

Dynamic Background Video Forgery Detection using Gaussian Mixture Model

Nugroho Satriyanto*, Rinaldi Munir†, Harlili‡
School of Electrical Engineering and Informatics
Institut Teknologi Bandung

Bandung, Indonesia
massatriya@gmail.com*, rinaldi@informatika.org†, harlili@informatika.org‡

Abstract— Video as evidence holds an important place in a court case and therefore the integrity of video must be proven. Various studies had been done in video forensics and most of them is only focused on a certain type of forgery, such as histogram correlation analysis that only focused on detecting temporally forged videos with static background. Improving histogram correlation analysis with foreground detection using Gaussian mixture model makes it possible to detect spatially forged videos and further improves its accuracy in detecting dynamic background video. Applying proposed method will yield 20.83% improved detection's accuracy, 34.42% increased localization's precision, and 39.17% increased localization's recall. Furthermore, object's definition introduced by foreground detection will also open up new possibility to detect spatially forged video.

Keywords— histogram, correlation, pixel line, forgery, Gaussian mixture model

I. INTRODUCTION

Evidence has an important place in criminal case trial. Existence of evidence can be used to testify crime or innocence and therefore has potential to be misused. This misuse can be done by manipulating evidence so it tells a story that never happens.

An example of evidence that often used is video. With the development of technology, such as development of more advance recording devices, video as an evidence is easily obtained. However, this also makes video's integrity becomes questionable with video editing software and technique development.

Nowadays, video forgery can be easily performed with such editing software. This software is usually intended for entertainment purposes but this obviously does not rule out the possibility such software used for manipulating evidence for court case.

Video forgery detection can be broadly classified into active and passive based approaches [1]. In active approach, pre-processing a video such as inserting a mark in video like watermark or digital signature is required. The mark inserted in active approach will be an identifier to decide whether the video is forged. Meanwhile in passive approach there is no pre-processing in video and it needs to observe certain characteristics in video to determine whether the video is forged.

II. FUNDAMENTAL CONCEPTS

A. Digital Video

Digital video is a representation of moving image element and voice element which represented in binary data. Unlike traditional analog video, which is captured frame by frame on a tape, digital video is recorded digitally, as ones and zeros. Since it is stored in a digital format, digital video

can be recognized and edited by a computer, which is also a digital device [3].

A digital video consist of two components, visual component and sound component. Visual component in a video represented by sequence of pictures called frame. A frame consist of pixels which is the smallest programmable color on a computer display or in a computer image.

B. Video Forgery

Video forgery is intentional modification/alteration of the digital video for fabrication [1]. Its implication depends upon the circumstance and where it is used. Particularly in the movie, political, and medical world, its impact is enormous where it is used to defame a personality, hide or forge important information to falsify or conceal actual event.

Sowmya et al [1] classify video forgery into three group, spatial, temporal, and spatio-temporal as shown in Fig. 1. Spatial attack performed on content of the frame(x-y axis) which present the visual information of the video. Whereas temporal attack performed on sequence of frames which present the sequence of events occurred in video. Meanwhile, spatio-temporal attack performed on both content of the frame and sequence of frames.

C. Gaussian Mixture Model

Gaussian mixture models are a probabilistic model for representing normally distributed sub populations within an overall population [4]. This model does not require knowing which sub population a data point belongs to, and intended to determine the sub population automatically. Therefore, this model constitutes as unsupervised learning.

Gaussian mixture model is parameterized with two values, component weight and mean and covariance. Each component k from components K will have mean μ_k , covariance σ_k , and weight ϕ_k which follows $\sum_{i=0}^K \phi_i = 1$.

Learning process of this model commonly using expectation-maximization technique. This technique consist of two main stages, expectation for calculating the probability of a data point is a part of a cluster and maximization for recalculating model's parameters with the calculated probability.

Before the expectation stage can be performed, an initialization of model's parameter must be performed. This can be done by splitting the data randomly in K clusters then calculating the mean and covariance of each cluster. Meanwhile, the weight of each component can be assigned with $\frac{1}{K}$.

