VIDEO FORGERY DETECTION USING HOG FEATURES AND COMPRESSION PROPERTIES

A.V. Subramanyam, Sabu Emmanuel

School of Computer Engineering, Nanyang Technological University
Singapore
subr0021@e.ntu.edu.sg
asemmanuel@e.ntu.edu.sg

***Abstract*—In this paper, we propose a novel video forgery detection technique to detect the spatial and temporal copy paste tampering. It is a challenge to detect the spatial and temporal copy-paste tampering in videos as the forged patch may drastically vary in terms of size, compression rate and compression type (I, B or P) or other changes such as scaling and filtering. In our proposed algorithm, the copy-paste forgery detection is based on HoG feature matching and video compression properties. The benefit of using HoG features is that they are robust against various signal processing manipulations. The experimental results show that the forgery detection performance is very effective. We also compare our results against a popular copy-paste forgery detection algorithm. In addition, we analyze the experimental results for different forged patch sizes under varying degree of modifications such as compression, scaling and filtering.**

***Index Terms*—Video forgery detection, Copy-paste tampering, HoG features**

I. Introduction

The intelligent use of digital video editing techniques is constantly increasing the difficulty in distinguishing the authentic video from the tampered one. For example, in [1] the authors refer to the forgery created in a popular film *Speed* by duplicating the frames thereby hiding any activity actually going on. The copy-paste tampering can be performed in a convincing manner without much of difficulty. And copy-paste tampering can be practically difficult to detect. Therefore, it is likely that copy-paste tampering can be often applied to forge a video. However, in order to detect such forgeries, intrinsic properties of captured media can be used [2]. In this paper, we use the intrinsic properties of the captured media to detect the copy-paste tampering in a video.

The copy-paste video forgeries can be classified into two different categories - spatial tampering and temporal tampering. In spatial tampering, a region may be copied from a location in a frame and pasted to a different location on the same frame or other frames possibly after some modification. For example, the trash bin in Figure 1-a has been enlarged and pasted in Figure 1-d. While in temporal tampering, a complete frame may be duplicated. In addition, a region may also be duplicated across the frames at the same spatial location. Since an action or object may be occurring in multiple frames, in order to create a convincing forgery the temporal tampering

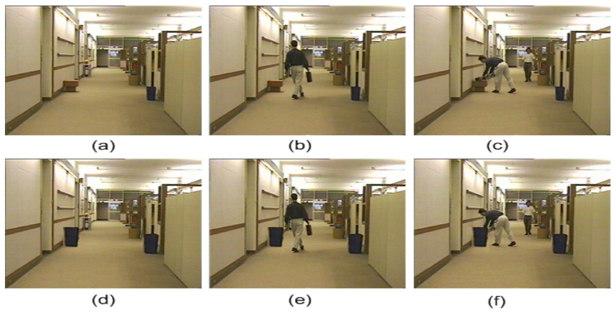


Fig. 1: (a - c) Sequence from original video (d - f) Sequence from forged video

may stretch over multiple frames. For example, the enlarged trash bin has been pasted across multiple frames as shown in figures 1-d to 1-f. In our case, to reliably detect a temporal tampering, we assume that a frame or a region is duplicated at least throughout a GOP (Group of Pictures). The forged regions may come from different videos as well. In both the spatial or temporal tampering, the motive may be to hide unfavorable actions or objects in a scene by pasting other objects or background, or to implant false evidences. From Figure 1-f it is not clear if the person is keeping the bag or stealing it away, although in Figure 1-c the person is keeping the bag.

Several forgery detection techniques related to image have been proposed till date, however, there are only few video forgery detection works. Among the image copy-paste forgery detection techniques, some of the promising algorithms are [3], [4], [5]. The techniques proposed in [3], [4], are based on Scale Invariant Feature Transform (SIFT) features matching. In [5] the authors use Fourier-Mellin Transform (FMT) to achieve robustness against geometric transformations. Some other forgery detection techniques can be found in [6]. Although the techniques [3] - [5] can be used for spatial copy-paste forgery detection, they cannot be easily extended for detecting temporal copy-paste forgery. This is because, the algorithms [3] - [5] assume that the tampered region is at a different spatial location than the source region from which it is copied. However, in temporal tampering the copy-paste regions are spatially colocated.

