

## ABSTRACT

Video surveillance has become an integral part of our lives. Low-cost camera technology has made it possible to deploy video surveillance in a wide range of applications, from monitoring secure areas to traffic-law enforcement. The most important aspect of automated video surveillance systems is monitoring the video channels for moving objects that may represent intruders or vehicles on the road that need to be registered and logged. Several algorithms already exist that can distinguish moving foreground objects from a static background, but these techniques cannot cope with panning or tilting cameras that are used to monitor a much wider field of view than a fixed camera. Various methods of moving object detection for non-static camera have been proposed, but none of them achieved adequate real-time performance. One well-known technique for detecting a moving foreground object with a panning camera is based on building a static mosaic image from a series of frames. It works well when the camera motion is very slow and where the target object motion is likely to be faster. In addition to the motion contrast requirements, the technique compensates for panning motion only, not tilting. However, it can tolerate some camera vibrations in the vertical direction. A second approach is to store a database of background objects and reference frames, which can then be compared with incoming images. However, it can only be used in a fixed environment and it is not suitable to real-time implementation. In this study, a novel approach that uses the fact that almost 80 per cent of the information in images is common to two successive frames was developed to construct a motion-corrected image. Thus, with panning and tilting, the bulk of the pixels can simply be translated horizontally or vertically. For this task, 2D correlation techniques were used to find the translation coordinates between two successive frames. There are also a number of tradeoffs that can be made to optimize for algorithm performance or to reduce compute time. The process of aligning images makes it possible to perform moving object detection using techniques developed for static cameras. The algorithm was implemented using a Texas Instrument DM6437 digital signal processor (DSP). By using the DMA with double buffering system, the overhead of the transfer between the internal and external memory could be substantially reduced. The DM6437 can sustain a performance of 28 frames per second, meeting its real-time constraints and allowing more algorithms to be integrated if needed.

# TABLE OF CONTENTS

<b>ABSTRACT .....</b>	<b>I</b>
<b>ACKNOWLEDGEMENTS.....</b>	<b>II</b>
<b>DECLARATION.....</b>	<b>III</b>
<b>TABLE OF FIGURES.....</b>	<b>VII</b>
<b>TABLE OF TABLES.....</b>	<b>IX</b>
<b>Introduction .....</b>	<b>1</b>
<b>Chapter I   Moving Object Detection .....</b>	<b>3</b>
I.1    Background Subtraction .....	3
I.1.1   Criteria of classification .....	4
I.1.2   Background Subtraction Methods.....	4
I.1.2.1   Temporal Difference.....	4
I.1.2.2   Temporal Median Filtering [15] [20].....	6
I.1.2.3   Running Gaussian Average [21] [22] .....	7
I.1.2.4   Mixture of Gaussians .....	8
I.1.2.5   Histogram-based Approach [15].....	9
I.1.2.6   Kernel Density Estimation.....	10
I.1.2.7   Cooccurrence of Image Variations .....	10
I.1.2.8   Eigenbackgrounds.....	10
I.1.3   Comparative Performance Analysis.....	10
I.1.3.1   Speed.....	11
I.1.3.2   Accuracy .....	11
I.1.3.3   Memory Requirements .....	11
I.2    Other Methods.....	12

<b>Chapter II</b>	<b>Image Registration and Sprite Generation</b>	13
II.1	Definition and examples	13
II.2	Image Registration Steps	14
II.3	Methods of Registration	15
II.3.1	Region features	16
II.3.2	Line features	16
II.3.3	Point features	16
II.4	Fundamentals of Sprite Generation	16
<b>Chapter III</b>	<b>Previous Work on Moving Object Detection for Panning/Tilting Surveillance Camera</b>	19
III.1	Sprite generation based Approach	19
III.2	Pre-recorded Image Database based Approach	21
III.3	Other Methods	22
<b>Chapter IV</b>	<b>DSP System Hardware and Software</b>	23
IV.1	The Choice of the DSP Platform	23
IV.2	The Platform Features	23
IV.3	The Processor Architecture	25
IV.4	The Memory Management	27
IV.4.1	The Internal Memory	27
IV.4.2	Integrated Direct Memory Access (IDMA)	27
IV.5	The Numerical issues	28
IV.5.1	Floating Point Processor	28
IV.5.2	Fixed Point Processor	28
<b>Chapter V</b>	<b>Proposed Approach</b>	29
V.1	Description	29
V.2	The Algorithm Stages	30
V.2.1	Frame Alignment and Matching Stage	30

V.2.1.1	Method 1 .....	31
V.2.1.2	Method 2 .....	33
V.2.1.3	Method 3 .....	34
V.2.1.4	Method 4 .....	35
V.2.2	Moving Object Detection Stage .....	36
<b>Chapter VI</b>	<b>System Implementation and Optimization .....</b>	<b>37</b>
VI.1	System Requirement.....	37
VI.2	System Hardware.....	38
VI.3	System Flow .....	39
VI.3.1	High Level System Flow .....	39
VI.3.2	Detailed System Flow.....	39
VI.4	System Optimisation.....	40
VI.4.1	Algorithm.....	40
VI.4.2	System .....	41
VI.4.3	Code.....	42
VI.5	System Implementation .....	42
VI.6	Code Description .....	43
<b>Chapter VII</b>	<b>Experimental Results and Discussion .....</b>	<b>44</b>
VII.1	Conventions and Assumptions .....	44
VII.1.1	Input Data .....	44
VII.1.2	Metrics .....	45
VII.2	Experimental Results .....	45
VII.2.1	Simulation Using Matlab.....	45
VII.2.1.1	Experiment 1 .....	46
VII.2.1.2	Experiment 2 .....	48
VII.2.1.3	Experiment 3 .....	48
VII.2.1.4	Experiment 4 .....	49

VII.2.1.5 Experiment 5 .....	50
VII.2.2 Real-time DSP Implementation.....	52
<b>Conclusion and Further Work</b> .....	54
<b>References</b> .....	57
<b>Appendix A: Paper</b> .....	62
<b>Appendix B: CD Contents</b> .....	68

## TABLE OF FIGURES

<b>Figure 1</b> Sensitivity of the background subtraction result to the threshold used. (a) original image, (b) the absolute difference, (c) threshold too high, (d) threshold too low .....	5
<b>Figure 2</b> Temporal Differencing performance issue .....	6
<b>Figure 3</b> Two input images from different sensors .....	14
<b>Figure 4</b> Output of the image registration process .....	14
<b>Figure 5</b> Image Registration Process.....	14
<b>Figure 6</b> Sprite plane orthogonal to z-axis .....	17
<b>Figure 7</b> Result of the foreground pixel rejection. (a) Original frames of the image, (b) the result of foreground rejection process.....	21
<b>Figure 8</b> TMS320DM6437 Functional Block Diagram [43].....	25
<b>Figure 9</b> TMS320C6437 CPU Data Path [42] .....	26
<b>Figure 10</b> Graph showing the proposed solution.....	30
<b>Figure 11</b> Shared Area between two successive frames.....	31
<b>Figure 13</b> The frame division process .....	32
<b>Figure 12</b> The proposed Method 1 of frame alignment and matching stage.....	32
<b>Figure 14</b> Final correlations applied to two successive frames.....	33
<b>Figure 15</b> The proposed Method 3 of frame alignment and matching stage.....	34

<b>Figure 16</b> Line positions.....	35
<b>Figure 17</b> The proposed Method 3 of frame alignment and matching stage.....	35
<b>Figure 18</b> Real-time Constraint .....	37
<b>Figure 19</b> System hardware components – The left image shows the input (DVD player) / output (TV monitor) devices and the DSP platform, the right image shows the panning / tilting device with its tripod .....	38
<b>Figure 20</b> EVM of the DM6437 DSP .....	39
<b>Figure 21</b> High Level System Flow .....	39
<b>Figure 22</b> Detailed System Flow .....	40
<b>Figure 23</b> Double Buffering System .....	41
<b>Figure 24</b> Main stages of the system implementation .....	42
<b>Figure 25</b> The results of Experiment 1 with a threshold value of 50 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames) ...	46
<b>Figure 26</b> The results of Experiment 1 with a threshold value of 30 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames) ...	47
<b>Figure 27</b> The results of Experiment 1 with a threshold value of 70 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames) ...	47
<b>Figure 28</b> The results of Experiment 2 with a threshold value of 25 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames) ...	48
<b>Figure 29</b> The results of Experiment 3 with a threshold value of 50 (The three top frames represent the original frames 25, 26 and 27; the three bottom frames represent the corresponding result frames).....	49
<b>Figure 30</b> the results of Experiment 4 with a threshold value of 50 (The three top frames represent the original frames 25, 26 and 27; the three bottom frames represent the result frames).....	50

<b>Figure 31</b> Relationship between the size of the mask in method 1 and the execution time .....	51
<b>Figure 32</b> Relationship between the mask width in method 2 and the execution time ..	51
<b>Figure 33</b> Repartition.....	52
<b>Figure 34</b> The results of DSP implementation without moving object in the scene, with a threshold value of 60 (The three top frames represent the original frames 15, 16 and 17; the three bottom frames represent the corresponding result frames) .....	53
<b>Figure 35</b> The results of DSP implementation with moving object in the scene, with a threshold value of 60 (The three top frames represent the original frames 48, 49 and 50; the three bottom frames represent the result frames).....	53

## TABLE OF TABLES

<b>Table 1</b> Summary of the comparative performance analysis .....	12
<b>Table 2</b> Some Features of the Davinci TMS320DM6437 DSP platform [43] .....	24
<b>Table 3</b> Description of the project source files.....	43
<b>Table 4</b> Input data characteristics .....	44
<b>Table 5</b> Convention for moving objects detection quality .....	45
<b>Table 6</b> Execution time comparison .....	50

## **Introduction**

The need of real-time object detection for video surveillance has become commonplace in our daily life, especially in some domains where it has received considerable attention, including criminology, sociology, statistics, traffic accident detection and military applications. Moving object detection is considered to be the most important task in automated video surveillance systems. It is the low level image processing technique [1], which is the integral part of automated video surveillance.

The aim of the moving object detection in video surveillance analytics is to distinguish moving foreground objects from the background. Some algorithms already exist that can distinguish moving foreground objects from a static background, but these techniques are not designed for panning and/or tilting cameras required for monitoring a much wider field of view than with fixed camera. In the case of panning/tilting camera, a global camera motion needs to be compensated for.

Object detection for a non-static (panning/tilting) camera has been the focus of interest of various applications. The aim of this project is to develop a novel real-time moving object detection approach that can efficiently handle the problem of a moving camera that performs panning and tilting motions to incorporate a wide field of view for increased security and a wider angle of view. The proposed system has to be implemented on an embedded platform; for this project the Texas Instrument DM6437 DSP platform was selected due to its suitability for encapsulation within the camera.

The present project thesis is outlined as follows:

Chapter I introduces the moving object detection task and outlines the different approaches achieved for moving object detection for static video camera. In addition, a comparative study between existing approaches is undertaken to highlight their strong and weak points.

Chapter II explains the process of image registration and sprite generation, an essential aspect of all main approaches used for resolving the problem of moving object detection for non-static camera.



Chapter III describes the previous achievements on moving object detection for non-static camera, including their advantages, drawbacks and their suitability for real-time implementation on an embedded system.

Chapter IV presents an overview of the device (DSP platform) that will be used in the implementation of the proposed system by illustrating its system hardware and software.

Chapter V proposes a novel real-time moving object detection approach for a moving camera that performs panning and tilting motions. A set of possible solutions that conform to the same principles is presented to incorporate more flexibility between the quality of detection and the system speed performance.

Chapter VI describes the system implementation, highlighting the different steps taken to achieve an optimal system that can satisfy the real-time constraints.

Chapter VII outlines the experimental results of both simulation and real-time implementation on the embedded system DSP.

The thesis is concluded with a summary of the undertaken study, highlights the key results and discusses further improvements and extensions of the present work.

# Chapter I Moving Object Detection

In almost all computer vision applications, it is essential to identify moving objects from a video sequence and then apply appropriate video processing algorithms to the selected material. A common approach for detecting moving objects of interest in a static scene is to perform background subtraction, which consists of forming a background model of the scene, and to subsequently use it as a reference to classify pixels as belonging to either the foreground or the background image [2]. Other techniques exist, such as those that are based on wavelet transform for temporal filtering [3], gradient based [3], wavelet-based neural network techniques [4], 3D Projection based techniques [5], etc.

In this chapter, presents some fundamental approaches of background subtraction, including “Temporal Differencing” (TD), “Running Gaussian Average” (RGA), “Temporal Median Filtering,” “Mixture of Gaussians” (MoG), “Histogram-based Approach,” “Kernel Density Estimation” (KDE), “Cooccurrence of Image Variations,” and “Eigenbackgrounds.” A comparative of their performance, suitability and feasibility for embedded system and real-time processing will be presented. In the final sections this chapter, others methods of moving object detection will be presented.

## I.1 Background Subtraction

The term “Background subtraction” refers to the subtraction of the observed image from an estimated image known as “Background Image,” followed by a thresholding of the result to classify the pixels into one of the two main classes, namely foreground and background [2].

Designing an efficient background subtraction approach encounters several challenges. Firstly, the changes in illumination both gradual and sudden have to be considered, in order to closely resemble the reality. Secondly, the background subtraction should be insensitive to background periodic movement, such as rain, snow, tree movement caused by the wind, sea waves, etc. Similarly, it should not interpret the shadow of a moving object as foreground objet. Finally, it should react

quickly to changes in the scene [6]. Furthermore, in the case of panning/tilting camera, the background model should be capable to adapt and update the background model reference for each angle (position) of the panning/tilting camera.

### **I.1.1 Criteria of classification**

The background subtraction techniques are usually defined by two parameters: the background modelling and the detection process. Background modelling is defined by three aspects [7]:

- **Representation** – defines the model used to represent the background.
- **Initialization** – defines the initialization of the chosen model.
- **Adaptation** – illustrates how the background model adapts his self to the changes.

For the representation aspect, several approaches have been proposed, such as reference frame, predictive models (Kalman [8], Wiener [9]), Mixture of Gaussians [10], etc. In [11], some update mechanisms for background adaptation are presented, including iterative update mechanisms (online expectation maximization (EM) algorithm), Bayesian update mechanism [12], models using image gradient and optical flow information [13], non-parametric kernel density estimation [14], etc.

### **I.1.2 Background Subtraction Methods**

There are several techniques of background subtraction; the following sub-sections we will present some examples – from simple to complex, depending on the background modeling.

#### **I.1.2.1 Temporal Difference**

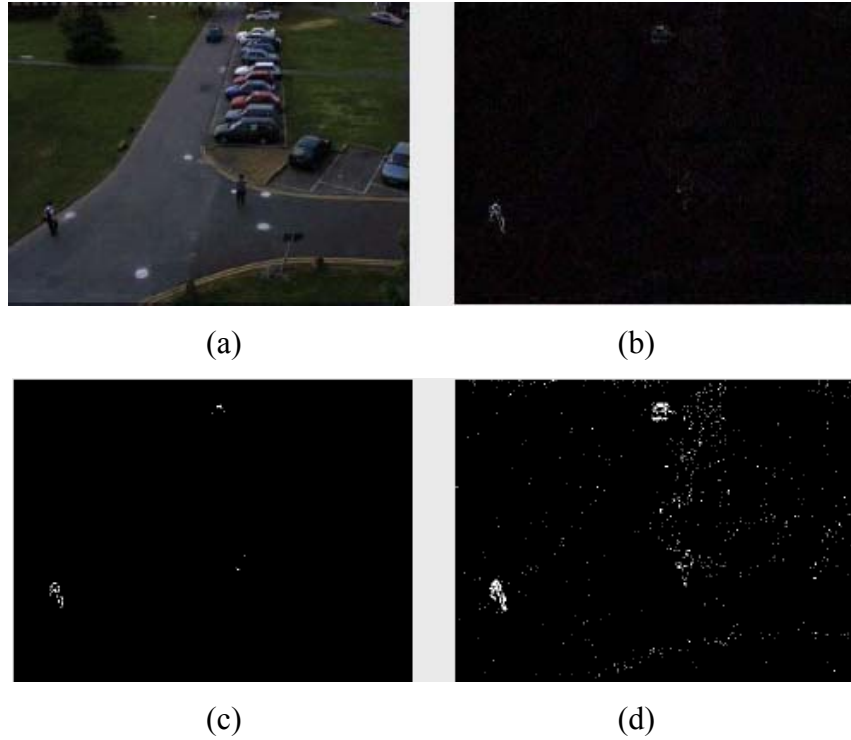
Temporal Difference (TD), also known as Frame Difference (FD), consists of subtracting the current image from the previous one – referred to as the reference image, followed by setting a threshold as follows [15] [16]:

$$|frame_i - frame_{i-1}| > Th \quad (1)$$

where  $Th$  is the threshold.

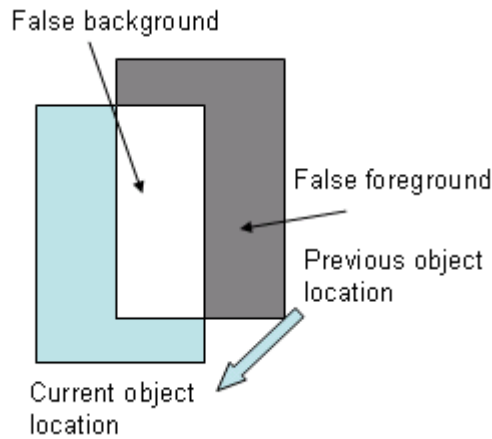
In this technique, the previous image is used as a background model.

This approach is rather inefficient and can only be reliably used in particular conditions of object velocity and frame rate. Another serious shortcoming of this approach is that it is very sensitive to the threshold [17] [18]. Figure 1 shows the sensitivity of the background subtraction result to the value of the threshold [18].



