IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 22, 2020, accepted October 2, 2020, date of publication October 12, 2020, date of current version October 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3030108

Fast-D: When Non-Smoothing Color Feature Meets Moving Object Detection in Real-Time

MD. ALAMGIR HOSSAIN, MD. IMTIAZ HOSSAIN, MD. DELOWAR HOSSAIN, NGO THIEN THU, AND EUI-NAM HUH, (Member, IEEE)
Department of Computer Science and Engineering, Kyung Hee University, Yongin 17104, South Korea

Corresponding author: Eui-Nam Huh (johnhuh@khu.ac.kr)

This work was supported in part by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. 2017-0-00294, Service Mobility Support Distributed Cloud Technology), and in part by the Institute for Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIT) (No. 2019-0-01615, Developed Digital Signage Solution for Cloud-Based Unmanned Shop Management that Provides Online Video Advertising).

ABSTRACT The moving object detection refers to the detection of physical moving objects from a video, which is applied in video surveillance, object recognition, object counting, human-computer interaction, and so on. Moreover, nowadays, real-time moving object detection is used as services in the cloud, edge, and fog computing. However, the existing methods do not meet the trade-off between accuracy and complexity. To address these issues, we present a background subtraction-based moving object detection method, called Fast-D. In this paper, we look at the 'non-smoothing color feature' to make the moving object detection more robust in real-time. Each color feature is given equal significance during the classification of a pixel. Background model and threshold are initialized for each pixel. And then, the background model and threshold are updated dynamically when there are changes in the background of the video. Adaptive post-processing is considered to discard salt and pepper noise and fill holes in the detected moving object silhouettes. The evaluation of our proposed method on four complex datasets exhibits the significance.

INDEX TERMS Real-time moving object detection, background subtraction, object segmentation, change detection.

I. INTRODUCTION

Our proposed approach detects moving objects based on background subtraction. The background subtraction (BGS) is a method that segments moving foreground (FG) objects and reconstructs the background (BG) from a video captured by a fixed or moving camera. The moving FG object detection has become very important research interest in the field of computer vision and image processing for many years because of its wide area of applications: object recognition, object tracking, activity recognition, video surveillance, airport and maritime monitoring, human-computer interaction, and so on [1]–[3]. Moreover, real-time moving object detection applications are used as services in cloud computing, IoT, fog computing, edge computing, smart cities, smart environment, smart home, robotics, drones, and so on [4]–[6].

Many moving object detection approaches have been proposed until now. We divide moving object detection approaches into two types: the traditional image processing-based method and the deep learning-based method. The traditional image processing-based methods cannot segment a video more accurately. On the other hand, according to Liu *et al.* [7], deep learning-based methods increase object detection accuracy. In practice, deep features extract pixel level to semantic level features which increase the expressive power of deep models [8]. However, Liu *et al.* [7] and Zhao *et al.* [8] analyzed that the success of deep learning-based object detection methods heavily depended on large scale training data, complex networks, and high computational cost during training as well as inference. In other words, extraction of deep features using a deeper architecture increases computational complexity in terms of running time, memory, and storage usage [7], [8]. Therefore, we conclude that the existing approaches are either computationally complex or less accurate in BG/FG segmentation. Therefore, it needs a trade-off between segmentation accuracy and algorithmic complexity. Recently, Bouwmans *et al.* [2] observed the demand of moving object detection in real-time

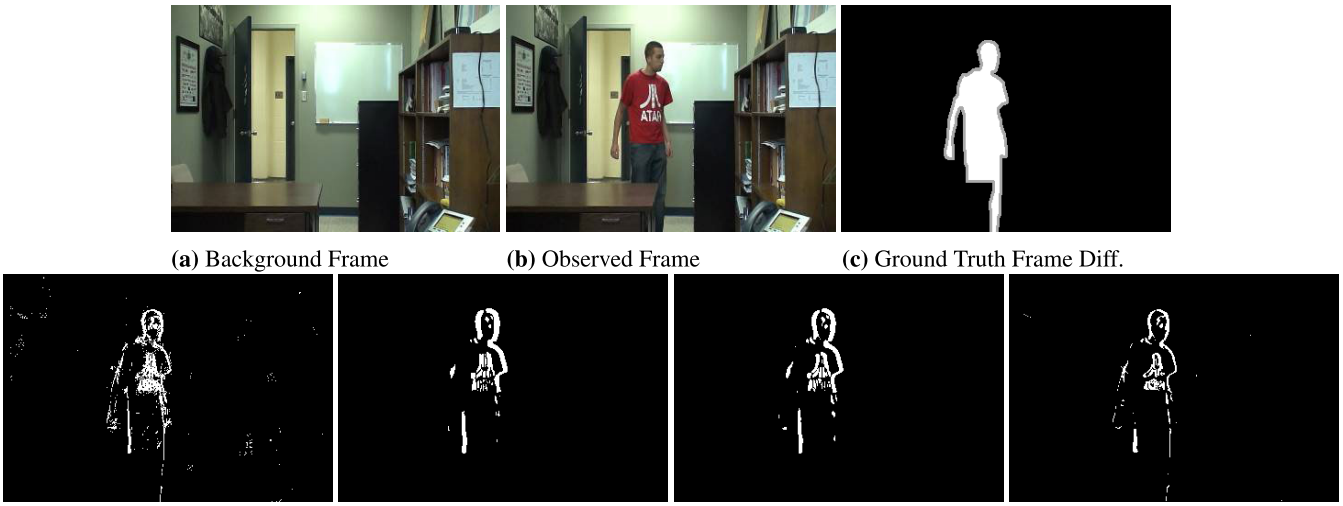
The associate editor coordinating the review of this manuscript and approving it for publication was Jiju Poovvancheri.

applications. Besides, Liu *et al.* [7] observes that object detection algorithm needs to achieve two competitive goals such as high accuracy and high computational efficiency. Consequently, we propose a moving object detection method, called Fast Detection (Fast-D), to more accurately segment BG/FG in real-time. Moreover, limiting computing devices and low main storage devices can use our proposed solution for effective and efficient moving object detection.

In the beginning, frame difference [9], [10] and some statistical-based BGS methods [11]–[13] were proposed for BG/FG segmentation. For the first, Wang and Suter [14] came up with a novel sample consensus strategy, called SACON, used only color features for BGS. Later on, SACON's sample consensus policy was used by methods [15], [16] to improve classification accuracy. Recently, Chen *et al.* [17] proposed sub-superpixel-based background subtraction (SBS) to increase detection accuracy. Additionally, in superpixel-level background estimation (SuperBE) [18], superpixel-based segmentation was proposed to reduce complexity. However, to reduce the complexity, SuperBE considerably sacrificed segmentation accuracy. Though the algorithmic complexity of the above BGS methods [9]–[17] was low, the segmentation accuracy was very poor. Afterward, Hofmann *et al.* [19] proposed a pixel-based adaptive segmenter (PBAS), newly created the dynamic background modeling policy, and used a sample consensus policy and gradient magnitude feature in addition to color feature. PBAS observed the little improvement of segmentation accuracy due to the use of gradient magnitude. Also, St. Charles *et al.* [20] used local binary similarity patterns (LBSP), and color features consensus to increase the classification accuracy. Then, St. Charles *et al.* [21] adopted the dynamic background modeling policy of [19] to extended their works [20]. The local binary similarity pattern (LBSP) was a combination of intra-LBSP and inter-LBSP used in [20], [21], which increased accuracy as well as complexity. The segmentation accuracy and complexity of the methods proposed by Hofmann *et al.* [19] and and St. Charles *et al.* [20], [21] was high compared with methods [9]–[17]. However, we see that none paid attention to find the trade-off between accuracy and complexity of the segmentation. Therefore, we propose a method to consider this trade-off.

Traditionally, smoothing functions are used in background subtraction to reduce pixels' noises due to camera sensors, light fluctuation during video capture, and so on. However, the smoothing functions also attenuate actual motion information of moving objects in addition to noise compensation. Fig. 1 depicts smoothing effects versus non-smoothing effects. To observe the differences between these effects, we obtained a non-smoothing frame difference and the smoothing frame differences. To experiment with the frame differences, we took a background frame #1, an observed frame #650, and a ground truth frame #650 from the office video file in baseline category. The frames are respectively shown in the upper row in Figs. 1a, 1b and 1c. Note that frame difference threshold was fixed to 17 and the window size of the average filter, Gaussian filter, and median filter was chosen 5×5, where the standard deviation of the Gaussian filter was taken 1.5. The non-smoothing frame difference and frame differences due to the average filter, Gaussian filter, and median filter are respectively shown in Figs. 1d, 1e, 1f and 1g. We see that the FG frame due to non-smoothing frames in Fig.1d preserved more accurate edges of the object silhouette, but incurred more salt and pepper noise than other FG frames in Figs. 1e, 1f and 1g. The salt and pepper noise is subsequently compensated by a post-processing operation. Also, we observe that average filter FG and Gaussian filter FG frames (Figs.1e and 1f) removed more salt and pepper noise than others. However, the FG frames in Figs. 1e and 1f attenuated edges (a portion of right-hand edge) and added more noises in the edges (thick edges in the mouth and left shoulder) of the object silhouette than the non-smoothing FG and median filter FG in Figs. 1d and 1g. Moreover, we numerically measured the effects explained in Section IV-D to more specifically analyze the differences. In that experiment, non-smoothing gained more accuracy than the smoothing functions. Therefore, we use non-smoothing input that keeps sharp edges.

In this paper, we propose a per-pixel sample consensus-based segmentation method. As we previously mentioned, we use non-smoothing color features instead of smoothing features of a pixel in order to keep all changes information. To create BG reference, a certain number of BG samples are temporarily stored for each pixel separately from the first video frame. The BG samples are neighbors of a pixel, which are updated in order to adapt to changes in the videos. The changes may occur due to dynamic background scenes, shadows, pan-tilt-zoom, and so on. Dynamic background scenes are very complex background scenes containing much intensity variation due to repetitive movements (shimmering water, fountain, tree shaken etc), varying illumination, shadow, and so on. After BG samples initialization from the first frame, BG/FG segmentation starts from the second frame. The intensity distance of each pixel's with its reference BG sample is measured in order to find similarity/dissimilarity. The intensity distance determines BG and raw FG pixels based on a certain threshold. We initially take the threshold value empirically. The threshold is further modified based on changes in a video like the BG samples to get more robust segmentation. The raw FG frame may contain more noises as we do not smooth input frames. Also, regions of object silhouette in the FG frame may not be detected more accurately. Therefore, the noises and unfilled regions of the object's silhouette are corrected by a post-processing operation. The key contributions of our work are as follows: (1) We propose a non-smoothing color feature-based moving object detection approach to effectively detect moving objects in complex videos. (2) Per-pixel thresholds are adapted dynamically to attain robust segmentation. (3) Three-color $\{R, G, B\}$ features are given equal significance during segmentation to exploit equal contribution of each feature. (4) Adaptive



(a) Background Frame (b) Observed Frame (c) Ground Truth Frame Diff.

(d) No-smoothing Frame Diff. (e) Average Filter Frame Diff. (f) Gaussian Filter Frame Diff. (g) Median Filter Frame Diff.

FIGURE 1. (a) Background frame, (b) observed frame, and (c) corresponding ground truth frame difference are shown left to right in the upper row. The background frame and the observed frame are smoothed by an average filter, a Gaussian filter, and a median filter separately. Then, the respective frame differences of the filtered frames are obtained. In the lower row, (d) frame difference of no-smoothing frames, (e) frame difference of average filter frames, (f) frame difference of Gaussian filter frames, and (g) frame difference of median filter frames are depicted from left to right.

post-processing operation is employed in order to compensate for raw segmentation noises. (5) We develop a moving object detection method that can be applied in limited computing devices.

In the remainder of this paper, in Section II, the most important aspects and drawbacks of related works are discussed. Section III describes the detailed description of our proposed method. Section IV presents the hyperparameters settings, the effects of a number of operations used in our model, evaluation scores of our method, and accuracy and the complexity comparison of our method with the existing approaches. Section V summaries the important aspects of the experimental results. In Section VI, we draw a conclusion and list the applications and future research directions of our proposal.