In this paper, we focus on copy-paste forgery detection in videos. The video copy-paste forgery has been addressed in

MMSP'12, September 17-19, 2012, Banff, Canada.
???-?-????-????-?/12/$??.?? ©2012 IEEE.

[1], [2]. In [1], the authors use temporal and spatial correlation in order to detect duplications. A temporal correlation matrix is computed between all frames in a given sub-sequence of frames and, spatial correlation matrix is computed for each frame in a given sub-sequence. The temporal and spatial correlation matrices are then used to detect the duplicated frames or regions. Although the detection performance is good while detecting a frame duplication, the region duplication detection efficiency is very low for small forged regions such as 64x64. In addition, this technique assumes that the forged regions belong to the same video sequence which may be a limitation when considering forged patches from other videos. However, the authors of [1] proposed another forgery detection scheme [7] which can detect forged regions belonging to different videos. However, this algorithm detects forgery only in interlaced or deinterlaced videos. In [2], the authors propose the detection of forged regions based on the inconsistencies of noise characteristics, which occurs due to the forged patches from different videos. However, as the noise characteristics depend on the intrinsic properties of the camera, the noise characteristics are not useful when the forged patch comes from the same video. In addition, the noise characteristics may not be estimated correctly under low compression rates.

In [8], the authors propose a forgery detection algorithm for the video in-painting tampering. The idea is to use the noise correlation properties between spatially colocated blocks to detect the forgery. Some other forgery detection techniques have also been proposed such as [9], [10]. In these techniques, the basic idea is that a forged video will be recompressed and the artifacts owing to double quantization of coefficients can be used to detect forgery. However, the techniques [8] - [10] suffer from the limitation that they are robust for a limited range of compression rates only.

One of the challenges in detecting video copy-paste forgery is to find the robust representations for the video frame blocks, so that the duplicated blocks can be identified even after modifications. In image forgery detection [3] - [5] features such as SIFT, transforms such as FMT have been used. While in video forgery detection normalized correlation [1], noise characteristics [2], [8], or quantization parameters [9], [10] have been used. Although features such as SIFT give good forgery detection performance [3], [4] owing to there robustness, other features such as SURF [11] or HoG [12] can also be used for detection purpose.

Second challenge is that, when a region is copy-pasted across the frames in a spatially colocated area, it may be very difficult to distinguish the forged regions and the authentic regions. This is because, in a video captured from a static camera, a sequence of frames may be highly correlated. And both the forged and authentic spatially colocated regions show a high correlation. For example, in a CCTV footage, an evidence might be implanted which need not be moving but just stay there such as a weapon in case of a crime. In this case the forged as well as non-forged parts will show a high correlation making it difficult to detect the tampering. Another challenge is the ability of detecting forgeries of small patch size such as a macroblock or of the order of few macroblocks. Although, features such as SIFT are highly robust, they may give less accurate forgery detection for small patch size such as macroblock when such a forged patch may have undergone transformations such as scaling [4]. This is because, SIFT algorithm may generate less robust keypoints in such a small area and the detection may not be effective. Therefore, features such as HoG which are dense image or blockwise descriptors are useful in detecting such form of tampering.

In this paper, we propose to use HoG features [12] extracted from the blocks of the frames for detection purpose. The HoG features are robust to scaling, compression, translation and other signal processing operations such as filtering etc. Due to the robustness and distinctiveness, the HoG features are apt for identifying the possibly modified duplicated regions with high accuracy. Further, the false positive (detecting authentic as forged) rates may be significantly lower for the dense grid approach such as HoG than keypoint based approaches such as SIFT [12]. Figures 2-a and 2-b shows the pipeline of the proposed algorithm. The HoG features extracted from blocks are matched against the HoG features extracted from the rest of the blocks in both the spatial and temporal domains for detecting spatial and temporal forgery respectively. In case of spatial forgery detection, the parameter cell size for HoG feature generation is set adaptively (Section III-A). The identification of the best match for each block is based on a nearest neighbor criteria for spatial forgery detection (Section III-A) or threshold criteria for temporal forgery detection (Section III-B). If the best match satisfies the nearest neighbor or threshold criteria, it is counted for forgery detection only when the nearest surrounding neighbors also fulfill the nearest neighbor or threshold criteria for spatial or temporal forgery detection respectively.