**Figure 1 Sensitivity of the background subtraction result to the threshold used. (a) original image, (b) the absolute difference, (c) threshold too high, (d) threshold too low**

As we can see from Figure 1, the result is highly dependent on the value of the threshold. Another issue encountered in this approach is that if the moving object has uniform colour occupying large area, the overlapping between two frames is usually considered as background due to the similarity and to the single value-based comparison for making decisions [18]. This situation can be shown in Figure 2 [19].



**Figure 2 Temporal Differencing performance issue**

In [16], the problem was resolved by assuming the trailing region as a part of the background. For each frame of the sequence, the centroids of the foreground objects are approximated in order to estimate their motion. The region of the foreground object, which is likely to be the transition, is subsequently disregarded.

In addition, this approach is based on the difference between two frames during the period where the object is moving, thus there are no apparent problems with the method's reasoning for objects in motion. However, once the object stops moving, it will be considered as a part of the background. A solution for this problem is proposed in [16] based on using the position of the moving object at the previous frame where it was still in motion.

### **I.1.2.2 Temporal Median Filtering [15] [20]**

This approach constructs its background model from the median of the  $N$  previous frames. Similarly to the Frame Difference approach, the Temporal Median Filtering uses a threshold on the frame difference of each pixel median value. Although this approach is relatively robust, it cannot be used in embedded systems and it is incapable of achieving real-time processing due to the need of storing the  $N$  previous frames of the video sequence.

### I.1.2.3 Running Gaussian Average [21] [22]

The previous approach uses a single value to model the background model, which makes it intolerant to changes to the value of a pixel. In the real world, where the camera will be employed, there are numerous internal and external interferences, such as noise and changing illumination. The Running Gaussian Average (RGA) consists of attributing a Gaussian probability density function ( $\mu$ ,  $\sigma$ ) to each pixel of the image. This method allows the pixels of the frame to have a range of values (within a normal distribution).

Hence, two parameters need to be updated (the mean  $\mu$  and the variance  $\sigma^2$ ). They are usually estimated by using the previous state; the following equations show the parameter update process:

$$\mu_n = \alpha X_n + (1 - \alpha)\mu_{n-1} \quad (2)$$

$$\sigma_n^2 = \alpha(X_n - \mu_n)^2 + (1 - \alpha)\sigma_{n-1}^2 \quad (3)$$

Where  $\alpha$  is a weighting value, usually empirical.

Depending on the value of  $\alpha$ , the model reacts differently; if this value is large, the model allows a wide range of pixel illumination changes, yet it will increase the vulnerability to system noise.

After the updating process, the following formulae are used to classify whether a pixel belongs to the background or not.

$$|X_n - \mu_n| < k\sigma_n \quad (4)$$

Where  $k$  controls how much deviation is allowed.

An improved Running Gaussian Average background model is proposed in [23]. It handles the varying luminance effects and the background updating problem more efficiently.

However, the Running Gaussian Average approach does not cope with multimodal backgrounds.

#### I.1.2.4 Mixture of Gaussians

In the adaptive mixture of Gaussians approach defined in [10], each pixel of the image is modelled independently by a combination of  $K$  Gaussians.

$$P(I_t) = \sum_{i=1}^K \omega_{i,t} \eta(I_t - \mu_{i,t}, \Sigma_{i,t}) \quad (5)$$

Where  $K$  is the number of Gaussian distributions and can take the values: 3, 4, and 5 or other [16]. In [16] [10] it is assumed that:

$$\Sigma_{i,t} = \sigma_{i,t}^2 I \quad (6)$$

The parameter  $\omega_{i,t}$  represents the weight of each distribution, where the associated probability is proportional to its size.

$\eta$  is the Gaussian probability density function and it is defined as:

$$\eta(I_t - \mu_{i,t}, \Sigma_{i,t}) = \frac{1}{2\pi^{t/2} |\Sigma_{i,t}|^{1/2}} \exp\left(-\frac{1}{2} (X_t - \mu_{i,t})^T \Sigma^{-1} (X_t - \mu_{i,t})\right) \quad (7)$$

In this approach, before proceeding to the detection of the foreground objects, the background is updated using one of two approaches, depending on whether the distribution matches or not:

1. If  $I_t$  matches component  $i$ , which implies  $I_t$  is within  $\lambda$  standard deviations of  $\mu_{i,t}$ , then the  $i$ th component is updated as follows:

$$\omega_{i,t} = \omega_{i,t-1} \quad (8)$$

$$\mu_{i,t} = (1 - \rho) \mu_{i,t-1} + \rho I_t \quad (9)$$

$$\sigma_{i,t}^2 = (1 - \rho) \sigma_{i,t-1}^2 + \rho (I_t - \mu_{i,t})^T (I_t - \mu_{i,t}) \quad (10)$$

Where  $\rho = \alpha Pr(I_t | \mu_{i,t-1}, \Sigma_{i,t-1})$

2. If  $I_t$  does not matches component  $i$  , the  $i$ th component is updated as follows:

$$\omega_{i,t} = (1 - \alpha)\omega_{i,t-1} \quad (11)$$

$$\mu_{i,t} = \mu_{i,t-1} \quad (12)$$

$$\sigma_{i,t}^2 = \sigma_{i,t-1}^2 \quad (13)$$

3. If  $I_t$  does not match any component, then in this case the least likely component is replaced with the one that has  $\mu_{i,t} = I_t$  ,  $\Sigma_{i,t}$  large, and  $\omega_{i,t}$  low.

After the process of updating, the weights  $\omega_{i,t}$  are renormalized.

Regarding the process of foreground detection, all the components in the mixture are sorted into the order of decreasing  $\omega_{i,t} / \|\Sigma_{i,t}\|$ . This implies that components with the evidence and lowest variance are assigned higher importance. Eventually, they are assumed to be the background.

Let us consider the following expression:

$$B = \underset{b}{\operatorname{argmin}} \left( \frac{\sum_{i=1}^b \omega_{i,t}}{\sum_{i=1}^K \omega_{i,t}} > T \right) \quad (14)$$

Components 1...B are assumed to be background. Therefore, the pixel that does not match one of these components is classified as foreground.

In sum, in order to handle the problem of multimodal backgrounds, adaptive mixture of Gaussian is proposed. This approach needs to update the background models at every frame of the video sequence, whereby an iterative update mechanism is used.

#### **I.1.2.5 Histogram-based Approach [15]**

In this approach, each pixel of the background image is attributed a histogram of its  $N$  previous values from the last  $N$  frames. Each histogram has to be reevaluated after every  $M$  frames. The value of  $M$  should be significantly lower than the value of  $N$ . Different thresholds are used for each pixel from the background model.



### **I.1.2.6 Kernel Density Estimation**

A method to model the background distribution is to use a histogram of the last recorded pixels belonging to the background. However, this technique might result with poor approximation of the real values [21]. For that reason, the authors in [14] have proposed a non-parametric model based on Kernel Density Estimation (KDE) to model the background distribution. This technique overcomes the drawbacks of using the histograms.

### **I.1.2.7 Cooccurrence of Image Variations**

This approach is proposed in [24]; the main idea is that similar variations over time should be observed in the neighbouring blocks of pixels classified as background. Obviously, it will generate false detections in border areas. The granularity of this algorithm is a vector of  $N^2$  elements representing a block of  $N \times N$  pixels. This characteristic makes it faster and more stable [21].

### **I.1.2.8 Eigenbackgrounds**

In [25], a similar approach as the one proposed in [24] is proposed, except that it is applied to the entire image instead to individual blocks, thus avoiding the effect of block partitioning.

## **I.1.3 Comparative Performance Analysis**

In [21], a comparative performance analysis has been conducted to highlight the advantages and the drawbacks of each method. This analysis is based on three performance aspects: Speed, Memory requirements and Accuracy. The comparative analysis does not include the Histogram-based approach.

### **I.1.3.1 Speed**

As Frame Difference and the Running Gaussian Average approaches require only the pixel value to process and update just one or two parameters to adapt the background model, they are the fastest techniques reviewed. In [21], time complexity is defined as  $O(1)$ . The Median Filter approach has a linear complexity as it has to consider the median of  $N$  values from the  $N$  previous frames, thus its complexity can be represented as  $O(N)$ . In the Mixture of Gaussian approach, linear complexity is defined by  $O(M)$ , where  $M$  represents the number of Gaussian distributions used to represent the background model. In comparison, Kernel Density Estimation approach has a linear time complexity  $O(N)$ , where  $N$  represents the  $N$  previous frames. The time complexity for the Cooccurrence of Image Variation approach is estimated as  $O(R/H^2)$ , where  $R$  is used for finding the nearest neighbours amongst the  $R$  variations and  $H^2$  represents the cost over the pixels in a block. Finally, the Eigenbackground approach has a complexity of  $O(S)$ , where  $S$  is the number of the best eigenvectors.

### **I.1.3.2 Accuracy**

In [21], the accuracy of the approach is coded in three categories (L, M, H) that respectively stand for Limited, Intermediate and High accuracy. The results are shown in Table 1.

### **I.1.3.3 Memory Requirements**

Almost all the treated approaches in this comparative performance analysis have the same memory complexity per pixel as the time complexity except for the Cooccurrence of Image Variation approach, which has a memory complexity of  $O(PV/H^2)$ . In this expression,  $P$  represents the number of variations in the training and  $V$  their associated dimension.

Table 1 summarizes all the above comparative performance analysis.

Approach	Speed	Accuracy	Memory requirements
Frame Difference	1	L/M	1
Running Gaussian Average	1	L/M	1
Temporal Median Filter	N	L/M	N
Mixture of Gaussians	M	H	M
Kernel Density Estimation (KDE)	N	H	N
Cooccurrence of Image Variations	$R/H^2$	M	$PV/H^2$
Eigenbackgrounds	S	M	N

**Table 1 Summary of the comparative performance analysis**

## **I.2 Other Methods**

Although Background Subtraction approaches remain the most dominant methods used for moving objects detection, there are other techniques that do not rely on a background model to perform moving object detection tasks and provide satisfactory results. In [3], a wavelet transform for temporal filtering is used to detect the dynamic object behaviour. Furthermore, another technique for moving objects detection based on gradient is presented in [3]. In [4], a wavelet-based neural network is proposed for the pre-crash safety system of vehicles. Another approach based on the analysis of the projection of the 3D objects' motion is proposed in [5], whilst in [26], a robust method for motion detection is proposed. It combines both temporal difference imaging and temporal filtered motion field and makes an assumption that the trajectory of the moving object remains constant.

Finally, a new algorithm for moving object detection is proposed in [27], making full use of the special information. Both colour segmentation and background model are used.

## **Chapter II Image Registration and Sprite Generation**

Manipulating a panning/tilting video camera requires processing images that are not similar. However they share a great amount of information that can be utilized by finding the commonality between the two frames. Image registration techniques perform this task. In this chapter, several techniques of image registration are presented and a detailed description of sprite generation process is provided.

### **II.1 Definition and examples**

Image registration is the technique of aligning two images or more taken from the same scene at different times, from different position (case of panning and tilting camera), or by different sensors. Basically, it aligns two images together, the reference and the sensed images [28]. The image registration process can be defined as the procedure of mapping points from one image to their corresponding points in another image [29].

Image registration is used widely in various applications, such as medicine – where images from different sources are combined in order to get more information and precision about the phenomena observed, cartography, military, monitoring, etc

Figure 3 shows an application example for image registration, representing two images of a specific location on earth (Aerial photo image and Ortho-photo image). The second image has some distortions and it is not aligned with the first image. As a result of the image registration process, the two images are matched by using specific feature points (Figure 4) [30].

Image registration is a vital process in video processing, especially when using a non-static camera, since the background is continuously changing due to the camera motion. Furthermore, if object detection approach is applied using camera configuration with adaptive background references, huge memory storage is needed to store the frames captured from each angle of the panning and/or tilting [31]. This substantial memory requirement is neither practical nor affordable, since the system will be implemented on an embedded platform of a digital signal processor (DSP).



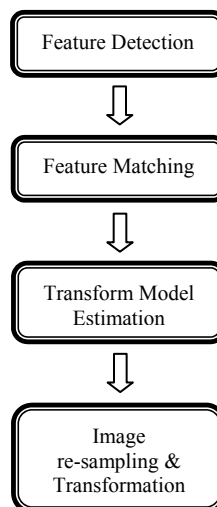
**Figure 3 Two input images from different sensors**



**Figure 4 Output of the image registration process**

## II.2 Image Registration Steps

As it is defined in [29], image registration process for feature-based techniques follows the steps shown in Figure 5.



**Figure 5 Image Registration Process**

The steps presented in the flow diagram above can be summarized as:

- Feature detection consists of extracting some distinctive characteristics from both reference and sensed images. Those characteristics could be the edges, line intersections, corners, etc.
- Feature matching consists of finding the correspondence between the features of the two images, namely the reference and the sensed images.
- In the transform model estimation, the type and the parameter mapping functions are estimated.
- In the Image resembling and transformation step, the sense image is transformed by using the mapping functions [29].

The implementation of each step passes through several decisions and considerations, such as the choice of the appropriate features that should be considered from the point of view of easiness of detection. The feature matching should be robust, so that it will not be affected by a mis-detection of features.

## **II.3 Methods of Registration**

The registration methods can be grouped into two categories: direct methods and feature-based methods. Direct methods consist of using pixel to pixel matching. However, feature-based methods begin by extracting characteristic feature from both reference and sensed images. This is followed by a feature matching process between the images, applied to highlight the correspondences [28].

The feature-based methods are more appropriate and suitable for the purpose of panning and tilting surveillance camera due to their capability to handle large motion and viewpoint changes between the images. Furthermore, they do not need an initialization stage required by the direct methods.

There are several approaches used for the feature-based image registration process depending on the nature of the extracted features [32].

### **II.3.1 Region features**

Since working directly with the intensity of the pixels present in the image is not suitable for more advanced applications, high contrast closed-boundary regions are employed as features of an image, such as lakes, buildings, urban area, forests, etc. Generally, the regions are defined by their centres of gravity in order to make them invariant with respect to rotation, scaling, illumination changes and eventually noise.

Region-based registration methods employ segmentation process to detect the region feature, which makes the accuracy of the registration dependent on the process of segmentation.

### **II.3.2 Line features**

Another approach is using lines or segments present in the images as significant features, which can be object contours, coastal lines, roads, etc.

Line-based registration methods make use of the different edge detection algorithm that can be first or second order approaches, such as Canny and methods based on Laplacian of Gaussian transformation respectively.

### **II.3.3 Point features**

This method of choosing features focuses on the intersections of the lines present in the images, such road crossings, and oil and gas pipelines.

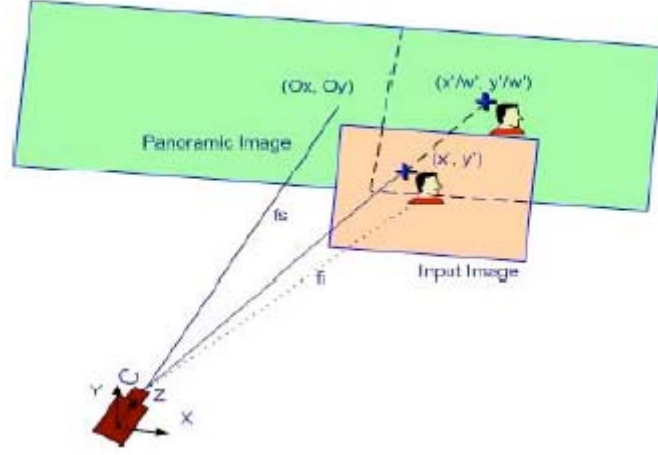
In the context of real-time implementation, it is judicious to use line feature approach for features, as they are less computationally complex than the region feature approach.

## **II.4 Fundamentals of Sprite Generation**

Static panoramic image (sprite), also known as mosaic, is a two-dimensional image that is extended to cover a larger scene by combining several images taken from

different angles or positions [33]. In the context of geometry, the sprite image can be considered as the projection of the 3D real word onto a 2D plan [34].

In order to understand the sprite generation process, we need to define the 3D world coordinates. The following figure illustrates the real world projection onto the plane, taking the camera position as the origin for the coordinate system.



**Figure 6 Sprite plane orthogonal to z-axis [34]**

Several frames are captured by the camera with different angles due to the panning motion. There is one distinctive frame along the z-axis; its pixels are located by the following triplet  $(x, y, f)$ . If we consider the virtual sprite plan orthogonal to the axis of the camera positioned at a distance defined by  $f_s$  [34], the projection can be defined as:

$$\begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} = \begin{pmatrix} f_s & 0 & o_x \\ 0 & f_s & o_y \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ f_i \end{pmatrix} \quad (15)$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = K_s \cdot R_i \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & f_i \end{pmatrix} \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = H_i \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} \quad (16)$$

where  $(O_x, O_y)$  refers to the coordinates of the principle point on the sprite plane. The following equation represents the projection of the homogeneous image coordinates onto the virtual sprite plane.



$$H_i = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} \cong \begin{pmatrix} a_{00} & a_{01} & t_x \\ a_{10} & a_{11} & t_y \\ p_x & p_y & 1 \end{pmatrix} \quad (17)$$

Or expressed as the Euclidean coordinates:

$$x' = \frac{a_{00}x + a_{01}y + t_x}{p_x x + p_y y + 1} \quad (18)$$

$$y' = \frac{a_{10}x + a_{11}y + t_y}{p_x x + p_y y + 1} \quad (19)$$

In general, a simplified version known as the affine motion model is used, it is a special case defined by setting  $p_x$  and  $p_y$  to 0 [34].

The obvious advantage of the affine model is that it is less complex due to its linearity. Yet, it cannot represent the camera pan and tilt accurately. For that reason the affine model needs to be supported by introducing some geometric distortions consideration [34] [35].