II. PRIOR WORKS

Many background subtraction (BGS) approaches have been proposed for moving object detection until now. BGS-based moving object detection approaches are divided into basic, sample consensus-based, statistical/cluster-based, and learning-based, which are shown in Fig. 2.

1) *Basic Approaches*. The frame differences- [9], block differences- [10], median- [22], and histogram-based [23] BGS are simple methods that are less complex, but less accurate compared to others. Also, this type of methods falsely detect moving objects when there are pan-tilt-zoom effects and dynamic background scenes containing tree leaves swinging, shadow, and illumination variations in a video.
2) *Sample Consensus-based BGS*. The sample consensus-based approaches [14]–[16], [19], [21], [24] stores features (color/gray intensity, gradient magnitude, LBSP, and the like) for each pixel from its neighbors, and each pixel is separately segmented as BG/FG pixel based on the pixel's features similarity/dissimilarity with its reference BG pixel. In 2007, for the first, a sample consensus-based BGS method called sample consensus (SACON) [14] was proposed where it kept color and motion features as background samples. However, SACON cannot detect occluded objects more accurately. Later on, like SACON, in 2011, Barnich *et al.* [15] also proposed a sample consensus-based approach called ViBe that only used color feature. Indeed, instead of a selective background sample update in SACON, ViBe took a random background update policy. Nevertheless, ViBe fails in pan-tilt-zooming videos and less accurately detect moving objects when there are dynamic background scenes and intermittent object motion in a video. Afterwards, Van *et al.* [16], [24] improved ViBe and called ViBe$^+$ and disruptive ViBe in 2012 and 2014, respectively. On the other hand, for the first, Hofmann *et al.* [19] proposed a dynamic background modeling technique in PBAS. The new technique is used to update the threshold and the background dynamically. However, the individual color dissimilarities or individual gradient magnitude distance were not utilized for this segmentation, and the fusion of color and gradient magnitude was static. Also, PBAS changes threshold a constant amount whether there are small or large dynamic scenes in background. Indeed, it needs to update the magnitude of the threshold according to the amount of background motion. Also, the method use a median filter for post-processing which is not enough for effective post-processing when much noise occurs. Therefore, the method cannot detect moving objects

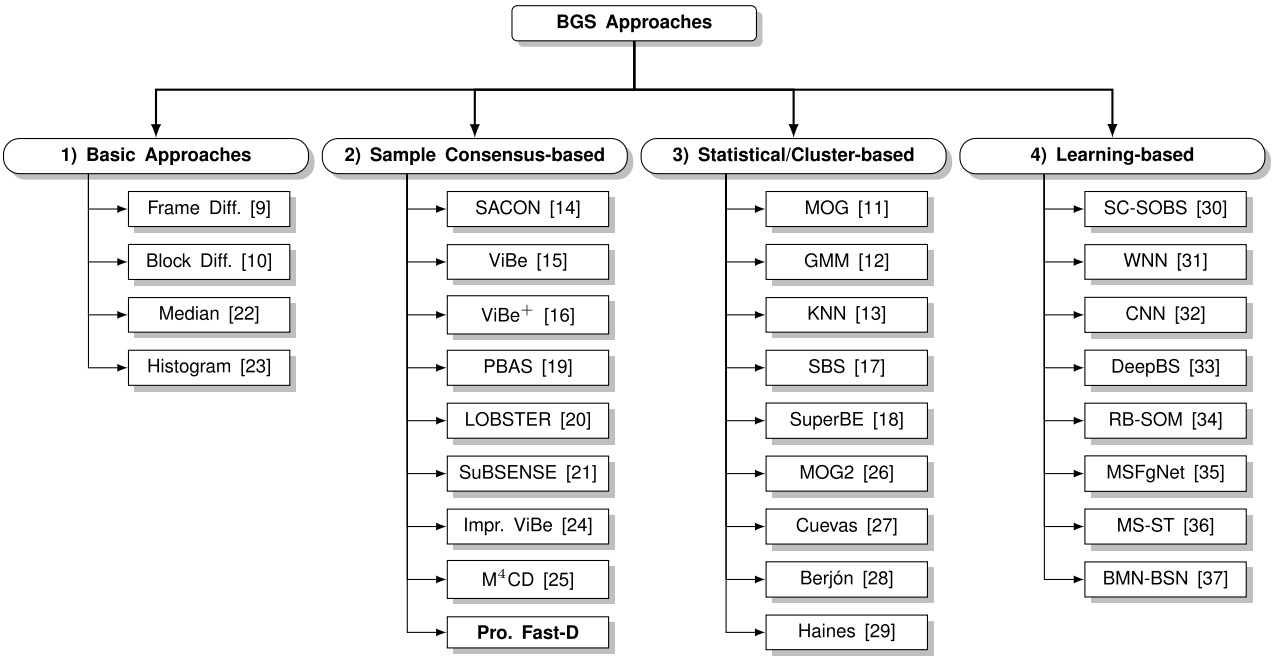


FIGURE 2. A taxonomy of background subtraction-based moving object detection approach.

more accurately because of the static fusion, ineffective weighted fusion of color and gradient magnitude, constant threshold increment, and very simple post-processing. Afterwards, St-Charles *et al.* came along local binary similarity segmenter (LOBSTER) [20] and self-balanced sensitivity segmenter (SuBSENSE) [21] which used color/gray intensity and local binary similarity pattern (LBSP) to describe a pixel. The LBSP, 32-bit patterns, was a combination of intra-LBSP and inter-LBSP. St-Charles *et al.* [20], [21] improved the dynamic background modeling mechanism proposed in PBAS which resulted in more accurate detection. However, color, LBSP, and dynamic background modeling mechanism incurred much computational cost. M^4CD [25] integrated chromaticity, brightness and texture features (local ternary pattern (LTP)), estimated FG based on Bayes rule, and used a Markov random field to make an optimized post-processing operation. Nevertheless, the use of multiple features and the complex post-processing operation incurred additional computational cost in addition to accuracy improvement. To simultaneously reduce complexity and increase accuracy of moving object detection in a video with dynamic background scenes, we propose a sample consensus-based method, called Fast-D.

3) *Statistical/Cluster-based BGS*. Statistical- and cluster-based BGS approaches were proposed to classify BG/FG in a video with non-dynamic background scenes. Butler *et al.* [38] proposed K-means clustered-based BGS where a pixel could reside in only one cluster at a time. Alternatively, Gaussian distribution-based method called Gaussian Mixture Model (GMM) [12] that took a pixel in more than one distribution at a time. Afterward, MOG [11] corrected the learning rate alteration of each mixture of Gaussian distribution by combining BGS approaches [12], [39], [40]. Also, MoG2 [26] combined [13] and [41] to further improve MoG. However, in reality, the standard deviation of Gaussian distribution can correct mild intensity variations. Therefore, GMM, MoG, and MoG2 cannot distinguish dynamic and non-dynamic background accurately. Also, GMM results in false segmentation when an object remains a long time in a location and then starts moving. Zivkovic *et al.* [13] considered dynamic kernel instead of a fixed kernel, called KNN. KNN methods use a global threshold which results in false segmentation because the pixel intensity changes are not identical in all the regions. Alternatively, pixel intensity hugely varies in case of changing backgrounds (tree leaves swinging, shadow, unstable video, and so on). Therefore, to model a non-stationary background, the codebook-based method [42] was proposed which captured the structural variation of a background. However, it needs long training videos to model the background and gives false classification in case of dynamic background scenes in a video like KNN and Gaussian distribution-based methods. Cuevas *et al.* [27] used an adaptive kernel bandwidth estimation technique to model BG and proposed a selective BG update policy to improve the accuracy of FG detection. Each BG sample was assigned a weight according to the selective BG update. However, the weight was non adaptively selected. Berjón *et al.* [28] improved [27] by adding a

particle filter to update the foreground pixel at the new frame sequences. Haines *et al.* [29] presented a Dirichlet process GMM for background modeling. Moreover, a Bayesian method was used to avoid overfitting and underfitting problem, and a continuous BG update technique was developed to adapt to BG scene changes. According to Haines *et al.* [29], DP-GMM cannot control sudden lighting changes which result in more false detection. In recent times, SBS [17] and SuperBE [18] reduced computational complexity by sacrificing much segmentation accuracy.

4) *Learning-based BGS*. Spatially Coherent Self-Organizing Background Subtraction (SC-SOBS) [30] and Weightless Neural Network (WNN) [31] methods were proposed to segment more accurately in a video with gradual/sudden lightning changes. In 2016, Braham *et al.* [32] came across a convolutional neural network (CNN)-based scene-specific BG/FG segmentation. The background generated by the frame difference method was used to train the CNN model. Later on, in 2018, Babaee *et al.* [33] only replaced the background creation portion of Braham's model and called DeepBS. In DeepBS, the background was created by the combination of SuBSENSE [21] and Flux Tensor [43] instead of by frame difference method. However, deep learning-based methods [32], [33] depend on the traditional computer vision and image processing algorithm because there is no available ground truth data set to effectively train the deep learning models. Moreover, we observed that the learning-based method incurred much computational cost to slightly increase accuracy. Ramirez-Quintana *et al.* [34] used a radial basis function to learn background pixels and took a retinotopic self-organizing neural network (RESOM) to detect gradual illumination changes, called RM-SOM. Although RB-SOM improved the background reconstruction, it failed to achieve the best foreground detection accuracy. Patil *et al.* [35] presented an end-to-end CNN architecture and motion saliency foreground network (MSFgNet) to segment BG/FG. In this method, a long video stream was partitioned into small video streams. Subsequently, MSFgNet reconstructed the background frame for each small video stream. However, Patil *et al.* [35] did not present how to segment a video in real-time. In MS-ST [36], a spatial deep model that extracted spatial deep features and a ConvLSTM model that obtained multi-scale temporal features. According to Yang *et al.* [36], MS-ST cannot segment low frame rate videos, night videos, and PTZ videos more accurately as non-challenging videos. Mondéjar-Guerra *et al.* [37] proposed two nested deep models, called BMN-BSN: the first deep model extracted background feature and the second one carried out subtraction operation between consecutive frames to model the background frame. Nevertheless, Mondéjar-Guerra *et al.* did not show how to segment a video with camera viewpoint variations.

III. COLOR CONSENSUS MODEL

We propose a background subtraction-based moving object detection method. In our method, we use some hyperparameters and variables described in Table 1. A few hyperparameters in our method are the same as ViBe [15], ViBe$^+$ [16], PBAS [19], LOBSTER [20], and SuBSENSE [21] and the rest of the hyperparameters we determine as those methods [15], [16], [19]–[21] obtained, which are explained in Section IV-C. Fig. 3 shows the moving object detection model. The model consists of three main blocks:

A) *Input*. Video/frame sequences are taken as input in our method. We use the red green blue (RGB) color space. Alternatively, our method can also detect moving objects in a grayscale video. Note that we do not smooth the video frame sequence as smoothing functions loses the essential change information, explained in Sections I and IV-D (See Section III-A).

B) *Mid-level Processing*. It is the main processing block. The more detail description is discussed in Section III-B. Moreover, we summarize the main processing tasks here. Non-smoothing frame sequences are the inputs and raw FG moving objects are the outputs in the mid-level processing. This processing block includes the following three sub-processes blocks:

   1) *BG Samples*. BG samples are neighbors of a candidate pixel. Before segmentation start, a certain number of BG samples of each pixel from the first video frame are temporarily stored, called BG samples initialization. Note that the initiated BG samples are replaced over the BG/FG segmentation based on dynamic or non-dynamic BG scenes. During segmentation, an observed pixel's proximity is compared with these BG samples (See Section III-B1).
   2) *Threshold*. A threshold is a certain value used to decide BG/FG pixels. Note that our method uses a per-pixel threshold for per-pixel segmentation. Initially, a value is assigned to each threshold. Then, like the BG samples update, the thresholds are also updated dynamically based on static/dynamic scenes in the background. According to our analysis, the threshold is a very important parameter that has a major impact on segmentation accuracy (See Section III-B2).
   3) *Segmentation Decision*. We propose a per-pixel BG/FG segmentation method. The classification starts with the second sequence of video frames. The BG/FG is performed based on the color/gray intensity difference between an observed pixel and its reference BG samples. If the intensity difference is less than a certain threshold, the observed pixel will be a BG pixel; otherwise, it will an FG pixel (See Section III-B3).

