Implementation of Motion Object Detection using BMNCKFCM on FPGA and ASIC

Siva Nagi Reddy Kalli¹, S Bhanu MurthyBhaskara²

¹Associate Professor, Sridevi Women’s Engineering College, ECE Department, JNTU HYDERABAD, Telangana, INDIA
²Professor, College of Computer and Information Sciences, Majmaah University, Kingdom of Saudi Arabia

ABSTRACT

Real time detection of moving objects was extremely crucial and fundamental step in the extraction of information concerning objects in motion and to stable the practical areas, such as tracking, classification, recognition, and so on. It is due to various factors like less competence of motion detection algorithm; increased hardware architecture, and poor memory access reduction schemes. To make efficient hardware architecture and increased hardware efficiency, of motion object segmentation, the proposed method used Background Modelling Neighborhood Coefficient Kernel based Fuzzy-C-Means (BMNCKFCM) algorithm was used in this paper. By the use of this algorithm keep a strong motion image to address variations on environmental changing provisions and utilized to eliminate the background interference information and separate the moving object from it. In fact, the consistency with which potential foreground objects in movement can be identified, directly impacts on the efficiency and performance level attained by subsequent processing steps of object recognition. The proposed algorithm has immense ability of anti-interference and maintains more accurate rate detection at the same time. The effectiveness of the presented algorithm for movement detection is displayed in a simulation environment and the assessment results of hardware like FPGA and ASIC are stated in this paper. In the view of Hardware realization of the motion object detection algorithms on FPGA and ASIC occupies more space, time and power consuming. To address all the issue of hardware architecture and observe effects from parameter settings in addition to fixed point quantization model can be performed with the help of FPGA and ASIC platform.

Keywords: Motion Object Segmentation, Morphology, Background Modelling, NCKFCM, FPGA, ASIC, Accuracy.

1. INTRODUCTION

Many algorithms have been proposed to detect moving objects under different situations in the past decades. In this paper, we propose a new method to detect moving objects in a non-stationary, complex background for an automatic traffic monitoring system. In Video surveillance schemes, automatic scene analysis is essential to identified person or object. Now a day these systems can automatically select the frames of concern in a video stream by using scene change Motion detection algorithms. The main dispute for designing these real time systems should be positioned in their performance constraints. The segmentation of moving objects recognition from video stream is an essential problem of tracking and traffic control etc applications. The recognition of the authentic outline of a moving object is appropriate, because it is affected by many complex factors like varying dynamic scene, light variation, presence of shadow, waving plants and rippling water, spouting fountain etc. In fact the objective of detecting a moving object is to realize the foreground of the moving target moreover in each video sequences or in the first video frame. The detection of object is to differentiate the foreground of the object from the stationary background and which is done through moving or fixed camera. The first method to represent the background statistically is to take for granted that over the past intensity values of a pixel can be modeled by a single Gaussian (SG) [1]. Nevertheless, the background being usually non-static, so this model cannot adequate to take out the background at wavering situation. To resolve this issue, the Mixture of Gaussians (MOG) [2] has been used to represent dynamic backgrounds. One of the drawback of this model is that it cannot predict precisely background with a less Gaussians (usually 3 to 5), causing troubles for sensitive detection. A new algorithm was developed called Kernel density estimation (KDE) [3] was developed to get precise estimation of background probabilities at each pixel, which was time consuming but a non-parametric technique. For resolving the problems of KDE, a new method introduced named as eigenbackground model, or SL-PCA in [4], which presents PDF
of the background, not the moving objects. This approach cannot produce proper results in various illumination changes in addition foreground object must be small. A method through Markov random field framework [5] was very successful for background estimation in enclosed surroundings. In accumulation the method merely predictable a static background model and was not used to remove foreground samples. In the similar proposal, a method established in [6] using Iterated Conditional Mode (ICM) technique. But the ICM method can’t able to provide the precise moving object. For motion object segmentation in video a new method was introduced in [7], which is the based on MOG produced fast video object segmentation. But it is ineffective to build accurate results. To full fill those kinds of problems, a hybrid model used, which is the combination of KDE-GMM [8]. This method deals with foreground detection and shadow removal concurrently by establish probability density functions (PDFs) of moving objects & non-moving objects. This model deals with highly dynamic backgrounds. In [9] recently developed a method for moving objects detection based Fuzzy system, which makes masks between the previous and the current foreground to enhance the efficiency detection. To get accurate results of moving object segmentation at each pixel, a new algorithm was build up based on hybrid technique [10], which is the combination of background modelling and Fuzzy-C-Means. One of the main issues to be pursued in background subtraction [11] is the uncertainty in the detection caused by the cited background maintenance issues. Usually, crisp settings are needed to define the method parameters, and this does not allow to properly dealing with uncertainty in the background model. Recently several authors have explored the adoption of fuzzy approaches to tackle different aspects of detecting moving objects. In [12] an approach using fuzzy Sugeno integral is proposed to fuse texture and color features for background subtraction, while in [13] the authors adopt the Choquet integral to aggregate the same features. In [14] a fuzzy approach to selective running average background modeling is proposed, and in [15] the authors model the background by the Type-2 Fuzzy Mixture of Gaussian Model proposed in [16]. In the above mentioned methodologies, most of the algorithms fail to be effective and fast but some algorithms have created effective foreground extraction results when conducted in a limited environment. Nevertheless, more robust and faster algorithms are required. As a preprocessing step, exact foreground removal generates good results in terms of detecting or tracking an object. So the main objective of motion detection algorithm is to create a balance between efficiency and difficulty.