The rest of the paper is organized as follows. In Section II, we give a brief summary of the HoG features. In Section III, we explain the proposed detection scheme. We discuss the effect of different parameter settings on detection performance as well as the time complexity of the algorithm in Section IV. The experimental results are given in Section V and Section VI concludes the paper.

II. Histogram of Oriented Gradients

In this section, we briefly describe the extraction of HoG features from a given image or frame. In [12], Dalal et. al. represented the image or frame by a set of local histograms. These histograms count the number of occurrences of gradient orientation in a local spatial region of the image known as cell. Typically, a cellsize may vary as 4x4, 6x6 or 8x8 pixels. In order to extract the HoG features, first the gradients of the image are computed followed by building a histogram of orientation at each cell. Finally the histogram obtained from each cell in a block is normalized which gives the HoG descriptor of that block, where a block may comprise of 2x2 or 3x3 cells. In the following we explain the sequence of steps in HoG feature extraction.

First the RGB frame is converted to gray-scale followed by gradient computation by convolving the image with the horizontal and vertical masks [-1 0 1] and [-1 0 1]^T respectively. The horizontal and vertical gradients of an image I can be represented as $G_x(x,y) = [-1 \ 0 \ 1] * I(x,y)$ and $G_y(x,y) = [-1 \ 0 \ 1]^T * I(x,y)$ respectively, where $'*'$ denotes convolution. Then the orientation Θ at each pixel is computed using the ratio of gradients in vertical and horizontal directions

$\Theta(x, y) = arctan(G_y(x, y)/G_x(x, y))$. In the next step, the frame is divided into overlapping blocks of size NxN. Further, each block is divided into MxM cells, where $M \leq N$. For each cell, each pixel calculates a weighted vote which is usually the gradient magnitude at that pixel and the votes are accumulated into orientation bins. In practice, the number of orientation bins L are evenly spaced over $0°$ - $180°$ for unsigned gradients or $0°$ - $360°$ for signed gradients and the value of l^{th} bin is updated as,

$$\Omega_l(x, y) = \begin{cases} G(x, y) & \text{if } \Theta(x, y) \in bin_l, \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where, $G(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}$ is the gradient magnitude at pixel (x, y). HoG features are extracted from each cell and the number of features generated from each cell is equal to the number of orientation bins. The features from all the cells in a block are normalized as,

$$f_{Cell} = \frac{\sum_{(x,y)\in Cell} \Omega_l(x, y) + \epsilon}{\sum_{(x,y)\in Block} G(x, y) + \epsilon} \tag{2}$$

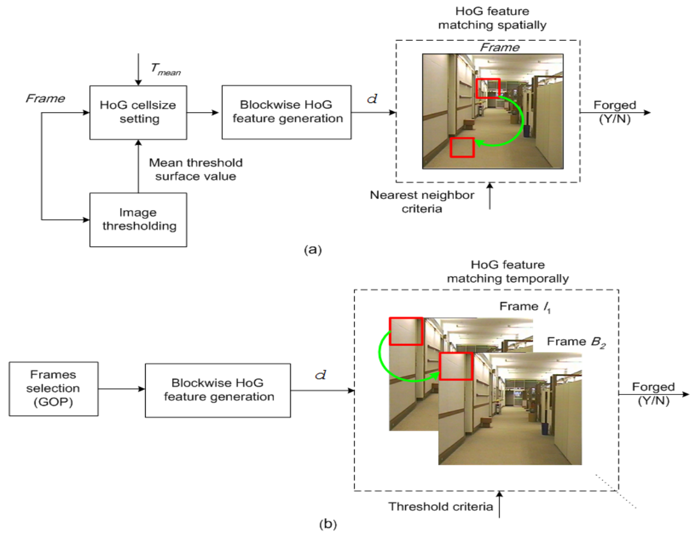
where ϵ is an infinitesimally small quantity. The normalized features from all the cells in a block form a vector d of dimension equal to the product of number of orientation bins and the number of cells in the block, which is the final HoG descriptor for that block. Next, we discuss the proposed scheme.