## **Chapter III Previous Work on Moving Object Detection for Panning/Tilting Surveillance Camera**

Moving object detection for a non-static (panning/tilting) camera is an enhanced video analytics application, since the background is not the same between two successive frames in the video sequence. Therefore, the background model has to handle the presence of two motions in the scene, namely: the object motion that has to be detected and a global motion caused by the camera motion. Furthermore, the background model needs to be robust enough to compensate the global motion and highlight only the object's motion.

In this chapter, various methods for moving object detection for non-static camera are cited and described with emphasis on their suitability for real-time performance.

### **III.1 Sprite generation based Approach**

Muhammad Asif and John Soraghan in [34] propose a technique for detecting moving foreground object in presence of a panning camera. This approach is able to determine and extract moving objects even in the presence of global motion.

The proposed approach follows two fundamental stages; the first stage is the foreground pixel rejection process, which consists on separating the object motion from the global motion of the camera, assuming that a small global motion is used. The second stage involves the generation of a static panoramic image (sprite) for panning camera motion. Eventually this sprite will be used as an adaptive background reference to detect moving objects in the scene.

This approach is applicable when the camera motion is very slight and much smaller than the object motion. The authors also consider the pixel changes due to the propensity of global (camera) motion to have a strong local correlation. The image is divided into block size of  $64 \times 64$ . A motion vector will be determined on those blocks of the image by using a phase correlation (FFT) technique. Let us define  $G_1$  and  $G_2$  as the DFT of two contiguous images  $I_t$  and  $I_{t-1}$ .

For spatial frequency  $(u,v)$ , the cross power spectrum is defined as follows:

$$e^{j(\phi_t(u,v)-\phi_{t-1}(u,v))} = \frac{G_t(u,v) * G_{t-1}^*(u,v)}{|G_t(u,v) * G_{t-1}^*(u,v)|} \quad (20)$$

In the case where the second image  $g_2$  (sub-image) is spatially shifted version of the first image  $g_1$ , then

$$g_t(x, y) = \alpha \cdot g_{t-1}(x - \delta_x, y - \delta_y) \quad (21)$$

Where  $\delta_x$  and  $\delta_y$  represent the displacement in x and y axis respectively, and  $\alpha$  is the contrast change value.

The phase correlation  $r$  is defined as the inverse DFT of cross power spectrum.

$$r = IDFT\{e^{j(\phi_t(u,v)-\phi_{t-1}(u,v))}\} \quad (22)$$

$$(\delta_x, \delta_y) = \max \arg\{r\} \quad (23)$$

The phase correlation surface is zero, except for the delta function at location  $(\delta_x, \delta_y)$ , which refers to the displacement between two sub-images. This allows for identification of peaks in the correlation surface. As a result, we will have identified two categories of sub-image blocks, namely those belonging to global motion or foreground object motion. A second level of processing is undertaken in order to achieve higher precision in detection of the object motion, whereby the blocks defined as foreground motion are segmented into a further small blocks. Blocks of size  $8 \times 8$  are used and the sum of absolute difference is computed for each block as follows:

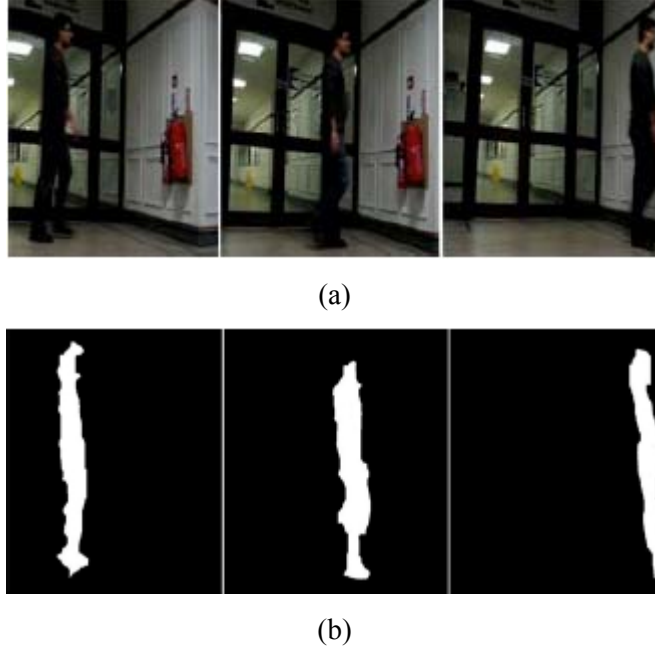
$$SAD = \sum_{i=0}^8 \sum_{j=0}^8 |e_{ij}| \quad (24)$$

where  $e_{ij}$  represents the difference in two perspective blocks.

The algorithm has two criteria for rejecting a block:

1. If SAD does not satisfy the threshold conditions
2. If in the 8 neighbour blocks, at least four of them have potential to be rejected.

The result of the first step of this approach is shown in figure 7.



**Figure 7 Result of the foreground pixel rejection. (a) Original frames of the image, (b) the result of foreground rejection process**

The second stage of this approach consists of constructing the mosaic of the scene by using the output of the first stage as an input. The output of the first stage is a sequence of frames containing only the background without the foreground objects. The second stage consists of using feature based image registration process. Features in the contiguous frames are detected and the matching feature points (control points) are computed. During the next step, a special mapping is determined using these matched feature points based on least squares regression. Finally, the two frames are merged together to form a part of the mosaic.

The defined feature should be unique in both images in order to have more robustness to distortion, thus features used in this approach are edge and corners.

As a final step of the sprite generation, a blending of the images is performed using pyramidal weight function. As a result of this process, a removal of all traces and edges between images is achieved.

### **III.2 Pre-recorded Image Database based Approach**

In another approach presented in [36], a set of the camera states known as **MCPS** (Minimum Camera Parameter Settings) is defined. Subsequently, for each state of

the set (**MCPS**), a background image of the camera view is attributed. The result of this procedure is an image database. The **MCPS** is basically used to facilitate object detection and tracking in a moving surveillance camera. Object detection is simply performed by comparing the current image with the corresponding reference image (Background image) related to the previously defined state from the database. This approach can only be applied to a fixed environment and the number of states has to be augmented to improve to smoothness of the detection.

This approach cannot be implemented in an embedded system and cannot achieve real-time performance, since it is impossible to store all the pre-recorded images when memory resources are limited. Furthermore, this approach is not able to cover a wide field of view since for each position, a separate image is needed.

### **III.3 Other Methods**

There are several other approaches that address the problem of using non-static camera for moving object detection. In [37], a novel approach based on Tree Dynamic Programming is used to perform the Background Modelling. In this approach the moving object detection and the correspondence estimation between the frames are considered as a labelling problem and its update mechanism is based on the optimal correspondence. This method suffers from some drawbacks manifested by its incapability of handling large depth disparity regions and some parallax problems. A solution is proposed in [38], based on the approach suggested in [37]. It uses a multilayer homography algorithm for building the background model. In contrast with the method presented in [37], it efficiently deals with large depth disparity and reduces the parallax issues considerably. In [39], a moving object detection algorithm for a non-static camera that uses motion clustering and classification is proposed. This algorithm uses only two consecutive images. It claims to be accurate within an accuracy of 7 pixels on average.

None of the presented methods are suitable for real-time implementation on an imbedded system such as a DSP (Digital Signal Processor).

## **Chapter IV DSP System Hardware and Software**

An efficient implementation on an embedded platform always requires a certain level of understanding of the involved platform, starting from the processor architecture to the peripherals used. Furthermore, it is crucial to know and control the strength and the vulnerabilities of the system hardware, as it will be useful to use the hardware in its best performance. In this study, the Texas Instrument Digital Media Processor Davinci TMS320DM6437 was used. In this chapter a detailed description of the involved DSP platform will be presented, focusing on all the aspect needed for an efficient real-time implementation. In addition, some implementation issues related to the DSP will be discussed.

### **IV.1 The Choice of the DSP Platform**

Texas Instrument (TI), the number one producer of Digital Signal Processors and Analog Semiconductors [40], offers to its clients a vast range of embedded processor platforms for the development of a rich variety of applications. The Davinci TMS320DM6437 platform consists of a DSP device and a set of ARM Cortex-A8/ARM9 processors. This platform is optimised for video analytic applications. Owing to its hardware and software supports, it allows the development of a wide range of video analytics based applications, such as machine-vision systems, video security, automotive vision applications, etc. [41].

### **IV.2 The Platform Features**

The Davinci TMS320DM6437 DSP is the highest-performance fixed-point in the TMS320C6000 DSP platform family. The DM6437 device is based on Very Long Instruction Word (VLIW) architecture, making it more suitable for video processing. These DSPs are able to process up to 5600 million instructions per seconds (MIPS) at a frequency of 700 MHz. Furthermore, they are considered to be low power devices [42] [43]. Table 2 depicts some of the main feature of the Davinci TMS320DM6437 DSP platform.

Hardware Feature	Description	
<b>CPU Frequency (MHz)</b>	700	
	660	
	600	
	500	
	400	
<b>Cycle Time (ns)</b>	2.5	
	2	
	1.67	
	1.51	
	1.43	
<b>Instructions/Cycle (MIPS)</b>	5600	
	5280	
	4800	
	4000	
	3200	
<b>Number of Functional Units</b>	<b>6 ALUs (32-40) bits</b>	Single 32-bit Arithmetic/Cycle Or Double 16-bit Arithmetic/Cycle Or Quad 8-bit Arithmetic/Cycle
	<b>2 Multipliers</b>	4 (16 x 16-bit) Multiplies (32-bit result)/Cycle Or 8 (8 x 8-bit) Multiplies (16-bit result)/Cycle

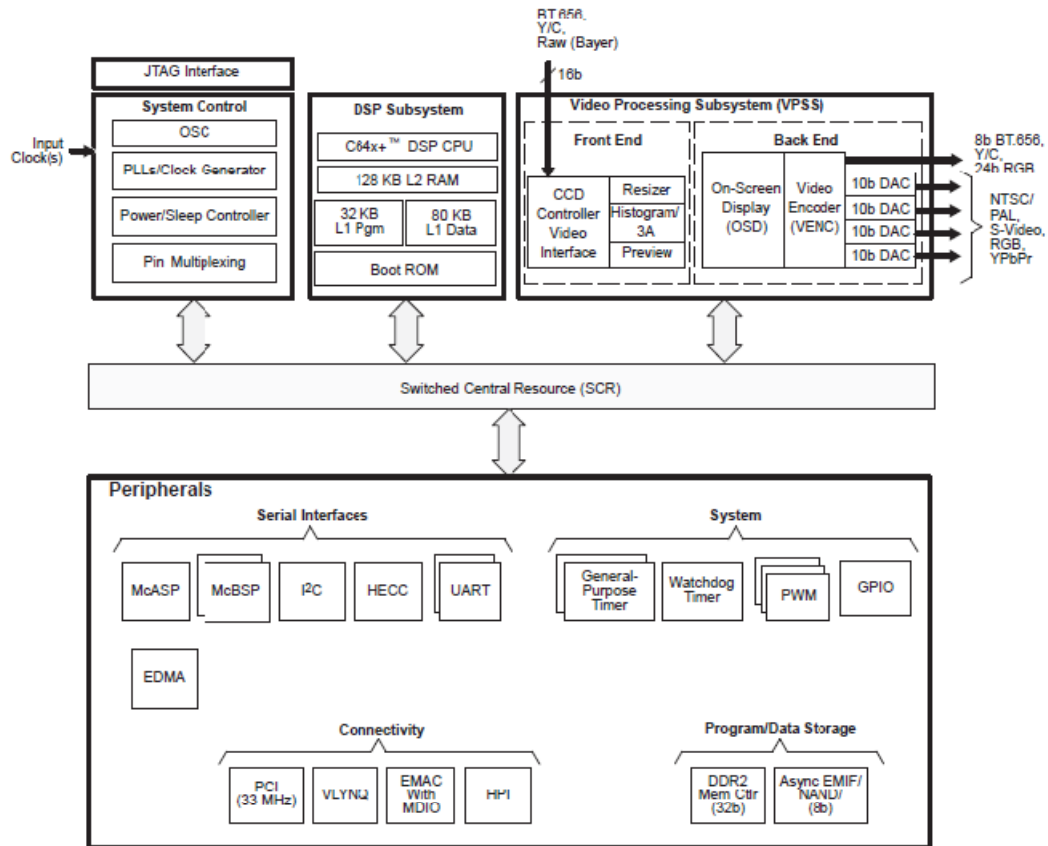
**Table 2 Some Features of the Davinci TMS320DM6437 DSP platform [43]**

The Davinci TMS320DM6437 DSP Platform consists of a rich set of peripherals, including two configurable video ports, a 4-bit transmit, 4-bit receive VLYNQ interface, a 10/100 Mb/s Ethernet MAC (EMAC), two 64-bit general purpose timers (with potential to configure each one as two separate 32-bit timers), one 64-bit watchdog timer, on-chip memory (more details can be found in later section), etc.

One of the reasons that make it more suitable for video analytics is the possession of a Video Processing Subsystem (VPSS) with two configurable Video/Imaging peripherals [44]:

- Video Processing Front-End (VPFE) input.
- Video Processing Back-End (VPBE) output.

Figure 8 illustrates the functional block diagram of the TMS320DM6437 DSP platform.



**Figure 8 TMS320DM6437 Functional Block Diagram [43].**

### IV.3 The Processor Architecture

The TMS320DM6437 CPU contains eight functional units (six ALUs and two Multipliers), two register files (A and B) and two data paths, as presented in Figure 9. Each register file contains 32 32-bit registers. All registers can be used to contain data or data address pointers [42].

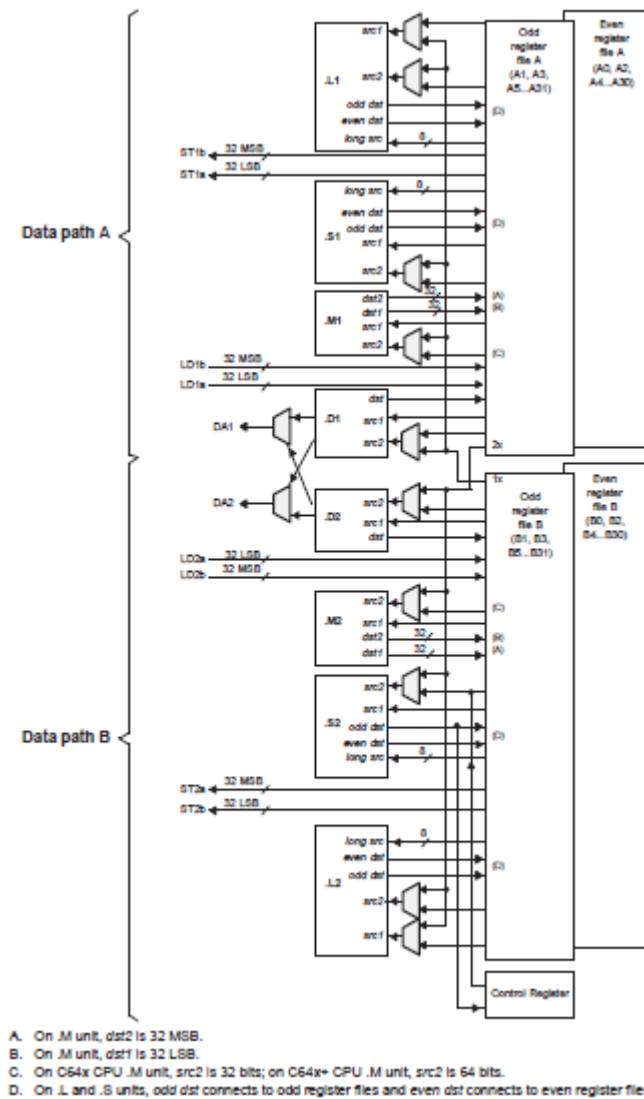
The eight functional units are designated as following:

- Multiply Operation (.M1 and .M2).
- General Arithmetic, Logical and Branch functions (.L1, .L2, .S1 and .S2).
- Load/ Store between the registers and the memory (.D1 and .D2).

All the functional units are able to execute one instruction per clock cycle.



Figure 9 shows the data path of the TMS320C6437 CPU.



**Figure 9 TMS320C6437 CPU Data Path [42]**

The following points summarize the instruction set features:

- Byte Addressable (8-16-32-64 bit data).
- 8-Bit Overflow Protection.
- Bit-Field Extract, Set, Clear.
- Normalization, Saturation, Bit-Counting.
- Compact 16-bit Instructions.
- Dedicated instructions to support complex multiplies.

## **IV.4 The Memory Management**

The memory management has always been the object of discussion in embedded systems, as the performances of the system rely on its robustness and efficiency. In video analytic applications, it is extremely important to dispose of a robust memory management, as huge amount of information has to be accessed. The TMS320DM6437 DSP platform adopts a two-level cache-based architecture, supported by a high performance memory transfer mechanism – the Direct Memory Access (DMA) [43].

### **IV.4.1 The Internal Memory**

The TMS320DM6437 DSP platform uses two levels of internal memory. The level 1 by itself is divided into two parts: the level 1 program memory/cache (L1P) and the level 1 data (L1D). The L1P cache consists of 256K-bit memory space and can considerably enhance the system performance, as it prevents external memory accesses that would be costly in terms of cycles. The L1D consists of 640K-bit memory space. The level 2 memory/cache (L2) consists of 1M-bit memory space shared between data and program. Furthermore, the L2 space memory can be set to be either a cache or RAM [45] [46].

### **IV.4.2 Integrated Direct Memory Access (IDMA)**

The TMS320DM6437 DSP platform includes an Integrated Direct Memory Access (IDMA) controller used to perform fast data transfers between the external and the internal memory and vice-versa [46] [47]. By using an IDMA mechanism, the data memory transfers can be achieved in parallel with the CPU operations.

To summarize, the IDMA is crucial for video analytic applications because:

- Optimized for contiguous memory blocks.
- Flexibility of access to and from any internal and external memory, except for some special cases.
- Use of programmable interrupts to signal transfer completion to the CPU.

