, "System-Level Power and Energy Estimation Methodology for Open Multimedia Applications Platforms," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2014, pp. 442–449.

[128] G. Hellestrand, "The Engineering of Supersystems," *Computer*, vol. 38, no. 1, pp. 103–105, 2005.

[129] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenstr, "The Worst-Case Execution Time Problem - Overview of Methods and Survey of Tools," *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, p. 36, 2008.

[130] S. Stattelmann, M. Oriol, and T. Gamer, “Execution Time Analysis for Industrial Control Applications,” *arXiv Prepr. arXiv1404.0847*, Apr. 2014.

[131] H. Theiling, C. Ferdinand, and R. Wilhelm, “Fast and Precise WCET Prediction by Separated Cache and Path Analyses,” *Real-Time Syst.*, vol. 18, no. 2–3, pp. 157–179, 2000.

[132] T. Lundqvist and P. Stenstrom, “An Integrated Path and Timing Analysis Method based on Cycle-Level Symbolic Execution,” *Real-Time Syst.*, vol. 17, no. 2–3, pp. 183–207, 1999.

[133] D. Zaparanuks and M. Hauswirth, “Accuracy of Performance Counter Measurements,” in *IEEE International Symposium on Performance Analysis of Systems and Software, ISPASS*, 2009, pp. 23–32.

[134] G. Bernat, A. Colin, and S. M. Petters, “WCET analysis of probabilistic hard real-time systems,” in *23rd IEEE Real-Time Systems Symposium, RTSS*, 2002, pp. 279–288.

[135] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, “A Statistical Response-Time Analysis of Real-Time Embedded Systems,” in *the 33rd IEEE Real-Time Systems Symposium*, 2012, pp. 351–362.

[136] C. Ferdinand and R. Heckmann, “aiT: Worst case execution time prediction by static program analysis,” in *Building the Information Society*, Springer, 2004, pp. 377–383.

[137] A. Colin and I. Puaut., “A Modular & Retargetable Framework for Tree-based WCET Analysis,” in *the 13th Euromicro Conference on Real-Time Systems*, 2001, pp. 37–44.

[138] M. France, “Timing Analysis for Instruction Caches,” *J. Real-Time Syst.*, vol. 18, no. 2–3, pp. 217–247, 2000.

[139] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. 2014.

[140] M. Mitchell, “Zynq for Video Applications,” 2012.

[141] “OpenCV | OpenCV.” [Online]. Available: http://opencv.org/. [Accessed: 14-Dec-2014].

[142] A. S. Neuendorffer, T. Li, and D. Wang, “Accelerating OpenCV Applications with Zynq-7000 All Programmable SoC using Vivado HLS Video Libraries,” 2013. [Online]. Available: http://www.xilinx.com/support/documentation/application_notes/xapp1167.pdf.

[143] “Image Processing and Computer Vision - MATLAB & Simulink Solutions - MathWorks France.” [Online]. Available: http://fr.mathworks.com/solutions/image-video-

processing/index.html;jsessionid=649f5b5f03a000705f80a1619981. [Accessed: 14-Dec-2014].

[144] "Zynq-7000 All Programmable SoC," 2015. [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[145] ARM Ltd, "Cortex -A9 Technical Reference Manual," 2010. [Online]. Available: http://www.arm.com/cortex-a9.php.

[146] "ARM Architecture Reference Manual," 2014. [Online]. Available: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0406c/index.html.

[147] N. Julien, J. Laurent, E. Senn, and E. Martin, "Power Consumption modeling and Characterization of the TI C6201," *IEEE Micro*, no. 5, pp. 40–49, 2003.

[148] L. Deng, K. Sobti, Y. Zhang, and C. Chakrabarti, "Accurate area, time and power models for FPGA-based implementations," *J. Signal Process. Syst.*, vol. 63, no. 1, pp. 39–50, 2011.

[149] "USB-TO-GPIO-USB Interface Adapter EVM." [Online]. Available: https://store.ti.com/USB-TO-GPIO-USB-Interface-Adapter-EVM-P960.aspx?DCMP=hpa_pwr_xilinxfpga&HQS=xilinx_usb. [Accessed: 16-Dec-2014].

[150] "Digital Power Software - FUSION_DIGITAL_POWER_DESIGNER - TI Software Folder." [Online]. Available: http://www.ti.com/tool/fusion_digital_power_designer&DCMP=hpa_pmp_general&HQS=NotApplicable+OT+fusion-gui. [Accessed: 16-Dec-2014].

[151] "Measuring Program Execution Time." [Online]. Available: http://gec.di.uminho.pt/Discip/MaisAC/CS-APP_Bryant/csapp.ch9.pdf.

[152] J. Aldrich, "Fisher and Regression," *Stat. Sci.*, vol. 20, no. 4, pp. 401–417, Nov. 2005.

[153] " Catapult Synthesis." [Online]. Available: http://calypto.com/en/products/catapult/overview. [Accessed: 09-Mar-2015].

[154] "Synopsys." [Online]. Available: http://www.synopsys.com/home.aspx. [Accessed: 09-Mar-2015].