III. Proposed Scheme

In this section we describe the proposed copy-paste forgery detection scheme in detail. We first discuss the spatial forgery detection. First of all, we apply an image thresholding mechanism which is used to set the cell size as shown in Figure 2-a. Then the HoG features are generated for each block and subsequently search for matches for individual block descriptors. Then we present the temporal forgery detection. Here, the frames in a GOP are selected based on compression properties. Then the HoG features are generated blockwise. Finally, the block descriptors are compared with the spatially colocated block descriptors to find if a match exists as shown in Figure 2-b.

A. *Spatial Forgery detection*

Figure 2-a gives the block diagram for spatial forgery. In the first step, we find a suitable cell size for HoG feature generation for each individual video frame. This is necessary because some videos or frames may have poor localized edges naturally or due to application of some signal processing techniques. As a result local intensity gradients or edge directions may not be captured well with the same parameter settings - number of orientation bins, cellsize, blocksize and block overlap percentage for HoG feature extraction. And the features may either be not very distinctive or may not be robust against different signal processing operations such as scaling. Therefore, we need an adaptive mechanism which can tune the parameters such as cell size for HoG feature generation based on the frame's statistical properties. We use the mean threshold surface value of the image, which is computed using [13] to decide the cellsize for HoG feature extraction. In practice, if the mean threshold surface value is lesser than T_{mean}, a


HoG feature matching spatially
Frame
T_{mean}
Frame
HoG cellsize setting
Blockwise HoG feature generation
d
Forged (Y/N)
Mean threshold surface value
Image thresholding
Nearest neighbor criteria
(a)
HoG feature matching temporally
Frame I_1
Frame B_2
Frames selection (GOP)
Blockwise HoG feature generation
d
Forged (Y/N)
Threshold criteria
(b)
Fig. 2: (a) Spatial forgery detection (b) Temporal forgery detection

large cellsize may not generate robust HoG features. On the other hand, if the mean threshold surface value is greater than T_{mean}, a small cellsize may give rise to false positives (detecting authentic as forged). Here, the value of T_{mean} is obtained experimentally and we set cell size MxM as given in equation (3).

$$M = \begin{cases} 8 & \text{if, Mean threshold surface} > T_{mean}, \\ 6 & \text{if, Mean threshold surface} < T_{mean}, \end{cases} \tag{3}$$

Once the cellsize is set according to equation (3), the HoG descriptors are generated as given in Section II. In the next step, the generated descriptors from the blocks are tentatively matched against the rest of the blocks' HoG features of the frame. Let i denote the spatial index of the block and N denote the total number blocks in a given frame. In order to select the best match for a block b_i $\forall$ $i = 1, 2, .., N$, we first find the angles between the HoG descriptor d_i of the block b_i and the descriptors d_j of the blocks b_j $\forall$ j as given in equation (4). Here, $j \in \mathcal{C}$ where $\mathcal{C}$ is a set whose elements lie outside 5x5 window centered at block b_i. Here $j|j \notin \mathcal{C}'$ because the blocks located nearby b_i may be highly correlated with b_i which may give rise to false positives.

$$\theta_i = arccos(\sum_k d_i(k)d_j(k)) \quad \forall j \in \mathcal{C} \tag{4}$$

where $d_i(k)$ indicates the k^{th} element of the row vector d_i. Then, θ_i which is a 1-D vector is sorted in ascending order. We select the match $\theta_i(1)$ for b_i only if it follows a nearest neighbor criteria given in equation (5).

$$\theta_i(1) < \tau_{spatial}\theta_i(2) \tag{5}$$

where $\tau_{spatial} \in$ [0, 1] and is obtained experimentally. However, if we count the best match for the block b_i based solely on the best match of b_i, the false positives may be high. In order to reduce the false positives, we count the best match for b_i only when b_i follows equation (5) and the m nearest surrounding neighbors in the positive and negative X-axis and Y-axis directions satisfy the respective nearest neighbor criteria. Similarly, we find the matches for all the blocks and we can also search for the matches between a pair of frames.

