DFC-D: A dynamic weight-based multiple features combination for real-time moving object detection

Md Alamgir Hossain\textsuperscript{1} \textsuperscript{\Letter} - Md Imtiaz Hossain\textsuperscript{1} \textsuperscript{\Letter} - Md Delowar Hossain\textsuperscript{1} \textsuperscript{\Letter} - Eui-Nam Huh\textsuperscript{1} \textsuperscript{\Letter}

Received: 30 April 2021 / Revised: 4 January 2022 / Accepted: 25 January 2022 / Published online: 13 April 2022
© The Author(s) 2022

Abstract
Real-time moving object detection is an emerging method in Industry 5.0, that is applied in video surveillance, video coding, human-computer interaction, IoT, robotics, smart home, smart environment, edge and fog computing, cloud computing, and so on. One of the main issues is accurate moving object detection in real-time in a video with challenging background scenes. Numerous existing approaches used multiple features simultaneously to address the problem but did not consider any adaptive/dynamic weight factor to combine these feature spaces. Being inspired by these observations, we propose a background subtraction-based real-time moving object detection method, called DFC-D. This proposal determines an adaptive/dynamic weight factor to provide a weighted fusion of non-smoothing color/gray intensity and non-smoothing gradient magnitude. Moreover, the color-gradient background difference and segmentation noise are employed to modify thresholds and background samples. Our proposed solution achieves the best trade-off between detection accuracy and algorithmic complexity on the benchmark datasets while comparing with the state-of-the-art approaches.

Keywords Moving object detection · Classification · Change detection · Foreground segmentation · Background subtraction

\Letter Eui-Nam Huh
johnhuh@khu.ac.kr

Md Alamgir Hossain
alamgir@khu.ac.kr

Md Imtiaz Hossain
hossain.imtiaz@khu.ac.kr

Md Delowar Hossain
delowar@khu.ac.kr

\textsuperscript{1} Department of Computer Science and Engineering, Kyung Hee University, Global Campus Yongin, South Korea
Introduction

Moving object detection (MoD) has become an important research interest for many years. It has applications in airport monitoring, maritime monitoring, video surveillance, object tracking, human-computer interaction, and so on [12]. Moreover, at present, the real-time MoD is used as a service in cloud computing, IoT, edge computing, fog computing [24], robotics, smart environment, smart home, smart city [19], autonomous driving [25], and the like. However, MoD faces many challenges, such as illumination variation, shadow, intermittent object motion, unstable video, dynamic background, low frame rate, bad weather, pan-tilt-jitter, background clutter, occlusion, and turbulence. Because of these challenges, state-of-the-art approaches [1, 3, 6, 7, 30, 33, 34, 46] cannot detect moving foregrounds more robustly in a video with a challenging background. Furthermore, accurate detection with a low computational cost is very challenging. Therefore, existing methods [1, 14, 34, 39, 46] did not consider segmentation accuracy and computational complexity at the same time. Besides, according to Bouwmans et al. [12], the complexity reduction of background/foreground segmentation is a present research trend. Therefore, we present a background subtraction-based MoD approach, called a Dynamic weight-based multiple Feature Combination for moving object Detection (DFC-D) to increase detection accuracy and decrease the algorithmic complexity simultaneously.

There are many algorithms for multi-feature fusion detection. Pixel-based adaptive segmenter (PBAS) [16] used color feature and gradient magnitude (GM) feature like our proposal but did a weighted combination of the features by keeping a fixed weight factor for the combination. Local hybrid pattern (LHP) [20] employed color feature and edge feature (a maximum gradient response of four directional Robinson compass masks) interchangeably to detect an inner/outer edge region of objects, respectively. However, this conditional fusion and PBAS’s weighted fusion do not fuse color intensity with edge feature more adaptively. St-Charles et al. [39] and Guo et al. [14] combined the color intensity with the texture feature using a logical AND operator, called AND fusion. The AND fusion decreases true positives since all the observed features need to be dissimilar at the same time in order to be foreground. In practice, the strength of edges/textures vary in the spatial domain, even a portion of a strong edge may attenuate to zero. Additionally, the gradient magnitude/texture approaches zero because of the very low-intensity discontinuity in the spatial domain (for example, sky, water surface, etc.). However, unlike the low-intensity discontinuity region, the gradient magnitude becomes high in texture regions. Moreover, the boundary edge of an object is thicker than the other edges of the object. Therefore, the importance of all the gradient magnitudes/textures to fuse with the color/gray intensities needs not to be the same. Therefore, we newly design an adaptive feature fusion technique that keeps more weight to GM when GM is high to merge the color feature with the GM feature. In our proposal, we put objects with motion in the foreground segment and keep static background (i.e., no changes in the background) and dynamic background (i.e., changing background) in the background segment. For example, DFC-D overlooks tree leaves swinging, shadows, etc. but detects moving objects. The foregrounds detected by our proposed adaptive fusion with DFC-D and the different static fusions with DFC-D are shown in Fig. 1. During the experiment, the input frame, baseline_office #750, of CDnet-2014 dataset [47] is selected, as shown in Fig. 1. According to the figure, the proposed adaptive fusion segments the object silhouette more effectively than the static fusions.
Many methods have been proposed for background/foreground detection. Frame difference-based detection method [51] can separate the background/foreground more effectively from videos with the static background. However, the methods cannot detect moving objects more accurately in videos with the changing background, light fluctuations, shadow, occlusion, and so on. Statistical/cluster-based detection approaches [7, 22, 31, 41, 45, 53] aimed to compensate for illumination variation. Like the frame difference-based methods, the statistical/cluster-based methods cannot detect moving targets more effectively in videos with a large background variation. Artificial intelligence (AI) and deep learning-based detection methods [1, 5, 10] increased detection accuracy moderately in a very high computational cost. Besides, methods [2, 14, 16, 18, 39, 40, 42, 43] took a sample consensus policy to reduce false detection. In addition to the sample consensus policy, local binary similarity segmenter (LOBSTER) [39] proposed an intra-local binary similarity pattern (LBSP) descriptor and an inter-LBSP descriptor to reduce spatial and temporal illumination variation, respectively. Afterwards self-balanced sensitivity segmenter (SuBSENSE) [40] extended LOBSTER. Likewise, Bilodeau et al. [4] used a local binary pattern (LBP) descriptor to compensate for intensity variation. Although LBP and LBSP descriptors can reduce illumination variations and detect textures, LBP and LBSP descriptors cannot distinguish the curved edge, corner edge, and line edge, which are depicted in Fig. 2. In theory, patterns are different for the different edges. However, LBP with a $3 \times 3$ window calculates the same patterns, 131, and LBSP with a $5 \times 5$ image patch also emerges the same patterns, 6789, for the three edges. Conversely, gradient magnitudes of the curved edge, corner edge, and line edge are 165, 223, and 516, respectively. In practice, the algorithmic cost of gradient magnitude (GM) computation is less than LBP and LBSP. Therefore, we use the GM feature for the background/foreground classification.

Fig. 2 Failure instances of LBP and LBSP descriptors for the curved, corner, and line edges
In this approach, we follow a sample consensus policy for a classification decision. We take the non-smoothing (NS) color/gray intensity sequence as one input in DFC-D like Fast-D [17]. Furthermore, the NS color frames we use to generate the GM sequence that is the second input. To explain the smoothing effects, we depict the NS GM frame difference and the various types of smoothing GM frame differences in Figs. 3c, 3d, 3e, 3f. For the experiment, we choose the input frame, baseline_office #741, from CDnet-2014 dataset [47]. Moreover, we consider a $5 \times 5$ kernel in the average, Gaussian, and median filters, 1.5 standard deviation in the Gaussian filter, and 15 as the threshold to get the frame differences. It is seen that the NS GM frame difference shown in Fig. 3c keeps more edges than all the smoothing frame differences. However, the NS frame differences segment more noises removed by a median filter. A color background sample and a GM background sample are initialized from the first color sequence and the first GM frame, respectively. Unfiltered segmentation results from the similarity/dissimilarity calculated for the color, GM, and adaptive combined feature. Additionally, the static or dynamic background is determined based on the segmentation noise and the consecutive background difference. Based on this background type and segmentation noise, the threshold factor and the background pixel in the background samples are updated simultaneously. Finally, the post-processing operation is used to remove noises and fill holes in the unfiltered detection. We summarize our main contributions as follows: (1) We consider the non-smoothing gradient magnitude feature to keep intact edge information. (2) Color-gradient background difference and detection noise-based thresholds (i.e., gradient threshold and color threshold) and background samples (i.e., gradient background sample and color background sample) are updated to reduce false detection. (3) We newly design an adaptive combination that combines color/gray intensity with gradient magnitude dynamically, to detect edge information effectively. (4) Our proposed method detects moving objects more accurately in videos with many challenges, including the thermal, pan-tilt-zoom, turbulence, and dynamic background while ensuring the real-time frame rate and low memory usage. (5) We present the analysis, design, and outcome of our proposal comprehensively and compare our method with the recent methods on the three datasets.

In the remaining of the paper, we discuss the contributions, advantages, and disadvantages of related approaches briefly in Section 2. Section 3 presents our proposed DFC-D method. This section also lists the key performance metrics (KPM) used to assess detection accuracy. In Section 4, we express the ablation experiment and the experimental outcome of the smoothing GM and non-smoothing GM, advantages of dynamic thresholds and background sample update, and various feature fusions. Moreover, we present the performance, comparisons, and a summary of the overall result in this section. Conclusion and future works we brief in Section 5.

Fig. 3 The NS GM frame difference and the smoothing GM frame differences are presented. (a) Input sequence, (b) ground truth, (c) foreground by the NS GM frame difference, (d) foreground by the average smoothing GM frame difference, (e) foreground by the Gaussian smoothing GM frame difference, and (f) foreground by the median smoothing GM frame difference are depicted from the left to right.
2 Literature survey

Moving object detection approaches classified into the following four categories are shown in Fig. 4.

1. **Simple method.** A few methods in this category subtract either consecutive frames or consecutive blocks, whereas the other methods use various types of color models to detect moving objects (See Section 2.1).

2. **Cluster/statistical-based method.** The methods in this category are based on the probability distribution/cluster algorithms (Gaussian distribution, KNN, K-means, etc.) (See Section 2.2).

3. **Sample consensus-based method.** This type of method segments an observed pixel as a background pixel if a certain number of background pixels are similar to the observation (See Section 2.3).

4. **Learning-based method.** Artificial intelligence (AI), machine learning, and deep learning models are trained to classify an observation into the background/foreground. There are a few hybrid methods that merge a deep learning-based algorithm with one or more simple methods or sample consensus-based methods to improve classification accuracy (See Section 2.4).