TABLE 1. Description of hyper-parameters, variables, and its notations.

Variable/ notation	Value	Description/Definition
x	-	The row number of a pixel
y	-	The column number of a pixel
C	$\{R, G, B\}$	The red, green, and blue channels
N	49	The number of BG samples
$I^c(x, y)$	-	The color/gray pixel, $c \in \{R, G, B\}$
$B_i^c(x, y)$	-	The background samples, $i \in [0, N-1]$, $c \in \{R, G, B\}$
$\|P - Q\|$	-	The Manhattan distance distance between P and Q; let the intensity of an observed pixel $= P$, and its corresponding BG samples $= Q$
$\min \|P - Q\|$	-	The minimum Manhattan distance in $\|P-Q\|$
$T(x, y)$	-	The threshold
S_r	-	The raw segmented frame
$S_r(x, y)$	-	The segmented pixel of S_r
$D(P, Q)$	-	The normalized minimum distance between P and Q
$D_{min}(P, Q)$	$[0, 1]$	The normalized minimum moving average
M_{int}	255	The maximum intensity of a pixel
$M_i(x, y)$	-	The number of intensity similarity of an observed pixel with BG samples, $i \in [0, N-1]$
$H(x, y)$	1	The threshold factor; initially, $H(x, y) = 1$
$U(x, y)$	$[2, 256]$	The background factor; initially, $U(x, y) = 2$
$P(x, y)$	$1/U(x, y)$	The probability of BG samples update
$Tmin$	15	The initial threshold value
M_{min}	2	The minimal number of required matches
γ	50	The moving average factor
H_s	0.01	The step size to update threshold factor
R_m	0.1	The minimum distance of background
U_{dr}	0.25	The step size of BG factor increment
U_{in}	0.50	The step size of BG factor decrement

C) *Output*. This output process block takes the raw segmentation frame from mid-level processing block as an input. Then, the raw segmentation frame is post-processed to fill holes and discard noises. Finally, we get filtered FG moving objects of interest (See Section III-C).

A. INPUT

Our model takes RGB color video frames as inputs. According to Sajid *et al.* [44], red green blue (RGB), hue saturation intensity (HSI), hue saturation value (HSV), and luma blue-difference red-difference chroma components (YCbCr) color models are mainly used for BG/FG segmentation. The color spaces such as RGB, YCbCr, HSV, and HSI affect the BG/FG segmentation accuracy [44]–[46]. Among the color models, although YCbCr is considered the best, the chance of miss classification of this color model increases when scenes contain dark pixels [44]. However, the RGB color model can be chosen for the reasons: (1) For good or bad lighting conditions, the RGB color space is the best [44]. (2) The color and brightness information are uniformly distributed in those $\{R, G, B\}$ channels [44]. (3) According to [45], this color model is robust against noises due to camera and environment. (4) The video format of the most camera is RGB, consequently, there is no need for extra color conversion cost [46].

Moreover, because of the above reasons, we choose the RGB color space. The channels, $C = \{R, G, B\}$, are used as inputs independently. On the other hand, our Fast-D also classifies a gray-scale video. We do not take any smoothing functions such as Gaussian filter, average filter, and median filter, because these smoothing functions not only reduce noises but also lose actual change information as explained in more detail in Sections I and IV-D.

B. MID-LEVEL PROCESSING

Mid-level processing begins with RGB color/grayscale video frames. Our approach initializes the BG samples from the first video frame. Per-pixel BG/FG segmentation starts with second frames. The similarity/dissimilarity intensity distance of an observed pixel, $I_o^c(x, y)$, is measured with its corresponding BG samples, $B_i^c(x, y)$, $i \in [0, N-1]$, $c \in \{R, G, B\}$. And, $I_o^c(x, y)$ is classified as BG/FG based on this similarity/dissimilarity distance. Based on the amount of background motion, BG samples are replaced by a segmented BG pixel, and thresholds are updated as well. After finishing the segmentation of all the pixels in a frame, we get a raw segmentation frame, $S_r(x, y)$. $S_r(x, y)$ is further processed according to the output block. The mid-level processing is described in more detail as follows.

1) BG SAMPLES

Before segmentation, BG samples, $B^c(x, y)$, are initialized. BG samples are $n \times n$ neighbors of a pixel at coordinate (x, y). In our method, a $7 \times 7 = 49 = N$ window is chosen empirically. Therefore, BG samples, $B^c(x, y)$, are initialized from the first video frame as

$$B^c(x, y) = \left\{B_0^c(x, y), B_1^c(x, y) \ldots B_{N-1}^c(x, y)\right\}, \tag{1}$$

where each $B_i^c(x, y)$, $i \in [0, N-1]$, $c \in \{R, G, B\}$, is called BG pixel of $B^c(x, y)$.

2) THRESHOLD

Segmentation frame, $S_r(x, y)$, is a binary image. In this binary image, white color and black color refer to FG and BG, respectively. These FG and BG pixels of an input video frame, $I^c(x, y)$, are determined if intensity distance between an observed pixel, $I_o^c(x, y)$, and its corresponding BG samples, $B_i^c(x, y)$, is respectively greater than or less than a threshold. We consider a per-pixel threshold, $T(x, y)$, for per-pixel BG/FG segmentation. $T(x, y)$ is calculated and dynamically updated as

$$T(x, y) = T_{min} H(x, y), \tag{2}$$

where T_{min} is the initial threshold. Note that T_{min} is determined according to the experiment discussed in Section IV-C. $H(x, y)$ is the threshold factor, as in (8).

3) SEGMENTATION DECISION

In our proposal, an observed pixel, $I_o^c(x, y)$, of an observed frame, I_o^c, is classified as either BG or FG. The segmented

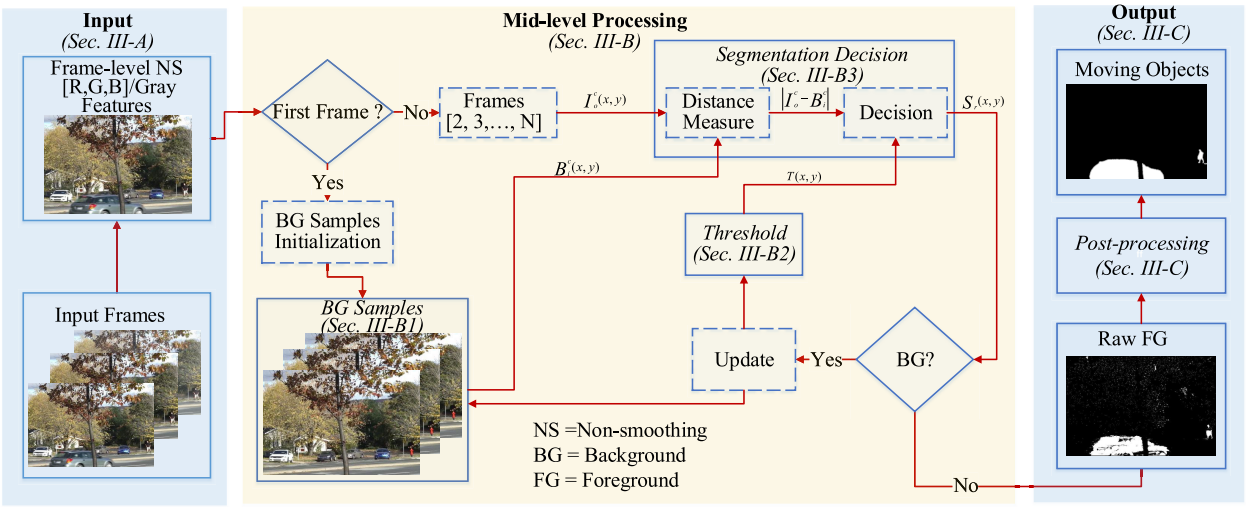


FIGURE 3. The non-smoothing color consensus-based moving object detection model.

BG and FG of $I_o^c(x, y)$ are represented as 0 and 1, respectively in the corresponding pixel, $S_r(x, y)$, of the corresponding segmented frame S_r. When $I_o^c(x, y)$ is dissimilar with its corresponding BG samples, $B_i^c(x, y)$, $I_o^c(x, y)$ is classified as FG (i.e., 1); otherwise, it is classified as BG (i.e., 0) as

$$S_r(x, y) = \begin{cases} 1, & \text{if } \sum_{i=0}^{N-1} M_i(x, y) < M_{min} \\ 0, & \text{otherwise,} \end{cases} \tag{3}$$

where M_{min} is the minimum number of required matches. We select $M_{min} = 2$ like the methods [15], [19]. $M_i(x, y)$ is the intensity similarity measurement counter which counts how many times the absolute intensity distance between an observed pixel, $I_o^c(x, y)$, and its reference BG samples, $B_i^c(x, y)$, is less than or equal to the threshold, $T(x, y)$. In other words, $M_i(x, y)$ calculates the number of BG pixels that are similar to an observed pixel, $I_o^c(x, y)$. When $\sum_{i=0}^{N-1} M_i(x, y)$ is less than the required number of matches, M_{min}, the observed pixel, $I_o^c(x, y)$, is segmented as FG (i.e., 1); otherwise, it is segmented as BG (i.e., 0). The similarity measurement counter, $M_i(x, y)$, $i \in [0, N-1]$, is expressed as

$$M_i(x, y) = \begin{cases} 1, & \text{if } \exists c |I_o^c(x, y) - B_i^c(x, y)| \leq T(x, y) \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $|I_o^c(x, y) - B_i^c(x, y)|$ is the Manhattan distance between $I_o^c(x, y)$ and $B_i^c(x, y)$, $i \in [0, N-1]$, $c \in \{R, G, B\}$. $T(x, y)$ is the threshold, as shown in (2). The statement $\exists c |I_o^c(x, y) - B_i^c(x, y)| \leq T(x, y)$ is false if and only if there is no absolute distance $|I_o^c(x, y) - B_i^c(x, y)|$ which is less than or equal to $T(x, y)$; otherwise, it is true. Put it another way, if $|I_o(x, y) - B_i(x, y)|$ is less than or equal to the threshold $T(x, y)$ for any channel $C = \{R, G, B\}$, the value of $M_i(x, y)$ will be 1; otherwise, the value of $M_i(x, y)$ will be 0. This fusion of $\{R, G, B\}$ color components is called OR fusion in our proposed method. Alternatively, to compare with the OR fusion, we present the AND fusion of $\{R, G, B\}$ color components as

$$M_i(x, y) = \begin{cases} 1, & \text{if } \forall c |I_o^c(x, y) - B_i^c(x, y)| \leq T(x, y) \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

where the statement $\forall c |I_o^c(x, y) - B_i^c(x, y)| \leq T(x, y)$ is true if and only if every absolute distances, $|I_o^c(x, y) - B_i^c(x, y)|$, are less than or equal to $T(x, y)$; otherwise, it is false. In other words, if $|I_o(x, y) - B_i(x, y)|$ is less than or equal to the threshold $T(x, y)$ for every channels C, the value of $M_i(x, y)$ will be 1; otherwise, the value of $M_i(x, y)$ will be 0. In our approach, we propose the OR fusion because OR fusion provided more accuracy over AND fusion, explained in Section IV-E.

The minimum moving average distance, $D_{\min}(x, y)$, is the normalized minimum Manhattan distance between an observed pixel, $I_o^c(x, y)$, and its corresponding BG Samples, $B_i^c(x, y)$, $i \in [0, N-1]$, $c \in \{R, G, B\}$, over recent frames. Like [21], $D_{min}(x, y)$ is used to measure the amount of background motion. $D_{\min}(x, y)$ is estimated as

$$D_{\min}(x, y) = (1 - \gamma) D_{\min}(x, y) + \gamma D(x, y), \tag{6}$$

where γ is the moving average factor. $D(x, y)$ is the normalized minimum absolute distance between $I_o^c(x, y)$ and $B_i^c(x, y)$, $i \in [0, N-1]$, $c \in \{R, G, B\}$. In other words, to estimate $D(x, y)$, the sum of minimum absolute distances between $I_o(x, y)$ and all samples in $B(x, y)$ for all channels, $C = \{R, G, B\}$, is calculated followed by normalization. $D(x, y)$ is expressed as

$$D(x, y) = \frac{1}{|C| M_{int}} \sum_{C} \min_{i \in [0, N-1]} |I_o^c(x, y) - B_i^c(x, y)|, \tag{7}$$

where $|C|$ denotes the cardinality of $C = \{R, G, B\}$, $c \in C$, and M_{int} is the maximum intensity of a pixel. At the beginning, $D_{\min}(x, y)$ is initialized to 0. It is bound to $[0, 1]$.