## **IV.5 The Numerical issues**

In the C6000 DSP family, two numerical representations are adopted: Fixed point arithmetic and Floating point arithmetic. The TMS320DM6437 DSPs are fixed point processors [42]. Therefore, any floating point operation is performed by software [48].

### **IV.5.1 Floating Point Processor**

In this type of processor, a Floating Point Unit is used to represent numbers with different decimal positions. It is a reliable system, as there is no ambiguity in the number interpretation [49].

### **IV.5.2 Fixed Point Processor**

A system adopting fixed point arithmetic is slow when required to perform floating point operations, due to the non native execution relying on external software. Usually  $Q_n$  format is used to represent the fractions in fixed point systems. The idea of the  $Q_n$  format is to associate a fixed number ( $n$ ) of bits representing the fraction. For instance a  $Q_7$  format implies that seven bits will be used for representing the fraction. One obvious effect of this representation is that the maximum number that it can be taken is considerably reduced. Therefore, two main issues might rise [49] [50]:

- Overflow on addition or multiplication due to the limited space.
- The result of an operation cannot be represented.

## **Chapter V Proposed Approach**

The aim of this project is to design a real-time system that can perform moving object detection in a scene that is captured by a non-static camera (panning/tilting) camera. A wide range of approaches have been proposed to achieve the task. However, they assume that the scene is captured by a static camera. For non-static (panning/tilting) camera, some designs have been proposed but none of them is suitable for real-time implementation, since most are based on sprite generation for background modelling or on databases. The sprite representing the adaptive background for moving object detection needs to be stored, placing significant storage demands on the system, which is not affordable in real-time embedded systems implementation.

The background model for moving object detection using a non-static (panning/tilting) camera is required to be able to take in consideration the camera motion and include it in its model.

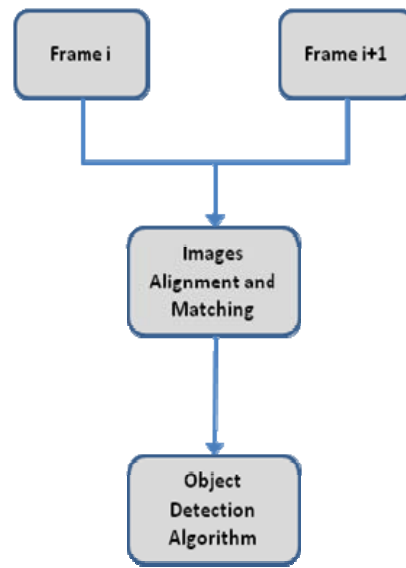
The proposed algorithm in this project is able to perform moving object detection, even in the presence of a global motion, which in our case is the camera motion (Pan and Tilt). It makes no assumptions about the camera motion (in the limits of panning and tilting) and the environmental conditions. Furthermore, the proposed algorithm does not need any initialisation stage.

In this chapter, a set of proposed solutions are illustrated in detail, with principle of their operation. All the solutions follow the same algorithm. However, they differ in the manner some parts of the algorithm are achieved.

### **V.1 Description**

The algorithm is performed between each two successive frames captured from a non-static (panning/tilting) camera. The process has to follow two stages: Frame alignment and matching stage; and object detection stage. The first stage consists of adjusting the input data in order to eliminate the global motion caused by the camera motion.

The proposed approach is illustrated in Figure10.



**Figure 10 Graph showing the proposed solution**

The proposed algorithm has different versions corresponding to various demands in precision and robustness in detriment of the resource requirements and the speed performance.

## **V.2 The Algorithm Stages**

The proposed solutions have two main stages:

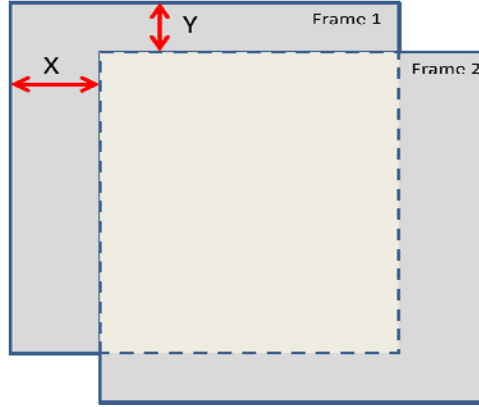
### **V.2.1 Frame Alignment and Matching Stage**

The main problem with a non-static video camera is that the background does not remain the same between two successive frames of the video. The background differences are not due to some changes that have occurred, such as luminance, but are there because the camera is no longer pointing to the same direction.

Assuming that the camera is used with a reasonable motion speed, which is the case for most video camera surveillance used for security, almost 80% of the information shared between two successive frames. In fact, they are only translated horizontally or vertically for panning and tilting motion respectively. In other words, almost all

the pixels of a specific frame have their correspondent pixel in the successive frame, except for the boundaries.

Figure 11 highlights the shared area between two successive frames of the surveillance video sequence.



**Figure 11 Shared Area between two successive frames**

The aim of the first stage is to compute the two displacement offsets in both directions, caused by the panning and/or tilting motion of the camera. In order to find the two displacement offsets ( $X$ ,  $Y$ ) between two successive frames of the video sequence, the approach proposed in this study uses the 2D correlation techniques.

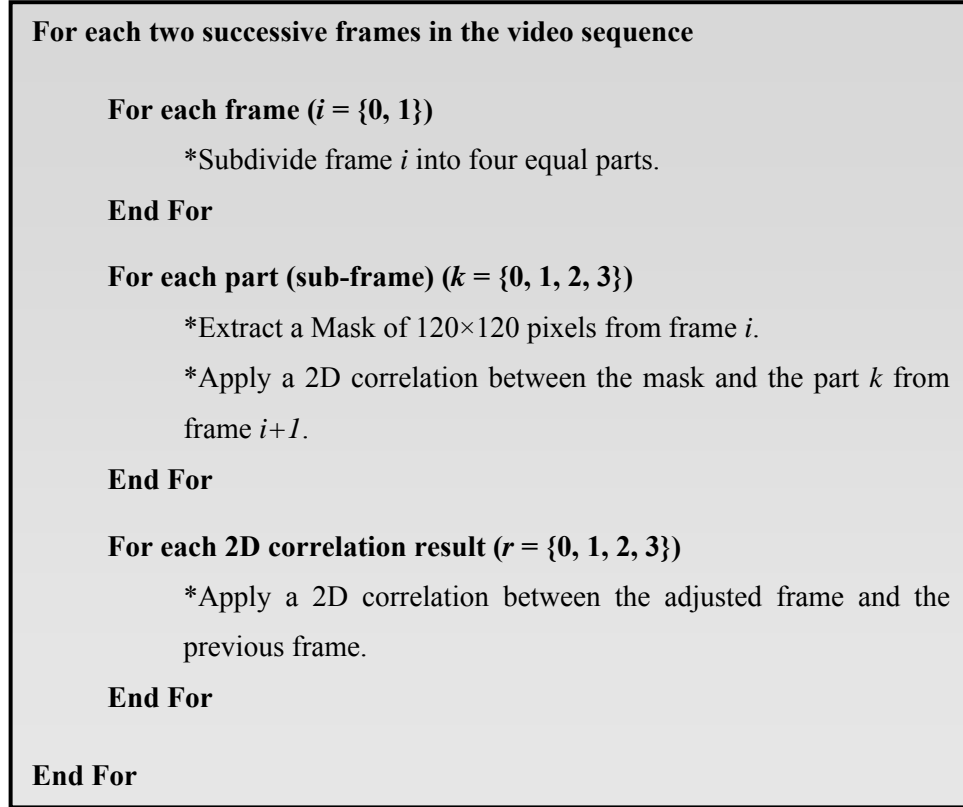
In the following sub-sections, some proposed approaches for frame alignment and matching stage will be presented.

### **V.2.1.1 Method 1**

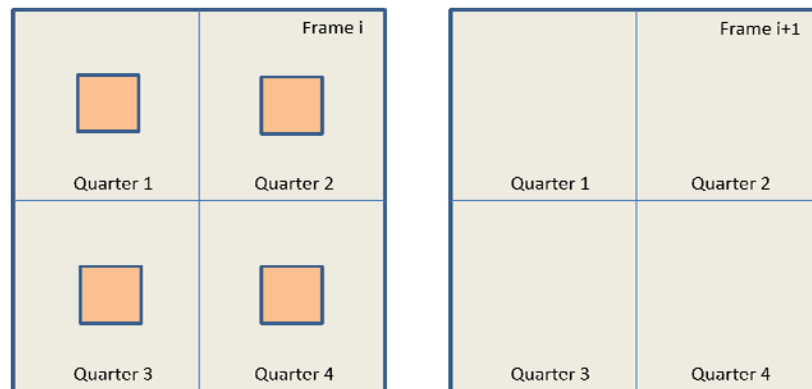
The process of finding the ( $X$ ,  $Y$ ) displacement offsets of two successive images consists of sub-dividing both successive frames (**frame  $i$**  and **frame  $i+1$** ) into four equal parts, resulting in four sub-frames (TLeft, TRight, BLeft and BRight). For each sub-frame of **frame  $i$** , a mask of  $120 \times 120$  pixels is taken from the centre as shown in Figure 13. Next, for each sub-frame, a 2D correlation between the complete sub-frame from **frame  $i+1$**  and the corresponding  $120 \times 120$  pixels mask from **frame  $i$**  is applied. The result of these operations is a set of displacement coordinates obtained from the four sub-frames. The next step is to select the more appropriate displacement coordinates that achieve the best frame matching. In order to achieve a

better precision, a 2D correlation between the corrected frame (adjusted using each displacement offset coordinates  $(X, Y)$ ) and the previous frame is applied. The displacement coordinates that yield the best correlation result are taken.

Figure 12 illustrates the proposed Method 1 of frame alignment and matching stage.

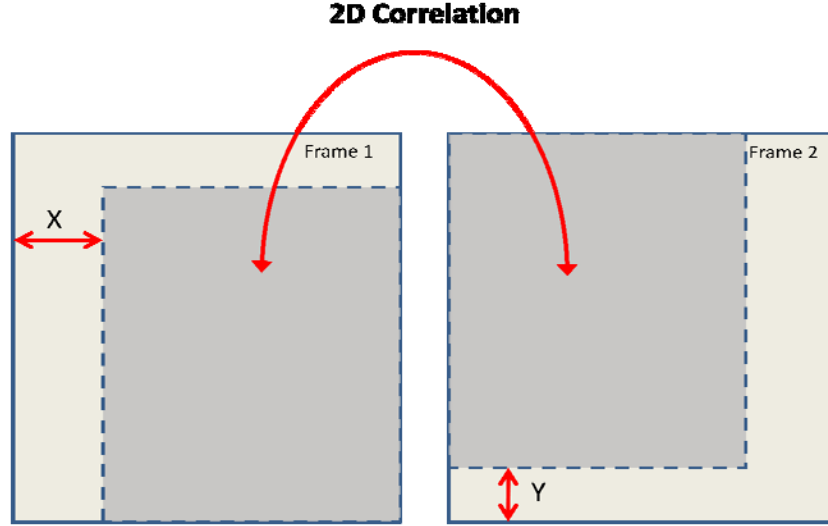


**Figure 12 The proposed Method 1 of frame alignment and matching stage**



**Figure 13 The frame division process**

Figure 14 illustrates the final correlations applied to the common area of the two successive frames.



**Figure 14 Final correlations applied to two successive frames**

To avoid the effect of luminance on the results, the 2D correlation has to be normalized.

### **V.2.1.2 Method 2**

This method is similar to the previous method, with only some changes in the configurations.

Instead of using a mask of  $120 \times 120$  pixels from frame  $i$ , a line of 60 pixels is used, which considerably reduces the amount of data to be processed. The correlation is performed between that line and a section from frame  $i+1$  determined by  $\pm 60$  pixels length. As another difference from the previous method, the final correlations (four correlations that use the set of displacement coordinates) are performed only for a part of the two images which is defined by a centred  $120 \times 120$  pixels mask. This operation will further reduce the computational load of the CPU.

Taking in consideration that the proposed system will be employed in a real-time environment that requires fast processing, especially the fact that it will be implemented in an embedded platform – namely Texas Instrument DM6437 DSP platform, this method attempts to minimise the memory access overhead between the external and the internal memory by loading fewer frame lines.

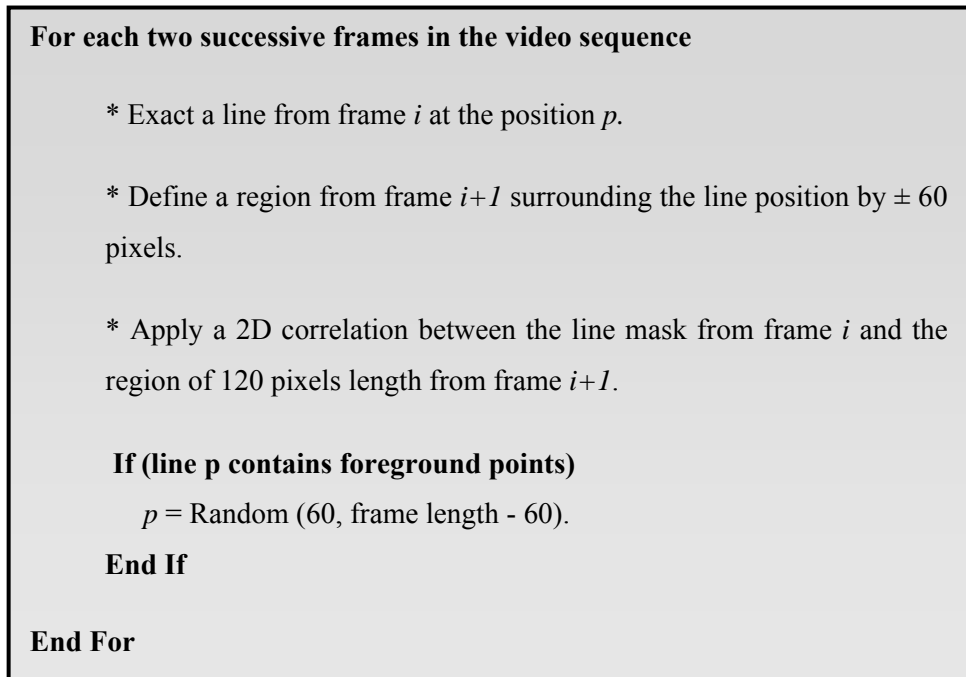


### V.2.1.3 Method 3

In this method, only one correlation is applied between a line of 520 pixels width from **frame i** and a section from **frame i+1** determined by  $\pm 60$  pixels length of the region surrounding the position of the chosen line (the mask). The choice of the line position follows a specific protocol. The line is initialized just over the centre of the frame (the centre of the image is avoided due to erroneous results generated by some imprecision of using fixed point processing). After performing the correlation and the object detection, the next step of this method is to check whether the line contains some foreground points. If that is the case, the line position will be changed randomly in a range of values that assures a panning/tilting motion tolerance of  $\pm 60$  pixels.

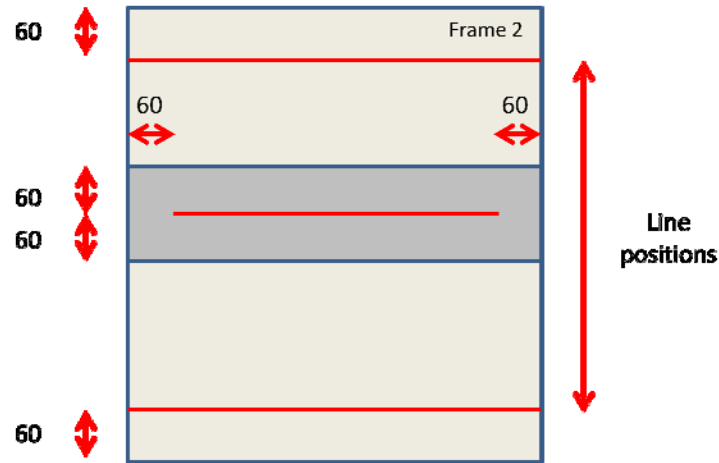
The line position is changed randomly in the case of presence of the moving object pixels due to the noise that can be introduced by the moving object pixels.

Figure 15 illustrates the proposed Method 3 of frame alignment and matching stage.



**Figure 15 The proposed Method 3 of frame alignment and matching stage**

Figure 16 demonstrates the line position range in Method 3:



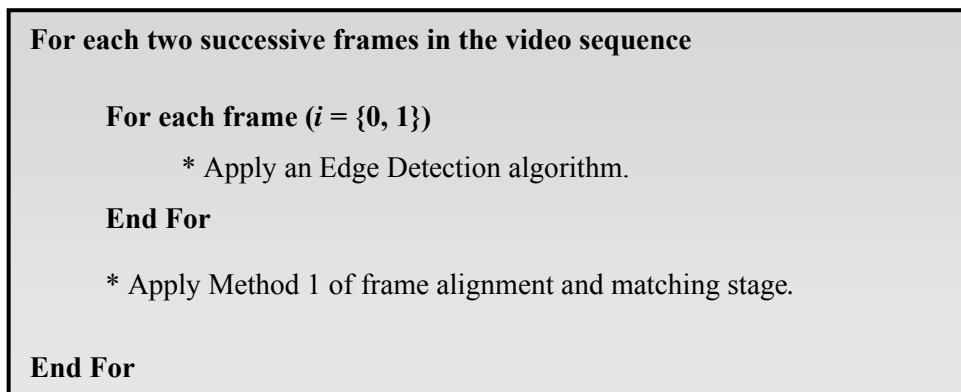
**Figure 16 Line positions**

#### **V.2.1.4 Method 4**

In this method, a pre-processing algorithm is applied to the successive frames in order to reduce the inter-frame alignment errors. This pre-processing stage consists on performing an edge detection algorithm to the successive frames. This operation will permit use of the edges present in the frames as features in 2D correlation. Once the edge frames are obtained, method 1 is applied to perform the frame alignment and matching.