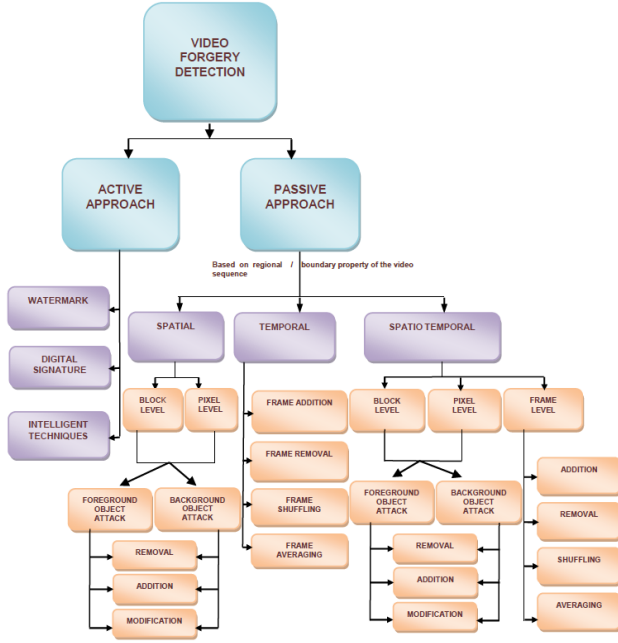


Fig. 1 video forgery classification [1]

After initializing model's parameter, the probability each data point is part of each cluster is calculated. The probability γ_{ik} which represents the probability of data point i is part of cluster k can be calculated with equation 1 and 2. The maximization stage then can be performed by recalculating weight, mean, and covariance of each component with equation 3, 4, and 5.

$$\gamma_{ik} = \frac{\phi_k \mathcal{N}(x|\mu_k, \sigma_k)}{\sum_{j=1}^K \phi_j \mathcal{N}(x|\mu_j, \sigma_j)} \quad (1)$$

$$\mathcal{N}(x|\mu_i, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{(x - \mu_i)^2}{2\sigma_i^2}\right) \quad (2)$$

$$\phi_k = \sum_{i=1}^N \frac{\gamma_{ik}}{N} \quad (3)$$

$$\mu_k = \frac{\sum_{i=1}^N \gamma_{ik} x_i}{\sum_{i=1}^N \gamma_{ik}} \quad (4)$$

$$\sigma_k = \frac{\sum_{i=1}^N \gamma_{ik} (x_i - \mu_k)^2}{\sum_{i=1}^N \gamma_{ik}} \quad (5)$$

III. RELATED WORK

In [2] the author proposed a method to detect video forgery based on histogram correlation between frames. The histogram is calculated from certain area in video called pixel belt. Pixel belt in video consist of several pixel lines that can be positioned horizontal or vertical. Fig. 2 show how pixel lines are made.

Every four continuous horizontal or vertical pixel lines make a horizontal or vertical pixel belts. b_h^i presents horizontal pixel belts and b_v^i presents vertical pixel belts as defined in equation 6 and 7 where l_h^i and l_v^i represents i^{th} horizontal and vertical pixel line and $i = 1, 2, 3, \dots, L - 3$ and L is the number of frame in video.

$$b_h^i = \langle l_h^i, l_h^{i+1}, l_h^{i+2}, l_h^{i+3} \rangle \quad (6)$$

$$b_v^i = \langle l_v^i, l_v^{i+1}, l_v^{i+2}, l_v^{i+3} \rangle \quad (7)$$

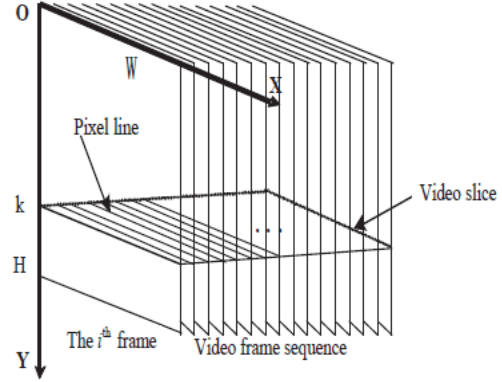


Fig. 2 pixel line[2]

The pixel belt defined then can be iterated to count the histogram that will be compared to another histogram from other frame. $H_{b_h^i}^i / H_{b_v^i}^i$ represents the ratio between the number of pixel with color p , n_p , with the number of pixel in pixel belt, N , as shown in equation 8 and 9.

$$H_{b_h^i}^i(p) = \frac{n_p}{N} \quad (8)$$

$$H_{b_v^i}^i(p) = \frac{n_p}{N} \quad (9)$$

From the histogram counted then correlation, $R_h(i)$, can be calculated with comparing histogram from pixel belt b_h^i / b_v^i with histogram from pixel belt b_h^{i+4} / b_v^{i+4} as shown in equation 10 and 11. The correlations calculated then can be aggregated as $A_h(R) / A_v(R)$ as shown in equation 12 and 13. The aggregated correlation is a time-series data which can be represented as a graph as shown in Fig. 3.