The presented algorithm is called as Background Modelling with Neighborhood Coefficient Fuzzy-C-Means (BMNCKFCM). This method automatically adjusts to different atmosphere or gradual illumination changes. The algorithm in background Modelling generates the background segmented object using combination of frame difference and background subtraction. The Neighborhood Coefficient Fuzzy-C-Means (NCKFCM) employ the array of grayscales in the neighborhood and utilize this measure for limited related information and substitute the standard Euclidean distance with Gaussian radial basis kernel functions, which increases the defending of image details, independence of clustering parameters, and reduced computational costs, produces the efficient motion object segmentation. In addition, the scheme uses morphological operations are dilation (expands an image) and erosion (shrinks an image). Most of the existing methods realized in software only and a few methods implemented on hard ware in real time.

The rest of paper is arranged as follows: In Section 2 explains some prior related methods. FPGA Architecture of motion object segmentation algorithm is presented in section 3, and in Section 4 Accurate motion extraction method using background modelling Neighbourhood Coefficient kernel Fuzzy-C-Means and its hardware designs presented. Experimental considerations mentioned in Section 5. Conclusion is in Section 6.

2. RELATED WORK

In corresponding with the improvement of algorithm, researchers published several research articles illustrated different types of hardware design based towards to deal with the problem of real-time performance of motion detection algorithms. These hardware based motion detection methods mentioned in literature that, each of design is differ from other due to methodologies/approaches and design tool used. Based on design methodologies, different hardware based methods can be classified as general purpose processor (GPU) based approach, digital signal processor (DSP) based approach [17–19], complex programmable logic device (CPLD) based approach [20], application specific integrated circuit (ASIC) based approach [21,22], FPGA based hardware design approach [23–29], and FPGA based hardware/software co-design approach [30–32]. From design tool view, the differences can be based on the use of VHDL/Verilog, high level hardware description language like Handle-C or System-C, MATLAB-Simulink software, an Embedded Development kit, and a System Generator tool.
In recent years, different schemes using a pipeline image processor, [33] propose a design which can process the Yosemite sequence of 252×316 size in 47.8ms. FPGAs have been used to process larger images at faster speed. An algorithm proposed by Horn and Schunck [34] was implemented [35]. It is an iterative algorithm where the accuracy depends largely on the number of iterations. The classical Lucas and Kanade approach was also implemented [36] for its good tradeoff between accuracy and processing efficiency. Two step search block matching algorithm [37] was first implemented and then ported onto an Altera NiosII processor [38] where some hardware-based support was used to meet application timing requirements. A fast and accurate motion estimation algorithm [39] was modified for FPGA hardware implementation [40]. This design is able to process images of size 640×480 at 64 frames per second (fps).

In the view of Hardware realization of the image processing algorithms on FPGA and ASIC occupies more space and time and power consuming. For address all the issue of hardware architecture and observe long term effects from parameter settings in addition to fixed point quantization simulation can be performed with the help of FPGA platform. By using a Xilinx FPGA reconfigurable device the moving object architecture can be implemented. The utilization of FPGA in this design, synthesis and development time can be reduced. The same architecture also implemented and verified the parameters in the form of power, area and delay in ASIC with TMSE 180 nm technology.

3. FPGA ARCHITECTURE FOR MOVING OBJECT SEGMENTATION USING NCKFCM

In this section how the algorithm can be implemented in hardware utilizing a dedicated architecture in order to provide an efficient and generic framework for real-time object detection. The figure shows system structure of a moving object detection system.

![Figure 1 Structure of FPGA based moving object detection system](image)

A CMOS sensor module was connected to CCD capture unit to acquire the video. Then the video is converted in to frames using frame conversion, the frame shading plan was changed from RGB to YCbCr, the foundation was built up by its Y, the Background Subtraction with NCKFCM was applied to compare the foreground and background’s Y, and moving object discovery was at long last accomplished. Moving object Segmentation incorporated into the framework actualizes Background Modeling with NCKFCM and the general operation can be executed in hardware utilizing subs, adds, shifters and multipliers. The Moving object Segmentation calculation utilizes hardware features, for example parallelism and pipelining. The Morphology utilizes hardware features as parallelism and pipelining, in an effort to parallelize the recurring calculations involved in the Erosion and Dilation operations, and utilizations upgraded memory structures to decrease the memory reading redundancy.