The value of $D_{\min}(x, y)$ indicates the amount of background motion. In case of completely static background region, $D_{\min}(x, y) \approx 0$ and for entirely dynamic background region, $D_{\min}(x, y) \approx 1$. $D_{\min}(x, y)$ is used to determine threshold factor, $H(x, y)$, and background factor, $U(x, y)$. Subsequently, $H(x, y)$ and $U(x, y)$ are respectively used to update threshold, $T(x, y)$, and BG samples, $B^c(x, y)$.

We are inspired by [19], [21] to determine threshold factor, $H(x, y)$, and background factor, $U(x, y)$. $H(x, y)$ is used to update threshold $T(x, y)$. $H(x, y)$ is calculated, as in (8), shown at the bottom of the page. In that equation, H_s is the step size used to increase/decrease $H(x, y)$. Initially, $H(x, y) = 1$. $H(x, y)$ is always $\geqslant 1$.

Background factor, $U(x, y)$, is presented, as in (9), shown at the bottom of the page. In that equation, R_m is the minimum background distance, U_{dr} is the step size used to decrease $U(x, y)$, and U_{in} is the step size used to increase $U(x, y)$. Initially, $U(x, y) = 2$. $U(x, y)$ is bound to $[2, 256]$.

BG factor, $U(x, y)$, determines probability used to update background samples, $B^c(x, y)$. The probability is calculated as $P(x, y) = 1/U(x, y)$. Based on $P(x, y)$, a background pixel in $B^c(x, y)$ is replaced by a segmented BG pixel determined, as in (3). Also, based on $P(x, y)$, the segmented BG pixel randomly replaces neighbors BG samples to maintain spatial consistency.

C. OUTPUT

The raw segmentation frame, S_r, is a binary image frame which is attained after the mid-level processing. This S_r is an unfiltered segmentation frame. Therefore, S_r may contain holes in the object silhouettes and salt and pepper noises. These holes and noises are effectively discarded by a post-processing operation discussed in the following.

1) POST-PROCESSING

We design a post-processing operation introduced in [21], [47]. The post-processing is accomplished as follows. The raw segmentation frame, S_r, becomes the input of the post-processing operation. Then, the morphological transformation performs the erosion and close operation on S_r. After that, flood fill operation fills the holes of object silhouettes. After filling holes in the FG objects, a complement operation is performed. Bitwise OR is carried out between S_r and the complemented frame. Finally, a median filter discards salt and pepper noises. Therefore, we get filtered FG moving objects, S_f.

IV. EXPERIMENTAL RESULTS

In this section, we discuss the datasets such as CD-2012 [48], CD-2014 [49], CMD [50], and LASIESTA [51] and list the key evaluation metrics (KEM) used to evaluate our proposed Fast-D. Moreover, we include (1) the hyperparameter setting, (2) the smoothing effects versus non-smoothing effects, (3) a fusion of RGB features, (4) accuracy on dynamic threshold versus static threshold, (5) accuracy on post-processing versus without post-processing, (6) the proposed method's performance on CD-2012, CD-2014, CMD, and LASIESTA datasets, (7) the comparisons of methods on CMD dataset, (8) the comparisons of methods on CD-2012 and CD-2014 datasets, (9) the comparisons of methods on LASIESTA dataset, (10) the qualitative comparison of methods, (11) an algorithmic complexity comparison, and (12) real-time testing.

A. DATASETS

CD-2012 [48] is one of the most used dataset for the performance evaluation of BGS approaches. It has 6 different number of video categories: baseline (BL), camera jitters (CJ), thermal (TR), intermittent object motion (IM), shadow (SD), and dynamic background (DB). And, CD-2014 [49] is a very complex dataset most frequently used for BG/FG segmentation accuracy, which is an extension of CD-2012 dataset. In addition to 6 categories in CD-2012 dataset, CD-2014 has 5 additional video categories: bad weather (BW), low frame rate (LR), PTZ, turbulence (TB), and night video (NV). CD-2012 and CD-2014 datasets consist of 31 and 53 videos, respectively, where each category has $4 \sim 6$ videos. Also, CMD [50] is a small dataset that has a video of 500 frames. Furthermore, LASIESTA [51] dataset has videos with real indoor and outdoor scenes of indoor simple sequences (I_SI), indoor camouflage (I_CA), indoor occlusions (I_OC), indoor illumination changes (I_IL), indoor modified background (I_MB), indoor bootstrap (I_BS), outdoor cloudy conditions (O_CL), outdoor rainy conditions (O_RA), outdoor snowy conditions (O_SN), and outdoor sunny conditions (O_SU). The challenges of indoor and outdoor videos include smooth shadows, moderate shadows, hard shadows, camouflage, dynamic background, partial occlusion, total occlusion, sudden illumination change, permanent changes in the background, abandoned object, bootstrap, cloud, rain, and snow. The resolution of the videos is 352×288.

To evaluate performance of object detection method, each input frame of all datasets [48]–[51] has a corresponding

$$H(x, y) = \begin{cases} H(x, y) + H_s, & \text{if } H(x, y) < (1 + D_{\min}(x, y))^2 \\ \max\big(1, (H(x, y) - H_s)\big), & \text{otherwise,} \end{cases} \tag{8}$$

$$U(x, y) = \begin{cases} \min\big(256, (U(x, y) + U_{in})\big), & \text{if } D_{\min}(x, y) < R_m \\ \max\big(2, (U(x, y) - U_{dr})\big), & \text{otherwise,} \end{cases} \tag{9}$$

ground-truth frame. In this ground truth frame, background and foreground pixels are assigned by 0 and 255 intensity values, respectively.

B. KEY EVALUATION METRICS

Key evaluation metrics (KEM) such as Recall (Re), Specificity (Sp), False Positive Rate (FPR), False Negative Rate (FNR), Percentage of Wrong Classification (PWC/PBC), Precision (Pr), and F-Measure (F1-score/FM), which are used to evaluate the performance of moving object detection [48]. Among KEM, F1-score is widely recognized as the best metric since F1-score is a harmonic mean of recall (Re) and precision (Pr). Moreover, according to [48], [49], among KEM, F1-score correlates most robustly to evaluate the moving object detection/background subtraction on average. Description of True Positive *(TP)*, True Negative *(TN)*, False Positive *(FP)*, and False Negative *(FN)* are listed as:

1) True Positive *(TP)*: The correct foreground pixel.
2) True Negative *(TN)*: The correct background pixel.
3) False Positive *(FP)*: The incorrect foreground pixel.
4) False Negative *(FN)*: The incorrect background pixel.

Given *TP*, *TN*, *FP*, and *FN*, formulas of KEM are expressed as:

$$\text{Specificity (Sp)}: \frac{TN}{FP + TN} \tag{10}$$

$$\text{False Positive Rate (FPR)}: \frac{FP}{FP + TN} \tag{11}$$

$$\text{False Negative Rate (FNR)}: \frac{FN}{FP + TN} \tag{12}$$

$$\text{Percentage of Wrong Classification (PWC)}: 100 \cdot \frac{(FN + FP)}{(TP + FN + FP + TN)} \tag{13}$$

$$\text{Recall (Re)}: \frac{TP}{TP + FN} \tag{14}$$

$$\text{Precision (Pr)}: \frac{TP}{TP + FP} \tag{15}$$

$$\text{F-Measure (F1-score/FM)}: \frac{(2 \cdot Pr \cdot Re)}{Pr + Re} \tag{16}$$

A small value of FPR, FNR, and PWC and a large value of Sp, Re, Pr, and F1-score are better. The categorical average and overall average (Avg.) are the simple averages (arithmetic averages) which are estimated as in [48].

C. HYPERPARAMETER SETTING

We inherited a few hyperparameters from previously published methods such as ViBe [15], ViBe$^+$ [16], PBAS [19], LOBSTER [20], and SuBSENSE [21]. We use the following parameters from those methods.

1) $M_{min} = 2$: The optimum number of required matches for segmentation is taken like ViBe, ViBe$^+$, and PBAS.
2) $H(x, y) = 1$: The initial value of the threshold factor is, $H(x, y) = 1$, the same as LOBSTER. After that, it is updated dynamically.
3) $U(x, y) = [2, 256]$: Initially, $U(x, y) = 2$. The values of the background factor are the same as SuBSENSE [21].

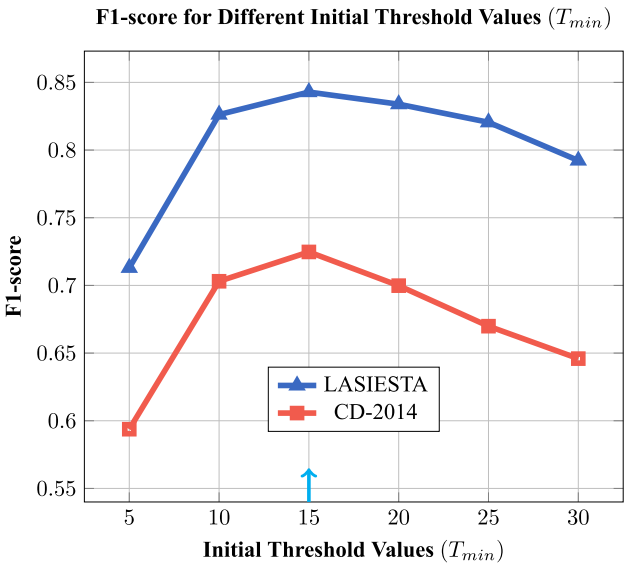


FIGURE 4. Accuracy in terms of F1-score for various initial threshold T_{min} on CD-2014 and LASIESTA datasets. Different values of T_{min} are shown along the x-axis. F1-scores due to the different values of T_{min} are shown along the y-axis. The cyan colored arrow indicates the best F1-scores due to T_{min}.

Besides, the remaining hyperparameters were decided the way the existing methods [15], [16], [19]–[21] determined. Note that the existing methods chose the parameters based on the set that provided the overall best scores across the dataset. Therefore, we did experiments using the same parameter sets on CD-2014 [49] and LASIESTA [51] datasets. Overall F1-scores on each dataset were estimated for the different values of a parameter. The optimum values of the parameters are chosen in Fast-D to obtain the overall best results across all these datasets. According to the experimental results displayed in Fig. 4, we found the optimal value of the initial threshold, $T_{min} = 15$. Besides, according to Fig. 5, we got the optimal value of the moving average factor, $\gamma = 50$. Moreover, in the same way, based on experiments, we achieved the optimal values of the remaining hyperparameters in our proposal.

D. SMOOTHING EFFECTS VERSUS NON-SMOOTHING EFFECTS

In a video, the intensity of a pixel is usually changed (because of noises) over time in a way that the intensity of the pixel is higher or lower than neighbors. The intensity of the pixel is primarily corrected by the image smoothing functions such as average filter, Gaussian filter, and median filter. Based on neighbors' intensities of a pixel, the smoothing functions decrease the higher intensity and increase the lower intensity to correct intensity. To compensate noises, the filtering functions lose the salient edge of the moving object explained in Section I. Moreover, we did the experiments to clearly observe the effects due to non-smoothing versus smoothing video input in our proposed approach in terms of F1-score. The experimental evaluations on CD-2014 dataset

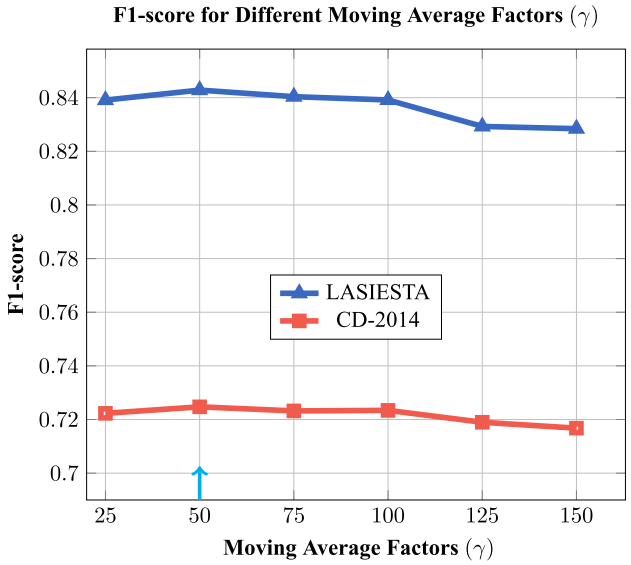


FIGURE 5. Accuracy in terms of F1-score for various moving average values of γ on CD-2014 and LASIESTA datasets. Different values of γ are shown along the x-axis. F1-scores due to the different values of γ are shown along the y-axis. The cyan colored arrow indicates the best F1-scores due to γ.