A second order approach for edge detection will be used, as it is giving relatively acceptable results.

Figure 17 illustrates the proposed Method 4 of frame alignment and matching stage.



**Figure 17 The proposed Method 3 of frame alignment and matching stage**

### **V.2.2 Moving Object Detection Stage**

After the first stage that consists of the aligning and matching of each successive frame from the video sequence, the input data will be ready to be considered as a classical static moving object detection problem. Thus, moving object detection approach for static camera, such as frame difference, or a mixture of Gaussian can be applied. Furthermore, some appropriate filters can be employed to improve the moving object detection quality such as dilatation and erosion.

## Chapter VI System Implementation and Optimization

Following the design of our proposed system, a prototype is built on an embedded platform – the Texas Instrument DM6437 fixed point DSP. This implementation allows it to be integrated into surveillance cameras. Due to the fact that our system is required to satisfy the real-time constraints, considerable amount of effort and time have been invested in trying to meet the requirements, since embedded platforms are limited on resources.

In this chapter, a detailed description of the implementation aspect of the project is given with an emphasis on the optimisation stage. An overview of the code structure is provided for additional information and clarity.

### VI.1 System Requirement

The system has to perform real-time moving object detection for moving (panning/tilting) camera. Real-time system in the context of video analytics implies the ability to achieve the required functionality in the time span of one frame rate. Figure 18 illustrates the real-time requirement of the system.

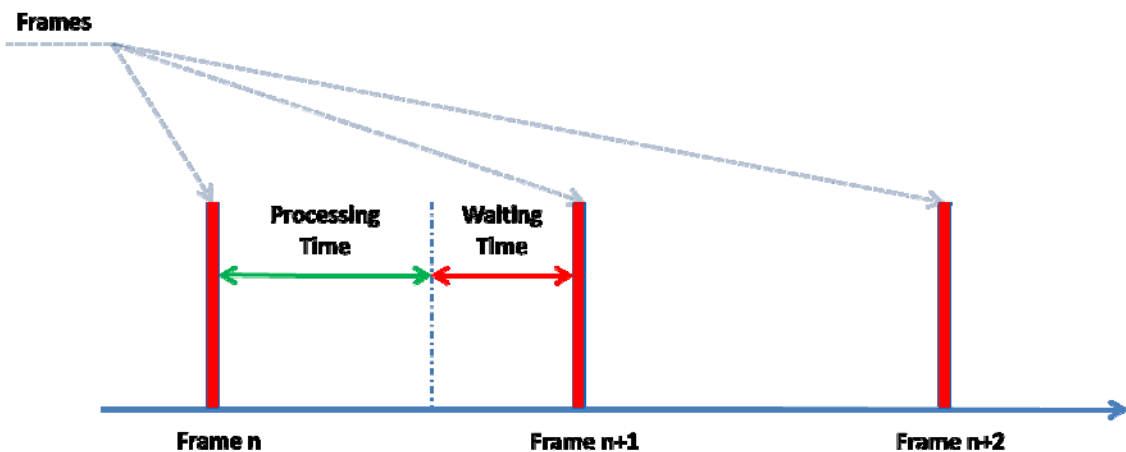


Figure 18 Real-time Constraint

## VI.2 System Hardware

A set of devices have been used to implement and evaluate the developed moving object detection for a panning/tilting camera approach.

The system hardware mainly includes:

- An Evaluation Module (EVM) that incorporates a Texas Instrument DM6437 DSP, some external hardware to support video input/output and an external memory
- A PC for debugging purpose
- A video camera / CD/DVD player
- A motorised tilt and pan system
- A display

Another device, not included above, is used in order to capture the output video sequence – a CCTV Recorder that has four (4) input channels and two (2) outputs. The CD/DVD player is used for debugging purposes. Figure 19 shows the system hardware components.



**Figure 19 System hardware components – The left image shows the input (DVD player) / output (TV monitor) devices and the DSP platform, the right image shows the panning / tilting device with its tripod**

The current real-time system implementation is based on the Texas Instrument DM6437 DSP. Figure 20 shows the Evaluation Module (EVM) for the Texas Instrument DM6437 DSP.



**Figure 20 EVM of the DM6437 DSP**

## **VI.3 System Flow**

This section describes the system flow, first globally by showing the high level system flow, and then with advanced details in a representation that include a lower level of abstraction.

### **VI.3.1 High Level System Flow**

The high level system flow of our project consists of three main functions: frame capture, video processing and the output display. The frame capture and the output display functions are performed by the Video Processing Subsystem (VPSS) that contains two configurable Video/Imaging peripherals: The Video Processing Front-End (VPFE) input and the Video Processing Back-End (VPBE) output. The video processing function represents the implementation of the current design.

Figure 21 shows the high level system flow.



**Figure 21 High Level System Flow**

### **VI.3.2 Detailed System Flow**

The detailed system flow can be approximated by four main stages. The first stage consists of extracting two consecutive frames from the video sequence. This is followed by the application of frame align and matching algorithm. The next stage is the decision making process on the most appropriate displacement coordinates for

the input data. Finally the object detection algorithm is performed. Figure 22 illustrates a detailed system flow of the implemented approach.

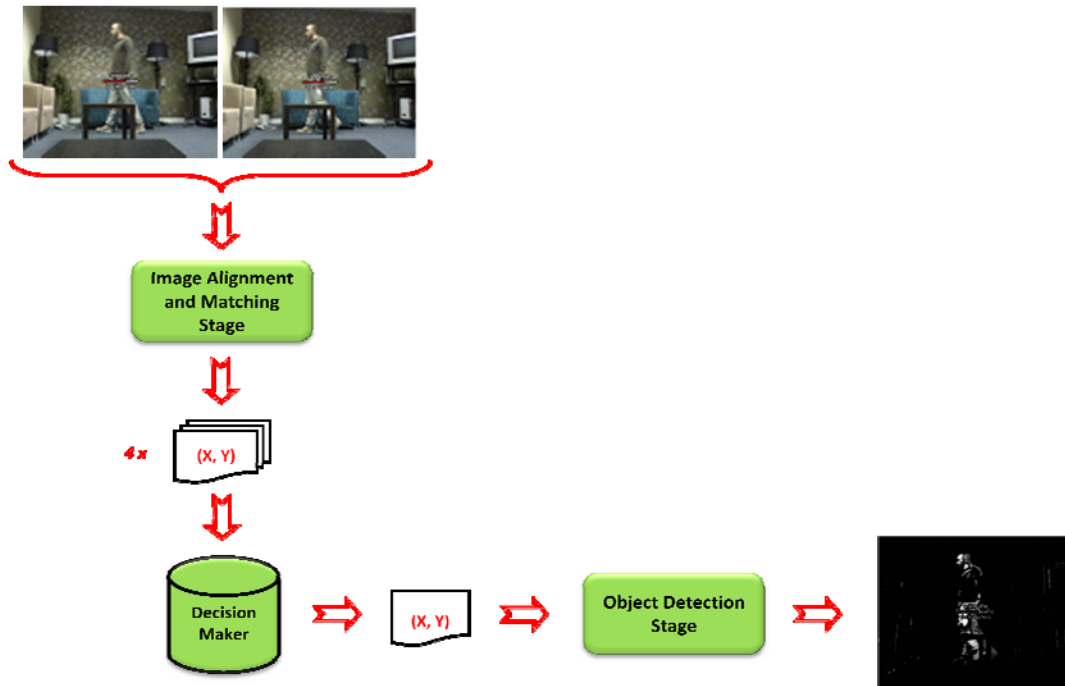


Figure 22 Detailed System Flow

## VI.4 System Optimisation

As embedded systems are limited on resources (computational capability, memory, etc.), complex algorithms cannot be implemented in real-time. For that reason, an optimisation stage is highly recommended.

The optimization process is broadly divided into three main stages: The first is related to the algorithm optimisation; the second stage of the optimisation process is concerned with the system management, mainly the memory management; and the third includes the code optimisation [50]. In the following sub-sections, a detailed description of the work achieved in the optimisation process of each part is provided.

### VI.4.1 Algorithm

The Algorithm optimisation is the first step in optimising an implementation on an embedded system such as a DSP. The algorithm should be optimal and efficient. The

aim of this optimisation process is to find a solution that yields acceptable results with less processed data.

For the implementation of solution developed in this study, method 2 has been selected for the frame alignment and matching stage due to its relatively lower computational complexity. For the second stage that consists of applying moving objects detection for static camera, the frame difference method has been selected, since the purpose of this project is to design an approach to solve the problem of real-time moving object detection for panning/tilting camera.

The first stage of the proposed solution (image alignment and matching) is designed to use normalised 2D correlations methods. However, the 2D correlation method that uses the sum of product (cross correlation) cannot be used due to the presence of divisions in the normalisation process. In fact, use of division should be avoided in programming such systems. For that reason, the Sum of Absolute Differences (SAD) has been selected, as it does not require any divisions.

#### VI.4.2 System

A considerable amount of time has been dedicated to the memory management so that image lines can be transferred fast enough from the external to the internal memory. For this purpose a DMA with a double buffering system was used. DMA has been used for data transfer to reduce the CPU usage. A scheme of double buffering is employed to improve the memory management of the system, as shown in Figure 23.

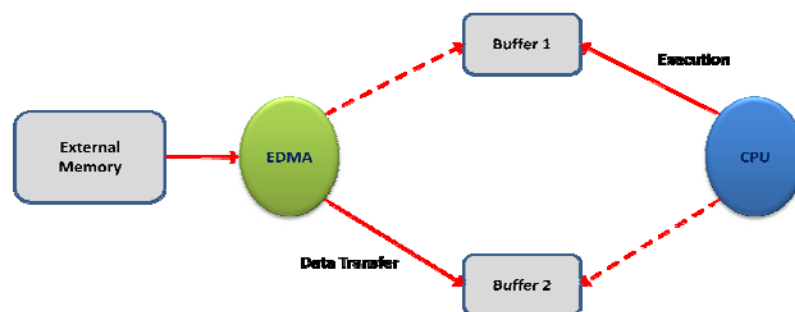


Figure 23 Double Buffering System



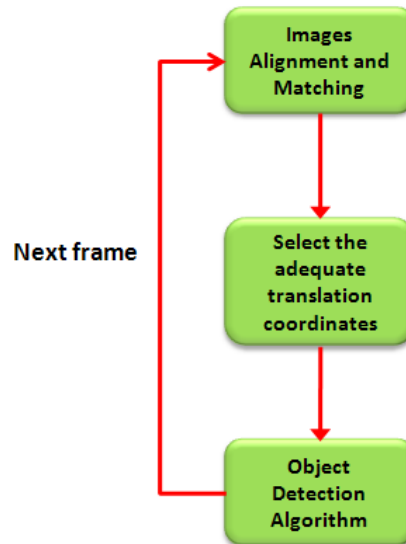
### VI.4.3 Code

The Texas Instrument DSPs are using Code Composer (CCS) software in order to implement and debug the programs. The Code Composer is using the C Language for coding the algorithms and has the capability to interface Linear Assembly and Handwritten Assembly with the C code. The main difference between the Linear Assembly and the Handwritten Assembly is that the former does not require precise specification of which functional unit will be used for each instruction.

In the present system implementation, linear assembly code will be used for the critical section of the processing only, followed by the handwriting assembly if necessary.

### VI.5 System Implementation

The system implementation can be sub-divided into three main stages, as shown in the following figure:



**Figure 24 Main stages of the system implementation**

The processing is performed line by line, whereby two lines are loaded simultaneously, one for each buffer. When buffer 1 is being processed, buffer 2 is receiving data from the external memory using the EDMA. The output of this stage

is a set of displacement positions from which the most adequate one will be selected in the following stage.

In the second stage, the choice of the adequate translation coordinate is achieved by selecting the sub-frame's result that has the least sum of absolute differences.

In the last stage, both frames are required. For that reason, a DMA channel is specified, where the current frame will be stored into the external memory to be used in the next iteration as a previous frame. This process is done similarly to the one used with the current frame (Double buffer technique).

## VI.6 Code Description

The project contains several files (Header, source and some configuration files). Table 3 presents the source files of the project with their brief description.

Function or Method (File)	Description/Contents
<b>video_preview.c</b>	The main file for the project. It contains all the initializations, including the DMA and the use of the internal memory and buffers, the frames fetching, the algorithm coding and the calls to different functions used for the project.
<b>EdmaTransfer.c</b>	The EDMA structure and configuration necessary for achieving transfers between two memory locations. For instance, between the internal and external memory and vice versa.
<b>Detect_Seg.c</b>	Code that detects the lines to be processed and/or stored.
<b>SAD.c</b>	The implementation of the Sum of Absolute Differences (SAD) correlation method.
<b>Trans_Val.c</b>	Code that detects which block has the lowest value of SAD.
<b>Min.c</b>	Basic procedure for calculating the minimum of a set of values contained in an array of integers of type <i>unsigned short</i> .

**Table 3 Description of the project source files**

## Chapter VII Experimental Results and Discussion

Following the design stage of moving object detection system for non-static (panning/tilting) camera, testing and tuning was performed through simulations using Matlab and subsequently implemented on a DSP.

In this chapter, the results of several proposed approaches are discussed and compared. Furthermore a set of experimental tests were performed on each method in order to identify the best configuration and setting. For instance, the tests strongly suggested the optimal size of the masks used for the correlations and the threshold that should be used for the Frame Difference technique in order to achieve better moving objects detection.

### VII.1 Conventions and Assumptions

In the following sections, some conventions and assumptions that have been taken in consideration during the experimentation phase will be described.

#### VII.1.1 Input Data

During the simulation and the real-time implementation on DSP phases, two input video sequences were used – Video\_test1 and Video\_test2.

Table 4 depicts some characteristics of the input data that have been used.

Input Data	Type	Number of frames/s	Length of the video (s)	Frame Width (pixels)	Frame height(pixels)
Video_test1	AVI	15	2	640	480
Video_test2	AVI	25	23	720	576

**Table 4 Input data characteristics**

### VII.1.2 Metrics

In order to evaluate the quality of moving object detection, a scale of evaluation grades was defined, starting from poor to excellent as explained in Table 5.

Grade	Description
Poor	Mediocre moving object detection or no detection at all.
Acceptable	Moving object detection achieved, but not distinguishable enough.
Good	Good moving object detection but with the presence of some noise.
Excellent	Excellent moving object detection with negligible noise.

**Table 5 Convention for moving objects detection quality**

## VII.2 Experimental Results

The study included two experimental stages. The first stage included several experimental tests based on simulation, using Matlab high-level computer language in order to validate and test the proposed approaches. The second stage implemented the most suitable approach (identified in the previous stage) on an embedded platform DSP. The results of this implementation will be shown in the real-time implementation section.

### VII.2.1 Simulation Using Matlab

A very intensive simulation phase was conducted using Matlab in order to identify optimal parameters and configurations for the system to be practically implemented. It has been one of the strongest factors in selecting the approach that fits the DSP requirements and constraints the best.

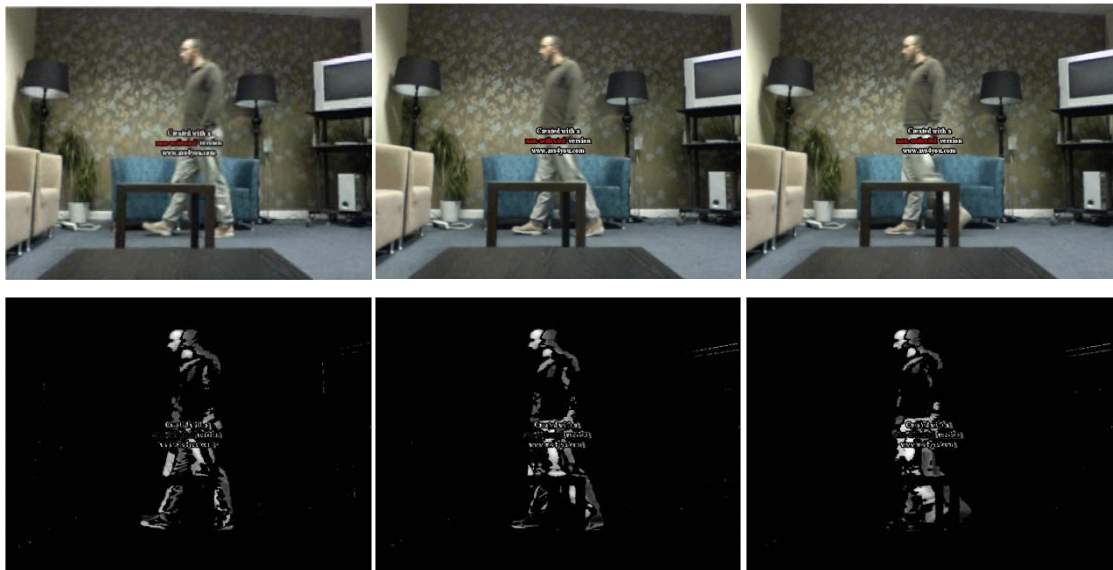
All simulations performed in Matlab incorporated normalised cross-correlation that uses the sum of products, since the aim was to test the system parameters, rather than to achieve the best real-time performance.

Only the input data (Video\_test1) were used for the simulation experiments. The simulations were running under a 2.26 GHz Core 2 Duo Processor. In the following sub-sections, a set of experiments with their results will be described and discussed.

### VII.2.1.1 Experiment 1

This experiment consists of testing the proposed approach using method 1 as a first stage of the current system (frame alignment and matching stage) and Frame Difference as a second stage (moving object detection stage).