3.1 Hardware Design for Background Modelling

The It includes 1) two stationary pixel extraction, such as stationary pixels via frame difference model (G_{fd} (u, v)) and stationary pixels via background subtraction method (G_{bg} (u, v)) to find the s By using these two methods to perform averaging operation and generate output G_{reg} (u, v). 2) Shift and add circuit is used to get the average generated stationary pixel 3) Back ground updated circuit is used to perform the ground truth 4) Absolute difference circuit provides initial motion field, which was shown in Figure 4.
To perform the update background process (G2) averaging pixel and some parameters like δ, γ, σ and φ are used in this method. Previous frame (Gref-1) also modernize the background through register bank. The absolute difference between the current frame (Ft) and updated background frame (G2) generates initial motion field.

i) Stationary Pixel calculation using Frame Difference:
The initial frame and reference background denoted as F(t) and Gref(i,j), respectively, which contains no foreground object. In this model the static pixels and non static pixel are isolated from the reference background frame and the frame difference. “Figure 3,” shows the stationary pixel from the frame difference method; this action can be done by using of threshold value comparison with frame difference. The set of stationery file is selected by using the difference between the current frame F(t) and previous frame F(t-1). The reference background frame Gref(i,j) follows:

\[ G_{t}^{fd}(i,j) = \begin{cases} 
G_{ref}(i,j), & \text{if } |F_t(i,j) - F_{t-1}(i,j)| < \tau_1 \\
G_{ref}(i,j) \times \text{sgn}(F_t(i,j) - F_{t-1}(i,j)), & \text{otherwise}
\end{cases} \]  

(1)

Where \( G_{t}^{fd} \) is a stationary pixel via frame difference model and \( \tau_1 \) cited as threshold value and the function Signum is defined as

\[ \text{sgn}(x) = \begin{cases} 
1, & \text{if } x > 0 \\
0, & \text{if } x = 0 \\
-1, & \text{if } x < 0
\end{cases} \]  

(2)

In equation (2) x is input value.
In frame difference hardware structure MUX helps to select the $G^{fd}_t$ from $G^{ref}_t$ of background frame with respect to the threshold value.

ii) Stationary Pixel calculation using Background Subtraction:

The current input frame $F_t(i,j)$ subtracts from reference background frame $G^{ref}_t(i,j)$ Used to investigate the stationary pixels using Equation (3).

$$G^{bg}_t(i,j) = \begin{cases} G^{ref}_t(i,j), & \text{if } |F_t(i,j) - G^{ref}_t(i,j)| < \tau_2 \\ F_t(i,j), & \text{otherwise} \end{cases}$$  

Here $G^{bg}_t(i,j)$ is the stationary pixel, which is measured by the background subtraction method and threshold $\tau_2$ respectively. The “Figure.4,” is the hardware design that provides the second stationary pixel by using multiplexer (MUX) and comparator.

![Figure 4 Stationary pixels through background subtraction method Hardware Design.](image)

ii) Averaging of stationary pixels:

The “Figure.7,” Provides average of foreground stationary pixel from the frame difference method $G^{reg}_t(i,j)$ and background stationary pixel from the background subtraction method $G^{bg}_t(i,j)$ design and it follows as:

$$G^{reg}_t(i,j) = \left(\frac{G^{fd}_t(i,j) + G^{bg}_t(i,j)}{2}\right)$$  

![Figure 5 Averaging stationary pixels Design](image)
iii) The updated current background frame:

The updated current frame can estimate by using Equation (5), as follows as

\[
G_t(i, j) = \begin{cases} 
\gamma \cdot G_{t-1}(i, j) + (1 - \gamma) \cdot (F_t(i, j) - G_{t-1}(i, j)), \\
\delta \cdot G_{t-1}(i, j) + (1 - \delta) \cdot (\sigma^2_d(i, j) - \sigma^2(i, j)), \\
\text{else if } |F_t(i, j) - G_{t-1}(i, j)| < \phi \cdot \sigma_d \\
\text{and } 1 < |F_t(i, j) - G^{\text{update}}_{t-1}(i, j)| < \tau_3 \\
G_{t-1}(i, j) + \text{sgn}(F_t(i, j) - G_{t-1}(i, j)), \\
\text{else } 0
\end{cases} 
\]  

(5)

Where \( \tau_3 \) is threshold value, which is user defined value. \( \delta, \gamma \) and \( \phi \) Values are ranging from 0.8 to 0.99, 0.999 for all videos, and 1 to 3 and the current updated background pixel hardware architecture shown in the Figure.8.