$$R_h(i) = \frac{\sum_{p=0}^{M-1} \min(H_{b_h^i}^i(p), H_{b_h^{i+4}}^{i+4}(p))}{\sum_{p=0}^{M-1} H_{b_h^i}^i(p)} \quad (10)$$

$$R_v(i) = \frac{\sum_{p=0}^{M-1} \min(H_{b_v^i}^i(p), H_{b_v^{i+4}}^{i+4}(p))}{\sum_{p=0}^{M-1} H_{b_v^i}^i(p)} \quad (11)$$

$$A_h(R) = \langle R_h(1), R_h(2), \dots, R_h(L-7) \rangle \quad (12)$$

$$A_v(R) = \langle R_v(1), R_v(2), \dots, R_v(L-7) \rangle \quad (13)$$

Fig. 4 represents a pixel belt from frame i to $i + 4$. If for instance frame $i + 4$ is forged, then correlation $R_h(i)$ will be the minimum among $R_h(i - 3)$ until $R_h(i + 3)$.

To determine forged frames' location, outlier detection must be applied. In [2], the author used interquartile range to determine outliers. The interquartile range can be calculated with sorting the aggregate in ascending order and determining three dividers based on median called Q1, Q2, and Q3. The outliers then can be defined as every value outside $Q1 - 1.5(Q3 - Q1)$ and $Q3 + 1.5(Q3 - Q1)$.

The problem from [2] is primarily pixel line's placement. When the pixel line is placed in a region with minimal change as shown in Fig. 5 both pixel line in (a) and (b). In Fig. 5(a), pixel line may cause a high correlation value because of minimal change in that region and cause a false positive. Whereas in Fig. 5(b), pixel line will not detect changes in the circle because it does not intersect with the object and this kind of case may cause false negative.

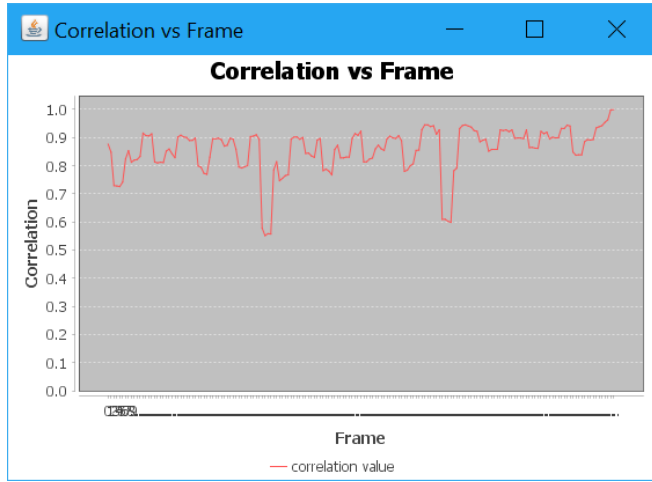


Fig. 3 Correlation aggregate chart

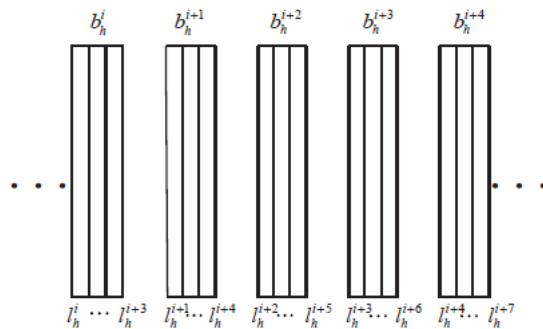


Fig. 4 pixel belt[2]

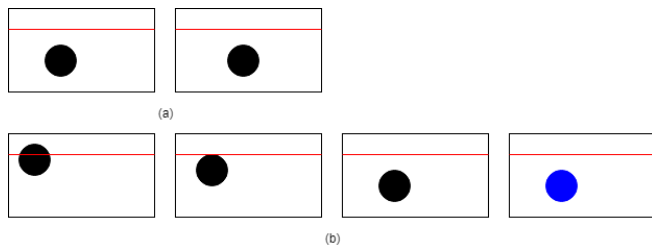


Fig. 5 pixel line placement problem

IV. PROPOSED METHOD

To solve previously mentioned problems, it is needed to place pixel line in an area that has a lot of changes. One way to do it is to use foreground detection to determine where pixel line should be placed. Since foreground detection works by classifying areas that have a lot of changes as foreground, it can be used in this case.