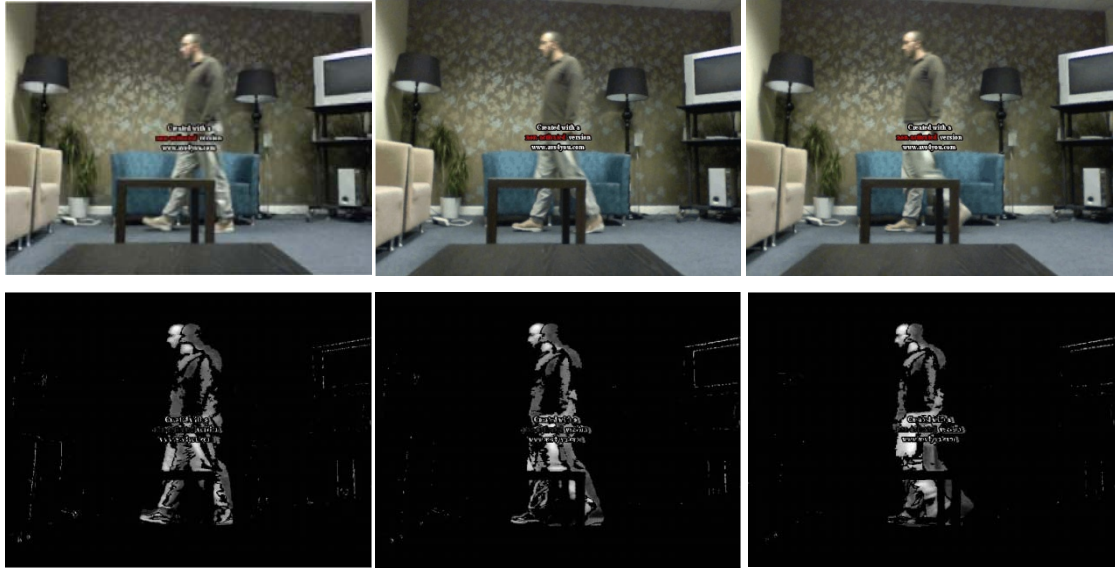
The moving object detection stage (Frame Difference) is using a threshold value of 50. The results of experiment 1 are shown in Figure 25.



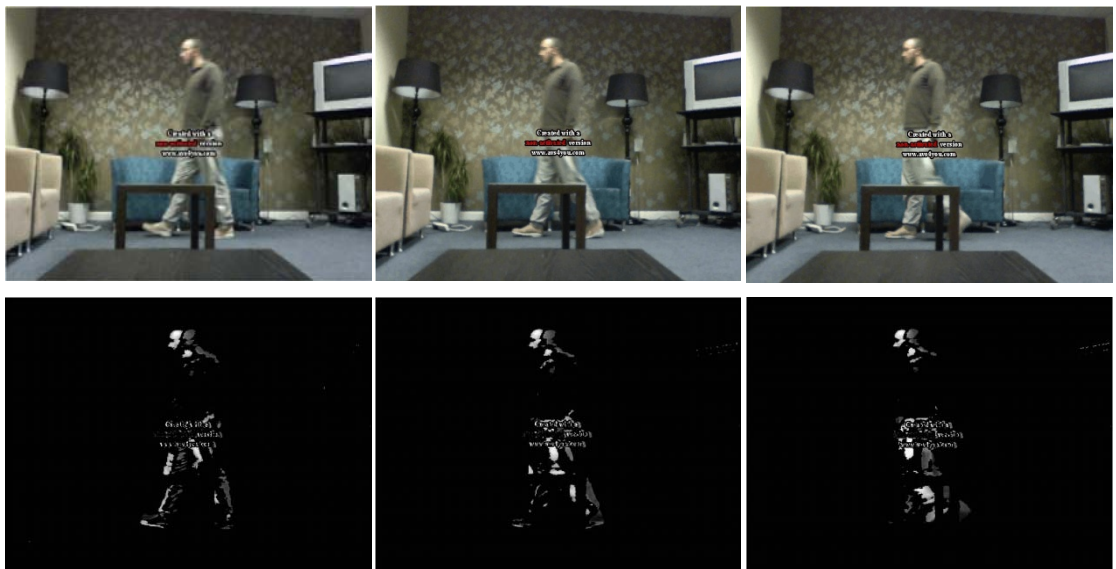
**Figure 25** The results of Experiment 1 with a threshold value of 50 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames)

The simulation was running for duration of 187.61 s. Quality of detection was placed in the category Excellent. The fact that the detected moving object has visible voids in the inner section is due to one of the drawbacks of the static moving object detection (Frame Difference) technique. Frame Difference technique is incapable of detecting the inner parts of homogeneous moving objects, since it is based on pixel by pixel comparison and models each pixel only in relation to its previous value.

Figures 26 and 27 show how the results change with the change of the threshold value. These results indicate that the Frame Difference technique is very sensitive to the threshold value.



**Figure 26 The results of Experiment 1 with a threshold value of 30 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames)**

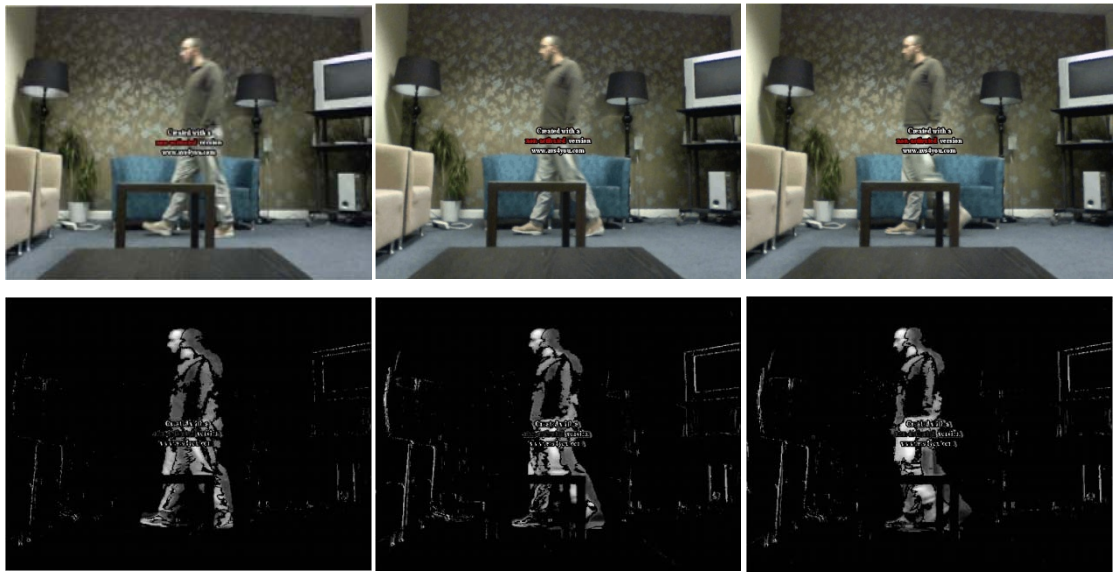


**Figure 27 The results of Experiment 1 with a threshold value of 70 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames)**

### VII.2.1.2 Experiment 2

This experiment includes testing the proposed approach using method 2 as a first stage of the current system (frame alignment and matching stage) and Frame Difference as a second stage (moving object detection stage).

The moving object detection stage (Frame Difference) is using a threshold value of 25. The results of experiment 2 are shown in Figure 28.



**Figure 28 The results of Experiment 2 with a threshold value of 25 (The three top frames represent the original frames 25, 26 and 27 respectively from left to right; the three bottom frames represent the corresponding result frames)**

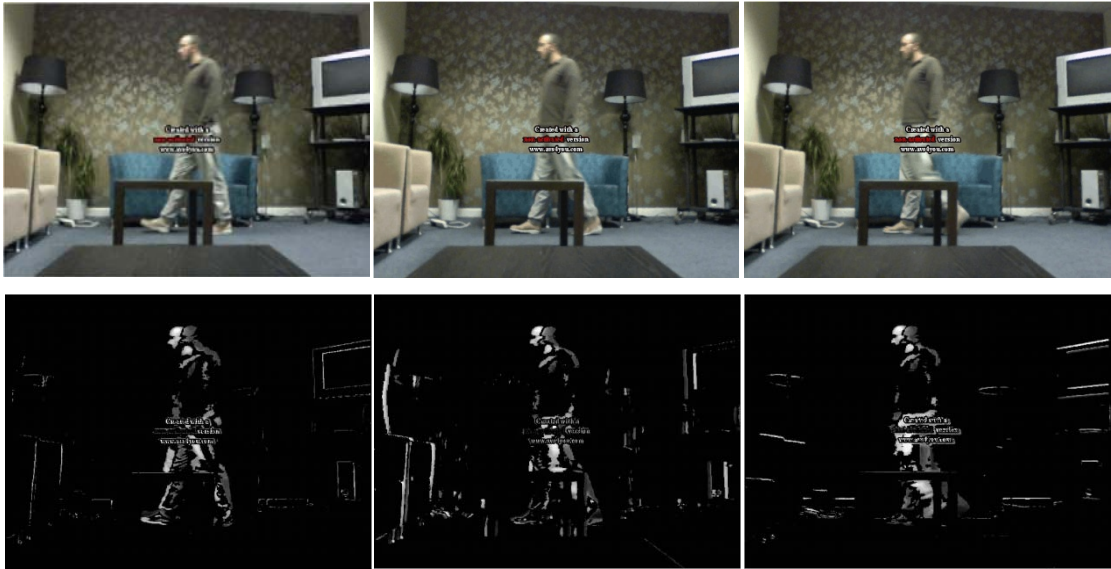
The simulation running time was 20.54 s. In this experiment the quality of detection was classified as Good. The results indicate presence of noise, due to the parallax effect and errors in the first stage.

### VII.2.1.3 Experiment 3

In this experiment, method 3 was used as a first stage (frame alignment and matching stage), whilst Frame Difference was again selected for the second stage (moving object detection stage).



Threshold of 50 was selected for the moving object detection stage (Frame Difference), and the results can be seen in Figure 29.



**Figure 29 The results of Experiment 3 with a threshold value of 50  
(The three top frames represent the original frames 25, 26 and 27;  
the three bottom frames represent the corresponding result frames)**

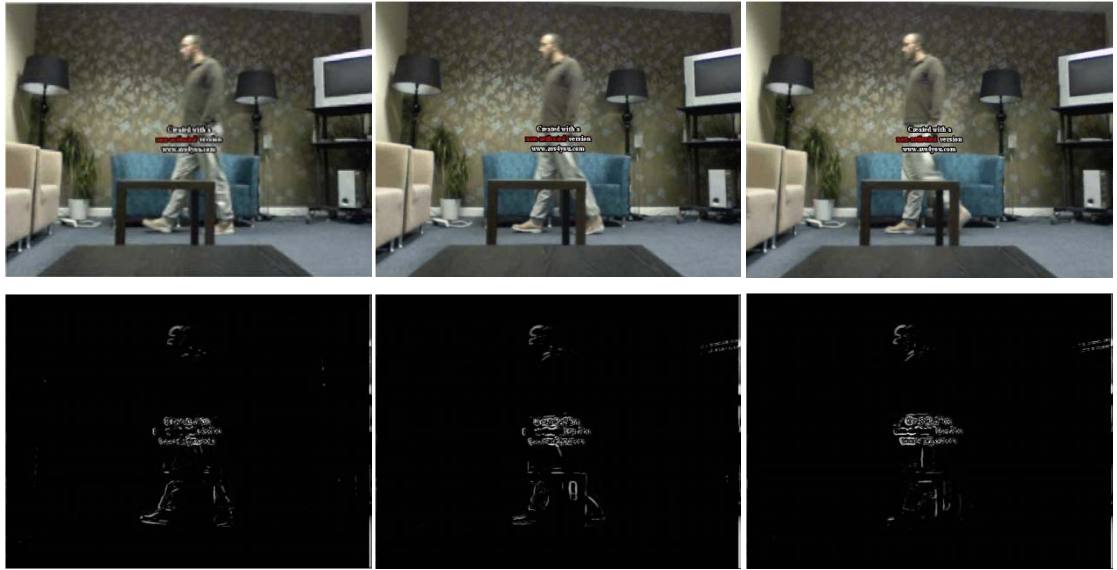
The simulation running time was 14.31 s. These settings resulted in quality of detection categorised as Poor, due to high level of noise arising from errors introduced in the first stage (frame alignment and matching). Those errors occur as a result of the presence of the detected moving object in the selected mask even when changed randomly, especially for objects of large size that occupy the entire image height.

#### **VII.2.1.4 Experiment 4**

This experiment consists of testing the proposed approach using method 4 as a first stage (frame alignment and matching stage) and, once again, Frame Difference as a second stage (moving object detection stage).

As in the previous experiment, the moving object detection stage (Frame Difference) was using a threshold value of 50. The results of experiment 4 are shown in Figure 30.





**Figure 30 the results of Experiment 4 with a threshold value of 50  
(The three top frames represent the original frames 25, 26 and 27;  
the three bottom frames represent the result frames)**

The simulation runtime was 268.01 s. In this case, the quality of detection was classified as Acceptable, as compared to the settings of the last experiment, this approach considerably reduces the noise; however, the detected moving object is not highlighted as well as in experiments 1 and 2.

### VII.2.1.5 Experiment 5

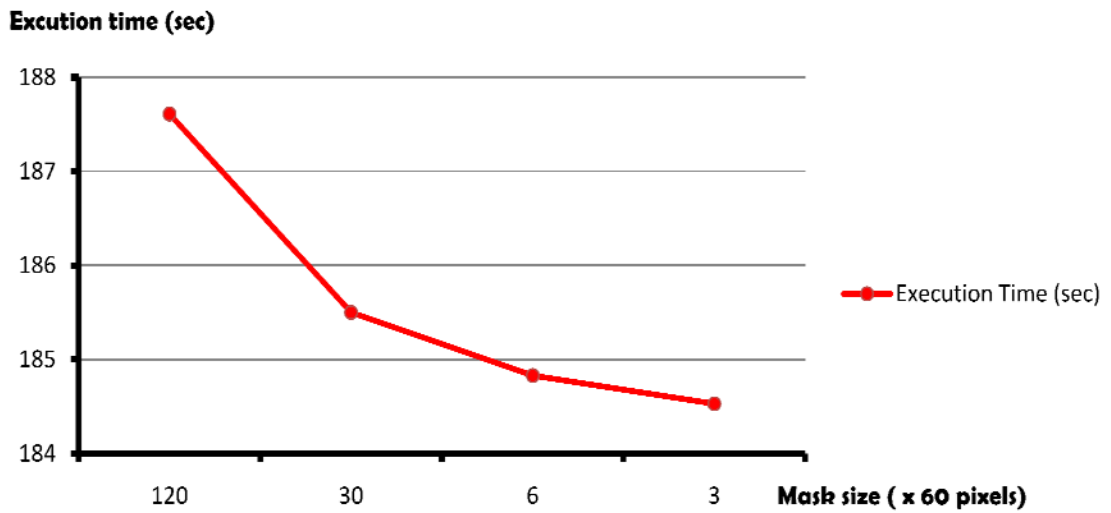
In this experiment, a set of tests have been conducted in order to analyse the speed performance of the proposed approaches. Table 6 compares their execution times.

Performance	Method 1	Method 2	Method 3	Method 4
Execution Time (s)	187.61	20.54	14.31	268.01

**Table 6 Execution time comparison**

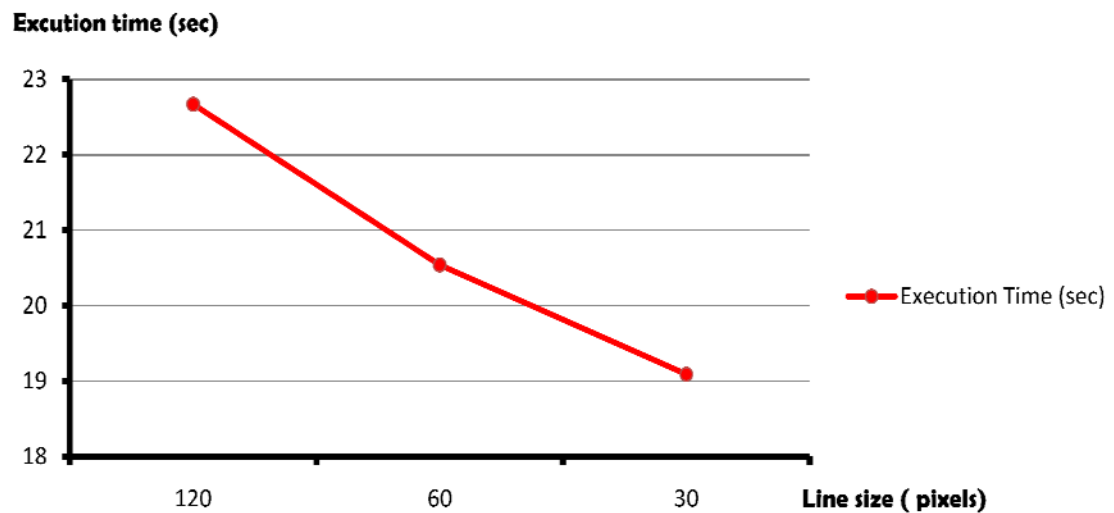
As we can see from Table 6, method 3 is the fastest method; it is running 10 times faster than the average execution time of all the proposed methods. However, method 2 presents better quality/speed ratio.

Figure 31 illustrates the relationship between the size of the mask and the execution time in method 1 for frame alignment and matching.



**Figure 31 Relationship between the size of the mask in method 1 and the execution time**

Figure 32 illustrates the relationship between the mask width and the execution time in method 2 for frame alignment and matching.



**Figure 32 Relationship between the mask width in method 2 and the execution time**

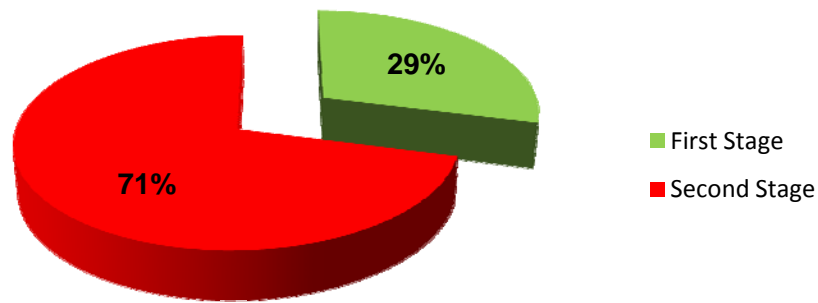
Figure 31 and 32 clearly demonstrate that the size of the mask does not dramatically influence the execution time.