![Figure 6 Updated backgrounds Estimation Circuit](image)

iv) Initial motion field:

It is the absolute difference between the current background and the first frame mentioned in equation (6) and its structure shown in Figure 7.
3.2 Neighborhood Coefficient Kernel based FCM (NCKFCM)

The objective function from [41] as follows as

$$I_{C2} = \sum_{i=1}^{N} \sum_{j=1}^{c} u_{ij}^m \left| x_i - v_j \right|^2 + \alpha \sum_{i=1}^{N} \sum_{j=1}^{c} u_{ij}^m \left| \bar{x}_i - v_j \right|^2$$  \hspace{1cm} (7)

The linearly-weighted sum image $\bar{x}$ [42] is in advance form from the original image and its neighborhood neighbor average image in terms of

$$\bar{x}_i = \frac{1}{1+\alpha} \left( x_i + \frac{\alpha}{N_i} \sum_{j \in N_i} x_j \right)$$  \hspace{1cm} (8)

Where $x_i$ denotes the gray level value of $i^{th}$ pixel of the image $x_i$ and $N_i$ stands for the set of neighbors ($x_j$) falling into a window around $x_i$. The clustering method [42] is performed on the gray level histogram of the newly generated image $\bar{x}$. As a result, the objective is defined as

$$J_5 = \sum_{i=1}^{N} \sum_{j=1}^{c} u_{ij}^m \left( \bar{x}_i - v_j \right)^2$$  \hspace{1cm} (9)

A) Median and Average Filters:

i) Kernel Function:

The Euclidean distance metric is easy and economical; however it is sensitive to perturbations and outliers. Recently, with trendy practice of support vector machine, a latest way come outs to exercise kernel functions. A kernel function can be viewed as bringing a nonlinear mapping from inputs $X$ to feature vectors $\Phi(x)$. The kernel figures the inner product in the induced feature space:

$$K(x, v) = \Phi(x) \cdot \Phi(v)$$  \hspace{1cm} (10)
ii) Kernelized fuzzy c-means method

The kernel methods [43] are widely used for pattern recognition and approximation. Advantages of kernel methods are 1) Non-Euclidean distance measures; 2) Enhancing from noise and outliers, and 3) computational simplicity. The kernel functions are able to project the data into higher dimensional space where the data could be more easily separated [44], it is called as kernel trick, it has been permitted that can transform linear algorithm to nonlinear using dot product [45]. Using the kernel trick, the Euclidean distance term \( \| x_i - v_j \|^2 \) can be substituted with \( \| \phi(x_i) - \phi(v_j) \|^2 \)

that is defined as

\[
\| \phi(x_i) - \phi(v_j) \|^2 = K(x_i, x_i) + K(v_j, v_j) - 2K(x_i, v_j)
\]

Where \( K \) is kernel function, and from \( K(x_i, x_i) = K(v_j, v_j) = 1 \) the Equation (11) rewritten as

\[
\| \phi(x_i) - \phi(v_j) \|^2 = 2 \left( 1 - K(x_i, v_j) \right)
\]

By considering Gaussian Kernel from [45] depicted as

\[
K(x_i, v_j) = \exp \left( -\frac{\| x_i - v_j \|^2}{2\sigma^2} \right)
\]

Where “\( \sigma \)” is the kernel width, which will be similar to [46] follows as:

\[
\sigma = \sqrt{\frac{\sum_{k=1}^{N} (d_i - \bar{d})^2}{N-1}}
\]

Where \( d_i = \| x_i - \bar{x} \| \) is distance from the grayscale of pixel i to the grayscale average of all pixel and \( \bar{d} \) is the average of all distance \( d_i \).

The objective function is defined by using Gaussian Kernel with FCM follows as

\[
J_{KFCM} = 2 \left[ \sum_{i=1}^{C} \sum_{j=1}^{N} u_{ij}^2 \left( 1 - K(x_i, v_j) \right) + \sum_{j=1}^{C} \sum_{i=1}^{N} u_{ij}^2 \left( 1 - K(x_i, v_j) \right) \right]
\]

B) Weighted filter:

i) Neighborhood Coefficient (NC):

To bring down the noise level of the pixel being prepared with Neighborhood Coefficient (NC) in this work, this is to evaluate the disparity of grayscale in the neighborhood window to be normalized as for the neighborhood average grayscale. Within the sight of noise to have high heterogeneity between the focal pixel and its neighbors, NC will enhance, which is characterized utilizing [47] as.

\[
\delta = \frac{\sum_{k=1}^{N} (x_k - \bar{x}_i)^2}{N_R \bar{x}_i^2}
\]

Where \( x_k \) is the gray scale of any pixel falling in the neighborhood window \( N_i \) around the pixel i, \( N_R \) is the basic of \( N_i \), and \( \bar{x}_i \) is its mean grayscale. To maintain the weights within the neighborhood window, Exponential function is applied to NC follows as:
The ultimate weight assigned to every pixel is associated with the average grayscale of the neighborhood window:

\[
\zeta_i = \exp\left(\sum_{k \in N_i} \omega_k \delta_k\right)
\]

(17)

\[
\omega_i = \frac{\zeta_i}{\sum_{k \in N_i} \zeta_k}
\]

(18)

The advantage of \( \phi \) can greatly reduce the computational cost. The second point is that the contextual information provided by \( \phi \) is based on the collection of grayscale distribution within the neighborhood. As a result, the proposed tends to yield uniform clustering according to neighborhood grayscale distribution.

ii) Weighted factor:

For Weighted filter NCKFCM in addition to making \( \mathbf{f} \) respectively, the grayscale of average/median filter of the original image \( \mathbf{f} \) can also be replaced with the grayscale of the newly formed weighted image \( \mathbf{z} \).