In order to detect if a frame contains a forged region, we first define a matrix D_S of the same dimensions of the frame and initialize it with zeros. At each index i where we get a match we update the matrix as,

$$D_S(x_i, y_i) = (D_S(x_i, y_i) + 1) \bigcup (D_S(x_j, y_j) + 1) \tag{6}$$

where (x_i, y_i) and (x_j, y_j) are the starting co-ordinates of blocks b_i and b_j respectively. We slide a window of size W_x x W_y by a pixel rowwise and then columnwise along the matrix D_S, and count the number of matches in each such window. If the number of matches in any such window is higher than a preset threshold $T_{spatial}$, then we declare the region to be forged.

B. Temporal Forgery detection

Figure 2-b gives the block diagram for temporal forgery detection. Here, we detect the forgery in a GOP, the size of which is set to 12 frames. Since the forgery occurs in a spatially colocated area, we cannot apply the technique given in Section III-A. In order to detect the forgery, we make use of the compression properties of the video coding. In video coding such as MPEG-2 [14], a video is divided into GOPs and each GOP is coded independently. A typical GOP comprises of 1 I frame, 3 P frames and 8 B frames. A typical GOP structure is

I_1 B_2 B_3 P_4 B_5 B_6 P_7 B_8 B_9 P_{10} B_{11} B_{12}

where the subscripts indicate the order of frames. Here, the I frame is encoded independently. P frames are encoded using the previous I or P frame as reference. B frames are encoded using previous and next I or P frames as references. When a region is copy-pasted across the frames in a GOP in a spatially colocated area, then the macroblocks belonging to this region show a high correlation as compared to the rest of the macroblocks. This is due to the fact that when a region is copy-pasted across the frames it contains the same amount of noise induced by the camera at the time of capture whereas the noise in the rest of the blocks may remain uncorrelated. Further, in case of B frames, when macroblocks are nearly identical to the macroblocks of reference frames, the decoded macroblocks of the reference frames are directly used for B frames i.e. encoding and decoding for B frame macroblocks is skipped. Therefore, the correlation between the duplicated blocks is particularly high in the frame pairs I_1B_2, P_4B_5, P_7B_8 and $P_{10}B_{11}$. Thus, these set of frames can be used to detect if a given region is forged or not.

Now that the frames in a GOP are selected, the HoG features are generated. In the following step, we compute the normalized correlation $C(.,.)$ between the descriptors d_i $\forall$ i of frame I_1 to the spatially co-located descriptors d_j $\forall$ j of the consecutive frame B_2.

$$C(d_i, d_j) = \frac{\sum_k (d_i(k) - \mu_{d_i})(d_j(k) - \mu_{d_j})}{\sqrt{\sum_k (d_i(k) - \mu_{d_i})^2}\sqrt{\sum_k (d_j(k) - \mu_{d_j})^2}}$$
$$\forall\ i = j$$

We select the match for a block b_i $\forall$ i only if $C(d_i, d_j) > \tau_{temporal}$, where $\tau_{temporal}$ and is obtained experimentally. Here again, as mentioned in Section III-A, we count the best match for b_i only when b_i follows a threshold criteria $C(d_i, d_j) > \tau_{temporal}$ and the m nearest surrounding neighbors in the positive and negative X-axis and Y-axis directions satisfy the respective threshold criteria.

In order to detect if a frame contains a forged region, we define a matrix D_T as in Section III-A of the same dimensions of the frame and initialize it with zeros. At each index i where we get a match as discussed above, we update the matrix as

$$D_T(x_i, y_i) = D_T(x_i, y_i) + 1 \tag{7}$$

where (x_i, y_i) are the starting co-ordinates of block b_i. Similarly, the matrix D_T is updated according to equation (7) for all the other three frame pairs P_4B_5, P_7B_8 and $P_{10}B_{11}$ with respect to the same spatial axes. After the matrix update step is complete, we perform

$$D_T(x_i, y_i) = \begin{cases} 1 & \text{if } D_T(x_i, y_i) = 4 \quad \forall\ i, \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

In equation (8), if a match is counted for a block with index i for all the four set of frames, then only we count that block for forgery detection. To detect the forgery, we slide a window of size W'_x x W'_y by a pixel rowwise and then columnwise along the matrix D_T, and count the number of matches in each such window. The number of matches is compared to a preset threshold $T_{temporal}$. If in any such window the number of matches is higher than $T_{temporal}$, then we declare the region to be forged.