2.1 Simple method

Wei et al. [50] compared consecutive blocks of two adjacent frames, and Panda and Meher [32] proposed a color difference histogram technique to detect the moving foreground. In the median-based method [21], if a moving object appears more than 50% along the temporal direction of the displayed frames, the object will be detected as a foreground; otherwise, it will be considered as a background. However, this technique fails when a moving object resides less than 50% in the temporal direction. Although methods [21, 32, 50, 51] require the least processing time, they show the least accuracy to classify objects in videos with small/large background variations. Sedky et al. [37] proposed a Spectral-360 classifier that used a spectral reflectance-based color descriptor to improve accuracy of the detection. However, Spectral-360 cannot detect objects effectively like the state-of-the-art methods. Moreover, according to Sedky et al. Spectral-360 is a computationally complex method so that it cannot be used in any real-time application.

2.2 Cluster/statistical-based method

Stauffer et al. [41] proposed a Gaussian mixture model (GMM) that kept a pixel in more than one Gaussian distribution and used real-time approximation to modify GMM. GMM determines the Gaussian distribution that mostly corresponds to background pixels. A pixel becomes foreground until it does not fit the background Gaussian distribution. However, GMM cannot detect an intermittent object motion and an inserted object into the scenes more accurately. Zivkovic et al. [53] used a balloon estimator approach and a uniform kernel to increase detection accuracy, called K-nearest neighbors (KNN). Although MoG2 [31] combined [53] and [52] to extend MoG approach, it cannot compensate for moderate and large intensity variation. Chen et al. [7] presented the sub-superpixel to correct intensity and further proposed sub-superpixel-based background subtraction (SBS) approach. Initially, this proposal generated superpixels from a first frame using the linear clustering algorithm. And then, SBS used a k-means algorithm to partition the superpixels into k units, called...
Fig. 4 Taxonomy of the moving object detection approach

Learning-based (Sect. 2.4)
- IMMC [33]
- RPCA/RLS [34]
- incPCP [35]
- CwisaDH [24]
- CNN [25]
- DeepBS [9]
- RB-SOM [10]
- MSFgNet [11]
- BMN-BSN [12]
- 3DCD [36]
- PgSegNet_v2 [37]

Sample Consensus-based (Sect. 2.3)
- ViBe [27]
- ViBe+ [28]
- LOBSTER [13]
- PBAS [15]
- PID [29]
- FBGS [30]
- SuBSENSE [26]
- SVDBP [14]
- M*CD [5]
- Fast-D [32]

Proposed DFC-D
- GMM [19]
- MoG2 [20]
- KNN [22]
- PTSG [23]
- SBS [6]
- SuperBE [7]
- Cuevas et al. [38]
- Berjón et al. [8]
- DP-GMM [39]
- Shuai Liu et al. [40]
- Shuai Liu et al. [41]
- Shuai Liu et al. [42]
- Wang et al. [43]
- Wang et al. [44]
- Frame Diff. [18]
- Block Diff. [45]
- Median [46]
- Spectral-360 [47]
- Histogram [48]
sub-superpixels. Furthermore, this proposed approach generated a background model of the sub-superpixels and updated based on the number of pixels in the cluster to handle ghost artifacts. Indeed, the weighting metric and sub-superpixel determined moving objects. However, in SBS, false segmentation increases when color/gray intensity varies randomly. In recent days, superpixel-level background estimation (SuperBE) [6] decreased algorithmic complexity and detection accuracy simultaneously. SuperBE constructed superpixels based on color and spatial coherency to describe the shapes of objects. This proposal used the color covariance metric and mean feature to compare a superpixel of the background model with an observed superpixel in order to segment the observation as foreground/background. SuperBE updated the background model to cope with the background changes. Cuevas et al. [8] modeled background to detect moving things. This proposal differentiated consecutive frames to calculate weighted distributions, thereby estimating kernel bandwidth. Moreover, this proposed approach employed a non-adaptive weight to update a pixel in the background model selectively. An extension of Cuevas et al. [8] was proposed by Berjón et al. [3] that used a particle filter to update a foreground pixel. However, according to Berjón et al. this method cannot provide real-time detection without a GPU. A Dirichlet process Gaussian mixture model (DP-GMM) to generate the background, a Bayesian technique to get a segmentation decision, and a new learning technique to adapt scene changes were proposed in [15]. Nevertheless, DP-GMM cannot handle illumination changes due to frequent light fluctuations. Shuai Liu et al. [27] proposed a fuzzy detection strategy to track targets in the case of complex environments. The strategy was used to conclude the tracking result. If the tracking result in the case of an observed frame is not expected enough, the stored template is employed for the tracking to keep away the template pollution. According to [27], it requires further research to improve more tracking accuracy under complex environments. To achieve robust visual monitoring, Shuai Liu et al. [28] presented a multi-layer template update strategy. This proposal made sure continuous monitoring the targets by utilizing the confidence memory, matching memory, and cognitive memory alternatively. However, this method cannot continue monitoring the targets if monitoring fails one time. Shuai Liu et al. [26] came up with a template update mechanism to enhance the accuracy of visual tracking in the case of clutter background. The proposed method stored an original template, employed both the original template and the current template, and chose the better one, whenever background clutter was identified. When there was no background clutter, the original template was used again. As stated by Shuai Liu et al. this proposal needs future research to enhance more visual tracking accuracy. Wang et al. [48] proposed a very effective method while using the context information of the spectral information for the efficient band selection. This work has a direct impact on band selection for the fast operation and execution. Wang et al. [49] proposed a novel proposal that can identify small objects from aerial scenes (sky as the background) videos. The proposed work is similar to our work in the sense of object detection.

2.3 Sample consensus-based method

Sample consensus-based BGS detects an observed pixel based on its neighbor pixels' similarities. Moreover, Barnich et al. [2] also took a simple consensus strategy to segment a pixel and used only a color feature, named visual background extractor (ViBe). ViBe initializes a background model from a single frame and updates the background model based on a blind update policy. One major drawback of the blind update policy is that detected foregrounds will never replace the background model throughout the classification. Van Droogenbroeck et al. [43] improved ViBe and named ViBe+. ViBe+ replaced the blind update strategy
of ViBe with a memoryless update strategy and considered few background samples for the classification. Hofmann et al. [16] took a sample consensus strategy and introduced a dynamic background modeling technique, called pixel-based adaptive segmenter (PBAS). PBAS used a color feature and a gradient magnitude feature simultaneously. Tiefenbacher et al. [42] improved the PBAS approach and proposed proportional-integral-derivative (PID) controllers. PID controllers are used to select a threshold factor and update a background model effectively. FBGS [18] proposed a fusion image and adjusted the dynamic thresholds. However, FBGS did not improve F1-score much on average. St-Charles et al. [39] used the LBSP pattern and the color feature in the proposed LOBSTER approach that is an improvement of ViBe. Afterwards St-Charles et al. [40] extended their LOBSTER and named SuBSENSE. Additionally, the dynamic background policy of PBAS was improved in SuBSENSE. Guo et al. [14] transformed an intensity image block into a singular value decomposition (SVD) image block, which was further represented by a local binary pattern (LBP) to develop a more illumination and shadow invariant descriptor, called SVD binary pattern (SVDBP). Finally, Guo et al. [14] replaced the LBSP descriptor of SuBSENSE with the SVDBP. In M4CD [46], chromaticity, brightness and local ternary pattern (LTP) features, Bayes rule for segmentation decision, and Markov random field-based post-processing were considered to improve detection accuracy. However, M4CD not only increases accuracy but also increases computational complexity. Fast-D [17] only took a non-smoothing color feature and adapted a color threshold. Additionally, for one channel grayscale input video, the BG/FG was detected using the similarity/dissimilarity between an observation and its corresponding background sample for the gray feature. In the case of RGB color input, the similarity/dissimilarity between an observation and its background sample was counted for any RGB channel. In DFC-D, a non-smoothing color feature and a non-smoothing gradient magnitude (GM) feature are incorporated. Moreover, when taking one channel grayscale input video, segmentation is performed based on the final similarity/dissimilarity measure that results from the logical OR operation of the three similarities/dissimilarities calculated between an observation and its background sample for the gray intensity feature, gray gradient magnitude feature, and adaptive combination of gray intensity–gray gradient feature. And in the case of three channels (i.e., RGB) color input, the final number of similarity results from a logical OR operation of the R, G, and B channel outcomes, and each R, G, and B channel outcome is calculated in the same way as the one channel grayscale sequence in DFC-D. In this proposal, one of our key contributions is the adaptive combination. The combination fuses each color/gray intensity feature with each GM feature for each pixel separately at run-time based on edge strength. Additionally, the adaptive gradient threshold and color threshold are calculated, and color and gradient background samples are updated based on the static/dynamic background scene.

2.4 Learning-based method

Farcas et al. [11] presented a subspace learning method, called incremental maximum margin criterion (IMMC). IMMC modified the eigenvalues and eigenvectors of the background model to adapt to changes. Consequently, IMMC outperformed the existing subspace learning methods. However, one of the main problems of IMMC is that it cannot initialize the background model without the ground truth. Robust principal component analysis (RPCA) and robust subspace learning (RSL) were called interchangeably in [44]. According to [44], the PCA-based learning method and RPCA-based learning method were used in the case of clean input data and outlier input data, respectively. For the background modeling,
Rodriguez et al. [35] came up with the incremental principal component pursuit (incPCP) method that processed one frame at a time instead of computing multiple frames simultaneously, thereby reducing a computational cost. However, according to the authors, incPCP would fail in the case of motion encompassing a large part of the background, a large size foreground, and an occluded background. Weightless neural network (WNN)-based CwisarDH [10] used a history of the color feature to correct intensity changes and took an adaptability technique to control the dynamic background. Nevertheless, CwisarDH cannot be trained in videos containing some challenges. DeepBS [1] used the CNN approach [5] and only replaced the frame difference technique to generate the background. In DeepBS, the background was detected by the SuBSENSE [40] and flux tensor with split Gaussian (FTSG) [45] approaches. Ramirez-Quintana et al. [34] proposed an approach, called radial basis RESOM (RBSOM) to learn background pixels and compensate for gradual illumination changes. However, RBSOM cannot detect moving foregrounds more robustly. Motion saliency foreground network (MSFgNet) [33] was proposed to detect moving foregrounds in non-real-time video. Initially, MSFgNet partitioned input video into small video streams (SVS), thereby creating a background frame for each SVS. And then, this proposal calculated saliency maps for these background frames and estimated the background model. Finally, the foreground network (FgNet) of MSFgNet determined the foreground from these saliency maps. Nevertheless, MSFgNet cannot classify input sequences captured in different weathers more effectively. Mondéjar-Guerra et al. [30] came up with BMN-BSN method that trained two nested networks together. The first network estimated background features and the second network carried out subtraction operation given the estimated background features and a target frame. However, BMN-BSN cannot classify input sequences captured with a shaky camera into the background/foreground more accurately. 3DCD [29] is a supervised method that is proposed for moving target detection in the case of scene independent videos. 3DCD adopted the end-to-end 3D-CNN-based network containing multiple spatiotemporal units. Nevertheless, the supervised method requires a large dataset to increase detection performance. FgSegNet_v2 [23] is an end-to-end trainable method that can extract multiple features and fuse them. The proposed network can take multi-scale inputs. However, FgSegNet_v2 needs much data to train to get more accurate segmentation results.