The proposed method consists of six stage, foreground detection, thresholding, denoising, pixel line placement, histogram correlation calculation, and analyzing calculated correlation. These processes can be represented in Fig. 6.

A. Foreground Detection

Gaussian mixture model can be used to do foreground detection by clustering pixels from each frame to several clusters. These clusters then can be used to determine which cluster classify as background and foreground. The probability calculated then can be used to transform video to grayscale that darker pixels more likely classify as background as shown in Fig. 7.

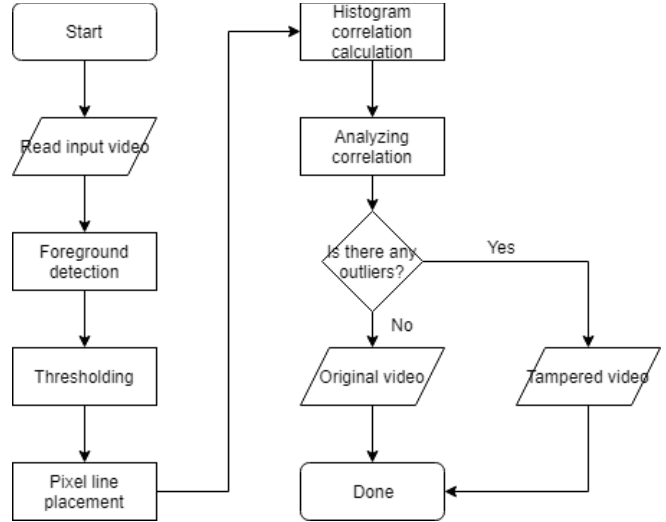


Fig. 6 proposed method's flowchart

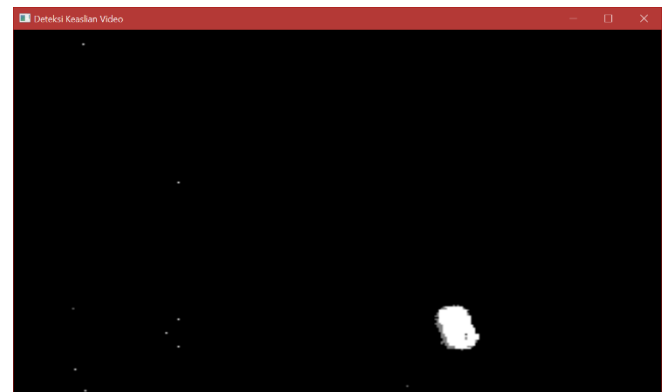


Fig. 7 grayscale video resulted from gaussian mixture model

B. Thresholding

The result of foreground detection using Gaussian mixture model is a grayscale video. Therefore, the video needs to be converted to binary video so it can be used to easily determine which pixel is a foreground.

One way to convert grayscale video to binary video is using a threshold. The threshold can be used to convert all pixel value below threshold to 0 or black and the other to 255 or white.

Threshold's calculation can be performed with Otsu method. This method iterates all possible threshold values then determine which threshold has the minimum distribution. The distribution between classes separated by threshold can be calculated with within-class variance w_2 as shown in equation 14 with $q_1(t)$ represents the number of pixels below t , $q_2(t)$ represents the number of pixels above t , and $\sigma_1^2(t)$ and $\sigma_2^2(t)$ represent covariance of the two groups which can be calculated using equation 15. The result of this process will be a binary video with white pixel represents foreground pixel as shown in Fig. 8.

$$\sigma_w^2 = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \quad (14)$$

$$\sigma_x^2 = \frac{\sum_{i=1}^N (X_i - \mu_x)^2}{N} \quad (15)$$

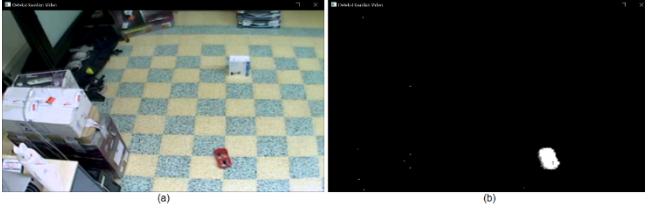


Fig. 8 thresholding result

C. Denoising

The video resulted from thresholding process will contains noises introduced by the insignificant movement of dynamic background. This can however be deleted using flood fill algorithm.