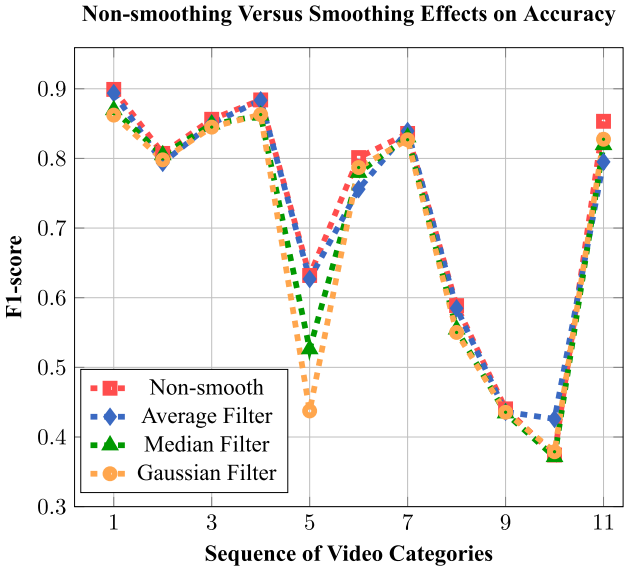


FIGURE 6. Non-smoothing versus smoothing effects of pixel's intensity on accuracy in terms of F1-score on CD-2014 dataset. The video categories are shown along the x-axis. The videos in the number of categories were filtered by the average filter, median filter, and Gaussian filter. F1-scores due to these filtering videos and non-filtering (non-smoothing) videos are shown along the y-axis.

are shown in Fig. 6. We observed that among the smoothing functions, the average filter showed the best accuracy. However, F1-scores of non-smoothing video categories on CD-2014 dataset were always higher than smoothing video cases except 10th (PTZ) video category. On an average, our proposed Fast-D with the non-smoothing color features obtained 0.79%, 2.47%, and 3.28% more F1-scores than the average filter, median filter, and Gaussian filter, respectively. Additionally, the smoothing functions incurred extra processing costs (memory usage and running time). Therefore, we do not smooth input frames in our proposed solution.

E. THE FUSION OF RGB FEATURES

RGB color features' based similarity/dissimilarity is calculated to segment BG/FG pixel. To measure similarity/dissimilarity, OR and AND fusion of $\{R, G, B\}$ channels were calculated as in (4) and (5). We experimented with these OR and AND fusions to observe the accuracy differences between them. F1-scores due to OR fusion and AND fusion on CD-2014 dataset are shown in Fig. 7. We observed that the AND fusion always exhibited lower accuracy than the OR fusion shown in Fig. 7. On an average, OR fusion escalated 8.39% more F1-score than AND fusion. Therefore, we choose the OR fusion to fuse $\{R, G, B\}$ channels to classify BG/FG.

F. ACCURACY ON DYNAMIC THRESHOLD VERSUS STATIC THRESHOLD

The dynamic threshold is modified based on the amount of dynamic background motion, as expressed in (6). Alternatively, the static threshold is not updated during the BG/FG

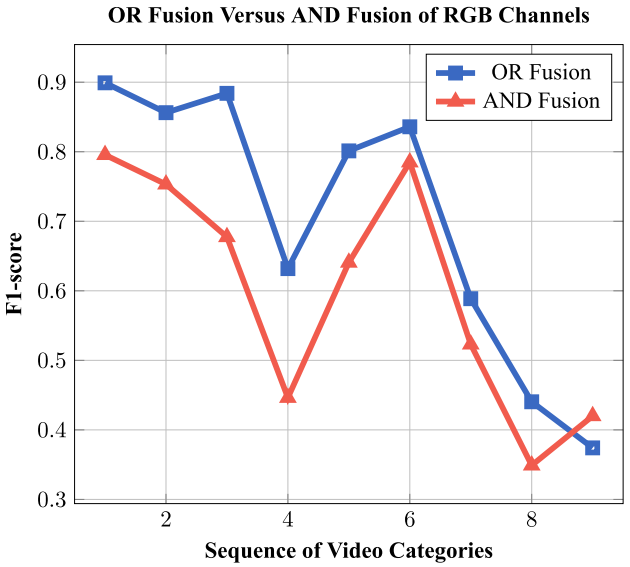


FIGURE 7. Effect of OR fusion versus AND fusion in terms of F1-score on CD-2014 dataset. The x-axis represents the video categories. F1-scores due to OR fusion versus AND fusion are depicted along the y-axis.

segmentation over frame sequences. However, a pixel's intensity varies over time. Hence, the static threshold increases more false positive and false negative than the dynamic threshold. Performance differences in terms of F1-score due to dynamic threshold and static threshold are displayed in Fig. 8. We saw that the dynamic threshold-based segmentation outperformed than static threshold-based segmentation except on 1th (BL), 9th (NV), 10th (PTZ), and 11th (TB) videos. Overall, the dynamic threshold gained 1.05% more

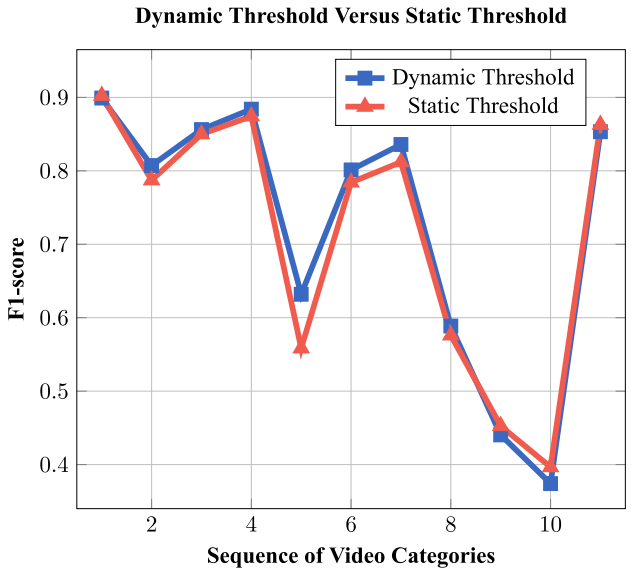


FIGURE 8. A comparison between the dynamic threshold and static threshold in terms of F1-score on CD-2014 dataset. The video categories are shown along the x-axis. F1-scores due to the dynamic threshold and the static threshold are depicted along the y-axis.

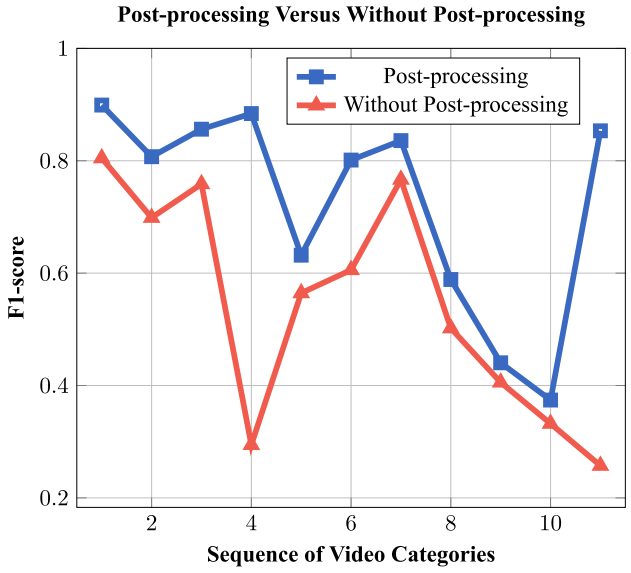


FIGURE 9. A comparison between post-processing and without post-processing in terms of F1-score on CD-2014 dataset. The video categories are shown along the x-axis. F1-scores due to the post-processing and without post-processing are depicted along the y-axis.

F1-score than the static threshold. Therefore, we consider the dynamic threshold, as presented in (2) in our approach.

G. ACCURACY ON POST-PROCESSING VERSUS WITHOUT POST-PROCESSING

Post-processing operation contributes more to accuracy improvement. Fig. 9 shows the accuracy due to post-processing and without post-processing in our proposed solution. We observe that there are sharp accuracy increments on all the video cases due to the use of post-processing operation. Moreover, the overall F1-score due to post-processed frames jumped 18.01% than the raw segmented frames. As we previously mentioned that we do not smooth the input frames which result in some blinking pixels in addition to sharp object detection. The blinking pixels (salt and pepper noise) are those foreground pixels that fluctuate between consecutive segmented frames. We see that these blinking pixels are compensated to a large extent with the use of post-processing operation.

TABLE 2. Experimental results on CD-2012, CD-2014, and CMD datasets in terms of key evaluation metric (KEM).

Cate./Method	Sp	FPR	FNR	PWC	Re	Pr	F1
BL	0.9966	0.0034	0.0868	0.7187	0.9132	0.8834	0.8991
TR	0.9927	0.0073	0.2116	1.4336	0.7884	0.8714	0.8072
SD	0.9904	0.0095	0.1002	1.4614	0.8998	0.8235	0.8562
DB	0.9915	0.0008	0.1180	0.1722	0.8820	0.8871	0.8840
IM	0.9922	0.0078	0.3546	3.3123	0.6453	0.8153	0.6319
CJ	0.9875	0.0125	0.1713	1.8580	0.8287	0.7854	0.8012
Avg.-2012	0.9931	0.0069	0.1738	1.4927	0.8262	0.8452	0.8133
BW	0.9988	0.0012	0.2202	0.4914	0.7798	0.9067	0.8359
LR	0.9869	0.0130	0.2246	1.8795	0.7754	0.5714	0.5887
NV	0.9613	0.0387	0.2459	4.3825	0.7541	0.3537	0.4404
PTZ	0.9516	0.0484	0.0597	4.8829	0.9403	0.2921	0.3741
TB	0.9997	0.0003	0.1997	0.1901	0.8003	0.9222	0.8533
Avg.-2014	0.9870	0.0130	0.1812	1.8893	0.8188	0.7379	0.7247
CMD	0.9961	0.0039	0.0286	0.5155	0.9714	0.9334	0.9520

TABLE 3. Key evaluation metric (KEM)-based experimental results on LASIESTA dataset.