3 Proposed DFC-D

We propose the DFC-D method that is depicted in both mathematical and technical forms. The proposed DFC-D model is shown in Fig. 5. DFC-D is a pixel-by-pixel moving target detection method. To maintain the flow and describe the end-to-end processing path accordingly, we discuss those sub-topics deeply through prior studies. Furthermore, to address the approaches and discuss their relevance (e.g., input processing, multiple features such as gradient magnitudes frames and their generations, background samples principles, segmentation error, and so on), we describe them concisely. Therefore, from input sequences to post-processing, we adopt some established approaches, hyperparameters, and experimental arrangements. The hyperparameters and symbols used in our proposal we list in Table 1. The pixel-level parameters and frame-level parameters we represent by the lowercase and uppercase letter, respectively. The values of the hyperparameters are set experimentally. Moreover, the subscript $t$ and $t + 1$ indicate a present/observed state and next state, respectively. Furthermore, for the gray intensity input, one channel calculation is carried out and
for the color input sequences, three channels are considered in the calculation. Our DFC-D method has the following nine process steps:

1. **Input sequences.** DFC-D takes non-smoothing RGB/grayscale input frames/videos (see Section 3.1).

2. **Gradient magnitude frames generation.** Gradient image frames are computed for the non-smoothing RGB/grayscale input frames/videos and then normalized. After that, the normalized gradient sequences are fed to our model in addition to the intensity sequences (see Section 3.2).

3. **Background samples.** Two background samples are generated: the color background sample is initialized from the first intensity sequence; the gradient background sample is also initialized from the first normalized gradient frame. Background pixels in a background sample are the neighbors of an observed pixel (see Section 3.3).

4. **Classification: Adaptive combination.** In classification, we describe how to classify an observed pixel into the background/foreground pixel. An observed pixel of an observed frame is segmented based on the correlation with its corresponding background pixels in the background samples. In addition, we describe the proposed color-gradient combination process in detail (see Section 3.4).

5. **Segmentation error.** It estimates the amount of classification error based on the salt and pepper noises in the classification. The amount of error helps to modify the background factor and the threshold factor simultaneously (see Section 3.5).

6. **Background factor and threshold factor update.** The background factor and the threshold factor result from the minimum distance between the consecutive backgrounds of consecutive frames and the amount of detection error (see Section 3.6).

7. **Color threshold and gradient threshold update.** The color threshold and the gradient magnitude threshold are determined and modified dynamically based on the threshold factor and static/dynamic background to reduce false detection due to intensity changes over time (see Section 3.7).

8. **Background sample update.** The intensity of input sequences changes over time; therefore, it needs to modify the background samples like the thresholds. The background

---

**Fig. 5** The proposed DFC-D model
Table 1 Description of parameters, values, and notations

| Parameter | Value | Description |
|-----------|-------|-------------|
| $x$       | -     | The row coordinate of each pixel |
| $y$       | -     | The column coordinate of each pixel |
| $I$       | -     | The color/gray frame |
| $G$       | -     | The gradient magnitude frame |
| $\mathcal{P}$ | $\{i, g\}$ | The color/gray and gradient pixels |
| $p$       | $p \in \mathcal{P}$ | The member of set, $\mathcal{P}$ |
| $\mathcal{C}$ | $\{R, G, B\}$ | The red, green, and blue color channels |
| $c$       | $c \in \mathcal{C}$ | The member of set, $\mathcal{C}$ |
| $b^s(x, y)$ | - | The gradient magnitude background sample |
| $b^l(x, y)$ | - | The color/gray background sample |
| $|.|$ | - | The L1 distance |
| $F(x, y)$ | - | The unfiltered segmentation frame |
| $F^f(x, y)$ | - | The filtered segmentation frame |
| $\times$ | - | The multiplication sign |
| $\alpha$ | - | The fusion weight factor |
| $l(x, y)$ | - | The blinking pixel |
| $n(x, y)$ | - | The segmentation error, initially, $n(x, y) = 0$ |
| $h^l(x, y)$ | - | The color/gray threshold |
| $h^s(x, y)$ | - | The threshold for gradient magnitude |
| $s(x, y)$ | - | The static/changing background region |
| $r(x, y)$ | - | The threshold factor, at first, $r(x, y) = 1$ |
| $a^m(x, y)$ | - | The minimum moving average distance between observation and its background samples |
| min(.) | - | The function resulting a minimum value |
| max(.) | - | The function calculating a maximum value |
| $o^{bf}(x, y)$ | - | The average number of BG-FG |
| $o^f(x, y)$ | - | The average number of foreground |
| $M_{min}$ | 2 | The required similarity to separate the BG/FG |
| $t(x, y)$ | $[2, 256]$ | The background factor, at first, $t(x, y) = 2$ |
| $N_{in}$ | 1 | The amount of noise increment |
| $N_{dr}$ | 0.1 | The amount of noise decrement |
| $R_m$ | 0.1 | The amount of minimum unstable ratio |
| $\beta$ | 50 | The moving average factor |
| $Q$ | 255 | The maximum pixel intensity |
| $R_s$ | 0.01 | The constant to modify the threshold factor |
| $T_{in}$ | 0.50 | The constant to increase the background factor |
| $T_{dr}$ | 0.25 | The constant to decrease the background factor |
| $M^I$ | 30 | The minimum initial intensity threshold |
| $O^I$ | $M^I/5$ | The intensity threshold offset |
| $M^G$ | 26 | The minimum initial gradient threshold |
| $O^G$ | $M^G/5$ | The gradient threshold offset |
| $N$ | 49 | The number of pixels in a background sample |
samples are updated by the segmentation pixels depending on the probability of background sample update and the segmentation error in the unfiltered segmentation (see Section 3.8).

9. **Post-processing.** This operation removes the blinking noises and fills the holes of the target silhouettes of the unfiltered segmentation sequence (see Section 3.9).

### 3.1 Input sequences

RGB/gray sequences, \( I_c, c \in \{R, G, B\} \), are the inputs in our proposed model. According to Sajid et al. [36], RGB color space does not take any extra color conversion cost and is more robust for object detection in indoor and outdoor videos on average; therefore, we use the RGB color model. Moreover, we do not filter the input sequences as smoothing functions (e.g., Gaussian smoothing, median smoothing, and average smoothing) distort essential change information in addition to noise reduction. However, non-smoothing \( I_c \) increases noises in the unfiltered segmentation which is filtered by the post-processing.

### 3.2 Gradient magnitude frames generation

Each gradient image frame, \( G_c, c \in \{R, G, B\} \), is computed for each non-smoothing RGB/gray intensity frame, \( I_c \). We discussed earlier that we do not smooth \( I_c \) to get intact gradient magnitude that is very important for accurate change detection. Sobel horizontal kernel, \( K_h \), as seen in Eq. 1 and Sobel vertical kernel, \( K_v \), as expressed in Eq. 2 are used to convolve with \( I_c \) to calculate the gradient magnitude frames in horizontal direction, \( H_c, c \in \{R, G, B\} \), as seen in Eq. 3 and vertical direction, \( V_c, c \in \{R, G, B\} \), as presented in Eq. 4, respectively. In these equations, \( * \) is the convolution sign.

\[
K_h = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}
\]  

(1)

\[
K_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}
\]  

(2)

\[
H_c = K_h * I_c
\]  

(3)

\[
V_c = K_v * I_c
\]  

(4)

After that, each non-normalized image gradient pixel, \( m_c, c \in \{R, G, B\} \), of each non-normalized gradient magnitude frame, \( M_c, c \in \{R, G, B\} \), is calculated as

\[
m_c = \sqrt{h_c^2 + v_c^2},
\]  

(5)

where \( h_c, c \in \{R, G, B\} \), and \( v_c, c \in \{R, G, B\} \), are the non-normalized gradient pixels of \( H_c \) and \( V_c \), respectively. Then, \( M_c \) is normalized so that each gradient pixel has a magnitude between 0 and 255. Normalized gradient magnitude frame, \( G_c, c \in \{R, G, B\} \), is calculated from \( M_c \) as follows: each \( m_c \) is multiplied by 255 followed by dividing by maximum image gradient of \( M_c \), \( \max(M_c) \), to get each \( g_c, c \in \{R, G, B\} \), of \( G_c \) as

\[
g_c = \frac{255 \times m_c}{\max(M_c)}.
\]  

(6)
Because of this normalization, for each channel, one-byte of memory space is required to keep each pixel, $g_c$, of $G_c$. Alternatively, we created a normalized gradient frame, $G_c$, by using the Robinson compass mask in eight directions. However, rather than increasing accuracy, the $G_c$-based DFC-D method incurred extra time complexity. Therefore, we only use Sobel mask to compute $G_c$.

### 3.3 Background samples

If $I_c$ and $G_c$ are the first frames, color/gray intensity background sample frame, $B^I_c(x, y), c \in \{R, G, B\}$, and GM background sample frame, $B^G_c(x, y), c \in \{R, G, B\}$, are initialized. Let us denote $b^i_c(x, y), c \in \{R, G, B\}$, and $b^g_c(x, y), c \in \{R, G, B\}$, are the two background samples of $B^I_c(x, y)$ and $B^G_c(x, y)$, respectively, at a pixel with coordinates $(x, y)$. Therefore, color/gray intensity background sample, $b^i_c(x, y)$, and gradient background sample, $b^g_c(x, y)$, are represented as

$$b^i_c(x, y) = \{b^i_{c,0}(x, y), b^i_{c,1}(x, y) \ldots b^i_{c,N-1}(x, y)\},$$

$$b^g_c(x, y) = \{b^g_{c,0}(x, y), b^g_{c,1}(x, y) \ldots b^g_{c,N-1}(x, y)\},$$

where $c \in \{R, G, B\}$ and 0, 1, 2, 3, ..., $N-1$ are the index of the background samples. Each element in the background samples is called a background element or a background pixel. For example, $b^i_{c,0}(x, y)$ is a background pixel of the background sample, $b^i_c(x, y)$. $N$ is the number of background element in a background sample, that is chosen arbitrarily from the neighbors at a pixel with coordinates $(x, y)$. The arbitrarily chosen background elements are buffered consecutively in 0, 1, 2, 3, ..., $N-1$th positions of the background samples.