## VII.2.2 Real-time DSP Implementation

Due to its good quality/execution time ratio, for the first stage of the proposed algorithm (frame alignment and matching) Method 2 has been selected for implementation.

In contrast to the simulation phase, where sum of products was used, in the DSP implementation, the Sum of Absolute Differences (SAD) was chosen, since it is more appropriate for real-time performances.

Only the input data (Video\_test2) was used for the DSP implementation testing. The implemented system was running at 28 frames/s under the frequency of 700 MHz. Figure 33 shows the CPU consumption allocation between the two stages of the proposed algorithm, namely: image alignment and matching stage; and moving object detection stage.



**Figure 33 Repartition**

It can be seen that the first stage required just a third of the CPU load. This result shows that more robust moving object detection algorithm for static camera can be used, as the adaptation stage is not computationally extensive.

The results of the DSP implementation are shown in Figure 34 and Figure 35. Figure 34 illustrates the results in a scene without moving objects, whereas in Figure 35 moving objects are present.

The actual system is running at 28 frames/s. Further improvement on the quality of detection can be achieved by using larger mask for more precision in the frame

alignment and matching stage; however, these improvements would result in inferior speed.



**Figure 34** The results of DSP implementation without moving object in the scene, with a threshold value of 60 (The three top frames represent the original frames 15, 16 and 17; the three bottom frames represent the corresponding result frames)



**Figure 35** The results of DSP implementation with moving object in the scene, with a threshold value of 60 (The three top frames represent the original frames 48, 49 and 50; the three bottom frames represent the result frames)

## Conclusion and Further Work

### Conclusion

In this study, a novel approach that achieves real-time moving object detection for non-static (panning/ tilting) surveillance camera is proposed. This approach is mainly based on detection of similar patterns from one frame to another using the 2D correlation. The algorithm follows two stages; the first stage attempts to align and match two consecutive camera frames, whilst the second takes the two matched frames and applies a static background subtraction algorithm. The reasoning that lead to the optimal design that makes the ideal compromise between the quality of detection and an acceptable quantity of data to be processed in an optimized execution time is explained below.

It is not necessary to perform a correlation across entire frames. A practical method that reduces the data requirement for correlation is to split the images into four sub-frames and then take a mask, measuring  $120 \times 120$  in our tests, from the centre of each one. The result of the correlations is a set of displacement coordinates. As these displacements will not be entirely accurate, a 2D correlation between the corrected frame and its neighbor is used for each displacement coordinate to select the best correlation result.

It is possible to reduce the computation time by reducing the square mask to a 60-pixel line, and for the second correlation using a mask of just  $120 \times 120$  pixels taken from the centre of the corrected image. This second method greatly reduces the amount of data that needs to be transferred between external memory and the processor's internal memory.

A third method attempts to reduce overhead even further by taking a 520-pixel line from one image and then correlating that with a block of  $120 \times 520$  pixels from the following frame centered on the position of the chosen line. Initially, the line is placed approximately in the centre of the image –the exact centre is not used to avoid accuracy problems caused by fixed-point processing. However, care must be taken to ensure that the line does not contain any pixels from a moving, foreground object. If

that is the case, the line position is randomly changed within a range of 60 pixels in any direction.

A fourth method combines the first method with a prior pre-processing phase applied to the frames. This approach is less sensitive to the parallax problem and errors due to the frame alignment and matching stage. However, the quality of the moving object detection is not satisfactory.

The process of aligning images makes it possible to perform moving object detection using techniques developed for static cameras. Several approaches exist for resolving the problem of object detection with static background. The simplest approach is to subtract one image from the other, and work out whether each pixel is a foreground or background object using a threshold function to. The problem with this simple approach is that the results depend heavily on the choice of threshold value.

For the final system implementation, the frame difference approach using a fixed threshold was chosen because of its lower complexity compared to the other techniques.

The algorithm was implemented using a Texas Instrument DM6437 digital signal processor (DSP). Memory management was an important aspect of this design, as it was critical that image lines could be transferred efficiently from external to internal memory and vice versa. A combination of direct memory access was finally implemented, making use of the DSP's enhanced DMA (EDMA) unit, and double-buffering to minimize processor load.

Processing for the production of the position-corrected frames and the moving object detection is performed line by line, whereby two lines are loaded simultaneously, one for each buffer. In this parallel process, while buffer one is being processed, buffer two is receiving data from the external memory using the EDMA. However, as the full frames are needed for the stage that detects moving foreground objects, the incoming frame is stored in external memory. Thus in order to avoid loading the buffers twice, the transfers are initiated during the frame alignment and matching stage, just after using the DMA to load the current frame line in the internal buffers. The proposed system is running at 19 frames per second under a frequency of 700 MHz, meeting its real-time constraints and allowing more algorithms to be integrated

if needed. This design has succeeded in massively reducing the compute overhead for performing moving-object detection in automated video surveillance systems that use panning and tilting cameras, making it possible to run the algorithms in real time on low-cost embedded hardware.

The proposed approach has been accepted for presentation in the European DSP Education and Research Conference (EDERC) that will take place in Nice, France on 1<sup>st</sup> and 2<sup>nd</sup> of December 2010. The modifications for the final version are due by 18<sup>th</sup> October 2010 (see Appendix A).

## **Further Work**

A more stable and robust system can be easily achieved by improving the first stage (frame alignment and matching). For instance, designing a strategy for the mask position somehow it will avoid areas belonging to moving objects. In addition, the proposed system can be further optimized, especially the coding stage, to allow faster moving object detection, tracking and/or extra features.

The proposed solution can be generalized to other camera motions and configurations, such as rotation or zoom. The system can be adapted to be used in automotive application where the forward car motion can be considered as a zoom between two successive frames.

More robust moving object detection approaches for static camera can be implemented as a second stage of our proposed approach, since the first stage requires only 29% of the overall computational complexity.

## References

- [1]. Hannah M. Dee · Sergio A. Velastin: "How close are we to solving the problem of automated visual surveillance?", *Machine Vision and Applications* (2008) 19:329–343 DOI 10.1007/s00138-007-0077-z.
- [2]. Alan M. McIvor, "Background Subtraction Techniques," Reveal Ltd PO Box 128-221, Remuera, Auckland, New Zealand.
- [3]. U.B. Desai, S.N. Merchant, Mukesh Zaveri, G. Ajishna, Manoj Purohit, and H.S. Phanish, "Small Object Detection and Tracking: Algorithm, Analysis and Application", First international conference on Pattern recognition and machine intelligence, Kolkata, India, December 20-22, 2005.
- [4]. JongBae Kim, "A REAL-TIME MOVING OBJECT DETECTION USING WAVELET-BASED NEURAL NETWORK," School of Computer Eng., Seoul Digital University, Seoul, South Korea.
- [5]. T. Xia, C. Liu, H. Li, "An Efficient Moving Object Detection and Description," Centre for Wavelets, Approximation and Information Processing, National University of Singapore.
- [6]. Computation, <https://computation.llnl.gov/casc/sapphire/background/background.html>, accessed 20<sup>th</sup> August 2010.
- [7]. Cristani M., Bicego M., Murino V. Multi-level background initialization using Hidden Markov Models. In First ACM SIGMM Int workshop on Video surveillance 2003; 11-20.
- [8]. C. Ridder, O. Munkelt, and H. Kirchner. Adaptive background estimation and foreground detection using kalman-filtering. In ICRAM, 1995.
- [9]. K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: principles and practice of background maintenance", *Proc. Int. Conf. Computer Vision*, pp. 255 - 261, 1999 .



- [10]. C. Stauffer and W. E. L. Grimson: Adaptive background mixture models for real-time tracking in: Computer Vision and Pattern Recognition Fort Collins, Colorado (Jun.1999) pp. 246-252.
- [11]. F. Porikli, Achieving real-time object detection and tracking under extreme conditions, Journal of Real-Time Image Processing, Springer, 2006.
- [12]. Porikli, F., Tuzel, O.: Bayesian background modeling for foreground detection. In: Proceedings of the ACM Visual Surveillance and Sensor Network (2005).
- [13]. O. Javed, K. Shafique, and M. Shah. A hierarchical approach to robust background subtraction using color and gradient information. In MVC, pages 22–27, Florida, December 2002.
- [14]. Ahmed Elgammal, David Harwood, Larry Davis “Non-parametric Model for Background Subtraction” 6th European Conference on Computer Vision. Dublin, Ireland, June/July 2000.
- [15]. Sonsoles Herrero, Jesús Bescós, “Background Subtraction Techniques: Systematic Evaluation and Comparative Analysis\*,” in J. Blanc-Talon et al. (Eds.): ACIVS 2009, LNCS 5807, pp. 33–42, 2009.
- [16]. . Lipton, H. Fujiyoshi, and R. S. Patil, "Moving target classification and tracking from real-time video," in Proceedings IEEE Workshop on Application of Computer Vision, 1998, pp. 8 - 14.
- [17]. R. Ewerth and B. Freisleben, “Frame Difference Normalization: An Approach to Reduce Error Rates of Cut Detection Algorithms for MPEG Videos,” in Proc. of IEEE International Conference on Image Processing, Vol. II, Barcelona, 2003, pp. 1009-1012.
- [18]. M. Piccardi, "Background subtraction techniques: a review," The ARC Centre of Excellence for Autonomous Systems (CAS), Faculty of Engineering, UTS, April 15, 2004.
- [19]. Yu-An Shih, “Real-time video object tracking and classification using an embedded DSP platform", University of Bristol, MSc Project 2009.

- [20]. S.S. Cheung, C. Kamath, Robust techniques for background subtraction in Urban traffic video, in: Video Communications and Image Processing. SPIE Electronic Imaging, vol. 5308, San Jose, California, Jan 2004.
- [21]. M. Piccardi, "Background subtraction techniques: a review," in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, vol. 4, 2004. pp. 3099-3104.
- [22]. C. Wren, A. Azarbayejani, T. Darrell and A. Pentland, "Pfnder: Real-Time Tracking of the Human Body," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 780-785, July 1997.
- [23]. S. Su and Y. Chen, " Moving Object Segmentation Using Improved Running Gaussian Average Background Model", Digital Image Computing: Techniques and Applications, 2008.
- [24]. M. Seki, T. Wada, H. Fujiwara, K. Sumi, "Background Subtraction based on Cooccurrence of Image Variations," Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03).
- [25]. N. Oliver, B. Rosario and A. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions," Proc. Int'l Conf. Vision Systems '99, Jan. 1999.
- [26]. Y.L.Tian and A.Hampapur, "Robust Salient Motion Detection with Complex Background for Real-time Video Surveillance", in Proc. Of IEEE Computer Society Workshop on Motion and Video Computing, January, 2005.
- [27]. X. Fang, W. Xiong, B. Hu, L. Wang, "A Moving Object Detection Algorithm Based on Color Information," International Symposium on Instrumentation Science and Technology, vol. 48, pp. 384-387, 2006.
- [28]. Xue Mei, Fatih Porikli : "Fast Image Registration via Joint Gradient Maximization: Application to Multi-Modal Data", TR2006-109 September 2006.
- [29]. Medha V. Wyawahare, Dr. Pradeep M. Patil, and Hemant K. Abhyankar: "Image Registration Techniques: An overview", International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 2, No.3, September 2009.

- [30]. Mathworks, <http://www.mathworks.com/access/helpdesk/help/toolbox/images/f20-14791.html>, accessed 28<sup>th</sup> August 2010.
- [31]. W. E. L. Grimson , C. Stauffer , R. Romano , L. Lee, “Using Adaptive Tracking to Classify and Monitor Activities in a Site”, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, p.22, June 23-25, 1998.
- [32]. Barbara Zitova', Jan Flusser: "Image registration methods: a survey", Image and Vision Computing 21 (2003) 977–1000.
- [33]. Wikipedia, "Sprite", [http://en.wikipedia.org/wiki/Sprite\\_\(computer\\_graphics\)](http://en.wikipedia.org/wiki/Sprite_(computer_graphics)), accessed 26<sup>th</sup> August 2010.
- [34]. Muhammad Asif, John J. Soraghan, 2008. MPEG7 Motion Descriptor Extraction for Panning Camera using Sprite Generated. IEEE Inter. Conf. on Advanced Video and Signal Based Surveillance, Sep. 2008.
- [35]. R. V. Babu and K. R. Ramakrishnan "Background sprite generation using MPEG motion vectors", Proc. 3rd Indian Conf. Comput. Vis., Graph. Image Process., pp. 7 2002.
- [36]. Yiming Ye, John K. Tsotsos, Karen Bennet, and Eric Harley: "Tracking a Person with Pre-recorded Image Database and a Pan, Tilt, and Zoom Camera", Lecture Notes in Computer Science, 1997.
- [37]. Naveed I. Rao, Huijun Di, Guangyou Xu: Joint Correspondence and Background Modeling Based on Tree Dynamic Programming. ICPR 2006, Vol 2: pp. 425-428.
- [38]. Yuxin Jin, Linmi Tao, Huijun Di, Naveed I. Rao, Guangyou Xu: Background modeling from a free-moving camera by Multi-Layer Homography Algorithm. ICIP 2008: 1572-1575.
- [39]. K. Jiman, K. Daijin, “Moving Object Detection under Free-moving Camera,” Computer Science and Engineering, POSTECH.

- [40]. Wikipedia, "Texas Instruments,"  
[http://en.wikipedia.org/wiki/Texas\\_Instruments#Texas\\_Instruments\\_TMS320](http://en.wikipedia.org/wiki/Texas_Instruments#Texas_Instruments_TMS320),  
accessed 3<sup>rd</sup> September 2010.
- [41]. TI.com, <http://focus.ti.com/docs/prod/folders/print/tms320dm6437.html#features>,  
accessed 2<sup>nd</sup> September 2010.
- [42]. Texas Instruments, "TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide," SPRU732J, July. 2010.
- [43]. Texas Instruments, "TMS320DM6437 Digital Media Processor," SPRS345D, June. 2008.
- [44]. Texas Instruments, "How to Use the VPBE and VPFE Driver on TMS320DM643x Devices," SPRAAP3A, November. 2007.
- [45]. Texas Instruments, "TMS320C64x+ DSP Cache User's Guide," SPRU862B, February. 2009.
- [46]. Texas Instruments, "TMS320C64x+ DSP Megamodule Reference Guide," SPRU871K, August. 2010.
- [47]. Texas Instruments, "TMS320DM643x DMP Enhanced Direct Memory Access (EDMA3) Controller User's Guide," SPRU987A, March. 2008.
- [48]. Z. Nikolic, "Implementation considerations for single-camera steering assistance systems on a fixed point DSP," in Proc. IEEE Intell. Vehicles Symp., Eindhoven, The Netherlands, Jun. 2008, pp. 697–702.
- [49]. Dheeshan Sasi, "Real-time video stabilisation on the TI DM642 platform", University of Bristol, MSc Project 2009.
- [50]. Naim Dahnoun, Digital Signal Processing Implementation Using the TMS320C6000 DSP Platform, Addison-Wesley Longman Publishing Co, Inc, Boston, MA, 200.

## Appendix A: Paper

### REAL-TIME OBJECT DETECTION FOR A PANNING/TILTING SURVEILLANCE CAMERA USING AN EMBEDDED DSP PLATFORM

M. L. Lemmou and N. Dahnoun

School of Electronics, University of Bristol

Merchant Venture Building, Woodland Road, Bristol, BS8 1UBR, Bristol, UK

phone: + (44) 1179545181, email: ml9267@bristol.ac.uk, Naim.Dahnoun@bristol.ac.uk

#### ABSTRACT

*Object detection for a non-static (Panning and Tilting) camera has been the centre of interests of various applications. In this paper, we present a novel real-time object detection approach that can efficiently handle the problem of a moving camera that performs panning and tilting motions to incorporate a wide field of view for more security and a wider angle of view. We present a set of possible solutions where the optimal is implemented on an embedded Texas Instrument DM6437 DSP platform which is suitable for video processing. In the presence of a non-static camera, classical object detection algorithms cannot be used. Therefore our idea consists of two main stages: The first stage attempts to align the two successive frames of the video by using a method based on 2D –correlation (Sum of absolute differences (SAD)); the second stage involves one of the classical background subtraction algorithms. The system was successively simulated in Matlab environment and then implemented in DSP platform.*

#### 1. INTRODUCTION

The need of real-time object detection for video surveillance has spawned a huge amount of our daily life, especially in some domains where it has received considerable attention, for instance: criminology, sociology, statistic, traffic accident detection and military applications. Moving object detection is considered to be the most important task in automated video surveillance systems. It represents the low level image processing technique which is the basic of automated video surveillance.

The aim of the object detection in video surveillance analytics is to distinguish moving foreground object from a static background. In the case of panning camera, a global motion which is the camera motion needs to be compensated for. Several approaches exist for resolving the problem of object detection with static background such as (1) “Background subtraction” which is the most common approach used in fixed camera scenarios. (2) “Predictive techniques” which are based on the temporal colour variations at each pixel of the video. (3) Others methods dedicated to multimodal background exist and often there are assigned as *Gaussian functions*; these approaches iterative update mechanisms are applied to fit the modal, for instance, online expectation maximization (EM)

algorithm. There are also Bayesian update mechanisms which are more appropriate to some changes in the scene such as illumination. We can also find variants of the mixture of model background that uses image gradient and optical flow information. All these techniques do not take in consideration the scenario of panning and tilting surveillance camera [1].