IV. Discussion

In this Section we discuss the effect of cell size, the dimensionality of descriptors, the number of neighbors used for detection on the forgery detection performance as well as the time complexity of the algorithm.

A. Spatial Forgery Detection Parameters

The cell size plays a significant role on forgery detection performance. Experimentally we find that keeping the cell size to 6x6 for all the videos increases the false positives for the spatial forgery case by $\approx$ 22.75% compared to the case when cell size is adaptively set. While, setting cell size to 8x8 for all videos decreases the true negative rate (detecting forged as forged) by $\approx$ 9% in case of detecting a forged region of size 40x40 scaled by a factor of 1.2 compared to the case when cell size is adaptively set. Therefore, we set cell size based on T_{mean} as discussed in Section III-A.

The number of neighbors chosen for counting the best match of a block b_i also affects the performance to a great extent. In our experiments we find that, using only two neighbors, the false positive rate increases by $\approx$ 45% compared to the case when decision is based on four neighbors.

Now we compute the time complexity of the algorithm. Here, each block's HOG features are compared against the rest of the N - 1 block's HOG features. Although we do not consider the surrounding blocks for matching in the proposed scheme, here for simplicity we assume that the matching is done for N - 1 blocks. For each such matching, m nearest

TABLE I: Spatial forgery detection performance

Scale		Detection accuracy		
$Scale_x$	$Scale_y$	40x40	60x60	80x80
1.1	1.1	93.3	96	96
1.2	1.2	89.6	96	96
1.1	1.4	88.6	94.3	96
1.5	1.25	87.1	91.7	96
1.3	1.3	80.35	91.7	91.7

TABLE II: Spatial forgery detection performance against compression

Bit rate (Mbps)	Detection accuracy		
	40x40	60x60	80x80
1	89.7	96	96
3	89.7	96	96
6	89.9	96	96
9	89.9	96	96

TABLE III: Spatial forgery detection performance against different signal processing operations

Tampering	Detection accuracy		
	40x40	60x60	80x80
Mean filter (3x3)	92.1	96	96
Median filter (3x3)	90.9	96	96
Gaussian (3x3, $\sigma = 1$)	91.7	96	96

TABLE IV: Comparison with Wang et. al.'s [1] scheme for a spatial forged region size 128x128

Bit rate (Mbps)	Detection accuracy		False Positives	
	Proposed	[1]	Proposed	[1]
3	99.8	70.4	0.002	0.006
6	99.8	80.0	0.002	0.004
9	99.8	82.2	0.002	0.002

neighbors of the source are also matched against the m nearest neighbors of the target. Therefore there are $m \times (N-1)$ by N (or $\approx O(m \times N^2)$) comparisons which can be considered to be small for the order of few hundreds of N.

B. Temporal Forgery Detection Parameters

In case of temporal forgery detection, as the forged regions are consistent in shape and size across the frames as they are temporally colocated, cell size need not be adaptively set. However, we find that the dimensionality of descriptors required to distinguish between authentic and forged regions is high as compared to spatial forgery detection. This is due to the fact that temporally colocated regions may exhibit a high correlation. Therefore, a low dimensional descriptor may not be able to differentiate between authentic and forged regions.

In terms of neighbors chosen for counting the best match, the false positive rate increases by $\approx$ 8% when only two neighbors are used relative to the case when decision is based on four neighbors.

Here, each block is compared against the spatially colocated block only. So there are N number of comparison for N number of blocks. Further, m nearest neighbors are also matched giving $\approx O(m \times N)$ comparisons.