### 3.4 Classification: Adaptive combination

After $B^I_c(x, y)$ and $B^G_c(x, y)$ initialization from the first frames, background/foreground segmentation starts with the first frames. An observed pixel with coordinates $(x, y)$ is primarily classified as unfiltered segmentation pixel, $f(x, y)$, of unfiltered segmentation frame, $F(x, y)$, as

$$f(x, y) = \begin{cases} 
1, & \text{if } \sum_{j=0}^{N-1} m_j(x, y) < M_{\min}, \\
0, & \text{otherwise},
\end{cases}$$

where 1 represents the foreground, 0 denotes the background, $M_{\min}$ is the required number of similarity to determine the BG and FG, and $m_j(x, y)$ is the counter that counts the number of similarity, as

$$m_j(x, y) = \begin{cases} 
0, & \text{if } \bigvee_{p \in \{i, g\}} (d_{c,j}^p(x, y) > h^p(x, y)) \lor (d_{c,j}^f(x, y) > h^g(x, y)), \exists c \in \mathcal{C}, \\
1, & \text{otherwise},
\end{cases}$$

where 0 indicates dissimilarity (i.e., change), 1 represents similarity (i.e., no change), $\lor$ is the logical OR operator, and $h^p(x, y), p \in \{i, g\}$, is the threshold that consists of color/gray intensity threshold, $h^i(x, y)$, and gradient threshold, $h^g(x, y)$. Additionally, $d_{c,j}^p(x, y), p \in \{i, g\}, c \in \{R, G, B\}, j \in [0, N - 1]$, is the similarity distance that consists of color/gray intensity similarity distance, $d_{c,j}^i(x, y)$, and gradient similarity distance, $d_{c,j}^g(x, y)$. This $d_{c,j}^i(x, y)$ results from the L1 norm between observation
(i.e., \( p_c(x, y), p \in \{ i, g \}, c \in \{ R, G, B \} \)) and its corresponding background sample 
(i.e., \( b^p_{c,j}(x, y), p \in \{ i, g \}, c \in \{ R, G, B \}, j \in [0, N - 1] \)), that is estimated as

\[
d^{p}_{c,j}(x, y) = |p_c(x, y) - b^p_{c,j}(x, y)|.
\]

In Eq. 10, \( d^{f}_{c,j}(x, y) \) is the adaptive fusion of \( d^{i}_{c,j}(x, y) \) and \( d^{g}_{c,j}(x, y) \). The adaptive combination is explained in below.

### 3.4.1 Adaptive combination

Our key idea is adaptive combination, \( d^{f}_{c,j}(x, y), c \in \{ R, G, B \}, j \in [0, N - 1] \), that is computed by taking a linear combination as

\[
d^{f}_{c,j}(x, y) = (1 - \alpha_c) \times d^{i}_{c,j}(x, y) + \alpha_c \times d^{g}_{c,j}(x, y),
\]

where \( \alpha_c, c \in \{ R, G, B \} \), is the combined weight factor, which is dynamically determined as follows:

\[
\alpha_c = \frac{g_c(x, y)}{|\max(G_c)|},
\]

where \( \max(G_c), c \in \{ R, G, B \} \), is the absolute maximum magnitude and \( g_c(x, y), c \in \{ R, G, B \} \), is the observed gradient magnitude of \( G_c \). The fusion factor, \( \alpha_c \), is calculated separately for each pixel of \( G_c \). In theory, the edges of an image are not distributed uniformly and the low gradient magnitude is erroneous. Hence, according to our proposed fusion, as expressed in Eq. 12, for the object’s boundary edge (i.e., for the high gradient value), the gradient information is weighted more than the color/gray intensity (i.e., \( \alpha_c \) becomes large); otherwise, for the rest of the object parts (i.e., for the weak gradient value), the gradient magnitude is weighted less than the color/gray intensity information (i.e., \( \alpha_c \) becomes small). However, in the static fusion, fusion weight factor, \( \alpha_c \), is set manually and it is constant throughout the segmentation.

### 3.5 Segmentation error

Like [18, 40], we calculate segmentation error, \( n(x, y) \), that indicates the classification error in \( f(x, y) \). In our proposal, \( n(x, y) \) is estimated by counting blinking pixel, \( l(x, y) \), that is the boolean variable (i.e., either true or false). In other words, \( l(x, y) \) is the segmentation error pixel that resides either in the present segmentation sequence or in the previous segmentation sequence, but does not exist in both sequences. If we directly do exclusive OR operation between observed unfiltered foreground, \( f_t(x, y) \), and previous post-processed foreground, \( f_{t-1}^f(x, y) \), it produces \( l(x, y) \) because of the brink edge of foreground objects in \( f_t(x, y) \). Therefore, we do exclusive OR operation between the dilated version of \( f_t(x, y) \) and \( f_{t-1}^f(x, y) \). Finally, \( n(x, y) \) is estimated at run-time as

\[
n_{t+1}(x, y) = \begin{cases} n_t(x, y) + N_{in}, & \text{if } (a^m(x, y) > R_m) \land l_t(x, y), \\ \max(n_t(x, y) - N_{dr}/2, N_{dr}), & \text{otherwise,} \end{cases}
\]

where \( n_t(x, y) \) is the current segmentation error, \( n_{t+1}(x, y) \) is the updated segmentation error, \( l_t(x, y) \) is the current state of \( l(x, y) \), \( N_{in} \) is the noise increment step size, \( N_{dr} \) is the noise decrement step size, and \( R_m \) is the minimum unstable ratio. Moreover, \( a^m(x, y) \) represents the present minimum moving average distance that is the normalized minimal L1 distance between observed pixels with coordinates \( (x, y) \) and their corresponding background samples over last \( \beta \) number of frames. We estimate \( a^m(x, y) \) to calculate
\( n(x,y) \), threshold factor, \( r(x,y) \), and background factor, \( t(x,y) \). Minimum moving average distance, \( a^m(x,y) \), is calculated as

\[
a^m_{t+1}(x,y) = (1 - \beta) \times a^m_t(x,y) + \beta \times k_t(x,y),
\]

where \( \beta \) represents the moving average factor, and \( a^m_{t+1}(x,y) \) and \( a^m_t(x,y) \) are the updated and current least moving average distances, respectively. Moreover, \( k_t(x,y) \) is the present state of \( k(x,y) \), that is the normalized least distance between color-gradient pixels and their corresponding background samples. Normalized minimum distance, \( k(x,y) \), is calculated as

\[
k(x,y) = \frac{1}{\mathcal{P}^\mathcal{C}} \sum_{p \in \mathcal{P}} \sum_{c \in \mathcal{C}} \min_{j \in [0, N - 1]} (|p_c(x,y) - b^p_{c,j}(x,y)|),
\]

where \( |\mathcal{P}| \) and \( |\mathcal{C}| \) are the cardinality of sets, \( \mathcal{P} \) and \( \mathcal{C} \), respectively, \( Q \) represents the maximal pixel intensity. Additionally, \( \min(|p_c(x,y) - b^p_{c,j}(x,y)|) \) is the least absolute distance between observation, \( p_c(x,y), p \in \{i, g\}, c \in \{R, G, B\} \), and its corresponding background samples, \( b^p_{c,j}(x,y), p \in \{i, g\}, c \in \{R, G, B\}, j \in [0, N - 1] \).

### 3.6 Background factor and threshold factor update

We calculate background factor, \( t(x,y) \), and threshold factor, \( r(x,y) \), like the methods, \([17, 40]\). In our proposed method, \( r(x,y) \) is estimated and modified at frame level as

\[
r_{t+1}(x,y) = \begin{cases} r_t(x,y) + R_t \times n_t(x,y), & \text{if } r_t(x,y) \leq (1 + 2 \times a^m_t(x,y))^2, \\ \max(1, r_t(x,y) - \frac{R_t}{n_t(x,y)}), & \text{otherwise}, \end{cases}
\]

where \( r_t(x,y) \) is the current threshold factor, \( r_{t+1}(x,y) \) is the updated threshold factor, and \( R_t \) represents the incremental/decremental step size of \( r(x,y) \). Moreover, \( r(x,y) \) is initialized to 1 and according to the equation it is \( \geq 1 \) all the time.

Additionally, background factor, \( t(x,y) \), is computed and updated at frame level, as

\[
t_{t+1}(x,y) = \begin{cases} \min\left(256, t_t(x,y) + \frac{T_{in}}{n_t(x,y) \times a^m_t(x,y)}\right), & \text{if } f_{t-1}(x,y) = 1 \lor \left((a^m_t(x,y) < R_m) \land f_{t-1}(x,y) = 1\right), \\ \max\left(2, t_t(x,y) - \frac{T_{dr} \times n_t(x,y)}{a^m_t(x,y)}\right), & \text{otherwise}, \end{cases}
\]

where \( t_t(x,y) \) is the current background factor, \( t_{t+1}(x,y) \) is the modified background factor, \( T_{in} \) represents the incremental step size of \( t(x,y) \), \( T_{dr} \) represents the decremental step size of \( t(x,y) \), \( \lor \) is the logical OR operator, \( \land \) indicates the logical AND operator, \( t(x,y) \) is initialized to 2, and the value of \( t(x,y) \) resides within a range of \([2, 256]\). Indeed, \( r(x,y) \) and \( t(x,y) \) we use to modify thresholds and background samples to adapt to the pixel intensity changes over time.
3.7 Color threshold and gradient threshold update

Color/gray intensity threshold, \( h^I(x, y) \), and gradient threshold, \( h^G(x, y) \), are used for classification decision. Thresholds, \( h^I(x, y) \) and \( h^G(x, y) \) are calculated as

\[
h^I(x, y) = \left( M^I \times r_I(x, y) - s_I(x, y) \times O^I \right) / 2 \tag{19}
\]
\[
h^G(x, y) = \left( M^G \times r_I(x, y) - s_I(x, y) \times O^G \right) / 2, \tag{20}
\]

where \( M^I \) represents the minimum color/gray intensity, \( M^G \) is the minimum gradient magnitude, \( O^I \) is the color/gray intensity offset, and \( O^G \) is the gradient magnitude offset. Moreover, \( r_I(x, y) \) is the current threshold factor, \( s_I(x, y) \) is the present state of \( s(x, y) \) that indicates the stability (i.e., \( s(x, y) = 1 \)) or instability (i.e., \( s(x, y) = 0 \)) of background, calculated as follows. When a pixel with coordinates \((x, y)\) is classified as a foreground pixel, the average number of occurrence of background and foreground, \( o^b(x, y) \), is estimated as

\[
o^b_{t+1}(x, y) = (1 - \beta) \times o^b_t(x, y) + \beta, \tag{21}
\]

where \( o^b(x, y) \) is set to 0 at the initial state, \( \beta \) is the moving average factor, and \( o^b_{t+1}(x, y) \) and \( o^b_t(x, y) \) are the updated and present average number of background-foreground, respectively. If a pixel with coordinates \((x, y)\) is segmented as a background pixel, \( o^b(x, y) \) is counted as

\[
o^b_{t+1}(x, y) = (1 - \beta) \times o^b_t(x, y), \tag{22}
\]

Therefore, according to Eqs. 21 and 22, a high value of \( o^b(x, y) \) indicates more foregronds and a low value of \( o^b(x, y) \) refers to the less number of foregronds.

After that, the only occurrence of foreground, \( o^f(x, y) \), is enumerated as

\[
o^f_{t+1}(x, y) = (1 - \beta) \times o^f_t(x, y) + \beta \times f^f_{t-1}(x, y), \tag{23}
\]

where \( o^f_{t+1}(x, y) \) and \( o^f_t(x, y) \) are the modified and current amount of the foreground pixel, respectively, and \( f^f_{t-1}(x, y) \) is the filtered foreground pixel of previous filtered foreground frame, \( F^F_{t-1}(x, y) \).