[155] "GAUT: High-Level Synthesis tool, From C to RTL." [Online]. Available: http://hls-labsticc.univ-ubs.fr/html/presentation.html. [Accessed: 09-Mar-2015].

[156] N. A. Gupta S, Gupta RK, Dutt ND, *SPARK: A Parallelizing Approach to the High-Level Synthesis of Digital Circuits*. Springer Science & Business Media, 2004.

[157] Xilinx Inc, "Introduction to FPGA Design with Vivado High-Level Synthesis," 2013. [Online]. Available:

http://www.xilinx.com/support/documentation/sw_manuals/ug998-vivado-intro-fpga-design-hls.pdf.

[158] "Reduce Power in Chip Designs with Sequential Clock Gating." [Online]. Available: http://electronicdesign.com/power/reduce-power-chip-designs-sequential-clock-gating. [Accessed: 25-Jun-2015].

[159] S. Huda, J. Muntasir Mallick, and J. H. Anderson, "Clock Gating Architecture For FPGA Power Reduction," in *the International Conference on Field Programmable Logic and Applications, FPL*, 2009, pp. 112–118.

[160] Mentor Graphics Inc, "Advanced Clock Gating Techniques in Catapult C Synthesis." .

[161] M. Pedram and J. M. Rabaey, *Power aware design methodologies*. Springer Science & Business Media, 2002.

[162] R. Zurawski, *Embedded Systems Handbook*. CRC Press, 2004.

[163] S. Eyerman and L. Eeckhout, "Fine-grained DVFS using on-chip regulators," *ACM Trans. Archit. Code Optim.*, vol. 8, no. 1, pp. 1–24, 2011.

[164] P. Pillai and K. G. Shin, "Real-time dynamic voltage scaling for low-power embedded operating systems," *ACM SIGOPS Oper. Syst. Rev.*, vol. 35, no. 5, pp. 89–102, 2001.

[165] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M. B. Srivastava, "Power optimization of variable-voltage core-based systems," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 18, no. 12, pp. 1702–1714, 1999.

[166] T. Pering, T. Burd, and R. Brodersen, "The simulation and evaluation of dynamic voltage scaling algorithms," in *Proceedings of the 1998 international symposium on Low power electronics and design*, 1998, pp. 76–81.

[167] L. Negri and A. Chiarini, "Power Simulation of Communication Protocols with StateC," in *Applications of Specification and Design Languages for SoCs*, Springer, 2006, pp. 277–294.

[168] S. Ahuja, A. Lakshminarayana, and S. K. Shukla, *Low Power Design with High-Level Power Estimation and Power-Aware Synthesis*. Springer Science & Business Media, 2011.

[169] J. Ayoub, O. Romain, Bertrand Granado, and Y. Mhanna, "Accuracy Amelioration of an Integrated Real-Time 3D Image Sensor," in *Conference on Design & Architectures for Signal and Image Processing*, 2008.

[170] F. Driessen, "Throughput Exploration and Optimization of a Consumer Camera Interface for a Reconfigurable Platform."

[171] P. S. Ong, Y. C. Chang, C. P. Ooi, E. K. Karuppiah, and S. M. Tahir, "An FPGA Implementation of Intelligent Visual Based Fall Detection," *Int. J. Comput. Information, Syst. Control Eng.*, vol. 7, no. 2, pp. 199–204, 2013.

[172] H. Rabah, A. Amira, and A. Ahmad, “Design and implementaiton of a fall detection system using compressive sensing and shimmer technology,” in *the 24th International Conference on Microelectronics (ICM)*, 2012, pp. 1–4.

[173] B. Senouci, I. Charfi, B. Heyrman, J. Dubois, and J. Miteran, “Fast prototyping of a SoC-based smart-camera: a real-time fall detection case study,” *J. Real-Time Image Process.*, pp. 1–14, 2014.

[174] F. R. Grüll, “Acceleration of Biomedical image processing and reconstruction with FPGAs,” PhD diss., Univ.-Bibliothek Frankfurt am Main, 2015.

[175] M. Kepski and B. Kwolek, “Human Fall Detection Using Kinect Sensor,” in *Proceedings of the 8th International Conference on Computer Recognition Systems CORES*, 2013, pp. 743–752.

[176] E. E. Stone and M. Skubic, “Fall detection in homes of older adults using the Microsoft Kinect,” *IEEE J. Biomed. Heal. informatics*, vol. 19, no. 1, pp. 290–301, Jan. 2015.

[177] M. Humenberger, S. Schraml, C. Sulzbachner, A. N. Belbachir, A. Srp, and F. Vajda, “Embedded fall detection with a neural network and bio-inspired stereo vision,” in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2012, pp. 60–67.

[178] B. S. Swan and C. D. Systems, “An Introduction to System Level Modeling in SystemC 2.0,” *Cadence Des. Syst. Inc., Draft Rep.*, 2001.

[179] “Philips Webcam SPC900NC VGA CCD with Pixel Plus.” [Online]. Available: http://www.p4c.philips.com/cgi-bin/dcbint/cpindex.pl?ctn=SPC900NC/00&scy=gb&slg=en. [Accessed: 12-Jan-2015].