V. Experimental Results

In our experiments, we collected 6000 frames from 15 different videos for spatial forgery and 150 GOPs of size 12 frames each for temporal forgery. Original video is compressed at 9mbps using MPEG-2 video codec. The frame resolution is 176x144. Tampering is carried by copying and pasting regions of size 40x40, 60x60 and 80x80 in the same or different frames and videos for spatial forgery and across the frames in spatially colocated regions for temporal forgery. The copied regions are chosen randomly over the frame. The false positives for spatial forgery detection and temporal forgery detection are determined using 6000 authentic frames and 150 authentic GOPs respectively. For determining true negatives, 4320 forged frames for each forged size are generated using 360 authentic frames for varying scale factor, compression rate and filtering. While for temporal forgery detection, an overall 2700 forged GOPs are generated from 150 authentic GOPs under varying compression rate, filtering and forged region size. The parameters for HoG feature generation are - 9 orientation bins evenly spaced over 0° - 360° for spatial forgery detection, 60 orientation bins evenly spaced over 0° - 180° for temporal forgery detection, cellsize M = 6 or 8 as given in Section III-A for spatial forgery and M = 6 for temporal forgery, number of cells in a block is 3x3, separation between two overlapping blocks is 4 pixels, W_x = 176, W_y = 144, W'_x = W'_y = 50, $\tau_{spatial} = 0.87$, $T_{spatial} = 10$, $\tau_{temporal} = 0.95$, $T_{temporal} = 20$, T_{mean} = 0.28, n = 4.

A. Spatial forgery detection performance

Table I gives the detection performance results when the duplicated region is scaled before pasting at a different location. The false positive rate is 0.08. From Table I, it is evident that the detection accuracy increases with the forged region size. This is because, with increased forged patch size, more number of forged blocks are present which increases the number of matches. The detection accuracy (DA) is measured as

$$DA = (TP + TN)/(TP + TN + FP + FN) \qquad (9)$$

where, TP refers to detecting authentic as authentic, TN refers to detecting forged as forged, FP refers to detecting authentic as forged and FN refers to detecting forged as authentic. For a 60x60 and 80x80 size forged patch, the maximum detection accuracy is 96% for $Scale_x$ = $Scale_y$ = 1.1 while minimum is 91.7% for $Scale_x$ = $Scale_y$ = 1.3. Also, we find that a forged region as small as 40x40 can be detected with as high as 93.3% accuracy for $Scale_x$ = $Scale_y$ = 1.1 and as low as 80.35% accuracy for $Scale_x$ = $Scale_y$ = 1.3. Table II, gives the detection performance under varying compression rate. Here, we find that the compression rate has a very mild effect on detection performance as the HoG features are robust against compression. The detection accuracy is $\approx$ 96% for both 60x60 and 80x80 forge regions while $>$ 89% for 40x40 forged region. Table III gives the detection accuracies for filtering operations such as mean, Gaussian and median in presence of 10% scaling. We can see that the detection accuracy is only slightly affected for 40x40 forged region size as compared to the detection accuracy when only scaling is present.

We give a comparison of our results against [1] for region duplication keeping the forged region size same in Table IV. We find that our algorithm performs better against the scheme

TABLE V: Comparison of spatial forgery detection performance under scaling modification

Scheme	Detection accuracy (%)		
	40x40	60x60	80x80
Proposed	87.79	93.7	93.85
[4]	61.2	88.2	90.4

TABLE VI: Temporal forgery detection performance against compression

Bit rate (Mbps)	Detection accuracy		False Positives
	60x60	80x80	
3	83.6	98.15	.037
6	84.5	99.05	.019
9	85.05	99.5	.009

TABLE VII: Temporal forgery detection performance against different signal processing operations

Tampering	Detection accuracy	
	60x60	80x80
Mean filter (3x3)	85.01	99.5
Median filter (3x3)	83.9	99.5
Gaussian (3x3, $\sigma = 1$)	84.6	99.5

TABLE VIII: Comparison with Wang et. al.'s [1] scheme for frame duplication

Bit rate (Mbps)	Detection accuracy		False Positive	
	Proposed	[1]	Proposed	[1]
3	99.5	87.9	.01	.06
6	100	84.8	0	0
9	100	84.4	0	0

given in [1] in terms of both detection accuracy and false positives. This increased performance is due to the reason that HoG features are robust against compression as well as generates distinct features for dissimilar objects.