Finally, static/dynamic background indicator, \( s(x, y) \), is computed as

\[
s(x, y) = \begin{cases} 
0, & \text{if } (r_I(x, y) > R_m) \lor \left( (o^b^f_t(x, y) - o^f_t(x, y)) > R_m \right), \\
1, & \text{otherwise},
\end{cases} \tag{24}
\]

where \( R_m \) is the minimum stable ratio. If \( r_I(x, y) \) or the difference between \( o^b^f_t(x, y) \) and \( o^f_t(x, y) \) is greater than \( R_m \), \( s(x, y) = 0 \); otherwise, \( s(x, y) = 1 \). Here, \( s(x, y) = 0 \) and \( s(x, y) = 1 \) refers to the unstable region (i.e., dynamic region) and the stable region (i.e., static region), respectively.

3.8 Background samples update

Background samples, \( b^I_c(x, y) \) and \( b^G_c(x, y) \), need to be replaced to adapt to the change due to dynamic background, camera jitters, light fluctuation, and so on. Conversely, for a static background, the background samples need not be updated to reduce false negatives. In practice, a background pixel in a background sample is replaced by a segmentation pixel (i.e., a foreground/background pixel). According to ViBe [2], there are two update policies: the
conservative policy, a foreground will never be a background pixel throughout the classification; the blind update policy, a segmentation foreground or a background pixel could be a background element. The conservative update policy may fail when wrongly classified moving objects are not considered to replace background elements to reduce false classification throughout the segmentation. Therefore, we select a blind update policy. However, the blind update policy may also fail because of the assimilation of slow-moving objects or intermittent-moving objects into the background samples. To address this problem, we use background factor, \( t(x, y) \), and threshold factor, \( r(x, y) \) like [17, 40]. Indeed, to reduce false negatives, \( t(x, y) \) is increased in a stationary background region, which in turn reduces the probability of background update, \( e(x, y) = 1/t(x, y) \). Alternatively, for a dynamic region, \( t(x, y) \) is decreased rapidly to assimilate the changes into background samples quickly and \( r(x, y) \) is increased sharply, both in turn lead to reducing false positives. Therefore, in our proposal, according to the blind update policy, the random \( ^{th} \) background elements (i.e., one background element in a color/gray background sample and one background element in a gradient background sample) in a color/gray background sample, \( b^g_c(x, y) \), and a gradient background sample, \( b^g_g(x, y) \), are updated by current pixels, \( i_c(x, y) \) and \( g_c(x, y) \), respectively, whether \( i_c(x, y) \) and \( g_c(x, y) \) are segmented as a foreground or a background (i.e., either \( f(x, y) = 1 \) or \( f(x, y) = 0 \)). However, this update is only carried out with probability, \( e(x, y) = 1/t(x, y) \), otherwise no update is performed at all.

3.9 Post-processing

A dilation operation is carried out on unfiltered segmentation, \( F(x, y) \), that is followed by an erosion operation to reduce noise and join broken parts of the detected objects. Then, the flood fill function is used to fill miss-detected regions in the detected silhouettes. In practice, a region can be filled without error if all the boundaries of the object holes in the segmentation are detected successfully. The flood fill operation in our proposed method can fill the miss-detected regions more accurately because of two main reasons: our proposed adaptive fusion detects the boundary of the object silhouette more effectively since the object boundary is weighted more during the fusion, as seen in Eq. 12; proposed DFC-D detects edges more robustly because of non-smooth intensity and non-smooth gradient magnitude. Finally, we compensate for salt and pepper noise using a spatial median filter of \( 9 \times 9 \) kernel to get filtered foreground, \( F^F(x, y) \).

3.10 Datasets

We use three datasets namely CDnet-2012 [13], CDnet-2014 [47], CMD [38], and LASIESTA [9] to compare our proposed method with the state-of-the-art methods. CDnet-2012 dataset has 31 videos of 6 categories which are baseline (BL), intermittent object motion (IM), shadow (SH), camera jitters (CJ), dynamic background (DB), and thermal (TH). Moreover, CDnet-2014 [47] is the extension of CDnet-2012. In addition to the six video categories of CDnet-2012, CDnet-2014 dataset has five more video categories including bad weather (BW), low frame rate (LF), night video (NV), pan-tilt-zoom (PTZ), and turbulence (TB). Additionally, CMD [38] has a video sequence of 500 frames captured by a camera with a small amount of jitter. Furthermore, LASIESTA [9] dataset includes indoor video categories such as bootstrap (I_BS), camouflage (I_CA), modified background (I_MB), illumination changes (I_IL), simple sequences (I_SI), and occlusions (I_OC). It also has outdoor video categories such as sunny conditions (O_SU), rainy conditions (O_RA), cloudy conditions (O_CL), and snowy conditions (O_SN). All the videos with a resolution of \( 352 \times 288 \)
have many challenges like partial and total occlusion, dynamic background, soft and hard shadows, camouflage, and sudden global illumination changes.

### 3.11 Key performance metrics

We use a key performance metric (KPM) to evaluate the accuracy of our proposed method. According to [13, 47], false positive rate (FPR), recall (Re), percentage of bad classification (PWC/PBC), F1-score (F1)/F-measure (FM), specificity (Sp), precision (Pr), and false-negative rate (FNR) are called KPM defined as

\[
\text{FPR} = \frac{FP}{FP + TN}, \quad (25)
\]

\[
\text{Re} = \frac{TP}{TP + FN}, \quad (26)
\]

\[
\text{PWC/PBC} = 100 \times \frac{(FN + FP)}{(TP + FN + FP + TN)}, \quad (27)
\]

\[
\text{F1-score/FM} = \frac{(2 \times Pr \times Re)}{Pr + Re}, \quad (28)
\]

\[
\text{Sp} = \frac{TN}{FP + TN}, \quad (29)
\]

\[
\text{Pr} = \frac{TP}{TP + FP}, \quad (30)
\]

\[
\text{FNR} = \frac{FN}{FP + TN}, \quad (31)
\]

where TP, TN, FP, and FN represents the correct foreground, correct background, incorrect foreground, and incorrect background, respectively. Re, Pr, Sp, and F1-score with a large value and FNR, PWC, and FPR with a small value are better. As mentioned by [13, 47] F1-score is more consistent than other metrics to evaluate the background/foreground detection. Therefore, we consider F1-score for the performance comparisons.

### 4 Experimental evaluation and discussion

This section includes (1) parameter initialization (see Section 4.1), (2) ablation experiment (see Section 4.2), (3) non-smoothing GM versus smoothing GM (see Section 4.3), (4) with the adaptive threshold versus without the adaptive threshold (see Section 4.4), (5) the effect of background sample update (see Section 4.5), (6) adaptive combination versus static combination (see Section 4.6), (7) experiment on four datasets (see Section 4.7), (8) comparisons on CDnet-2012, CDnet-2014, LASIESTA, and CMD datasets (see Section 4.8), (9) visual comparisons (see Section 4.9), (10) processing speed and memory usage comparisons (see Section 4.10), and (11) trade-off between detection accuracy and complexity (see Section 4.11). In this subsection, we present the results and analysis. Note that the bold text style represents the best score in the table.

#### 4.1 Parameter initialization

In our proposed method, we set the optimum value of the hyper-parameter for which we achieve the best F1-score. According to Fig. 6 we find the best F1-score when the value
of $M_{\text{min}}$ is 2. Therefore, the optimal value of $M_{\text{min}}$ is assigned to 2. Moreover, we get the best F1-score at threshold factor modifier, $R_s = 0.01$, as shown in Fig. 7. Therefore, we take $R_s = 0.01$. Threshold factor, $r(x, y)$, is initialized to 1 like [39]. The range of background factor, $t(x, y)$, is assigned to $[2, 256]$ like [40]. Background factor increment constant, $T_{in} = 0.50$, and background factor decrement constant, $T_{dr} = 0.25$, are chosen, the same as [17]. Furthermore, we use a $9 \times 9$ spatial median filter like PBAS [16] that also uses a $9 \times 9$ spatial median filter. Like our proposal, PBAS also took the color and image gradient features. However, we experimented to set the optimal kernel size of the spatial median filter. The experimental result is shown in Fig. 8. According to the figure, the best detection accuracy is observed for the $9 \times 9$ kernel of the filter. Therefore, we decide to set a $9 \times 9$ spatial median filter in all the experiments. Finally, the rest of the parameter in our proposed method is set based on the best F1-score.

![Figure 6](image-url)

**Fig. 6** F1-scores are plotted in the y-axis for the various values of $M_{\text{min}}$ in the x-axis. The red color indicates the best accuracy for the value of $M_{\text{min}}$.
4.2 Ablation experiment

The multiple feature fusion strategy has a great impact on accurate object detection. However, a fusion of too many features can decrease the performance of object detection (see Table 2). We carried out three ablations to analyze the effect of multiple features in Table 2. In this study, we show moving target detection accuracy in terms of F1-score for color, gradient magnitude (GM), and local binary pattern (LBP) features on CDnet-2012 [13] and CDnet-2014 [47] datasets. In Table 2, we observe the best detection accuracy while considering color and GM features simultaneously. The object detection performance decreases while using color, GM, and LBP features at the same time. This happens because these multiple features detect more true positives but generate more false positives. Therefore, based on the experiment, we take color and GM features in our proposed method.

4.3 Non-smoothing GM versus smoothing GM

Accuracy due to smoothing GM and non-smoothing GM inputs is shown in Table 3 in terms of F-measure. In this table, we observe that the non-smoothing GM input sequences

| Features                  | CDnet-2012 | CDnet-2014 |
|---------------------------|------------|------------|
| Color GM^1 LBP^2          | 0.8133     | 0.7247     |
| ✓                         | 0.8275     | 0.7347     |
| ✓ ✓                       | 0.7889     | 0.6995     |

^1GM=Gradient Magnitude
^2LBP=Local Binary Pattern
Table 3 Performance comparisons between smoothing GM and non-smoothing GM input sequences in terms of F1-score

| Function                          | F1-score  |
|----------------------------------|-----------|
|                                  | CDnet-2012 | CDnet-2014 |
| Gaussian filter smoothing GM     | 0.8018    | 0.7178     |
| Median filter smoothing GM       | 0.8135    | 0.7286     |
| Average filter smoothing GM      | 0.7890    | 0.7073     |
| Non-smoothing GM                 | **0.8275**| **0.7344** |

obtain 2.57%, 1.40%, and 3.85% more F-measures on CDnet-2012 dataset and 1.66%, 0.58%, and 2.71% more F1-scores on CDnet-2014 dataset than the Gaussian smoothing, median smoothing, and average smoothing input sequences, respectively. Because the non-smoothing GM input sequences keep all the changes, detection accuracy increases.

4.4 With the adaptive threshold versus without the adaptive threshold

To observe the effect of the adaptive threshold, we experiment with the adaptive threshold and without the adaptive threshold on CDnet-2012 and CDnet-2014 datasets. The resulted accuracy is shown in Table 4 in terms of F1-score. It is seen from the table that, while using the adaptive threshold it achieves 5.69% more F1-score on CDnet-2012 dataset and obtains 3.31% more F1-score on CDnet-2014 dataset compared to not using the adaptive threshold. In conclusion, according to the experiment, the adaptive threshold is effective in our proposed DFC-D.