Category	Sp	FPR	FNR	PWC	Re	Pr	F1
I_SI	0.9961	0.0039	0.0500	0.5481	0.9500	0.9084	0.9287
I_CA	0.9939	0.0061	0.1415	2.2584	0.8585	0.9375	0.8924
I_OC	0.9949	0.0052	0.0214	0.5859	0.9786	0.8669	0.9194
I_IL	0.8653	0.1347	0.0987	13.104	0.9013	0.3762	0.5021
I_MB	0.9910	0.0090	0.0205	1.0102	0.9795	0.9109	0.9430
I_BS	0.9972	0.0027	0.4643	1.2397	0.5357	0.7666	0.6182
O_CL	0.9931	0.0069	0.0121	0.6359	0.9879	0.8912	0.9368
O_RA	0.9973	0.0027	0.0162	0.3205	0.9838	0.8961	0.9378
O_SN	0.9923	0.0078	0.0469	0.8095	0.9531	0.8154	0.8789
O_SU	0.9976	0.0024	0.0830	0.3279	0.9170	0.8319	0.8710
Average	0.9819	0.0181	0.0955	2.0840	0.9045	0.8201	0.8428

H. OUR METHOD's PERFORMANCE ON CD-2012, CD-2014, CMD, AND LASIESTA DATASETS

Tables 2 and 3 show evaluation of our method on CD-2012 [48], CD-2014 [49], CMD [50], and LASIESTA [51] datasets in terms of key evaluation metrics (KEM) such as specificity (Sp), false positive rate (FPR), false negative rate (FNR), percentage of wrong classification (PWC), recall (Re), precision (Pr), and F1-score (F1). However, F1-score is emphasized more to measure the accuracy of BG/FG segmentation as explained in Section IV-B. According to Tables 2 and 3, our method achieved 81.33%, 72.47%, 95.20%, and 84.28% overall F1-scores on CD-2012, CD-2014, CMD, and LASIESTA datasets, respectively. Because we effectively used non-smoothing color features, dynamic threshold, features' fusion strategy, and

post-processing operations in our proposed approach, segmentation accuracy improved. However, due to the most dynamic background scenes in CD-2014 dataset, segmentation accuracy in terms of F1-score on this dataset was lower than CD-2012, CMD, and LASIESTA datasets. Besides, F1-score of Fast-D on indoor illumination changes (I_IL) category was lower than other categories in Table 3. Note that our Fast-D is not illumination invariant; therefore, it cannot compensate for sudden global illumination changes of the I_IL videos which result in reduced F1-score and increased PWC. Also, our method's percentage of wrong classification (PWC) on LASIESTA dataset shown in Table 3 was higher than other datasets presented in Table 2. It happened because I_IL category in Table 3 contributed the largest PWC score which led to an increased average PWC score. Moreover, recall (Re) scores were always greater than precision (Pr) scores in Tables 2 and 3 as our proposed non-smoothing color features detected more changes than smoothing color features.

TABLE 4. Comparisons of methods on CMD dataset [50] in terms of key evaluation metric (KEM).

Method	Year	Sp	FPR	FNR	PWC	Re	Pr	F1
Fast-D	-	0.9961	0.0039	0.0286	*0.5155*	**0.9714**	0.9334	**0.9520**
SuBS [21]	2015	0.9962	0.0038	0.0405	0.5738	*0.9594*	0.9335	*0.9462*
SBS [17]	2019	0.9975	0.0024	**0.0014**	**0.3862**	0.9170	0.8672	0.8914
PBAS [19]	2012	*0.9981*	*0.0019*	0.1872	1.1500	0.8128	**0.9585**	0.8797
LOB [20]	2014	0.9980	0.0020	0.1893	1.1910	0.8107	*0.9565*	0.8776
ViBe [15]	2011	0.9939	0.0060	*0.0018*	0.7800	0.8950	0.7209	0.7986
SuperBE [18]	2019	**0.9994**	**0.0005**	0.0105	1.0844	0.5343	0.9552	0.6852

I. COMPARISONS OF METHODS ON CMD DATASET

Table 4 shows the comparison of our method with the state-of-the-art approaches in terms of key evaluation metric (KEM). Moreover, recall (Re), precision (Pr), and F1-score (F1) are the most important metrics in KEM. As in the previous table, the bold, italic, and underlining text styles denote the same ranks. Our method achieved the best accuracy with 97.14% Re and 95.20% F1-score, as displayed in Table 4. However, in terms of precision (Pr) and percentage of wrong classification (PWC), our method attained the third rank. Besides, other KEM measures have not improved in our proposed method like Re and F1. The reason is that our proposed non-smoothing color features keep all changes which increase false positive in addition to true positive. Overall, our method outperformed compared with the existing approaches.

J. THE COMPARISON OF OUR METHOD ON CD-2012 AND CD-2014 DATASETS

Table 5 shows the comparison of our proposed approach with the existing methods in terms of F1-score on CD-2012 and CD-2014 datasets. As we mentioned before, F1-score is the most important metric to evaluate the accuracy of BG/FG segmentation. Therefore, we use F1-score for performance comparison. Bold, italic, and underlining text represent first, second, and third ranks, respectively. In Table 5, MS-ST method outperformed for all video cases. However, our proposed Fast-D achieved the second-best accuracy on dynamic background (DB), pan-tilt-zoom (PTZ), and turbulence (TB) videos. Moreover, our method showed comparable accuracy to DeepBS, and SuBSENSE (SuBS). We observed that more pixel features (color + edge or texture features) incurred more false detection on videos with dynamic background scenes, pan-tilt-zoom, and turbulence. Therefore, we use only non-smoothing color features in our proposed Fast-D.

K. COMPARISONS OF METHODS ON LASIESTA DATASET

Table 6 presents F1-score based category-wise comparisons. Like previous Tables 4 and 5, the bold, italic, and underlining text provide first, second, and third ranks, respectively. According to Table 6, our method achieved the best rank on I_SI, I_MB, O_CL, O_RA, and O_SU videos with 92.87%, 94.30%, 93.68%, 93.78%, and 87.10% F-measures, respectively. It happens because our proposed non-smoothing color features with the OR fusion do not decay any changes and keep more edge information of input frames, which increase true positive; the adaptive threshold regulates false detection; the post-processing operation decreases false positive by discarding blinking pixels (The blinking pixel is a pixel that either resides in the observed segmented frame or the previous segmented frame, but it does not stay in both frames.) and increases true positive by filling holes in the object silhouettes simultaneously. However, our proposal attained 50.21% F1-score on I_IL video category. Note that I_IL category has videos with sudden global illumination changes. As we previously mentioned that our method is not illumination invariant; therefore, Fast-D detected more false positive in addition to true positive, which led to decreased F-measure on I_IL videos. Moreover, on average, Fast-D performed 88.07% F1-score whereas MSFgNet showed 86.92% F-measure when excluding I_IL videos.

L. METHODS' QUALITATIVE COMPARISON ON CD-2014 DATASET

For qualitative comparison, we choose M^{4}CD [25], SuBSENSE [21], and DeepBS [33] methods. We select four frames with challenging scenes such as TR_library #2575, DB_overpass #2406, PTZ_intermittenobjctmotion #1241, and TB_turbulence2 #2455 from CD-2014 [49] dataset to show in Fig. 10. Input frames with their corresponding ground truth frames and segmented frames of proposed Fast-D, M^{4}CD, SuBSENSE, and DeepBS methods are shown from left to right. It is clearly observed that our method more accurately detects object silhouettes than the other methods. The reasons for improved moving object detection are that the non-smoothing color features do not lose any changes in the input; therefore it keeps intact all edge information; the OR fusion detects more changes than other fusions; the floodfill operation in post-processing fills the holes in the raw segmentation. All the segmented results on CD-2014 dataset are available at our URL.[1]

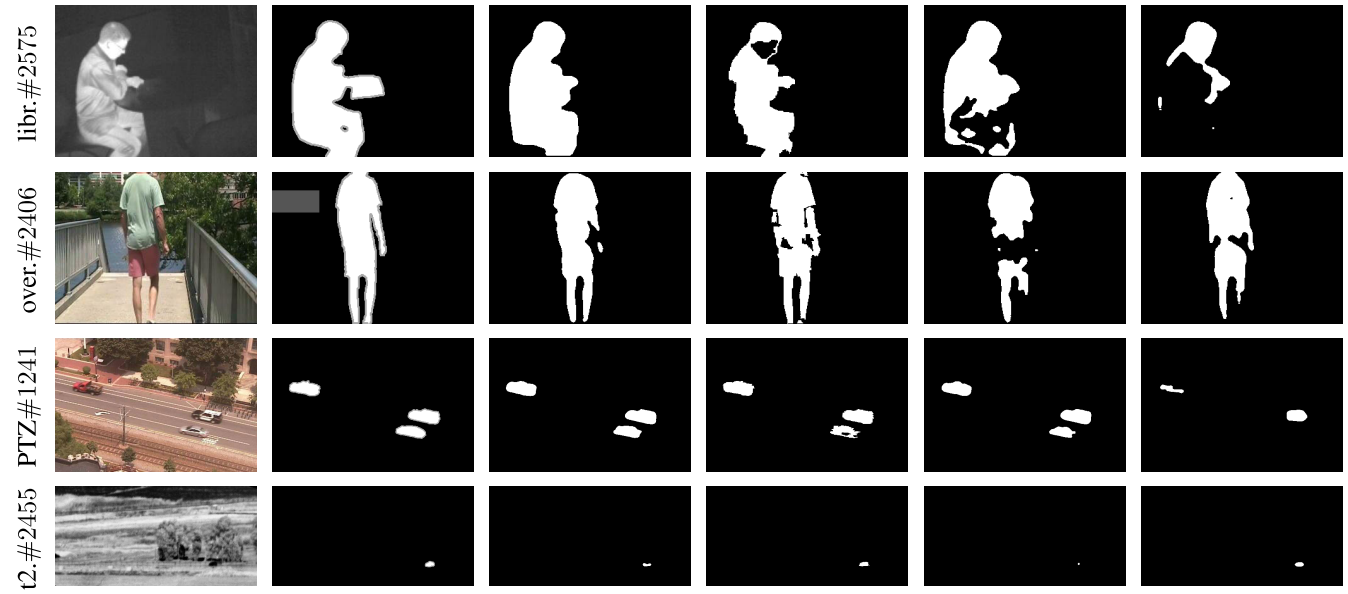
[1] https://github.com/alamgir39/Fast-D

TABLE 5. Category-wise comparisons of methods on CD-2012 [48] and CD-2014 [49] datasets in terms of F1-score.

Method	Year	BL	TR	SD	DB	IM	CJ	Avg.CD-2012	BW	LR	NV	PTZ	TB	Avg.CD-2014
MS-ST [36]	2019	**0.9895**	**0.9840**	**0.9874**	**0.9791**	**0.9893**	**0.9802**	**0.9849**	**0.9846**	**0.9013**	**0.9390**	**0.9314**	**0.9569**	**0.9657**
DeepBS [33]	2018	*0.9580*	0.7583	*0.9304*	0.8761	0.6098	*0.8990*	*0.8386*	0.8301	0.6002	*0.5835*	0.3133	0.8455	*0.7458*
SuBS [21]	2015	0.9480	0.8184	0.8890	0.8138	0.6523	0.7694	0.8152	*0.8528*	0.6437	0.5390	0.3185	0.8197	0.7331
Fast-D	-	0.8991	0.8072	0.8562	*0.8840*	0.6319	0.8012	0.8133	0.8359	0.5887	0.4404	*0.3741*	*0.8533*	0.7247
M^4CD [25]	2018	0.9322	0.7448	0.8969	0.6857	*0.6939*	0.8231	0.7961	0.8136	0.6275	0.4946	0.2322	0.7978	0.7038
BMN-BSN [37]	2019	0.9521	0.7849	0.8588	0.6371	0.6369	0.6962	0.7610	-	-	-	-	-	-
LOB [20]	2014	0.9242	*0.8248*	0.8728	0.5679	0.5770	0.7423	0.7515	0.6378	*0.6738*	0.4293	0.0875	0.5299	0.6230
PBAS [19]	2012	0.7363	0.6887	0.7732	0.6080	0.3618	0.5600	0.6280	0.7863	0.5366	0.3795	0.1130	0.7236	0.5734
MoG [11]	2002	0.6423	0.4354	0.6766	0.6855	0.3373	0.6316	0.5681	0.6512	0.6110	0.3748	0.2623	0.7650	0.5521
SC-SOBS [30]	2012	0.9333	0.6923	0.7784	0.6739	0.5918	0.7051	0.7283	-	-	-	-	-	0.5961
$ViBe^+$ [16]	2012	0.8710	0.6650	0.8150	0.7200	0.5090	0.7540	0.7220	-	-	-	-	-	-
SBS [17]	2019	-	-	-	-	-	-	0.7197	-	-	-	-	-	-
ViBe [15]	2011	0.8700	0.6650	0.8030	0.5650	0.5070	0.6000	0.6680	-	-	-	-	-	-
RB-SOM [34]	2018	-	-	-	-	-	-	-	-	-	-	-	-	0.5591
SuperBE [18]	2019	-	-	-	-	-	-	0.5356	-	-	-	-	-	-

TABLE 6. F1-score based category-wise methods' comparisons on LASIESTA dataset [51].