\[
\mathbf{z}_i = \frac{1}{2 + \max(\mathbf{z})} \left( \mathbf{x}_i + \frac{1 + \max(\mathbf{z})}{N_R - 1} \sum_{\mathbf{z} \in N_i} \mathbf{x}_j \right)
\]

(20)

Where \( x_i \) and \( N_i \) are respectively, the grayscale and neighborhood of pixel \( i \) and \( N_R = N_i \).

\[
J_{\text{NCKFCM}} = 2 \left[ \sum_{i=1}^{N} u^w_{ij} \left( 1 - K(x_i, v_j) \right) + \sum_{j=1}^{N} \sum_{i=1}^{N} \omega_i u^w_{ij} \left( 1 - K(x_i, v_j) \right) \right]
\]

(21)

\[
L_m = 2 \left[ \sum_{i=1}^{N} \sum_{j=1}^{N} u^w_{ij} \left( 1 - K(x_i, v_j) \right) + \sum_{j=1}^{N} \sum_{i=1}^{N} \omega_i u^w_{ij} \left( 1 - K(x_i, v_j) \right) \right] + \sum_{j=1}^{N} i_j \left( 1 - \sum_{i=1}^{N} u_{ij} \right)
\]

(22)

By taking the derivative of \( L_m \) w.r.t \( u_{ij}, v_j, \mathbf{z}^w_{i} \mathbf{z}^w_{j} = 1 \) and setting the result to zero, for \( m > 1 \), then

\[
u_{ij} = \frac{\left( (1-K(x_i, v_j)) + \phi (1-K(x_i, v_j)) \right)^{-\frac{m-1}{m-2}}}{\sum_{i=1}^{N} \left( (1-K(x_i, v_j)) + \phi (1-K(x_i, v_j)) \right)^{-\frac{m-1}{m-2}}}
\]

(23)

\[
\nu_j = \frac{\sum_{i=1}^{N} u^w_{ij} \left( K(x_i, v_j) x_i + \phi K(x_i, v_j) r_i \right)}{\sum_{i=1}^{N} u^w_{ij} \left( K(x_i, v_j) + \phi K(x_i, v_j) \right)}
\]

(24)

When \( \mathbf{f} \) is replaced with the grayscale of the average/median filter of the original image, the algorithm is denoted as NCKFCM1/NCKFCM2. When \( \mathbf{z} \) is replaced with the weighted image \( \mathbf{z} \) defined in (20), the algorithm is denoted as NCKFCMw.

3.2.1 NCKFCM Architecture

The NCKFCM architecture in Figure.8 shows two units, one is the mean computation unit and other is the fuzzy clustering unit. The goal of the mean computation unit is to evaluate the mean value \( \mathbf{z} \) defined in Equation (20).
The main architecture of NCKFCM is the fuzzy clustering unit, which computes the membership coefficients and centroids of NCKFCM. To evaluate and simplify the fuzzy clustering unit let us consider the following equations as:

\[
J = 2 \left[ \sum_{i=1}^{N} \sum_{j=1}^{c} u_{ij}^m \left( 1 - K(x_i, v_j) \right) + \sum_{i=1}^{N} \sum_{j=1}^{c} \alpha u_{ij}^m \left( 1 - K(\bar{x}_i, v_j) \right) \right]
\]  
(25)

\[
\|x_k - v_i\|^2 = 2 \left( 1 - K(x_i, v_j) \right)
\]  
(26)

\[
\|\bar{x}_k - v_i\|^2 = 1 - K(\bar{x}_i, v_j)
\]  
(27)

\[
\alpha = \varphi, \bar{x} = \bar{x}_k m = \frac{a}{b}, r = b \& n = a - b,
\]  
(28)

Where both a, b are integers. Because m should be larger than 1, it follows that a > b > 0. Let

Using Equations (26), (27) and (28), we can rewrite the membership coefficients of NCKFCM defined in Equation (25) as

\[
\left( (\|x_k - v_i\|^2 + \alpha \|\bar{x}_k - v_i\|^2)^{1/n} P_k^{-1/(n+r)} \right)^{-(n+r)}
\]  
(29)

Where

\[
P_k = \sum_{j=1}^{c} \|x_k - v_j\|^2 + \alpha \|\bar{x}_k - v_j\|^2 - r/n
\]  
(30)

Figure 11 shows the architecture for the computation of each \((\|x_k - v_i\|^2 + \alpha \|\bar{x}_k - v_i\|^2)^{1/n}\) implementation with 4-stage pipeline. In that there are two squared distance calculation units and one adder at the first stage of the pipeline. Figures 12–14 depict the architecture for membership coefficients updating, centroids updating and cost function computation for NCKFCM based on (29), respectively. In the NCKFCM, the incremental centroid for i-th cluster up to data point x_k is defined as

\[
v_i(k) = \frac{\left( \sum_{n=1}^{k} u_{in}^m x_n + \alpha \bar{x}_n \right)}{(1 + \alpha \left( \sum_{n=1}^{k} u_{in}^m \right))}
\]  
(31)