4.5 The effect of background sample update

The pixel in the background sample is replaced by the segmentation pixel throughout the segmentation based on the background factor, as in Eq. 18. We experiment with and without the background sample update. The results are presented in Table 5. We observe from the figure that F1-scores increase 10.59% and 15.13% on CDnet-2012 and CDnet-2014 datasets, respectively, because of the background sample replacement.

4.6 Adaptive combination versus static combination

We experiment with the various static weighted combinations, static combination of PBAS [16], conditional combination of LHP [20], logical AND combination used in LOBSTER [39] and SVDBP [14], and our proposed dynamic combination on CDnet-2012 and CDnet-2014 datasets. The accuracies are shown in Table 6 in terms of F1-score. During the

Table 4 Without threshold factor versus with adaptive threshold factor

| Function                | F1-score  |
|-------------------------|-----------|
|                         | CDnet-2012 | CDnet-2014 |
| Without adaptive threshold | 0.7706    | 0.7013     |
| **With adaptive threshold** | **0.8275**| **0.7344** |
experiment with all the combinations, color/gray threshold, $h^l(x, y)$, and gradient magnitude threshold, $h^g(x, y)$, were set up the same as our proposed method. To compare with the conditional combination of LHP [20], we implemented the conditional combination of LHP. Moreover, we developed the static fusion of PBAS according to PBAS and the logical AND operator-based fusion like LOBSTER [39] and SVDBP [14].

According to the average F1-scores in Table 6 our proposed combination outperforms all the combinations. It happens because our proposed adaptive combination gets more weight on GM whenever it becomes high and gets less weight on GM if it becomes low. However, the proposed adaptive fusion shows better accuracy on CDnet-2012 dataset than on CDnet-2014 dataset. The reason is that GM shares neighbor information that is affected by intensity changes, large camera movement, and the like more in CDnet-2014 dataset than in CDnet-2012 dataset. Except for 20% color + 80% GM and 30% color + 70% GM combinations, all the static weighted combinations show better F1-scores than the PBAS's fusion, LHP's fusion, and logical AND fusion used in LOBSTER and SVDBP. Moreover, among the various static weighted combination, a more weighted color combination gains better accuracy than a less weighted color combination.

**Table 5** An accuracy comparison without the update and with the update of background sample in terms of FM

| Function                              | F1-score   |
|---------------------------------------|------------|
|                                       | CDnet-2012 | CDnet-2014 |
| Without background sample update      | 0.7216     | 0.5831     |
| With background sample update         | **0.8275** | **0.7344** |

**Table 6** F1-score-based comparisons of the static combination and proposed adaptive combination of color and gradient magnitude (GM)

| Function                              | F1-score   |
|---------------------------------------|------------|
|                                       | CD.-2012   | CD.-2014   |
| 50% Color + 50% GM                    | 0.7781     | 0.7099     |
| 20% Color + 80% GM                    | 0.6521     | 0.6070     |
| 80% Color + 20% GM                    | 0.7819     | 0.7107     |
| 40% Color + 60% GM                    | 0.7610     | 0.6960     |
| 60% Color + 40% GM                    | 0.7836     | 0.7131     |
| 30% Color + 70% GM                    | 0.7274     | 0.6679     |
| 70% Color + 30% GM                    | 0.7879     | 0.7151     |
| Conditional combination of LHP        | 0.7248     | 0.6778     |
| Static combination of PBAS            | 0.7146     | 0.6771     |
| Logical AND fusion (LOB., SVDBP)      | 0.1035     | 0.1145     |
| **Proposed adaptive combination**     | **0.7989** | **0.7181** |
## Table 7  key performance metric (KPM)-based performance of our proposed DFC-D Method on CDnet-2012, CDnet-2014, and CMD datasets

| Category | Sp  | FPR | FNR | PWC | Re  | Pr  | F1  |
|----------|-----|-----|-----|-----|-----|-----|-----|
| BL       | 0.9971 | 0.0029 | 0.0416 | 0.4129 | 0.9584 | 0.8981 | 0.9266 |
| TH       | 0.9918 | 0.0082 | 0.1748 | 1.4221 | 0.8252 | 0.8218 | 0.8199 |
| SH       | 0.9906 | 0.0094 | 0.0534 | 1.1551 | 0.9466 | 0.8243 | 0.8796 |
| DB       | 0.9991 | 0.0009 | 0.1151 | 0.2326 | 0.8849 | 0.8708 | 0.8762 |
| IM       | 0.9842 | 0.0158 | 0.2587 | 3.2532 | 0.7413 | 0.7844 | 0.6653 |
| CJ       | 0.9865 | 0.0135 | 0.1520 | 1.9033 | 0.8480 | 0.7588 | 0.7972 |
| A. CD.-12 | 0.9915 | 0.0085 | 0.1326 | 1.3965 | 0.8674 | 0.8264 | 0.8275 |
| BW       | 0.9973 | 0.0027 | 0.1850 | 0.5992 | 0.8150 | 0.8463 | 0.8200 |
| LF       | 0.9904 | 0.0096 | 0.1990 | 1.5926 | 0.8010 | 0.5773 | 0.5899 |
| NV       | 0.9694 | 0.0306 | 0.2521 | 3.6552 | 0.7478 | 0.3909 | 0.4688 |
| PTZ      | 0.9311 | 0.0689 | 0.0526 | 6.9067 | 0.9474 | 0.2807 | 0.3594 |
| TB       | 0.9997 | 0.0003 | 0.1551 | 0.1672 | 0.8449 | 0.9172 | 0.8758 |
| A. CD.-14 | 0.9852 | 0.0148 | 0.1490 | 1.9364 | 0.8510 | 0.7246 | 0.7344 |
| CMD      | 0.9948 | 0.0052 | 0.0156 | 0.5774 | 0.9844 | 0.9128 | 0.9473 |

### 4.7 Experiment on three datasets

The performance of our proposed method on CDnet-2012 [13], CDnet-2014 [47], and CMD [38] datasets in terms of KPM is shown in Table 7. According to the table, our proposal achieves 82.75%, 73.44%, and 94.73% average F1-scores on CDnet-2012, CDnet-2014, and CMD datasets, respectively. Additionally, our proposed method shows a low score, with 00.85% FNR on CDnet-2012 dataset. Moreover, accuracies of our method on the non-challenging videos and the challenging videos such as thermal (TH), camera jitters (CJ), shadow (SH), dynamic background (DB), bad weather (BW), and turbulence (TB) are very similar. However, our DFC-D decreases accuracy on PTZ videos compared to the other videos as the PTZ videos have large pan-tilt-zoom effects.

Table 8 presents our method’s category-wise KPM scores on LASIESTA [9] dataset. As stated by Table 8 our method shows very good average KPM scores. Most importantly, our proposed DFC-D obtains more than 90% F1-scores on all the videos except for I_IL, I_BS, and O_SU videos. The reasons for the enhanced performance on the challenging videos of CDNet-2012, CDNet2014, and LASIESTA datasets are: (1) The proposed adaptive fusion detects edge information more effectively, which results in more true positives and less false positives simultaneously. (2) Based on the background type (static or dynamic), the adaptive threshold is determined; therefore, it decreases false detection. (3) Based on the background type, the background samples are replaced by background pixels; therefore, it decreases false positives and increases false negatives simultaneously. However, our proposal shows low scores, with 51.44% F1-score on I_IL videos and 62.90% F1-score on I_BS videos. The reason is that I_IL and I_BS categories have videos with large global illumination changes and bootstrap effects, respectively, which our proposed method cannot compensate more effectively. From Tables 7 and 8, it is observed that our DFC-D method provides more recall...
Table 8  Accuracy of our proposal on LASIESTA dataset based on key performance metric (KPM)

| Category | Sp    | FPR   | FNR   | PWC   | Re    | Pr    | F1    |
|----------|-------|-------|-------|-------|-------|-------|-------|
| I_SI     | 0.9953| 0.0047| 0.0382| 0.5921| 0.9619| 0.8961| 0.9278|
| I_CA     | 0.9913| 0.0087| 0.0319| 1.1196| 0.9681| 0.9209| 0.9435|
| I_OC     | 0.9945| 0.0055| 0.0177| 0.6083| 0.9822| 0.8609| 0.9176|
| I_IL     | 0.8667| 0.1333| 0.0729| 12.861| 0.9271| 0.3816| 0.5144|
| I_MB     | 0.9891| 0.0108| 0.0129| 1.1000| 0.9871| 0.8988| 0.9397|
| I_BS     | 0.9968| 0.0032| 0.4428| 1.2475| 0.5572| 0.7496| 0.6290|
| O_CL     | 0.9932| 0.0068| 0.0033| 0.6076| 0.9967| 0.8910| 0.9405|
| O_RA     | 0.9973| 0.0027| 0.0112| 0.3067| 0.9888| 0.8857| 0.9342|
| O_SN     | 0.9934| 0.0066| 0.0390| 0.7194| 0.9610| 0.8504| 0.9023|
| O_SU     | 0.9953| 0.0047| 0.0352| 0.5060| 0.9648| 0.7190| 0.8236|
| Average  | 0.9813| 0.0187| 0.0705| 1.9669| 0.9295| 0.8054| 0.8473|

(Re) than precision (Pr) as the non-smoothing input sequences in DFC-D keep track of all the change information.

4.8 Comparisons on CDnet-2012, CDnet-2014, LASIESTA, and CMD datasets

Comparisons of our proposal with the state-of-the-art methods on CDnet-2012 [13] and CDnet-2014 [47] datasets are shown in Table 9. In this table, F1-score is observed for each video category separately. As stated by Table 9 our method obtains the best F1-score on TH and TB video cases. Note that these are the challenging videos in CDNet-2012 and CDNet-2014 datasets. We show the comparisons of our method with the existing approaches on LASIESTA dataset [9] in Table 10. This table also presents category-wise F-measure like Table 9. According to Table 10 our proposal outperforms the existing approaches on I_CA, O_CL, and O_SN videos with more than 90.00% F-measure. KPM score-based comparisons of our proposed method with the recent methods on CMD dataset [38] are shown in Table 11. We observe from Table 11 that our proposed method is the best, with 98.44% recall (Re) and 94.73% F1-score. Because our proposed adaptive fusion detects edges of an object more accurately and the non-smoothing input sequences keep intact all the edge information, we get a very high recall (Re). Moreover, the proposed method provides a very high F1-score due to our proposed adaptive fusion.