Table V gives a comparison of the spatial forgery detection of the proposed scheme and [4]. In [4], the forgery detection is performed using SIFT features. From Table V we find that small forged patch size such as 40x40 can not be detected with high accuracy using the algorithm given in [4]. While for a higher forged patch size the detection accuracy is slightly better for the proposed scheme. And therefore, HoG features based detection gives better performance under scaling modification.

B. *Temporal forgery detection performance*

Table VI gives the detection performance of temporal forgery under varying compression rate. The false positive rates respective to the three bit rates 3, 6 and 9 mbps are 0.037, 0.019 and 0.009. Here, we find that false positives differ for different bit rates as the dimensionality of HoG descriptors is high and HoG descriptors are not as robust in such a case as compared to low dimensional HoG descriptors. It is clear from Table VI that with the increasing forged region size the detection performance also increases. And the detection performance is slightly affected by the compression rate. The detection accuracy is $\approx$ 84.5% for 60x60 forged region while $\approx$ 99% for 80x80 forged region. Table VII shows that the detection performance is good under different filtering operations - mean, Gaussian and median.

We compare the results against [1] for frame duplication keeping the false positive rate similar in Table VIII. From Table VIII we can say that our algorithm performs better against the scheme given in [1]. In addition, the proposed algorithm detects the frame duplication using only the frames in a GOP whereas the algorithm given in [1] needs 30 frames to detect frame duplication.

VI. Conclusions

In this paper we propose a novel video forgery detection technique using HoG features and video compression properties. The parameter cellsize of the Hog feature generation is set adaptively to reduce the false positive rate and increase the detection accuracy for spatial forgery detection. In temporal forgery detection, the frames with high correlation for duplicated regions compared to authentic regions are selected for detection purpose. Experimental results show that the proposed algorithm gives good detection accuracy under unfavorable operations such as compression, scaling and filtering for spatial forgery detection while compression and filtering for temporal forgery detection. Further, the proposed algorithm performs better than the popular video forgery detection technique which deals with the copy-paste tampering.

References

[1] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting duplication," in *Proc 9th workshop on ACM Multimedia & Security*, 2007, pp. 35–42.

[2] M. Kobayashi, T. Okabe, and Y. Sato, "Detecting video forgeries based on noise characteristics," *Lecture Notes in Computer Science, Advances in Image and Video Technology*, vol. 5414, pp. 306–317, 2009.

[3] I. Amerini, L. Ballan, R. Caldelli, A. Del Bimbo, and G. Serra, "A sift-based forensic method for copy-move attack detection and transformation recovery," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 3, pp. 1099–1110, Sep. 2011.

[4] X. Pan and S. Lyu, "Region duplication detection using image feature matching," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 857–867, Dec. 2010.

[5] W. Li and N. Yu, "Rotation robust detection of copy-move forgery," in *Proc. IEEE International Conference on Image Processing ICIP'10, 2010*, pp. 2113–2116.

[6] T. Van Lanh, K. Chong, S. Emmanuel, and M. Kankanhalli, "A survey on digital camera image forensic methods," in *Proc. IEEE International Conference on Multimedia and Expo ICME'07, 2007*, pp. 16–19.

[7] W. Wang and H. Farid, "Exposing digital forgeries in interlaced and deinterlaced video," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 438–449, Sep. 2007.

[8] C. Hsu, T. Hung, C. Lin, and C. Hsu, "Video forgery detection using correlation of noise residue," in *Proc. 10th Workshop on IEEE Multimedia Signal Processing*, 2008, pp. 170–174.

[9] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting double quantization," in *Proc. 11th ACM workshop on Multimedia and Security, 2009*, pp. 39–48.

[10] W. Chen and Y. Shi, "Detection of double mpeg compression based on first digit statistics," *Lecture Notes in Computer Science, Digital Watermarking*, vol. 5450, pp. 16–30, 2009.

[11] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," *Computer Vision–ECCV 2006*, pp. 404–417, 2006.

[12] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. CVPR'05, 2005*.

[13] B. Saha and N. Ray, "Image thresholding by variational minimax optimization," *Pattern Recognition*, vol. 42, no. 5, pp. 843–856, 2009.

[14] B. Haskell, A. Puri, and A. Netravali, *Digital video: an introduction to MPEG-2*. Kluwer Academic, 1997.