In addition, the incremental cost function J (k) up to data point x_k is defined as

\[
J(k) = \sum_{i=1}^{c} \sum_{n=1}^{k} u_{in}^m \left( \|x_n - v_i\|^2 + \alpha \|\bar{x}_n - v_j\|^2 \right)
\]  
(32)

As shown in Figures 11 and 12, the goals of the centroids updating unit and the cost function computation unit are to compute vi (k) and J (k), respectively. As k = t, the vk (i) and J (k) in Equations (31) becomes vi.
Figure 9 \((||x_k - v_j||^2 + \alpha ||x_k - v_j||^\alpha)^{1/\alpha}\) Evaluating Circuit

Figure 10 $$u^{m_k}$$ Evaluating Circuit
Figure 11 \( v_1(k) \) Calculation Circuit for NCKFCM

Figure 12 Cost function \( J(k) \) calculation Circuit for NCKFCM
The membership coefficients, centroids and cost function for NCKFCM as the extension of those for original FCM by replacing \( \|x_k - v_j\|^2 \) with \( \|x_k - v_j\|^2 + \alpha \|x^k - v_j\|^2 \). Therefore, the membership coefficients updating unit, centroids updating unit and cost function computation unit for NCKFCM also have similar architectures to those of their counterparts in original FCM. The circuits in NCKFCM require only additional squared distance unit and adder for computing \( \|x_k - v_j\|^2 + \alpha \|x^k - v_j\|^2 \).

4. PERFORMANCE OBSERVATIONS

Typical performance evaluation takes the output of some visual surveillance algorithm and compares it with ground truth data as illustrated in this work. The performance of any surveillance algorithm will depend on the choice of the internal parameters. A test sequence with outdoor conditions of Traffic highway video has been considered in this work. The choice of such different scenes was prepared to highlight the consistency and robustness of the presented method in outdoor circumstances. A standard performance metrics were considered to be analyze the observed pixels about to the ground truth image depend on True-positive (TP) pixels, True-negative pixels (TN), False-positive pixels (FP), and False-negative pixels (FN). True-positive pixels (TP) were the correctly detected pixels by the algorithm of the moving object. The sensitivity standards of the suggested algorithm can find out by the following parameters. The relevant pixels (Recall) and irrelevant pixels (Precision) of the detected object can be initiated as follows as:

\[
\text{Recall} = \frac{TP}{(TP + FN)} \quad (33) \\
\text{Precision} = \frac{TP}{(TP + FP)} \quad (34) \\
\text{Similarity} = \frac{TP}{(TP + FP + FN)} \quad (35)
\]

\[
\text{False Measure} = 2 \times \text{Recall} \times \text{Precision} / (\text{Precision} + \text{Recall}) \quad (36)
\]

\[
\text{Accuracy} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad (37)
\]

True positive rate and True negative Rate:

\[
\text{TPR} = \frac{TP}{(TP + FN)} \quad (38) \\
\text{TNR} = \frac{TN}{(TN + FP)} \quad (39)
\]

False positive rate and False negative Rate:

\[
\text{FPR} = \frac{FP}{(FP + TN)} \quad (40) \\
\text{FNR} = \frac{FN}{(TP + FN)} \quad (41)
\]

The positive predictive value and negative predictive values:

\[
\text{PPV} = \frac{TP}{(TP + FP)} \quad (42) \\
\text{NPV} = \frac{TN}{(TN + FN)} \quad (43)
\]

<table>
<thead>
<tr>
<th>Video’s</th>
<th>Sample frame</th>
<th>Ground Truth</th>
<th>Motion Object Segmentation</th>
</tr>
</thead>
</table>

Figure 13 Motion mask generated by the proposed method and other baseline methods
The algorithm described algorithm BMNCKFCM has been experienced with outdoor video stream. The ground truth reference has been prepared for a video stream by physically extracting frame-by-frame of every pixel of each moving vehicle. Comparing all the existing with BM-NCKFCM, the developed approach output gives clear object information without noise. For all of the above images are numerically shown in Table 1.

![Figure 14](image-url)  
**Figure 14** Object Motion Segmentation and Mask generation of the Background Modeling with NCKFCM FCM method. (a) Median Filter Output (b) Average Filter Output (c) Weighted Filter Output

The Figure14 shows robustness of the projected method to hold with outdoor circumstances. The Figure 16 consists of background frame, previous frame and current frames are taken as 50th, 119th and 120th frame are reference frames in video. The segmented output and ground truth frame are taken as 150th frame from video delivered the final motion object mask.
Table 1: Summary of existing method with defending video, BMSE vs GMM
ICM, BMFCM, BMBIFCM, BMNCKFCM