4.9 Visual comparisons

We show the segmentation object silhouettes to compare our proposal qualitatively with the existing approaches in Fig. 9. Our proposed DFC-D, Fast-D [17], DeepBS [1], and M4CD [46] segments thermal_corridor #610, dynamic_background_fall #3547, and PTZ_twoPositionPTZCam #1019 input sequences of CDnet-2014 dataset [47]. Note that the input sequences are selected from the most challenging videos of the dataset. The corresponding segmentation sequences and ground truth sequences are depicted in Fig. 9. From the figure, we observe that our proposed approach separates the moving objects from the input sequences without any noise and obtains the best segmentation object silhouettes on the three frames.
| Category Name         | Method                        | BL  | TH  | SH  | DB  | IM  | CJ  | A_CDnet-12 | BW  | LF  | NV  | PTZ | TB  | A_CDnet-14 |
|----------------------|-------------------------------|-----|-----|-----|-----|-----|-----|-------------|-----|-----|-----|-----|-----|-------------|
|                      | Proposed DFC-D                | 0.9266 | 0.8199 | 0.8796 | 0.8762 | 0.6653 | 0.7972 | 0.8275 | 0.8200 | 0.5899 | 0.4688 | 0.3594 | **0.8758** | 0.7344 |
|                      | Fast-D [17] (2020)           | 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 |
|                      | FBGS [18] (2020)             | 0.9112 | 0.7924 | 0.8623 | 0.8832 | 0.6450 | 0.7588 | 0.8088 | **0.8517** | 0.5982 | 0.4615 | 0.3564 | 0.8317 | 0.7230 |
|                      | M^4CD [46] (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 |
|                      | PBAS [16] (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 |
|                      | BMN-BSN [30] (2019)          | 0.9521 | 0.7849 | 0.8588 | 0.6371 | 0.6369 | 0.6962 | 0.7610 | - | - | - | - | - | - |
| Learning-based (Deep learning) | DeepBS [11] (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** |
|                      | RB-SOM [34] (2018)           | - | - | - | - | - | - | - | - | - | - | - | - | 0.5591 |
| Cluster-based        | SBS [7] (2020)               | - | - | - | - | - | - | 0.7197 | - | - | - | - | - | - |
|                      | SuperBE [6] (2019)           | - | - | - | - | - | - | 0.5356 | - | - | - | - | - | - |
| Category Name            | Method Name                  | LSI  | LCA  | LOC  | LIL  | LMB  | LBS  | O_CL | O_RA | O_SN | O_SU | Average |
|--------------------------|------------------------------|------|------|------|------|------|------|------|------|------|------|---------|
|                         |                              |      |      |      |      |      |      |      |      |      |      |         |
| Consensus-based          | Proposed DFC-D               | 0.9278 | **0.9435** | 0.9176 | 0.5144 | 0.9397 | 0.6290 | **0.9405** | 0.9342 | **0.9023** | 0.8236 | 0.8473 |
|                         | Fast-D [17] (2020)           | **0.9287** | 0.8924 | 0.9194 | 0.5021 | **0.9430** | 0.6182 | 0.9368 | **0.9378** | 0.8789 | **0.8710** | 0.8428 |
|                         | FBGS [18] (2020)             | 0.9140 | 0.8687 | 0.9016 | 0.3806 | 0.9291 | 0.6139 | 0.9303 | 0.9102 | 0.8621 | 0.8204 | 0.8131 |
|                         | 3DCD [29] (2021)             | 0.8612 | 0.7156 | 0.7512 | 0.8868 | 0.8274 | **0.8188** | 0.8268 | 0.8027 | 0.6841 | 0.8681 | 0.8043 |
| Learning-based (Deep learning) | FgSegNet_v2 [23] (2020) | 0.6547 | 0.6411 | 0.3995 | 0.4112 | 0.5291 | 0.3619 | 0.5170 | 0.4268 | 0.1229 | 0.1110 | 0.4175 |
|                         | MSFgNet [33] (2019)          | 0.9264 | 0.9213 | 0.9163 | **0.8967** | 0.9143 | 0.7157 | 0.8806 | 0.8659 | 0.8952 | 0.7869 | **0.8717** |
|                         | Berjón et al. [3] (2018)     | 0.8805 | 0.8444 | 0.7806 | 0.6487 | 0.9373 | 0.6644 | 0.9276 | 0.8669 | 0.7786 | 0.7221 | 0.8051 |
| Cluster-based            | DP-GMM [15] (2014)           | 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. [8] (2013)     | 0.7859 | 0.7361 | 0.8527 | 0.7915 | 0.7288 | 0.5836 | 0.8638 | 0.8085 | 0.4555 | 0.7305 | 0.7335 |
### Table 11  Our method’s performance on CMD dataset [38]

| Method                | Sp   | FPR  | FNR  | PBC   | Re    | Pr   | F1-score |
|-----------------------|------|------|------|-------|-------|------|----------|
| Proposed DFC-D        | 0.9948 | 0.0052 | 0.0156 | 0.5774 | **0.9844** | 0.9128 | **0.9473** |
| SBS [7] (2020)        | 0.9975 | 0.0024 | **0.0014** | **0.3862** | 0.9170 | 0.8672 | 0.8914 |
| SuperBE [6] (2019)    | **0.9994** | **0.0005** | 0.0105 | 1.0844 | 0.5343 | 0.9552 | 0.6852 |
| SuBS [40] (2015)      | 0.9962 | 0.0038 | 0.0405 | 0.5738 | 0.9594 | 0.9335 | 0.9462 |
| PBAS [16] (2012)      | 0.9981 | 0.0019 | 0.1872 | 1.1500 | 0.8128 | **0.9585** | 0.8797 |
| LOBSTER [39] (2014)   | 0.9980 | 0.0020 | 0.1893 | 1.1910 | 0.8107 | 0.9565 | 0.8776 |
| ViBe [2] (2011)       | 0.9939 | 0.0060 | **0.0018** | 0.7800 | 0.8950 | 0.7209 | 0.7986 |

**Fig. 9** Ground truth frames, and segmentation results of M³CD [46], DeepBS [1], Fast-D [17] and our proposed DFC-D on (a) thermal_corridor #610, (b) dynamic_background_fall #3547 and (c) PTZ_twoPositionPTZCam #1019 frames of CDnet-2014 dataset [47]
Table 12  Experimental setup of the methods

| Method   | Processing unit | GPU       | IDE       | OS         | RAM     | Language          |
|----------|-----------------|-----------|-----------|------------|---------|-------------------|
| Pro. DFC-D/ Fast-D [17] / FBGS [18] | Intel Core i5- 8400 2.80 GHz | -         | MS Visual Studio 2015 | Windows 10 (64 bit) | 16 GB | C++ with OpenCV 3.3.0 |
| MSFgNet [33] | Intel Corei7 4.20 GHz NVIDIA | GTX 1080 11GB | -         | -         | -       | -                 |
| DeepBS [1] | Intel Xeon (R) GeForce CPU E5-1620 v3 Titan X | - @3.5GHz × 8 | -         | -         | -       | Lua               |
| M4 CD [46] | -3210M | -         | -         | -         | -       | Matlab            |
| RB-SOM [34] | Intel Core i5 2.7GHz | -         | -         | -         | 8 GB    | Matlab-2016       |

4.10 Processing speed and memory usage comparisons

The experimental setup information of our proposed method and the state-of-the-art approaches is presented in Table 12. We observe the memory usage by byte per pixel (bpp) and the running time in terms of frame per second (fps) to compare computational complexity. From CDnet-2014 dataset, we take the baseline highway and baseline pedestrians video sequences with the frame size of 320 × 240 and 360 × 240, respectively to compare the frame rate. According to Table 13 our DFC-D uses the second-lowest memory with 6 bpp, and processes the third-highest number of frames with 41.46 fps and the second-highest number of frames with 38.89 fps on the baseline highway video and baseline pedestrians video, respectively. Indeed, our method processes a higher number of frames than a real-time frame rate.

Table 13  Comparisons in terms of memory usage and running time

| Category          | Method Name | bpp | fps |
|-------------------|-------------|-----|-----|
|                   |             |     | 256 × 256 resolution | 320 × 240 resolution | 360 × 240 resolution |
| Consensus-based   | DFC-D       | 6   | 41.46 | 38.89 |
|                   | Fast-D [17] | 3   | 74.24 | 69.51 |
|                   | FBGS [18]   | 6   | 42.78 | -    |
|                   | M4 CD [46]  | 6   | 00.21 | -    |
|                   | MSFgNet [33]| -   | 59.88 | -    |
| Learning-based (Deep) | DeepBS [1] | -   | 10.00 | -    |
|                   | RB-SOM [34] | -   | 07.00 | -    |
4.11 Trade-off between detection accuracy and complexity

Our method considers the trade-off between detection accuracy and algorithmic complexity simultaneously. We show the running time and segmentation accuracy at the same time in Fig. 10. According to the figure, our proposed DFC-D method achieved the best trade-off between detection accuracy and processing speed. However, other methods such as Fast-D [17], DeepBS [1], and RB-SOM [34] did not provide the trade-off. Fast-D reduced computational cost as compared to our DFC-D but provided more false detection. DFC-D expresses its superiority by sacrificing very negligible performance but having very low per-frame processing time.

5 Conclusions and future works

We present a novel, highly effective, dynamic multiple features fusion technique, called an adaptive combination, to increase detection accuracy in our proposed DFC-D. The proposed adaptive combination of color/gray intensity and gradient magnitude contributes as follows: (1) Contours of detection objects are detected more robustly since edges of the detection objects get more weight dynamically than non-edges. (2) As the contours are detected more accurately; therefore, the flood fill function in the post-processing can fill holes of the unfiltered segmentation objects more accurately, which results in more accurate detection. Additionally, the non-smoothing color-gradient input, and adaptive threshold and background sample update increase more segmentation accuracy. Our proposed method achieves the best accuracy on TH and TB videos of CDnet-2014 dataset, and on O_CL, I_CA, and O_SN videos of LASIESTA dataset. Furthermore, our proposal requires less memory and processes more frames than a real-time frame rate. Therefore, our DFC-D can be used in less complex devices. Last but not least, our multi-feature fusion technique can also be used to fuse a color/gray intensity feature with an edge-based feature in other research areas of computer vision and image processing.

In the future, we will present a condition-dependent background sample modification policy instead of an arbitrary sample modification policy. Moreover, we will add deep features in our proposed method to improve classification accuracy.
Acknowledgements The authors would like to express thanks to all members of the Intelligent-distributed Cloud and Security (ICNS) Laboratory, Kyung Hee University, Republic of Korea, for their support and cooperation. The authors also pay thanks to all the anonymous reviewers for their comments and suggestions consolidating this research work.

Author Contributions Conception: Md Alamgir Hossain; supervision: Eui-Nam Huh; writing-original draft: Md Alamgir Hossain, Md Imtiaz Hossain, Md Delowar Hossain, and Eui-Nam Huh. All the authors agreed to publish this manuscript.

Funding This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Grand Information Technology Research Center support program(IITP-2021-0-00742) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation) and Project No.2021-0-00818, Machine Learning Based Low Bandwidth Image Communication Edge Computing System for Proactive Anomaly Detection on Smart Plant Environment.