Various methods of moving object detection for non-static camera have been cited [2] [3], but none of them achieved real-time performance. Then we propose a novel approach based on applying some image transformations to bring the object detection for panning/tilting camera problem to a classical object detection algorithm (Frame Difference, Mixture of Gaussian ...etc).

This paper is divided into six sections. The first section is the present introduction which aims to provide the context of the paper and a brief overview of the classical techniques used for objects detection. In the second section, a review of some of the previous work related to non-static (Panning/Tilting) surveillance camera is presented. A detailed description of the proposed approach is investigated in the third section. This section might be divided into two main sub-sections that illustrate the different approach’s stages. The fourth section describes all the considerations taken to implement the proposed solution. The fifth section states the undertaken experimentations and highlights the results. Conclusions are given in the sixth section.

#### 2. PREVIOUS WORK ON OBJECT DETECTION FOR PANNING/TILTING SURVEILLANCE CAMERA

One of the consistent approaches used to handle the difficulty of moving object detection for panning camera, is the generation of the sprite (Mosaic). Asif and Soraghan in [2] proposed a technique for detecting moving foreground object in a presence of a panning camera. This approach is able to determine and extract moving objects even in the presence of another exterior motion which is the camera motion, usually known as Global Motion. Their approach follows two fundamental stages; the first stage is the foreground pixel rejection process, which consists on separating the object motion from the global motion of the camera, assuming that a small global motion is used. The

second stage involves the generation of a static panoramic image (sprite) for panning camera motion. Eventually this sprite will be used as an adaptive background reference to detect moving objects. This approach is applicable when the camera motion is very low, precisely much lower than the object motion. They also consider the pixel changes due to global (camera) motion to have a strong local correlation. The image is divided into block sizes of 64x64. A motion vector will be determined on those blocks by using a technique of phase correlation (FFT). An obvious disadvantage of this approach is the huge computational complexity which is not suitable for real-time application, where lots of processes are merged together to provide the complete system. Another important point, this approach handles only the panning motion of the camera and not the tilting motion. However, it can tolerate some vertical camera's vibrations due to the panning in a non-stable platform.

In another approach presented in [3], a set of the camera's states known as **MCPS** (Minimum Camera Parameter Settings) is defined. Then, for each state of the set (**MCPS**), a background image of the camera view is attributed. The result of this procedure is an image database. The **MCPS** is basically used to facilitate object's detection and tracking in a moving surveillance camera. Object detection is simply performed by comparing the current image with the corresponding reference image (Background image) related to the state from the database previously defined. This approach can only be applied to a fixed environment.

### 3. PROPOSED APPROACH

The proposed algorithm in this paper is able to perform moving object detection even in the presence of a global motion which is in our case the camera motion (Pan and Tilt). The process has to follow two stages: Frames alignment and matching stage; and the object detection stage. The first stage consists of adjusting the input data. The proposed approach is illustrated in the following diagram.

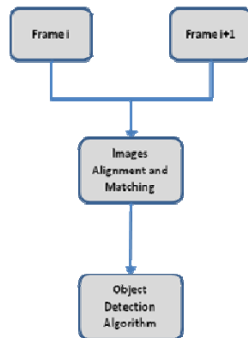


Figure 1 Graph showing the proposed solution.

#### 3.1 Frames Alignment and Matching

The fact with a non-static video surveillance camera is that the background is not really the same between two successive frames of the video. Another point is that with a

reasonable motion speed of the camera, almost 90% of the information is common to both frames. In fact, there are only translated horizontally or vertically respectively for panning and tilting motion. Figure 2 highlights the common area between the two successive frames of the surveillance video.

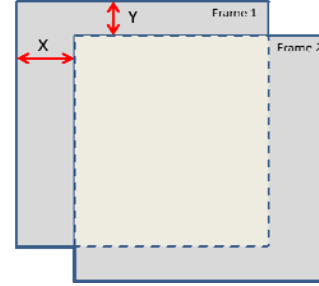


Figure 2 Shared Area between two successive frames.

The aim of the first stage is to compute the two displacements in both directions caused by the panning and tilting motion of the camera.

Our approach uses the 2D correlation techniques in order to find the translation coordinates ( $X$ ,  $Y$ ) between two successive frames of the video.

In the following sub-sections, we will be presenting some proposed approaches for frames alignment and matching, some of them will not be implemented for optimisation reasons.

##### 3.1.1 Method 1

The process of finding the ( $X$ ,  $Y$ ) displacements of two successive images consists of sub-dividing each successive frames (**frame i** and **frame i+1**) in four equal frames resulting in four sub-frames (TLeft, TRight, BLeft and BRight). Then for each sub-frame of **frame i**, a mask of 120 x 120 pixels is taken from the centre as shown in Figure 3. Next, for each part, we apply a 2D correlation between the sub-frame from **frame i+1** and the corresponding 120 x 120 mask from **frame i**. The results of these operations are a set of displacement coordinates obtained from sub-frames. The next step is to select the more appropriate displacement coordinates that gives the best matching of the frames. In order to achieve better precision, a 2D correlation between the corrected frame (Adjusted using a translation of ( $X$ ,  $Y$ )) and the previous frame is applied. Then the displacement coordinates that gives the best correlation result are taken.

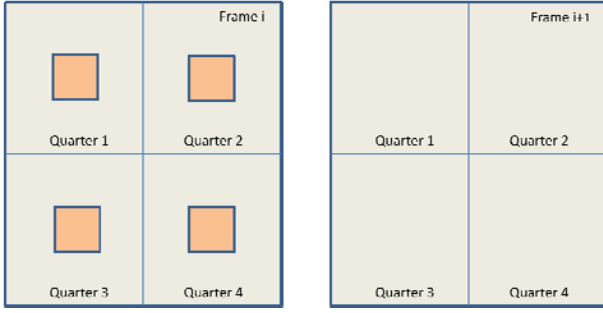


Figure 3 The dividing process of the frames.

Figure 4 illustrates the final correlations applied to the common area of the two successive frames.

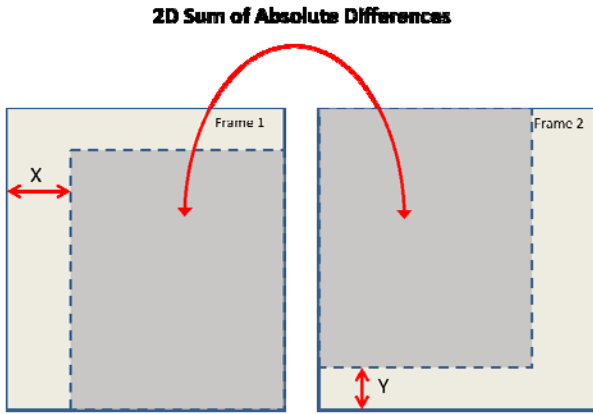


Figure 4 Final correlations applied to two successive frames.

### 3.1.2 Method 2

This method is similar to the previous methods, only some configurations changed.

Instead using a mask of 120 x 120 pixels, we will be using a line of 60 pixels which reduces considerably the amount of data to be preceded. Another difference from the previous method, the final correlations (four correlations that use the set of displacement coordinates) are performed only for a part of the two images which is defined by a centred 120 x 120 mask.

Taking in consideration that our system will be employed in a real-time environment that requires fast processing, especially the fact that it will be implemented in an embedded platform namely Texas Instrument DM6437 DSP platform, this method attempts to minimise the memory transfer overload between the external memory and the internal memory by loading less frame lines.

### 3.1.3 Method 3

In this method, only one correlation is applied between a line of 520 pixels width from **frame i** and a section from **frame i+1** determined by  $\pm 60$  pixels surrounding the position of the chosen line (the mask). The choice of the line's position follows a specific protocol. The line is initialized just over the centre of the frame (we avoid the

centre of the image due to wrong results generated by some inexactitudes of using fixed point processing). After performing the correlation and the object detection, the next step of this method is to check whether the line contains some foreground point. If it is the case, the position of the line will be changed randomly in a range of value that assures a panning/tilting motion tolerance of  $\pm 60$  pixels.

Figure 5 demonstrates the line position in Method 3:

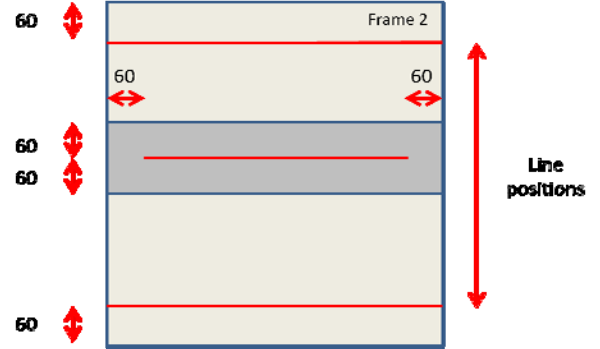


Figure 5 Line positions.

## 3.2 Moving Object Detection

After the first stage which consists on aligning and matching each successive frames from the video, the input data will be ready to be considered as a classical static moving object detection problem, and moving object detection approach such as frame difference or a mixture of Gaussian which is a more robust approach can be used.

In our research, we were focusing on the two following approaches: Frame difference and Mixture of Gaussian.

Frame difference is a mechanism of deciding whether a specific pixel belongs to the foreground or background, by subtracting the current image from the previous one as following:

$$|\text{frame}_i - \text{frame}_{i-1}| > Th$$

Where ( $Th$ ) is the threshold.

A serious problem of this approach is that it is very sensible to the threshold value ( $Th$ ) [4].

In the adaptive mixture of Gaussians approach defined in [5] [6], each pixel of the image is modelled independently by a mixture of  $K$  Gaussians, where  $K$  is the number of Gaussian distributions and in practical situation can take the following values 3,4, and 5 or other [5]. In the context of video surveillance, a value of 3 is sufficient [7].

In fact, the mixture of Gaussians approach models the temporal variations of each pixel present in the image. Each Gaussian of the  $K$  mixture of Gaussians is represented by a mean ( $\mu$ ) and a variance ( $\sigma^2$ ). Also, a weight ( $w$ ) is attributed to each  $K$  Gaussian distributions. The larger the weight is, the higher the associated probability is. The most likely distribution is determined by the narrowest variance and highest associated weight [6].

Calculating the square of the difference between the input pixel and the mean of the distribution and comparing it to 6.25 times the variance of the distribution, is the criteria used to decide whether a pixel belong to the background or foreground images [6].

To sum up, in order to handle the problem of multimodal backgrounds, adaptive mixture of Gaussian is proposed. This approach need to update the background models at every frame of the video sequence. An iterative update mechanism is used. A review of updating mechanisms showing their advantages and disadvantages can be found in reference [1].

#### 4. IMPLEMENTATION

The system hardware mainly contains:

- An Evaluation Module (EVM) that incorporates a Texas Instrument DM6437 DSP, some external hardware to support video input/output and an external memory.
- A PC for debugging purpose.
- A video camera / DVD player.
- A motorised tilt and pan system.
- A display.

Figure 6 shows the system hardware components.

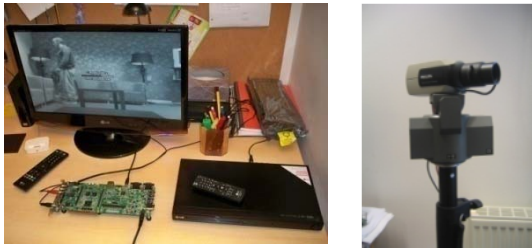


Figure 6 System hardware components (The left image shows the input (DVD player) / output (TV monitor) devices and the DSP platform, the right image shows the panning / tilting device with its tripod).

A considerable amount of time has been dedicated to the memory management so that image lines can be transferred fast enough from the external memory to the internal memory. To achieve this, a DMA was used with a double buffering.

The optimization process is divided mainly into three parts: The first part is related to the algorithm optimisation. The second part of the optimisation process is concerned with the memory management. The third part concerns the code optimisation. In our system implementation, we will be using linear assembly code for the critical section of the processing and then the handwriting assembly if necessary [8].

For the frames alignment and matching algorithm, the Sum of Absolute Differences (SAD) has been selected. The SAD

does not require any division compared to the normalized cross-correlation which requires a large number of divisions.

EDMA has been used for data transfer to reduce the CPU usage. A scheme of double buffering is employed to improve the memory management of the system.

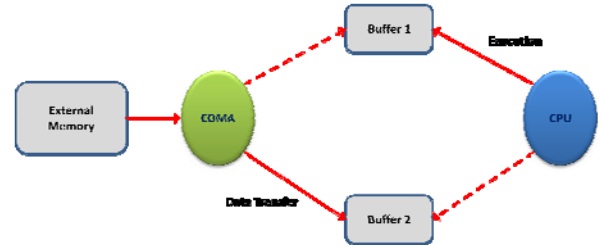


Figure 7 Double buffering.

The implementation of the system can be sub-divided into three main parts as shown in the following figure:

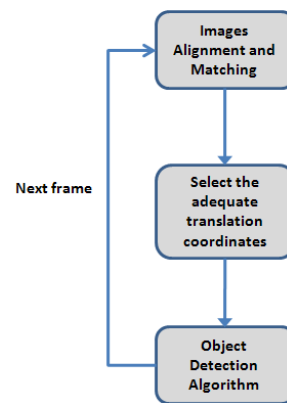


Figure 8 Main parts of the system implementation.

The processing is performed line by line, each time we load two lines, one for each buffer. When buffer 1 is being processed, buffer 2 is getting data from the external memory using the EDMA. The output of this stage is a set of displacement positions that the most adequate one will be elected by the following stage.

In the implementation side, the choice of the adequate translation coordinate is achieved by selecting the sub-frame's result that has the least sum of absolute differences value.

In the last stage, both frames are needed. For that reason, we specify a DMA channel to store the current frame into the external memory to be used in the next iteration as a previous frame. The process is done similarly to the one used with the current frame (Double buffer technique).

For the object detection algorithm, we opt for the frame difference approach due to its reasonable complexity comparing with the mixture of Gaussian.



## 5. EXPERIMENT AND RESULTS

The experiments were conducted for a video sequence where both camera and object are in motions. The system is designed to work in real-time, receiving the input frames from the DSP platform's input, processing them and then sending them to the output.

### 5.1 Simulation results

Results have been compared to the simulation results. Table 1 illustrates the results of our proposed approach using the three methods for the frames alignment and matching (a threshold value of 50 for the frame difference method has been used).

Method 2 is very optimal from the point of view of execution time and its subjective results are comparable to the Method 1 results. It improves the execution time by a factor of approximately 10 times.

Method 3 is giving a good time performance. However the quality of the detection is degraded.

### 5.1 Real-time implementation

The code was initially written to allow an input/output video using a double buffer. Using the benefits of such memory management, the algorithm has been coded taking in consideration that we can only load line by line. Double buffering is used to minimise the CPU load. However, since we need to save the current frame in the external memory, it is judicious to initiate the transfers within the frames alignment and matching stage, just after using the DMA to load the current frame line in the internal buffers. This is to avoid loading the buffers twice. This has been achieved in 7.19 ms (CPU Clock: 700 MHz). There after the frame is adjusted and the frame difference is performed. This is achieved in 19 frames / sec.

The system meets the real-time constraints and allows more algorithms to be integrated if needed.

## 6. CONCLUSION

In this paper, a novel approach that achieves real-time motion object detection for non-static (panning/ tilting) surveillance camera is proposed. This approach is mainly based on detection of similar patterns from one frame to another. the algorithm follows two stages; the first stage attempts to align and match two consecutive frames from the camera; the second stage takes the two matched frames and applies a static background subtraction algorithm.

Different methods have been simulated and compared. Method 2 has been implemented in real-time and it has been found that the frames alignment and matching consumes 7.19 ms (CPU Clock: 700 MHz) and the number of frames per second for the complete system is 19. The system is running in real-time and could be further improved by writing the code in linear assembly or handwritten assembly.

## REFERENCES

- [1]. Hannah M. Dee, Sergio A. Velastin: "How close are we to solving the problem of automated Visual surveillance?" *Machine Vision and Applications* (2008) 19:329– 343 DOI 10.1007/s00138-007-0077-z.
- [2]. Muhammad Asif, John Soraghan: "Video analytics for panning camera in dynamic surveillance environment", 50th International Symposium ELMAR-2008, 10-12 September 2008, Zadar, Croatia.
- [3]. Yiming Ye, John K.Tsotsos, Karen Bennet, and Eric Harley: "Tracking a Person with Pre-recorded Image Database and a Pan, Tilt, and Zoom Camera", *Lecture Notes in Computer Science*, 1997.
- [4]. Computation, <https://computation.llnl.gov/casc/sapphire/background/background.html>, accessed 15<sup>th</sup> Mars 2010.
- [5]. C. Stauffer and W. E. L. Grimson: Adaptive background mixture models for real-time tracking. In: *Computer Vision and Pattern Recognition Fort Collins, Colorado* (Jun.1999) pp. 246-252.
- [6]. Ierodiconou, S.P.; Dahnoun, N.; Xu, L.Q. "Implementation and Optimization of a Video Object Segmentation Algorithm on an Embedded DSP Platform" , *Crime and Security, 2006. The Institution of Engineering and Technology Conference on 13-14 June 2006* Page(s):432 – 437.
- [7]. B.Lei and L-Q. Xu, "Real-time outdoor video surveillance with robust foreground extraction and object tracking via multi-state transition management", *Pattern Recognition Letters* (in Press). (2006).
- [8]. Naim Dahnoun, *Digital Signal Processing Implementation Using the TMS320C6000 DSP Platform*, Addison-Wesley Longman Publishing Co, Inc, Boston, MA,200.

















Methods	Execution Time (Sec)	Frame 25	Frame 26	Frame 27	Frame 28
					
Method 1	200.95				
Method 2	22.11				
Method 3	13.73				

Table 1 Different methods for the frames alignment and matching (a threshold value of 50

## **Appendix B: CD Contents**

A CD is attached with this dissertation. It contains the following contents:

- Dissertation.pdf
- Simulation: All the Matlab simulation.
- DSP Code
- Test Video
- References: All the cited references.