The flood fill algorithm used to calculate the area of white contour. Then some threshold value can be applied to filter small contour that can be defined as noise. This whole process can be summarized in algorithm 1.

Algorithm 1 Denoising algorithm

- 1: **procedure** DENOISE(*frame*, *threshold*)
- 2: **for all** *pixel* in *frame* **do**
- 3: **if** isForeground(*pixel*) **then**
- 4: *frameFloodFill* \leftarrow flood fill from white to black starting from pixel location
- 5: *count* \leftarrow number of pixels filled with black
- 6: **if** *count* < *threshold* **then**
- 7: *frame* \leftarrow *frameFloodFill*

D. Pixel Line Placement

After the denoising process, pixel line can be placed intersecting the foreground detected. One way to do that is logging the bounding rectangle of foreground contour using the flood fill algorithm as shown in Algorithm 2. Pixel line then placed inside the bounding rectangle.

Algorithm 2 Bounding flood fill

- 1: **procedure** FLOODFILL(*node*, *targetcolor*, *replacementcolor*, *rect*)
- 2: **if** color(*node*) \neq *targetcolor* **then return**
- 3: getNewBound(*rect*, *node*)
- 4: *rect1*, *rect2*, *rect3*, *rect4* \leftarrow *rect*
- 5: FloodFill(southOf(*node*), *targetcolor*, *replacementcolor*, *rect1*)
- 6: FloodFill(northOf(*node*), *targetcolor*, *replacementcolor*, *rect2*)
- 7: FloodFill(westOf(*node*), *targetcolor*, *replacementcolor*, *rect3*)
- 8: FloodFill(eastOf(*node*), *targetcolor*, *replacementcolor*, *rect4*)
- 9: getMaximumRect(*rect*, *rect1*, *rect2*, *rect3*, *rect4*)

Foreground movement in the video can make pixel line obsolete in frame *i*. This is caused by the case pixel line is not intersecting the foreground anymore. Therefore, new pixel line must be introduced to intersect the moving object. Thus, pixel line *l_{hi}* and *l_{hi+1}* will have different location.

To solve this problem can be simply by making those pixel lines overlapping. Therefore, pixel line from frame *i+1* to *i+3* will have two locations. This whole process of pixel line placement shown in Fig. 9.

The foreground detection applied can open new opportunity which is detecting object duplication or copy-move. This can be done by simply comparing two foreground's histograms by making pixel line filling foreground objects. This again can be done by modifying flood-fill algorithm to log each pixel color in object as shown in Algorithm 3.

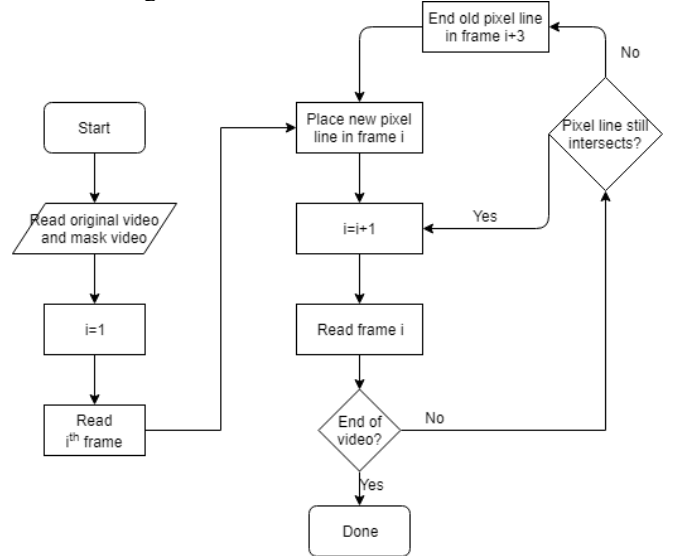


Fig. 9 pixel line placement flowchart

Algorithm 3 Pixel line placement

- 1: **procedure** FLOODFILL(*node*, *targetcolor*, *replacementcolor*)
- 2: **if** color(*node*) \neq *targetcolor* **then return**
- 3: addToHistogram(colorOf(*node*))
- 4: FloodFill(southOf(*node*), *targetcolor*, *replacementcolor*)
- 5: FloodFill(northOf(*node*), *targetcolor*, *replacementcolor*)
- 6: FloodFill(westOf(*node*), *targetcolor*, *replacementcolor*)
- 7: FloodFill(eastOf(*node*), *targetcolor*, *replacementcolor*)

E. Histogram Correlation Calculation

Calculating correlation between pixel belt will be mostly the same as [2]. The calculation for spatial tampering detection however need to be redefined. Unlike frames that have linkages between them, foregrounds does not have and thus a full comparison between foregrounds must be performed.