Method	Year	I_SI	I_CA	I_OC	I_IL	I_MB	I_BS	O_CL	O_RA	O_SN	O_SU	Average
MSFgNet [35]	2018	*0.9264*	**0.9213**	0.9163	**0.8967**	0.9143	**0.7157**	0.8806	0.8659	**0.8952**	0.7869	**0.8717**
Fast-D	-	**0.9287**	0.8924	*0.9194*	0.5021	**0.9430**	0.6182	**0.9368**	**0.9378**	*0.8789*	**0.8710**	0.8428
Berjón *et al.* [28]	2018	0.8805	0.8444	0.7806	0.6487	*0.9373*	0.6644	*0.9276*	0.8669	0.7786	0.7221	0.8051
Haines *et al.* [29]	2013	0.8876	*0.8938*	**0.9223**	*0.8491*	0.8440	*0.6809*	0.8267	*0.8908*	0.1750	*0.8568*	0.7826
Cuevas *et al.* [27]	2013	0.7859	0.7361	0.8527	0.7915	0.7288	0.5836	0.8638	0.8085	0.4555	0.7305	0.7335



(a) Input Frame **(b)** Ground truth **(c)** Proposed Fast-D **(d)** M^4CD [25] **(e)** SuBSENSE [21] **(f)** DeepBS [33]

FIGURE 10. Typical segmented frames for various input frame sequences on CD-2014 dataset; input frames and corresponding ground truth frames, and segmented frames of Fast-D, M^4CD, SuBSENSE, and DeepBS are depicted from left to right.

M. THE ALGORITHMIC COMPLEXITY COMPARISON

We implemented proposed Fast-D, $ViBe^+$ [16], PBAS [19], LOBSTER (LOB) [20], and SuBSENSE (SuBS) [21] in a desktop computer with the configuration shown in Table 7. To compare computational complexity, we used baseline_higway, baseline_pedestrians, and shadow_peopleinshade videos with the size of 320×240, 360×240, and 380×244, respectively. Moreover, experimental setup of M^4CD [25], SC-SOBS [30], DeepBS [33], RB-SOM [34], MSFgNet [35], and MS-ST [36] methods is shown in Table 8. Memory usage and processing time are respectively measured in terms of bytes per pixel (bpp) and frames per second (fps). First, second, and third ranks are respectively denoted by bold, italic, and underlining as shown

TABLE 7. Experimental setup of proposed Fast-D, ViBe$^+$ [16], PBAS [19], LOBSTER (LOB) [20], and SuBSENSE (SuBS) [21] for complexity comparison.

Hardware/Software	Description
Operating system	Windows 10 (64 bit)
CPU	Intel Core i5-8400 2.80 GHz
RAM	16 GB
Programming language	C++
Additional library	OpenCV 3.3.0
IDE	Microsoft Visual Studio 2015
Webcam	@INFO ALC-M800

TABLE 8. Experimental setup of existing methods for the comparison.

Method	CPU	GPU/RAM	Language
M^{4}CD [25]	Intel Corei5-3210M	-	Matlab
DeepBS [33]	Intel Xeon (R) CPU E5-1620 v3 @3.5GHz × 8	GeForce Titan X	Lua
RB-SOM [34]	IntelCore i5 2.7GHz	8 GB RAM	Matlab-2016
MSFgNet [35]	Intel Corei7 4.20 GHz	NVIDIA GTX 1080 11GB	-
MS-ST [36]	-	NVIDIA 1080Ti	Tensorflow
SC-SOBS [30]	Core i3-330M 2.13GHz	-	C

in the previous tables. Note that a small value for bpp and a large value for fps are better.

Table 9 shows a comparison of computational complexity in terms of bpp and fps. In this comparison, we see that our proposed Fast-D takes only 3 bpp whereas PBAS, LOBSTER, and SuBSENSE use 15 bpp and ViBe$^+$ allocates 4 bpp. In other words, our proposal achieved the first best rank in terms of bpp. According to Table 9, our proposed method also outperformed in terms of fps like memory usage. More explicitly, the frame rates of our approach were 74.24 fps, 69.51 fps, and 59.64 fps for the frames dimension of 320 × 240, 360 × 240, and 380 × 244, respectively. Put it another way, our method processed frames faster than real-time frame rate. Alternatively, PBAS and LOBSTER took the second-best rank and the third-best rank, respectively. However, MSFgNet attained 59.88 fps for the frame size of 256 × 256 while using a GPU.

N. REAL-TIME TESTING

Real-time testing refers to the process of evaluating a real-time system. For real-time testing, we implemented our proposed Fast-D on a desktop computer with a webcam. The detailed experimental setup is shown in Table 7. Video frames captured from the webcam were the size of 320 × 240. Video capture and moving object detection were performed concurrently. In each video frame, there was only one moving foreground which was detected by our developed system. Figs. 11a and 11b show input frames and their corresponding segmented binary frames, respectively. Moreover, our Fast-D

TABLE 9. Methods' complexity comparisons in terms of bytes per pixel (bpp) and frames per second (fps).

Method	Year	bpp	fps			
			256 × 256 resolution	320 × 240 resolution	360 × 240 resolution	380 × 244 resolution
Fast-D	**-**	**3**	**-**	**74.24**	**69.51**	**59.64**
MSFgNet [35]	2018	-	59.88	-	-	-
PBAS [19]	2012	15	-	*47.71*	*46.88*	*41.98*
LOB [20]	2014	15	-	35.09	38.75	36.52
SuBS [21]	2015	15	-	23.60	25.69	22.79
SC-SOBS [30]	2012	-	-	23.00	-	-
ViBe$^+$ [16]	2012	*4*	-	15.22	16.76	13.04
MS-ST [36]	2019	-	-	11.00	-	-
DeepBS [33]	2018	-	-	10.00	-	-
RB-SOM [34]	2018	-	-	07.00	-	-
M^{4}CD [25]	2018	6	-	00.21	-	-

(a) Input Frames **(b)** Segmented Frames

FIGURE 11. The real-time moving object detection for the webcam input. (a) input frames and (b) corresponding segmented frames are shown side by side. White and black pixels represent foreground and background in the segmented frames, respectively.

processed 77.05 frames per second. Therefore, we observe that our proposed system can accurately detect moving FG and ran frames faster than real-time frame rate.

V. DISCUSSION

In this section, we summarize the experimental outcomes from Section IV and list the advantages of our proposal.

Our proposal consisted of non-smoothing input frames, segmentation strategy, dynamic threshold, and adaptive post-processing operation increased the BG/FG segmentation accuracy. We briefly describe the important aspects of the

experiments of non-smoothing frames, segmentation strategy, dynamic threshold, and adaptive post-processing operation below.

1) Smoothing functions are generally used to reduce noises in an image. In addition to filtering noises, the smoothing functions also modify essential information about moving objects, which results in decreasing segmentation accuracy. However, according to the experiment shown in Fig. 6, the average filter, median filter, and Gaussian filter, respectively achieved 0.79%, 2.47%, and 3.28% less F1-scores than non-smoothing frames on an average. Therefore, we do not smooth input video frames.
2) We observed much impact of the fusion of $\{R, G, B\}$ color features on segmentation accuracy. We evaluated the logical AND versus logical OR fusion of $\{R, G, B\}$ color features. The evaluation experiment is shown in Fig. 7. In the experiment, the OR fusion increased 8.39% more F1-score than the AND fusion on an average. Hence, we take a logical OR operation to combine the pixel features during the classification.
3) Throughout the classification, the static threshold is not modified whereas the dynamic threshold is altered based on static/dynamic background motion. Overall, the dynamic threshold increased 1.05% more F1-score than the static threshold which is shown in Fig. 8. Therefore, we use the dynamic threshold in our proposed method.
4) Like the previous operations, a post-processing operation can change classification accuracy that is depicted in Fig. 9. On an average, the post-processing increased 18.01% more F1-score than the raw segmentation. This happens because the non-smoothing frame does not lose any information which leads to an accurate boundary edge detection of the object silhouette. Note that a flood fill operation incurs additional errors if all boundaries of an object are not completely detected. Therefore, our used flood fill in the post-processing operation made the least errors to fill the unfilled region of the object silhouette. Consequently, this adaptive post-processing operation is used in our proposed approach.

We combine the above four operations to increase video segmentation accuracy. Our proposed approach obtained 95.20% F1-score on CMD dataset (Table 4). Additionally, overall, our proposal showed 81.33%, 72.47%, and 84.28% F1-scores on CD-2012, CD-2014, and LASIESTA datasets, respectively (Tables 5 and 6). However, we observed that the classification accuracy on the most challenging videos fairly decreased than the least challenging videos (Tables 5 and 6). In terms of computational complexity, according to Table 9, our method used 3 bytes per pixel (bpp), and displayed 74.24, 69.51, and 59.64 frames per second (fps) for the video frames dimension of 320×240, 360×240, and 380×244, respectively.

Tables 4, 6, and 9 presented the advantages of proposed Fast-D as follows. According to Table 4, proposed Fast-D provided the best recall (Re) and the best F1-score (F1) on CMD dataset [50]. Moreover, in Table 6, our proposal achieved the best F1-score on I_SI, I_MB, O_CL, O_RA, and O_SU videos of LASIESTA dataset [51]. However, Fast-D showed lower performance on I_IL videos with sudden global illumination fluctuations. Therefore, when I_IL videos were excluded, our method obtained 88.07% F-measure whereas MSFgNet performed 86.92% F1-score on average. On the other hand, in terms of complexity comparisons in Table 9, Fast-D obtained the best performance. Put it another way, our method processed more frames per second than real-time frame rate and used very limited memory so that it can be used in very limited resource devices.

Segmentation accuracy and computational complexity are simultaneously used to compare each method in our paper. The segmentation accuracy of Fast-D is not as promising as the deep learning-based models such as MS_ST, DeepBS, and MSFgNet and the shallow learning-based approach such as SuBSENSE. However, according to Table 9, proposed Fast-D provides lower computational complexity than MS_ST, DeepBS, MSFgNet, and SuBSENSE. Also, our method presents better accuracy than DeepBS on TR, DB, IM, BW, PTZ, and TB videos in Table 5. Moreover, according to Tables 4 and 5, proposed method obtains better F-measures than SubSENSE on DB, CJ, PTZ, and TB videos and achieves better recall (Re) and F1-score on CMD dataset. Furthermore, Fast-D shows better F1-scores than MSFgNet on I_SI, I_OC, I_MB, O_CL, O_RA, and O_SU videos in Table 6.

VI. CONCLUSION

We propose an effective and efficient moving object detection method based on background subtraction. In our proposed Fast-D, non-smoothing RGB features' fusion, dynamic threshold, and adaptive post-processing increased the BG/FG classification accuracy at a low computational cost. Experimentally, our non-smoothing color features-based moving object detection method outperformed on I_SI, I_MB, O_CL, O_RA, and O_SU videos on LASIESTA dataset. Moreover, our proposal achieved the best accuracy on CMD dataset. On the other hand, the proposed approach is less complex compared to the existing approaches. Our Fast-D processed 67.80 fps on an average. Hence, Fast-D displays frames faster than real-time frame rate.

Our proposed solution worked better for a video with the effects of strong BG motion with cloudy, rainy, and sunny conditions, camouflage, hard shadows, camera motion, patrolling, and zoom and used limited resources. Therefore, security robots can use our method for video surveillance during autonomous patrolling. Also, this solution can be applied in outdoor scenarios with strong BG motion (video with moving objects in a tree shaken by the wind, video with moving objects on the shimmering water, and so on) and videos that are captured by a moving camera. Moreover, most

importantly, Fast-D can be used in limited computing and low memory devices.

In the future, we will modify the threshold determination technique to more effectively choose the threshold dynamically. Also, instead of random BG samples update, we will replace less reliable BG samples by the more reliable BG pixels. Moreover, we will use deep features in addition to the non-smoothing color feature to improve the segmentation performance.

REFERENCES

[1] N. K. Jain, R. Saini, and P. Mittal, "A review on traffic monitoring system techniques," in *Soft Computing: Theories and Applications*. Cham, Switzerland: Springer, 2019, pp. 569–577.