Declarations

Conflict of interests/competing interests The authors have no conflict of interests/competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Babaee M, Dinh DT, Rigoll G (2018) A deep convolutional neural network for video sequence background subtraction. Pattern Recognit 76:635–649
2. Barnich O, Van Droogenbroeck M (2011) ViBe: A universal background subtraction algorithm for video sequences. IEEE Trans Image Process 20(6):1709–1724. https://doi.org/10.1109/TIP.2010.2101613
3. Berjón D, Cuevas C, Morán F et al (2018) Real-time nonparametric background subtraction with tracking-based foreground update. Pattern Recognit 74:156–170
4. Bilodeau GA, Jodoin JP, Saunier N (2013) Change detection in feature space using local binary similarity patterns. In: 2013 International conference on computer and robot vision, pp 106–112. https://doi.org/10.1109/CRV.2013.29
5. Braham M, Van Droogenbroeck M (2016) Deep background subtraction with scene-specific convolutional neural networks. In: 2016 international conference on systems, signals and image processing, pp 1–4. https://doi.org/10.1109/IWSSIP.2016.7502717
6. Chen ATY, Biglari-Abhari M, Kevin I et al (2019) SuperBE: computationally light background estimation with superpixels. J Real Time Image Process 16(6):2319–2335
7. Chen QY, Sun ZL, Lam KM (2020) An effective subsuperpixel-based approach for background subtraction. IEEE Trans Ind Electron 67(1):601–609. https://doi.org/10.1109/TIE.2018.2893824
8. Cuevas C (2013) Improved background modeling for real-time spatio-temporal non-parametric moving object detection strategies. Image Vis Comput 31(9):616–630
9. Cuevas C, Yáñez EM, García N (2016) Labeled dataset for integral evaluation of moving object detection algorithms: LASIESTA. Comput Vis Image Underst 152:103–117
10. De Gregorio M, Giordano M (2014) Change detection with weightless neural networks. In: 2014 IEEE Conference on computer vision and pattern recognition workshops, pp 409–413. https://doi.org/10.1109/CVPRW.2014.66
11. Farcas D, Marghes C, Bouwmans T (2012) Background subtraction via incremental maximum margin criterion: a discriminative subspace approach. Mach Vis Appl 23(6):1083–1101
12. Garcia-Garcia B, Bouwmans T, Silva AJR (2020) Background subtraction in real applications: Challenges, current models and future directions. Comput Sci Rev 35:100204
13. Goyette N, Jodoin PM, Porikli F et al (2012) Changedetection.net: a new change detection benchmark dataset. In: 2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, pp 1–8. https://doi.org/10.1109/CVPRW.2012.6238919
14. Guo L, Xu D, Qiang Z (2016) Background subtraction using local SVD binary pattern. In: 2016 IEEE Conference on computer vision and pattern recognition workshops, pp 1159–1167. https://doi.org/10.1109/CVPRW.2016.148
15. Haines TS, Xiang T (2014) Background subtraction with Dirichlet process mixture models. IEEE Trans Pattern Anal Mach Intell 36(4):670–683. https://doi.org/10.1109/TPAMI.2013.239
16. Hofmann M, Tiefenbacher P, Rigoll G (2012) Background segmentation with feedback: the pixel-based adaptive segmenter. In: 2012 IEEE Computer society conference on computer vision and pattern recognition workshops, pp 38–43. https://doi.org/10.1109/CVPRW.2012.6238925
17. Hossain MA, Hossain MI, Hossain MD et al (2020) Fast-D: When non-smoothing color feature meets moving object detection in real-time, vol 8, pp 756–186,772. https://doi.org/10.1109/ACCESS.2020.3030108
18. Hossain MA, Nguyen V, Huh EN (2020) The trade-off between accuracy and the complexity of real-time background subtraction. IET Image Process 15(2):350–368. https://doi.org/10.1049/ipt2.12026
19. Hu L, Ni Q (2018) IoT-driven automated object detection algorithm for urban surveillance systems in smart cities. IEEE Internet Things J 5(2):747–754. https://doi.org/10.1109/JIOT.2017.2705560
20. Kim J, Rivera AR, Ryu B et al (2015) Simultaneous foreground detection and classification with hybrid features. In: Proceedings of the IEEE Int J Comput Vis, pp 3307–3315
21. Laugraud B, Piérad S, Braham M et al (2015) Simple median-based method for stationary background generation using background subtraction algorithms. In: Image analysis and processing – CIAP 2015 Workshops, Springer Cham., pp 477–484
22. Lee DS (2005) Effective Gaussian mixture learning for video background subtraction. IEEE Trans Pattern Anal Mach Intell 27(5):827–832. https://doi.org/10.1109/TPAMI.2005.102
23. Lim LA, Keles HY (2020) Learning multi-scale features for foreground segmentation. Pattern Anal Appl 23(3):1369–1380
24. Liu Y, Bellay Z, Bradsky P et al (2019) Edge-to-fog computing for color-assisted moving object detection. In: Big Data: Learning, analytics, and applications, international society for optics and photonics, p 1098903
25. Liu D, Cui Y, Chen Y et al (2020) Video object detection for autonomous driving: motion-aid feature calibration. Neurocomputing 409:1–11
26. Liu S, Liu D, Muhammad K et al (2021) Effective template update mechanism in visual tracking with background clutter. Neurocomputing 458:615–625. https://doi.org/10.1016/j.neucom.2019.12.143
27. Liu S, Wang S, Liu X et al (2021) Fuzzy detection aided real-time and robust visual tracking under complex environments. IEEE Trans Fuzzy Syst 29(1):90–102. https://doi.org/10.1109/TFUZZ.2020.306520
28. Liu S, Wang S, Liu X et al (2021) Human memory update strategy: a multi-layer template update mechanism for remote visual monitoring. IEEE Trans Multimed 23:2188–2198. https://doi.org/10.1109/TMM.2021.306580
29. Mandal M, Dhar V, Mishra A et al (2021) 3DCC: Scene independent end-to-end spatiotemporal feature learning framework for change detection in unseen videos. IEEE Trans Image Process 30:546–558. https://doi.org/10.1109/TIP.2020.3037472
30. Mondéjar-Guerra VM, Rouco J, Novo J et al (2019) An end-to-end deep learning approach for simultaneous background modeling and subtraction. In: Br Mach Vis Conf, p 266
31. OpenCV.org (2021) OpenCV: cv::backgroundsubtractormog2 class reference. https://docs.opencv.org/master/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html. Accessed 29 Jan 2021
32. Panda DK, Meher S (2016) Detection of moving objects using fuzzy color difference histogram based background subtraction. IEEE Signal Process Lett 23(1):45–49. https://doi.org/10.1109/LSP.2015.2498839
33. Patil PW, Murula S (2019) MSFgNet: A novel compact end-to-end deep network for moving object detection. IEEE Trans Intell Transp Syst 20(11):4066–4077. https://doi.org/10.1109/TITS.2018.2880096
34. Ramirez-Quintana JA, Chacon-Murguia MI, Ramirez-Alonso GM (2018) Adaptive background modeling of complex scenarios based on pixel level learning modeled with a retinotopic self-organizing map and radial basis mapping. Appl Intell 48(12):4976–4997
35. Rodriguez P, Wohlberg B (2016) Incremental principal component pursuit for video background modeling. J Math Imaging Vis 55(1):1–18
36. Sajid H, Cheung SCS (2017) Universal multimode background subtraction. IEEE Trans Image Process 26(7):3249–3260. https://doi.org/10.1109/TIP.2017.2695882
37. Sedky M, Moniri M, Chibelushi CC (2014) Spectral-360: A physics-based technique for change detection. In: 2014 IEEE Conference on computer vision and pattern recognition workshops, pp 405–408. https://doi.org/10.1109/CVPRW.2014.65
38. Sheikh Y, Shah M (2005) Bayesian modeling of dynamic scenes for object detection. IEEE Trans Pattern Anal Mach Intell 27(11):1778–1792. https://doi.org/10.1109/TPAMI.2005.213
39. St-Charles PL, Bilodeau GA (2014) Improving background subtraction using local binary similarity patterns. In: IEEE Winter Conf Appl Comput Vis, p 509–515. https://doi.org/10.1109/WACV.2014.6836059
40. St-Charles PL, Bilodeau GA, Bergevin R (2015) SuBSENSE: A universal change detection method with local adaptive sensitivity. IEEE Trans Image Process 24(1):359–373. https://doi.org/10.1109/TIP.2014.2378053
41. Stauffer C, Grimson WEL (1999) Adaptive background mixture models for real-time tracking. In: Proceedings. 1999 Proceedings of the IEEE computer society conference on computer vision and pattern recognition (Cat. No PR00149), pp 246–252. https://doi.org/10.1109/CVPR.1999.784637
42. Tiefenbacher P, Hofmann M, Merget D et al (2014) PID-based regulation of background dynamics for foreground segmentation. In: 2014 IEEE International conference on image processing, pp 3282–3286. https://doi.org/10.1109/ICIP.2014.7025664
43. Van Droogenbroeck M, Paquot O (2012) Background subtraction: experiments and improvements for ViBe. In: 2012 IEEE Computer society conference on computer vision and pattern recognition workshops, pp 32–37. https://doi.org/10.1109/CVPRW.2012.6238924
44. Vaswani N, Bouwmans T, Javed S et al (2018) Robust subspace learning: Robust PCA, robust subspace tracking, and robust subspace recovery. IEEE Signal Process Mag 35(4):32–55
45. Wang K, Gou C, Wang FY (2018) M4CD: A robust change detection method for intelligent visual surveillance. IEEE Access 6(15):505–15,520. https://doi.org/10.1109/ACCESS.2018.2812880
46. Wang J, Zhang G, Zhang K et al (2020) Detection of small aerial object using random projection feature with region clustering. IEEE Trans Geosci Remote Sens 59(6):5028–5039. https://doi.org/10.1109/TGRS.2020.3011002
47. Wang Q, Li Q, Li X (2020) A fast neighborhood grouping method for hyperspectral band selection. IEEE Trans Geosci Remote Sens 59(6):5028–5039. https://doi.org/10.1109/TGRS.2020.3011002
48. Wang Q, Li Q, Li X (2020) A fast neighborhood grouping method for hyperspectral band selection. IEEE Trans Geosci Remote Sens 59(6):5028–5039. https://doi.org/10.1109/TGRS.2020.3011002
49. Wang Q, Li Q, Liu X (2020) Detection of small aerial object using random projection feature with region clustering IEEE Trans Cybern, p 1-14. https://doi.org/10.1109/TCyb.2020.3018120
50. Wei H, Peng Q (2018) A block-wise frame difference method for real-time video motion detection. Int J Adv Robot Syst 15(4):1729881418783, 633
51. Zhang Y, Wang X, Qu B (2012) Three-frame difference algorithm research based on mathematical morphology. Procedia Eng 29:2705–2709
52. Zivkovic Z (2004) Improved adaptive Gaussian mixture model for background subtraction. In: Proceedings of the 17th Int Conf Pattern Recognit, ICPR 2004., pp 28–31. https://doi.org/10.1109/ICPR.2004.133992
53. Zivkovic Z, Van Der Heijden F (2006) Efficient adaptive density estimation per image pixel for the task of background subtraction. Pattern Recognit Lett 27(7):773–780

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.