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recall</td>
<td>0.9349</td>
<td>0.8763</td>
<td>0.9030</td>
<td>0.9293</td>
<td>0.9247</td>
<td>0.8577</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8749</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8866</td>
</tr>
<tr>
<td></td>
<td>Precision</td>
<td>0.4530</td>
<td>0.5132</td>
<td>0.4703</td>
<td>0.5610</td>
<td>0.5701</td>
<td>0.6157</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.6095</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5539</td>
</tr>
<tr>
<td></td>
<td>Similarity</td>
<td>0.4367</td>
<td>0.4780</td>
<td>0.4472</td>
<td>0.5380</td>
<td>0.5453</td>
<td>0.5569</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5585</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5569</td>
</tr>
<tr>
<td></td>
<td>F-Measure</td>
<td>0.6047</td>
<td>0.6465</td>
<td>0.6175</td>
<td>0.6990</td>
<td>0.7049</td>
<td>0.7149</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.7159</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.7125</td>
</tr>
<tr>
<td></td>
<td>Accuracy</td>
<td>0.9514</td>
<td>0.9731</td>
<td>0.9687</td>
<td>0.9776</td>
<td>0.9784</td>
<td>0.9804</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9807</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9797</td>
</tr>
<tr>
<td></td>
<td>TPR</td>
<td>0.9049</td>
<td>0.8763</td>
<td>0.9030</td>
<td>0.9293</td>
<td>0.9247</td>
<td>0.8577</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8749</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8866</td>
</tr>
<tr>
<td></td>
<td>TNR</td>
<td>0.9704</td>
<td>0.9760</td>
<td>0.9707</td>
<td>0.9790</td>
<td>0.9798</td>
<td>0.9834</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9840</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9821</td>
</tr>
<tr>
<td></td>
<td>FPR</td>
<td>0.0276</td>
<td>0.1237</td>
<td>0.0293</td>
<td>0.0210</td>
<td>0.0202</td>
<td>0.0166</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0160</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.0179</td>
</tr>
<tr>
<td></td>
<td>FNR</td>
<td>0.0981</td>
<td>0.0240</td>
<td>0.0970</td>
<td>0.0707</td>
<td>0.0753</td>
<td>0.1423</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1251</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1134</td>
</tr>
<tr>
<td></td>
<td>PPV</td>
<td>0.4620</td>
<td>0.5132</td>
<td>0.4703</td>
<td>0.5610</td>
<td>0.5701</td>
<td>0.6157</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.6095</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5991</td>
</tr>
<tr>
<td></td>
<td>NPV</td>
<td>0.9980</td>
<td>0.9962</td>
<td>0.9969</td>
<td>0.9978</td>
<td>0.9978</td>
<td>0.9964</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9961</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9969</td>
</tr>
</tbody>
</table>

The above results show the potential of the developed advance. It was necessary to estimate the efficiency of the method with a ground-truth. The purpose of the projected method was not only for simply detection improvement or discrimination of shadow pixels; it’s for efficient and precise object detection for further relevance. The efficiency of the numerical values of presented algorithm evaluates to the existing methods shows parameters Recall, Similarity and F-Measure in Table 1, for the given Traffic car video. From the Table 1, the proposed approaches are relatively superior to the existing methods.

Table 2: FPGA (Virtex4 - xc4vx12) Implantmentation Performance for proposed Architecture with previous methods

<table>
<thead>
<tr>
<th>Target FPGA</th>
<th>Circuit</th>
<th>LUT (10,944)</th>
<th>Flip flop (10,944)</th>
<th>Slice (5,472)</th>
<th>DSP- MULT (32)</th>
<th>BRAM/ DPRA M</th>
<th>Frequency (MHz)</th>
<th>HD Fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtex4 (xc4vx12)</td>
<td>BM-NCKFCM(w)</td>
<td>191</td>
<td>80</td>
<td>105</td>
<td>1</td>
<td>16</td>
<td>179.387</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BM-BIFCM [50]</td>
<td>190</td>
<td>79</td>
<td>104</td>
<td>1</td>
<td>16</td>
<td>214.891</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BM-FCM [49]</td>
<td>192</td>
<td>77</td>
<td>108</td>
<td>0</td>
<td>16</td>
<td>243.891</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BMSE [48]</td>
<td>1.850</td>
<td>38</td>
<td>1,009</td>
<td>9</td>
<td>0</td>
<td>111.101</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>ICM [6]</td>
<td>1865</td>
<td>76</td>
<td>960</td>
<td>27</td>
<td>0</td>
<td>10.238</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>GMM [24]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
The presented background modelling with FCM is synthesized and implemented on Virtex 6 (xc6vfx20), Virtex 5 (xc5vlx50), and Virtex 4 (xc4vlx75t), Xilinx (VIVADO) FPGA devices. By using ISE tool Fitting and Place Route have been carried out and to perform the circuit simulation Model-Sim has been used. In general for better performance, the cost of the FPGA should be less and it can be measured by using number of slices in the FPGA. Similarly, frequency also important parameters while designing the FPGA. The Table II, III, and IV provide the results of the proposed method implementation targeted to FPGA compared to other methods.