[2] T. Bouwmans and B. Garcia-Garcia, "Background subtraction in real applications: Challenges, current models and future directions," 2019, *arXiv:1901.03577*. [Online]. Available: http://arxiv.org/abs/1901.03577

[3] A. Aggarwal, S. Biswas, S. Singh, S. Sural, and A. K. Majumdar, "Object tracking using background subtraction and motion estimation in MPEG videos," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2006, pp. 121–130.

[4] S. Tuli, N. Basumatary, and R. Buyya, "EdgeLens: Deep learning based object detection in integrated IoT, fog and cloud computing environments," in *Proc. 4th Int. Conf. Inf. Syst. Comput. Netw. (ISCON)*, Nov. 2019, pp. 496–502.

[5] L. U. Khan, I. Yaqoob, N. H. Tran, S. M. A. Kazmi, T. N. Dang, and C. S. Hong, "Edge computing enabled smart cities: A comprehensive survey," *IEEE Internet Things J.*, early access, Apr. 10, 2020, doi: 10.1109/jiot.2020.2987070.

[6] G. Javadzadeh and A. M. Rahmani, "Fog computing applications in smart cities: A systematic survey," *Wireless Netw.*, vol. 26, no. 2, pp. 1433–1457, Feb. 2020.

[7] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, Feb. 2020.

[8] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[9] Y. Zhang, X. Wang, and B. Qu, "Three-frame difference algorithm research based on mathematical morphology," *Procedia Eng.*, vol. 29, pp. 2705–2709, 2012.

[10] H. Wei and Q. Peng, "A block-wise frame difference method for real-time video motion detection," *Int. J. Adv. Robotic Syst.*, vol. 15, no. 4, Jul. 2018, Art. no. 1729881418783633, doi: 10.1177/1729881418783633.

[11] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*. Cham, Switzerland: Springer, 2002, pp. 135–144.

[12] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 246–252.

[13] Z. Zivkovic and F. van der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern Recognit. Lett.*, vol. 27, no. 7, pp. 773–780, May 2006.

[14] H. Wang and D. Suter, "A consensus-based method for tracking: Modelling background scenario and foreground appearance," *Pattern Recognit.*, vol. 40, no. 3, pp. 1091–1105, Mar. 2007.

[15] O. Barnich and M. Van Droogenbroeck, "ViBe: A universal background subtraction algorithm for video sequences," *IEEE Trans. Image Process.*, vol. 20, no. 6, pp. 1709–1724, Jun. 2011.

[16] M. Van Droogenbroeck and O. Paquot, "Background subtraction: Experiments and improvements for ViBe," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 32–37.

[17] Y.-Q. Chen, Z.-L. Sun, and K.-M. Lam, "An effective subsuperpixel-based approach for background subtraction," *IEEE Trans. Ind. Electron.*, vol. 67, no. 1, pp. 601–609, Jan. 2020.

[18] A. T.-Y. Chen, M. Biglari-Abhari, and K. I.-K. Wang, "SuperBE: Computationally light background estimation with superpixels," *J. Real-Time Image Process.*, vol. 16, no. 6, pp. 2319–2335, Dec. 2019.

[19] M. Hofmann, P. Tiefenbacher, and G. Rigoll, "Background segmentation with feedback: The pixel-based adaptive segmenter," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 38–43.

[20] P.-L. St-Charles and G.-A. Bilodeau, "Improving background subtraction using local binary similarity patterns," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, Mar. 2014, pp. 509–515.

[21] P.-L. St-Charles, G.-A. Bilodeau, and R. Bergevin, "SuBSENSE: A universal change detection method with local adaptive sensitivity," *IEEE Trans. Image Process.*, vol. 24, no. 1, pp. 359–373, Jan. 2015.

[22] B. Laugraud, S. Piérard, M. Braham, and M. Van Droogenbroeck, "Simple median-based method for stationary background generation using background subtraction algorithms," in *Proc. Int. Conf. Image Anal. Process.* Cham, Switzerland: Springer, 2015, pp. 477–484.

[23] D. K. Panda and S. Meher, "Detection of moving objects using fuzzy color difference histogram based background subtraction," *IEEE Signal Process. Lett.*, vol. 23, no. 1, pp. 45–49, Jan. 2016.

[24] M. Van Droogenbroeck and O. Barnich, "ViBe: A disruptive method for background subtraction," in *Background Modeling and Foreground Detection for Video Surveillance*, T. Bouwmans, F. Porikli, B. Hoferlin, and A. Vacavant, Eds. London, U.K.: Chapman & Hall, Jul. 2014, ch. 7, pp. 7.1–7.23, doi: 10.1201/b17223-10.

[25] K. Wang, C. Gou, and F.-Y. Wang, "M^4CD : A robust change detection method for intelligent visual surveillance," *IEEE Access*, vol. 6, pp. 15505–15520, 2018.

[26] opencv.org. *Opencv: CV:: Backgroundsubtractormog2 Class Reference*. Accessed: Jul. 3, 2020. [Online]. Available: https://docs.opencv.org/master/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html

[27] C. Cuevas and N. García, "Improved background modeling for real-time spatio-temporal non-parametric moving object detection strategies," *Image Vis. Comput.*, vol. 31, no. 9, pp. 616–630, Sep. 2013.

[28] D. Berjón, C. Cuevas, F. Morán, and N. García, "Real-time nonparametric background subtraction with tracking-based foreground update," *Pattern Recognit.*, vol. 74, pp. 156–170, Feb. 2018.

[29] T. S. F. Haines and T. Xiang, "Background subtraction with DirichletProcess mixture models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 670–683, Apr. 2014.

[30] L. Maddalena and A. Petrosino, "The SOBS algorithm: What are the limits?" in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 21–26.

[31] M. D. Gregorio and M. Giordano, "Change detection with weightless neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 403–407.

[32] M. Braham and M. Van Droogenbroeck, "Deep background subtraction with scene-specific convolutional neural networks," in *Proc. Int. Conf. Syst., Signals Image Process. (IWSSIP)*, Bratislava, Slovakia, May 2016, pp. 1–4.

[33] M. Babaee, D. T. Dinh, and G. Rigoll, "A deep convolutional neural network for video sequence background subtraction," *Pattern Recognit.*, vol. 76, pp. 635–649, Apr. 2018.

[34] J. A. Ramirez-Quintana, M. I. Chacon-Murguia, and G. M. Ramirez-Alonso, "Adaptive background modeling of complex scenarios based on pixel level learning modeled with a retinotopic self-organizing map and radial basis mapping," *Int. J. Speech Technol.*, vol. 48, no. 12, pp. 4976–4997, Dec. 2018.

[35] P. W. Patil and S. Murala, "MSFgNet: A novel compact end-to-end deep network for moving object detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 11, pp. 4066–4077, Nov. 2019.

[36] Y. Yang, T. Zhang, J. Hu, D. Xu, and G. Xie, "End-to-end background subtraction via a multi-scale spatio-temporal model," *IEEE Access*, vol. 7, pp. 97949–97958, 2019.

[37] V. M. Mondéjar-Guerra, J. Rouco, J. Novo, and M. Ortega, "An end-to-end deep learning approach for simultaneous background modeling and subtraction," in *Proc. Brit. Mach. Vis. Conf.*, 2019, p. 266.

[38] D. E. Butler, V. M. Bove, and S. Sridharan, "Real-time adaptive foreground/background segmentation," *EURASIP J. Adv. Signal Process.*, vol. 2005, no. 14, Dec. 2005, Art. no. 841926.

[39] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee, "Using adaptive tracking to classify and monitor activities in a site," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 1998, pp. 22–29.

[40] C. Stauffer and W. E. L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 8, pp. 747–757, 2000.

[41] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," in *Proc. 17th Int. Conf. Pattern Recognit. (ICPR)*, vol. 2, 2004, pp. 28–31.
[42] K. Kim, T. H. Chalidabhongse, D. Harwood, and L. Davis, "Real-time foreground–background segmentation using codebook model," *Real-Time Imag.*, vol. 11, no. 3, pp. 172–185, 2005.
[43] R. Wang, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Static and moving object detection using flux tensor with split Gaussian models," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 414–418.
[44] H. Sajid and S.-C.-S. Cheung, "Universal multimode background subtraction," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3249–3260, Jul. 2017.
[45] F. Kristensen, P. Nilsson, and V. Öwall, "Background segmentation beyond RGB," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, 2006, pp. 602–612.
[46] M. Balcilar, F. Karabiber, and A. C. Sonmez, "Performance analysis of Lab2000HL color space for background subtraction," in *Proc. IEEE INISTA*, Jun. 2013, pp. 1–6.
[47] D. H. Parks and S. S. Fels, "Evaluation of background subtraction algorithms with post-processing," in *Proc. IEEE 5th Int. Conf. Adv. Video Signal Based Surveill.*, Sep. 2008, pp. 192–199.
[48] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar, "Changedetection.Net: A new change detection benchmark dataset," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2012, pp. 1–8.
[49] Y. Wang, P.-M. Jodoin, F. Porikli, J. Konrad, Y. Benezeth, and P. Ishwar, "CDnet 2014: An expanded change detection benchmark dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 387–394.
[50] Y. Sheikh and M. Shah, "Bayesian modeling of dynamic scenes for object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 11, pp. 1778–1792, Nov. 2005.
[51] C. Cuevas, E. M. Yáñez, and N. García, "Labeled dataset for integral evaluation of moving object detection algorithms: LASIESTA," *Comput. Vis. Image Understand.*, vol. 152, pp. 103–117, Nov. 2016.

MD. ALAMGIR HOSSAIN received the bachelor's and master's degrees from the Department of Information and Communication Engineering (ICE), Islamic University, Kushtia, Bangladesh, in 2003 and 2004, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. He is also an Assistant Professor with Islamic University. Previously, he was a Lecturer with the Department of Computer Science and Engineering, Institute of Science Trade & Technology (Under National University), Dhaka, Bangladesh, from October 2007 to April 2010. His research interests include computer vision, machine learning, and image processing.

MD. IMTIAZ HOSSAIN received the B.Sc. degree from the Department of Information and Communication Technology (ICT), Islamic University, Bangladesh. He is currently pursuing the M.S. degree with the Department of Computer Science and Engineering, Kyung Hee University, Suwon, South Korea. He has been a Research Student with the Intelligent Network and Cloud Security (ICNS) Laboratory, Kyung Hee University, since 2019. His research interests include deep learning, image processing, visual-based human action recognition in videos, object detection and recognition, and video surveillance systems.

MD. DELOWAR HOSSAIN received the B.Sc. and M.Sc. degrees from the Department of Information and Communication Engineering (ICE), Islamic University, Bangladesh, in 2004 and 2005, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. From 2011 to 2013, he was the Chairman with the Department of Computer Science and Engineering. He was the Visiting Scholar with Infosys, Bengaluru, India. He is also serving as an Associate Professor with the Department of Computer Science and Engineering, Hajee Mohammed Danesh Science & Technology University, Dinajpur, Bangladesh. His current research interests include cloud/edge/fog computing, big data, machine learning, and the Internet of Things. He was a recipient of Best Paper Award in KSC 2018 and KSC 2019, South Korea.

NGO THIEN THU received the B.S. degree in management information system from Vietnam National University, Ho Chi Minh City, in 2010. She is currently pursuing the combined Ph.D. degree with the Department of Computer Science and Engineering, Kyung Hee University. Her research interests include visual saliency, intelligent vision systems, and deep learning.

EUI-NAM HUH (Member, IEEE) received the B.S. degree from Busan National University, South Korea, the master's degree in computer science from The University of Texas at Austin, USA, in 1995, and the Ph.D. degree from Ohio University, USA, in 2002. He is currently a Professor with the Department of Computer Science and Engineering, Kyung Hee University, South Korea. His research interests include cloud computing, the Internet of Things, future Internet, distributed real-time systems, mobile computing, big data, and security. He is also a review board of National Research Foundation of Korea. He has also served many community services for ICCSA, WPDRTS/IPDPS, APAN Sensor Network Group, ICUIMC, ICONI, APIC-IST, ICUFN, and SoICT as various types of chairs. He is also a Vice-Chairman of Cloud/Bigdata Special Technical Group of TTA, and an Editor of ITU-T SG13 Q17.