F. Analyzing Correlation

Interquartile range that used in [2] used to analyze the outliers in correlation values. However, to detect spatial tampering, the lower limit $Q1-1.5(Q3-Q1)$ can't be used as a limit. This abandonment can be done since two objects can be very different from another and does not imply any tampering done.

V. EXPERIMENTAL RESULTS

The proposed method was tested with 18 videos consisting of 4 original videos, 8 temporally forged videos, 6 spatially forged videos that downloaded from SULFA dataset [5]. Each video was tested twice in the same environment.

Without foreground detection, the result yield 16 correctly detected and 8 wrongly detected as classified in Table I. Mean-while with foreground detection, the result yield 21 correctly detected and 3 wrongly detected as classified in Table II. These results show that there is significant improvement from 66.67% to 87.5% using foreground detection.

TABLE I. TEMPORAL DETECTION RESULT WITHOUT FOREGROUND DETECTION

	Positive	Negative
True	13	3
False	3	5

TABLE II. TEMPORAL DETECTION RESULT WITH FOREGROUND DETECTION

	Positive	Negative
True	16	5
False	0	3

The same experiments were also carried in tampering localization. Without foreground detection, the result shows 10 frames correctly detected and 29 frames wrongly detected as classified in Table III. Meanwhile with foreground detection, the result shows 18 correctly detected and 10 wrongly detected as classified in Table IV. These results show that there is significant improvement both from recall that increases from 55.56% to 94.73% and from precision from 32.25% to 66.67% using foreground detection.

TABLE III. TEMPORAL LOCALIZATION RESULT WITHOUT FOREGROUND DETECTION

	Positive	Negative
True	10	-
False	21	8

TABLE IV. TEMPORAL LOCALIZATION RESULT WITH FOREGROUND DETECTION

	Positive	Negative
True	18	-
False	9	1

The significant improvement is obtained from extra process required in the proposed method. This extra process, however, makes significant changes in the time needed to do

detection. Without the foreground detection, the average time needed to do detection is 14845.06 ms. Meanwhile, with foreground detection, the average time needed is 20541.94 ms. These results show that there is 38.37% increased time needed. However, compared to the improvements made, this can be tolerated.

Another experiments using spatially tampered video were also carried and yield 77.78% accuracy as classified in Table V. This result was worse than some other method in spatial tampering detection such as cross-correlation method that published in [6] which yield 85% accuracy. However, the proposed method has 90% recall that is higher than 82% recall in [6].

TABLE V. SPATIAL DETECTION RESULT WITH FOREGROUND DETECTION

	Positive	Negative
True	9	5
False	3	1

VI. CONCLUSION

The proposed method improved the method proposed by Jie Xu [2] both in detecting temporally and spatially forged video. The proposed method offers 20.83% increased accuracy on detecting temporally forged video and 34.42% increased precision and 39.17% increased recall on localizing temporally forged video.

The proposed method also makes detecting spatially forged video possible. This method is worse than the other spatial forgery detection method in term of accuracy. However, this method's recall is can be considered better than the others that means the proposed method can identify forgery better.

REFERENCES

- [1] Sowmya K. N and H.R. Chennamma, "A Survey on Video Forgery Detection," A survey on video forgery detection, vol. 9, no. 2, pp. 17–27, Mar. 2015.
- [2] J. Xu, Y. Liang, X. Tian and A. Xie, "A Novel Video Inter-frame Forgery Detection," 2016.
- [3] P. Christensson, "DV," P2P (Peer To Peer) Definition. [Online]. Available: <https://techterms.com/definition/dv>. [Accessed: 20-Jan-2019].
- [4] "Gaussian Mixture Model," Brilliant Math & Science Wiki. [Online]. Available: <https://brilliant.org/wiki/gaussian-mixture-model/>. [Accessed: 20-Jan-2019].
- [5] G. Qadir, S. Yahaya, and A. T. S. Ho, "Surrey university library for forensic analysis (SULFA) of video content," in IET Conference on Image Processing (IPR 2012), 2012.
- [6] M. Mathai, D. Rajan, and S. Emmanuel, "Video forgery detection and localization using normalized cross-correlation of moment features," 2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), 2016.