### Table 3: FPGA (Virtex5 - xc5vlx50) implementation performance for proposed Architecture with previous methods

<table>
<thead>
<tr>
<th>Target FPGA</th>
<th>Circuit</th>
<th>LUT (28800)</th>
<th>Flip flop (28800)</th>
<th>Slice (7200)</th>
<th>DSP-MULT (48)</th>
<th>BRAM</th>
<th>Frequency (MHz)</th>
<th>HD Fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtex5 (xc5vlx50)</td>
<td>BM-NCKFCM(w)</td>
<td>152</td>
<td>80</td>
<td>53</td>
<td>1</td>
<td>0</td>
<td>199.005</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BM-BIFCM [50]</td>
<td>153</td>
<td>68</td>
<td>43</td>
<td>0</td>
<td>0</td>
<td>259.175</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BM-FCM [49]</td>
<td>162</td>
<td>77</td>
<td>57</td>
<td>0</td>
<td>0</td>
<td>272.298</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BMSE [48]</td>
<td>317</td>
<td>148</td>
<td>132</td>
<td>9</td>
<td>2</td>
<td>308.166</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>ICM [6]</td>
<td>1240</td>
<td>72</td>
<td>355</td>
<td>35</td>
<td>0</td>
<td>9.279</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>GMM [24]</td>
<td>742</td>
<td>0</td>
<td>303</td>
<td>3</td>
<td>0</td>
<td>79.5</td>
<td>38</td>
</tr>
</tbody>
</table>

### Table 4: FPGA (Virtex6 - Xc6vlx75t) IMPLEMENTATION PERFORMANCE FOR PROPOSED ARCHITECTURE WITH PREVIOUS METHODS

<table>
<thead>
<tr>
<th>Target FPGA</th>
<th>Circuit</th>
<th>LUT (46,560)</th>
<th>Flip flop (93120)</th>
<th>Slice (11640)</th>
<th>DSP-MULT (288)</th>
<th>BRAM</th>
<th>Frequency (MHz)</th>
<th>HD fps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtex6 (xc6vlx75t)</td>
<td>BM-NCKFCM(w)</td>
<td>112</td>
<td>74</td>
<td>38</td>
<td>3</td>
<td>0</td>
<td>92.901</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BM-BIFCM [50]</td>
<td>107</td>
<td>65</td>
<td>34</td>
<td>3</td>
<td>0</td>
<td>125.492</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BM-FCM [49]</td>
<td>114</td>
<td>71</td>
<td>41</td>
<td>3</td>
<td>0</td>
<td>131.376</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>BMSE [48]</td>
<td>261</td>
<td>140</td>
<td>111</td>
<td>9</td>
<td>2</td>
<td>282.135</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>ICM [6]</td>
<td>1,860</td>
<td>74</td>
<td>540</td>
<td>35</td>
<td>0</td>
<td>101.965</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>GMM [24]</td>
<td>692</td>
<td>0</td>
<td>279</td>
<td>3</td>
<td>0</td>
<td>85.7</td>
<td>41</td>
</tr>
</tbody>
</table>
Table 5: ASIC Parameters Comparisons of Proposed Architecture with Existing State Art

<table>
<thead>
<tr>
<th>Method</th>
<th>Cell Area (um²)</th>
<th>Power (nw)</th>
<th>Delay (ps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM-NCKFCM (w)</td>
<td>5.9×10⁻⁵</td>
<td>1618.75</td>
<td>7088</td>
</tr>
<tr>
<td>BM-BIFCM [50]</td>
<td>5.9×10⁻⁵</td>
<td>1643</td>
<td>7237</td>
</tr>
<tr>
<td>BIFCM [49]</td>
<td>5.8×10⁻⁵</td>
<td>1391</td>
<td>6659</td>
</tr>
<tr>
<td>Block MSE [48]</td>
<td>11×10⁻⁵</td>
<td>1847.42</td>
<td>14261</td>
</tr>
<tr>
<td>ICM [6]</td>
<td>492×10⁻⁵</td>
<td>7439.53</td>
<td>102822.40</td>
</tr>
</tbody>
</table>

The design of moving object segmentation modeled in Verilog HDL, synthesized via TSMC 180-nm standard-cells library, placed, routed and chip-finished. This process accomplished by using Cadence Encounter RTL Compiler. The designs have been simulated with NCSim and the Toggle Count File (.tcl) has been generated in order to obtain an exact view of the power dissipation. The Table v will provide cadence synthesis outcomes of design with the cell area/chip area.

5. Conclusion

In this work the projected method reduced the noise and developed accurate segmentation by reduction of false pixel using hybrid algorithm over the other methods. After realization of the Background Modelling with different NCKFCM algorithms in FPGA achieved real-time capability with 30 fps with frame size of 640 × 480 in live video and also architecture used almost equal logic resources over BMFCM and BIFFCM. In addition hardware design implemented in ASIC performed almost equal logic resources over BMFCM and BIFFCM parameters in terms of area, power and delay. By the observation of the projected method is more efficient in terms of F-measure, Recall and precision values. So that it reduces false negative pixels with neighborhood pixel and the Computation delay also reduced. But this method does not offer automated selection of the clusters centroid, such that it cannot able to provide much hardware efficiency.

References


[14] Baf, F.E., Bouwmans, T., Vachon, B.: Type-2 Fuzzy Mixture of Gaussians Model:


