Exposing Digital Forgeries in Video by Detecting Duplication

Weihong Wang
Department of Computer Science
Dartmouth College
Hanover, NH 03755
whwang@cs.dartmouth.edu

Hany Farid
Department of Computer Science
Dartmouth College
Hanover, NH 03755
farid@cs.dartmouth.edu

ABSTRACT

With the advent of high-quality digital video cameras and sophisticated video editing software, it is becoming increasingly easier to tamper with digital video. A common form of manipulation is to clone or duplicate frames or parts of a frame to remove people or objects from a video. We describe a computationally efficient technique for detecting this form of tampering.

Categories and Subject Descriptors

I.4 [**Image Processing**]: Miscellaneous

General Terms

Security

Keywords

Digital Tampering, Digital Forensics

1. INTRODUCTION

In the popular movie *Speed*, the characters created a doctored video by simply duplicating a short sequence of frames, Figure 1. In so doing, they were able to conceal activity on the bus. This common and relatively easy to perform manipulation can be used to remove people or objects from a video sequence or simply remove an undesired event from a video. When done carefully, this form of tampering can be very difficult to detect.

Techniques for detecting image duplication have previously been proposed [2, 5]. These techniques, however, are computationally too inefficient to be applicable to a video sequence of even modest length. We describe two computationally efficient techniques for detecting duplication. In the first, we show how to detect duplicated frames, and in the second, we show how to detect duplicated regions across frames. We also describe how these techniques can be adapted to yield a more computationally efficient image duplication algorithm. In each case we show the efficacy on several real video sequences.

Figure 1: By duplicating frames on a surveillance video, the characters in the movie *Speed* create a doctored video to conceal activity on the bus.

2. METHODS

2.1 Frame Duplication

Shown in Figure 2 is a portion of a video where three frames are duplicated to remove the flight attendant. This type of manipulation is fairly easy to perform and can be difficult to detect visually particularly in a video taken from a stationary surveillance camera. Given a video sequence, $f(x, y, t), t \in [0, L-1]$, of length L, it would be computationally intractable to search for duplication by comparing all possible sub-sequences of arbitrary length and positions in time. An additional difficulty in searching for duplication is that compression artifacts (e.g., MPEG) introduce differences between initially identical duplicated frames. We, therefore, describe a computationally efficient algorithm for detecting duplicated video frames that is robust to compression artifacts.

Our basic approach is to partition a full-length video sequence into short overlapping sub-sequences. A compact

ACM Multimedia and Security Workshop '07 Dallas, Texas

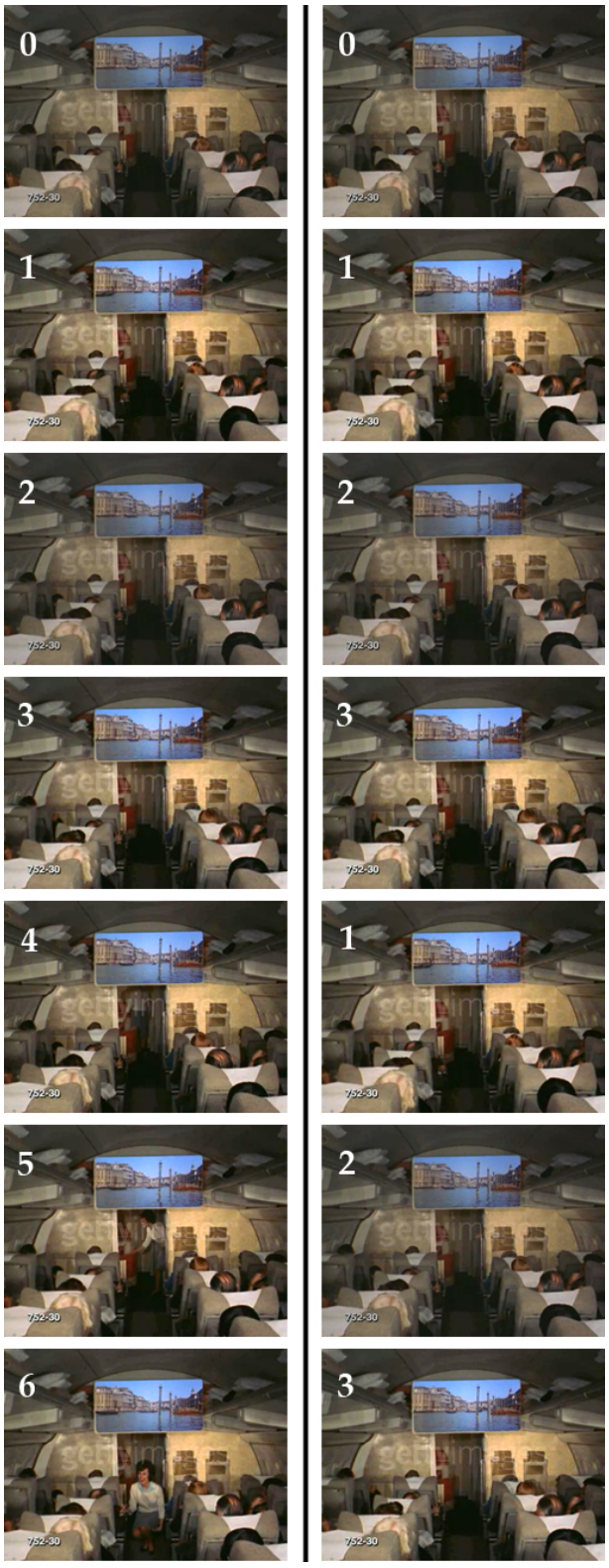

Figure 2: Shown in the left column are seven frames of an original video. Shown on the right are the results of duplicating three frames so as to remove the flight attendant from the scene.

and efficient to compute representation that embodies both the temporal and spatial correlations in each sub-sequence is then extracted and compared throughout the entire video. Similarity in the temporal and spatial correlations are then used as evidence of duplication.

Throughout, we will use the correlation coefficient as a measure of similarity. The correlation coefficient between two vectors $\vec{u}$ and $\vec{v}$ (or matrices or images strung out in vector form) is given by:

$$C(\vec{u},\vec{v}) \quad = \quad \frac{\sum_i (u_i-\mu_u)(v_i-\mu_v)}{\sqrt{\sum_i (u_i-\mu_u)^2}\sqrt{\sum_i (v_i-\mu_v)^2}}, \qquad (1)$$

where u_i and v_i are the i^{th} element of $\vec{u}$ and $\vec{v}$, and μ_u and μ_v are the respective means of $\vec{u}$ and $\vec{v}$.

Denote a sub-sequence which starts at time τ and of length n frames as:

$$S_\tau(t) \quad = \quad \{f(x,y,t+\tau) \mid t \in [0,n-1]\}. \qquad (2)$$

We define the temporal correlation matrix T_τ to be a $n \times n$ symmetric matrix whose $(i,j)^{th}$ entry is the correlation coefficient between the i^{th} and the j^{th} frame of the sub-sequence, $C(S_\tau(i), S_\tau(j))$. This temporal correlation matrix embodies the correlations between all pairs of frames in a sub-sequence. If, for example, there is little change across the sub-sequence then the matrix entries will each have a value near 1, whereas, if there is significant change, then the matrix entries will have values closer to -1.

The spatial correlations of each frame within a sub-sequence, $S_\tau(t)$, can be embodied in a similar spatial correlation matrix, $B_{\tau,k}$, for $k \in [0, n-1]$. To compute this matrix, a frame is first tiled with m non-overlapping blocks. The spatial correlation matrix is a $m \times m$ symmetric matrix whose $(i,j)^{th}$ entry is the correlation coefficient between the i^{th} and j^{th} blocks.

The temporal and spatial correlation matrices, embodying the correlations of short sub-sequences, are used to detect duplicated frames in a full-length video. In the first stage of detection, the temporal correlation matrix for all overlapping (by one frame) sub-sequences is computed. The correlation coefficient between pairs of these matrices, T_{τ_1} and T_{τ_2}, is then computed, $C(T_{\tau_1}, T_{\tau_2})$. Any two sub-sequences with a correlation above a specified threshold (close to 1) is considered a candidate for duplication. In the second stage, the spatial correlation matrices, $B_{\tau_1,k}$ and $B_{\tau_2,k}$ for $k \in [0, n-1]$, of these candidate sub-sequences are compared. If the correlation coefficient, $C(B_{\tau_1,k}, B_{\tau_2,k})$, between all pairs of these matrices is above a specified threshold (close to 1), then the sub-sequences are considered to be temporally and spatially highly correlated, and hence duplicated. Multiple sub-sequences with the same temporal offset are combined to reveal the full extent of the duplicated frames. See Figure 3 for a detailed algorithmic description.

A stationary video surveillance camera recording a largely static scene will seemingly give rise to numerous duplications. To avoid this problem, we ignore sub-sequences when the minimum element in the temporal correlation matrix T_τ is above a specified threshold (close to 1). Such a correlation matrix implies that all pairs of frames in the sub-sequence are nearly identical, and hence the scene is static.

FRAMEDUP($f(x, y, t)$)

```
▷ f(x, y, t): video sequence of length N

▷ n: sub-sequence length
▷ γ_m: minimum temporal correlation threshold
▷ γ_t: temporal correlation threshold
▷ γ_s: spatial correlation threshold

for τ = 1 : N − (n − 1)
    do S_τ = {f(x, y, t + τ) | t ∈ [0, n − 1]}
       build T_τ ▷ temporal correlation

for τ_1 = 1 : N − (2n − 1)
    do for τ_2 = τ_1 + n : N − (n − 1)
          do if (min(T_τ1) > γ_m & C(T_τ1, T_τ2) > γ_t)
                build B_τ1,k ▷ spatial correlation
                build B_τ2,k ▷ spatial correlation
                if ( C(B_τ1,k, B_τ2,k) > γ_s ) ▷ ∀k
                   do ▷ Frame Duplication at τ_1
```

Figure 3: Pseudo-code for detecting frame duplication.

2.2 Region Duplication

In the previous section we showed how to detect duplicated frames in a video sequence. Shown in Figure 4 is an example where only part of several frames are duplicated. Note that this form of duplication will not typically be detected using the algorithm described above. Here we describe how this form of tampering can be efficiently detected. We begin by assuming that a subset of the pixels in $f(x, y, \tau_1)$ of unknown location are duplicated and placed in another frame at a different spatial location, $f(x + \Delta_x, y + \Delta_y, \tau_2)$. We also assume that these pixels undergo no other geometric or significant intensity changes. Next we describe how, given a pair of frames, to estimate the shift (Δ_x, Δ_y), and then how to verify that this estimated shift corresponds to a duplication.

2.2.1 *Stationary Camera*

For simplicity, we begin by assuming that our video was taken with a stationary camera (this assumption will be relaxed later). Given a pair of frames $f(x, y, \tau_1)$ and $f(x, y, \tau_2)$, we seek to estimate a spatial offset (Δ_x, Δ_y) corresponding to a duplicated region between these frames. Phase correlation [4, 1] affords a robust and computationally efficient mechanism for this estimation. We begin by defining the normalized cross power spectrum:

$$P(\omega_x, \omega_y) = \frac{F(\omega_x, \omega_y, \tau_1)F^*(\omega_x, \omega_y, \tau_2)}{\|F(\omega_x, \omega_y, \tau_1)F^*(\omega_x, \omega_y, \tau_2)\|}, \quad (3)$$

where $F(\omega_x, \omega_y, \tau_1)$ and $F(\omega_x, \omega_y, \tau_2)$ are the Fourier transforms of the two frames, $*$ is complex conjugate, and $\|\cdot\|$ is complex magnitude. Let $p(x, y)$ be the inverse Fourier transform of $P(\omega_x, \omega_y)$. Phase correlation techniques estimate spatial offsets by extracting peaks in $p(x, y)$. In our case, since the video camera is stationary, we expect a significant peak at the origin $(0, 0)$. Peaks at other positions denote secondary alignments that may be the result of duplication (and translation). Region duplication can therefore

Figure 4: Shown in the left column are seven frames of an original video. Shown on the right are the results of region duplication so as to add another zebra into the scene.

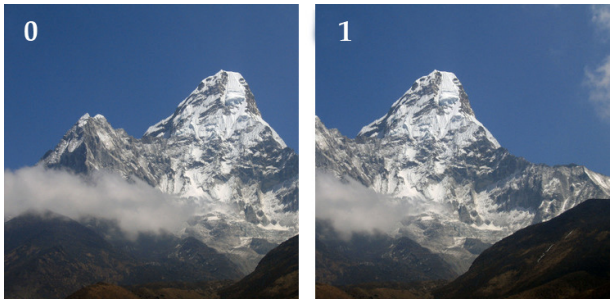

Figure 5: Two frames of a a mountain peak as the camera pans from left to right.

be detected by simply extracting peaks in $p(x, y)$ that are not at or near the origin. The spatial location of a peak corresponds to the spatial offset (Δ_x, Δ_y).

For each spatial offset, (Δ_x, Δ_y), we compute the correlation between $f(x, y, \tau_1)$ and $f(x + \Delta_x, y + \Delta_y, \tau_2)$ to determine if an offset is likely to correspond to a duplicated region. More specifically, each frame is tiled into 16×16 overlapping (by 1 pixel) blocks, and the correlation coefficient between each pair of corresponding blocks is computed, Equation (1). All blocks whose correlation is above a specified threshold (close to 1) are flagged as possibly belonging to a duplicated region. In order to remove spurious correlations a binary image is constructed whose pixel value is 1 at the center of each block whose correlation is above a specified threshold (close to 1), and 0 elsewhere. This binary image is subjected to a connected components labeling using a connectivity of eight [3]. Any remaining connected regions with pixel value 1 whose area is above a specified threshold are considered to be the result of region duplication.

See REGIONDUP1 in Figure 6 for a detailed algorithmic description.

2.2.2 *Moving Camera*

The above algorithm describes how to detect region duplication from a stationary camera. We next describe how to extend this technique to a largely stationary scene filmed with a moving camera (e.g., a panning surveillance camera). Given our current approach, there is no way of differentiating between region duplication from a stationary camera, and no duplication from a moving camera where objects simply appear in different spatial locations due to camera motion. Shown in Figure 5 for example is a scene that will appear to our above algorithm to contain region duplication because the mountain peak appears in different positions on successive frames. A similar problem may arise when an object moves across an otherwise static scene – we do not consider this case here, although we do note that nearby objects will change appearance as they move away from or towards the camera, and hence will not necessarily be seen as duplications.

In order to contend with this ambiguity, we compute a rough measure of the camera motion to determine if the field of view between $f(x, y, \tau_1)$ and $f(x, y, \tau_2)$ are sufficiently different so as to not contain any overlap, and hence the same objects. The camera motion between successive frames is approximated as a global translation. The motion between all pairs of successive frames between time τ_1 and τ_2 are computed and summed to give an overall estimate of camera motion. If, for example, between τ_1 and τ_2, the camera continually moves to the right displacing 10 pixels in the image, then the estimated motion in the horizontal direction will be $10(\tau_2 - \tau_1)$. If, on the other hand, the camera pans to the right and then to the left returning to its starting position, then the estimated motion will be close to 0. In the latter case, region duplication cannot be detected because of the frame overlap between τ_1 and τ_2, whereas in the former case, assuming an overall large camera motion, a detected region duplication is unlikely to be caused by an overlap in the field of view. Phase correlation, as described above, is used to estimate this camera motion – the largest peak is assumed to correspond to the camera motion. If the accumulated motion is within a specified factor of the size of a frame, then it is assumed that the frames at time τ_1 and τ_2 share a common field of view, and detection of region duplication is not applicable. If, on the other hand, the frames do not share a field of view, then we detect region duplication as described above. That is, peaks in the phase correlation between $f(x, y, \tau_1)$ and $f(x, y, \tau_2)$ are considered as candidate duplications, the correlation at the estimated offsets is computed to verify duplication, and the connected components are computed.

See REGIONDUP2 in Figure 6 for a detailed algorithmic description.

3. RESULTS

Shown in Figure 7 are every 1000^{th} frame from two video sequences, each captured with a SONY-HDR-HC3 digital video camera. Each frame is 480×720 pixels in size, and the length of each sequence is $10,000$ frames (approximately 5 minutes in length). For the first video, the camera was placed on a tripod and kept stationary throughout. For the second video, the camera was hand-held as the observer walked through the Dartmouth College campus. These videos were subjected to various forms of duplication, the results of which are reported below. Throughout, each frame was converted from color to grayscale.

3.1 Frame Duplication

Frame duplication was simulated by selecting a random location in each video sequence, and duplicating 200 frames to another non-overlapping position in the video sequence. This entire process was repeated 100 times, each time randomly selecting a different region to be duplicated to a new random location.

The duplication algorithm was configured as follows: for run-time considerations, each frame was first down-sampled by a factor of 8; the temporal correlation matrix was computed from sub-sequences of length $n = 30$ frames; the minimum correlation for classifying the frames of a subsequence as stationary was $\gamma_m = 0.96$; the temporal correlation threshold was $\gamma_t = 0.99$; the spatial correlation threshold was $\gamma_s = 0.99$; and the spatial correlation matrix was computed for each frame partitioned into 15×15 pixel blocks (i.e., $m = 24$);

For the uncompressed video taken from a stationary camera, an average of 84.2% of the duplicated frames were detected with only an average of 0.03 false positives (a non-duplicated frame classified as duplicated). For the uncompressed video taken from a moving camera, 100% of the duplicated frames were detected with 0 false positives. The

Figure 7: Sample frames from two video sequences captured from a stationary camera (top) and hand-held camera (bottom).

improvement in performance over the stationary camera sequence is because duplication cannot be detected in largely static frames which occur more often with a stationary camera.

To test the sensitivity to compression, each video was subjected to MPEG compression with a bit rate of either 3, 6, or 9 Mbps. Below are the average detection accuracies and false positives (averaged over 50 random trials).

	detection			false positive		
video	3	6	9	3	6	9
stationary	87.9%	84.8%	84.4%	0.06	0.0	0.0
moving	86.8%	99.0%	100.0%	0.0	0.0	0.0

These results show that the frame duplication algorithm is effective in detecting duplications, and is reasonably robust to compression artifacts.

Running on a 3.06 GHz Intel Xeon processor, a $10,000$ frame sequence requires 45 minutes of processing time. This run-time could be greatly reduced by distributing the calculations and comparisons of the temporal and spatial correlation matrices to multiple nodes of a cluster. The run-time complexity of this algorithm is $O(N^2)$ where N is the total number of frames. The run-time is dominated by the pairwise comparison of temporal and spatial correlation matrices.

3.2 Region Duplication

Region duplication was simulated by selecting a local region from one frame and duplicating it into a different location in another frame. The regions were of size 64×64, 128×128, and 256×256. For each region size, this process was repeated 500 times, each time randomly selecting a different region location and pair of frames.

3.2.1 Stationary Camera

Assuming that the pair of frames containing the duplicated regions are known, the duplication algorithm, REGIONDUP1, was configured as follows: the phase correlation threshold was $\gamma_p = 0.015$; the phase correlation offset was $\gamma_o = 15$ pixels; the block neighborhood size was $b = 16$ pixels; the block correlation threshold was $\gamma_b = 0.7$; and the minimum area threshold was $\gamma_a = 1,500$ pixels.

To test the sensitivity to compression, each video was subjected to MPEG compression with a bit rate of either 3, 6, or 9 Mbps. Below are the average detection accuracies and false positives (a non-duplicated region classified as duplicated) for the video from the stationary camera (top row of Figure 7).

region	detection			false positive		
size	3	6	9	3	6	9
64	8.6%	23.4%	35.0%	0.004	0.000	0.000
128	70.4%	80.0%	82.2%	0.006	0.004	0.002
256	100.0%	100.0%	100.0%	0.000	0.000	0.000

These results show that the region duplication algorithm is effective in detecting relatively large regions, but, not surprisingly, struggles to find small regions (a region of size 64×64 occupies just over 1% of the pixels in a frame of size 480×720). In addition, detection is reasonably robust

```
REGIONDUP1(f(x, y, τ1), f(x, y, τ2))
 1  ▷ f(x, y, τ1), f(x, y, τ2): two video frames
 2
 3  ▷ γp: phase correlation threshold
 4  ▷ γo: phase correlation offset threshold
 5  ▷ b: block neighborhood size
 6  ▷ γb: block correlation threshold
 7  ▷ γa: minimum area threshold
 8
 9  compute p(x, y) ▷ phase correlation
10  for each (Δx, Δy), s.t. p(Δx, Δy) > γp and
                            (|Δx| > γo or |Δy| > γo)
11        do for each i, j
12              do i' = i + Δx
13                 j' = j + Δy
14                 b1 = f(i : i + b − 1, j : j + b − 1, τ1)
15                 b2 = f(i' : i' + b − 1, j' : j' + b − 1, τ2)
16                 if ( C(b1, b2) > γb ) ▷ correlation
17                       do mask(i, j) = 1
18           mask = connected_components(mask)
19           if ( area( mask(x, y) ) > γa )
20                 do ▷ Region Duplication at (x, y)
```

```
REGIONDUP2(f(x, y, t), τ1, τ2)
 1  ▷ f(x, y, t): video sequence
 2  ▷ τ1, τ2: two frame indices
 3  ▷ frame size: Nx × Ny pixels
 4
 5  ▷ s: motion camera offset threshold
 6
 7  for τ = τ1 : τ2 − 1
 8        do compute p(x, y) for f(x, y, τ) and f(x, y, τ + 1)
 9           (δx, δy) = arg max_{x,y}(p(x, y))
10           (Δx, Δy) = (Δx, Δy) + (δx, δy)
11  if ( Δx > sNx and Δy > sNy )
12        do REGIONDUP1( f(x, y, τ1), f(x, y, τ2) )
```

Figure 6: Pseudo-code for detecting region duplication from a stationary (REGIONDUP1) and moving camera (REGIONDUP2).

to compression artifacts, and overall, the number of false positives is small.

Shown in Figure 8 are two frames with region duplication from a 800 frame video taken with a stationary camera (the zebra was copied from one frame to another). Assuming the first frame known, the entire video sequence was searched for duplication. Shown in this same figure are the results of REGIONDUP1. In the third panel is the detected duplication and in subsequent panels are false positives corresponding to the largely uniform sky. These false positives could be reduced because the duplicated regions are almost entirely overlapped. Note also that the phase correlation for the actual duplication was 0.21, while that for the false positives was on the order of 0.05. As such, the phase correlation can be used to rank-order the suspicious regions.

3.2.2 Moving Camera

A new video of length 4,200 frames was captured to test region duplication from a moving camera. The camera was hand-held as the operator walked along the sidewalk for one half of the sequence and then back to his starting position. In this way, we are able to select frames with and without overlap.

Region duplication was applied to two random frames from the first 2,000 frames of the video sequence that were at least 1,000 frames apart, and hence had no overlap in their views. This process was repeated 500 times, each time randomly selecting a pair of frames and different region locations.

Assuming that the pair of frames containing the duplication regions is known, the duplication algorithm REGIONDUP2 was configured as follows: the camera motion offset thresholds was $s = 1.2$ and the parameters to REGIONDUP1 were the same as that described above except that the phase correlation offset threshold was set to $\gamma_o = -1$ to allow all possible phase correlations to be considered. Below are the average detection accuracies and false positives for the video from the moving camera.

region	detection			false positive		
size	3	6	9	3	6	9
64	16.6%	30.8%	39.4%	0.012	0.006	0.004
128	47.8%	67.2%	72.8%	0.002	0.000	0.000
256	72.4%	83.4%	87.8%	0.004	0.004	0.002

Note that the detection accuracies are, on average, not as good as those from the stationary camera. The reason for this is that the compression artifacts are more severe in the presence of motion which introduce larger differences in the originally duplicated regions. Note, however, that for the 64×64 region size, the performance is slightly better than in the stationary camera. The reason for this is that the phase correlation is more likely to detect the duplicated region when the overall background is not stationary.

The above experiment was repeated where the pairs of frames were selected so that they shared a portion of their field of view. Here we tested the ability of our camera motion estimation algorithm to determine that these pairs of frames were not suitable for duplication detection. Below are the percentage of frames (averaged over 500 random trials) that were correctly classified.

region	detection		
size	3	6	9
64	100.0%	100.0%	100.0%
128	100.0%	100.0%	100.0%
256	97.6%	100.0%	100.0%

This result shows that the camera motion estimation is reliable, allowing us to disambiguate between duplication and an object simply appearing twice in a scene due to camera motion.

In our final experiment, we tested the ability of our technique to detect region duplication when only one of the frames containing duplication was known. Fifty sequences with region duplication were generated from the first 2,000 frames of the 4,200 frame video described above. In each case, two random frames that were at least 1,000 frames apart were selected, and a randomly selected region of size 128x128 was copied from one frame to a different randomly selected location of the second frame. Each sequence was subjected to MPEG compression with a bit rate 9 Mbps. Assuming that one of the duplicated frames, τ_1, is known,

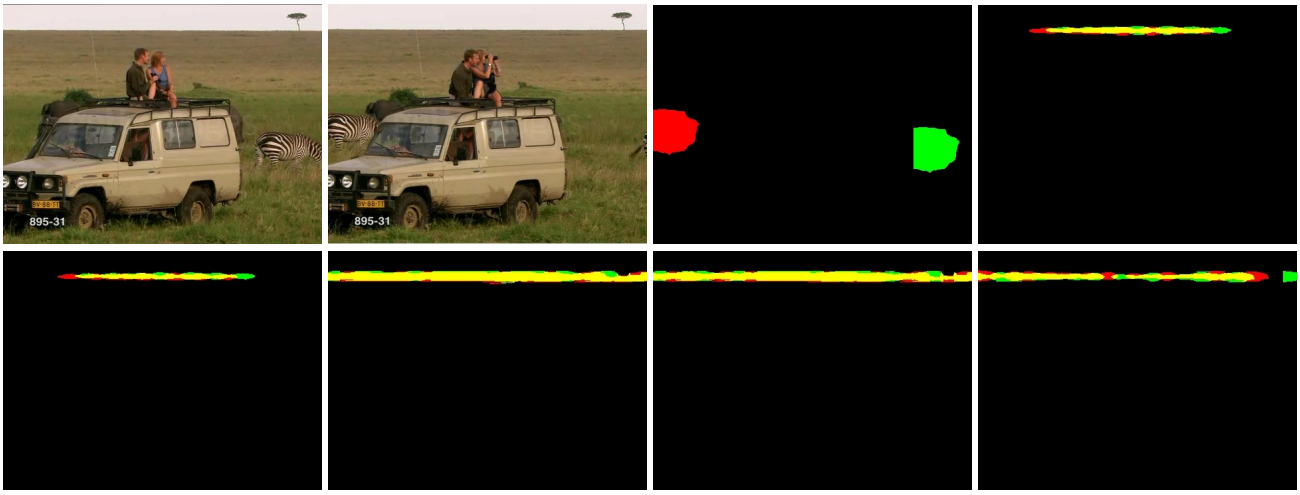

Figure 8: Shown in the first panel is one frame of a 800 frame video taken from a stationary camera. Shown in the second panel is the result of region duplication where the zebra from the first panel was copied into this frame. Shown in the subsequent panels are the detection results: in the third panel is the duplication corresponding to the zebra, and in the subsequent panels are false positives corresponding to the largely uniform sky.

the function REGIONDUP2 was applied. The block correlation threshold was increased from $\gamma_b = 0.7$ to 0.9 so as to reduce the number of false positives. All other parameters remained unchanged. The duplicated regions were detected 94% of the time, with an average of 0.1 false positives.

Running on a 3.06 GHz Intel Xeon processor, region duplication between two pairs of frames requires about 1 second of processing time. The run-time complexity of this algorithm is $O(P \log(P))$ where P are the total number of image pixels. The run-time is dominated by the Fourier transform needed to compute phase correlation.

3.3 Image Duplication

With only a few minor adjustments, the region duplication algorithm described above can be adapted to detect tampering in a single static image (or video frame). This approach is more computationally efficient than our earlier approach to detecting duplication in images [5].

An image is first partitioned into non-overlapping sub-images of equal size. The function REGIONDUP1 is then evaluated for all pairs of sub-images. This approach will not detect a duplicated region if it is contained within a single sub-image. To contend with this, each sub-image can be processed recursively to detect more and more spatially localized duplication.

Shown in Figure 9 are several original images (left) and the results of duplication (right) to remove an object or person from the image. Also shown in this figure are the results of duplication where the duplicated image was saved with JPEG compression of 100 (left) and 50 (right), on a scale of 0 to 100. In each example, REGIONDUP1 was configured as follows: the phase correlation threshold was $\gamma_p = 0.05$; the phase correlation offset threshold was $\gamma_o = -1$ to allow all possible phase correlations to be considered; the block neighborhood size was $b = 16$ pixels; the block correlation threshold was $\gamma_b = 0.8$; and the minimum area threshold was $\gamma_a = 1,000$ pixels. The image was partitioned into 4 sub-images (2×2), and each was recursively processed 2 times. In each example the duplication was detected.

For a grayscale image of size 512×512 pixels, the run-time on a 3.06 GHz Intel Xeon processor is approximately 1 second as compared to 10 seconds for our earlier implementation [5].

4. DISCUSSION

We have described two techniques for detecting a common form of tampering in video. The first technique detects entire frame duplication and the second detects if only a portion of one or more frames was duplicated. In each case, central to the design of the algorithms was the issue of computational cost since even a video of modest length can run into the tens of thousands of frames. In addition to being computationally efficient, each algorithm can easily be distributed to a cluster for more efficient processing. Results from both real and simulated tampering suggests that these algorithms can detect most duplications in both high- and low-quality compressed video, with relatively few false positives. We have also shown how these techniques can be adapted to efficiently detect duplication in a single image or video frame.

We expect these techniques, in conjunction with earlier work in video forensics [6, 7] to make it increasingly harder to doctor digital video.

Acknowledgments

This work was supported by a Guggenheim Fellowship, a gift from Adobe Systems, Inc., a gift from Microsoft, Inc., a grant from the United States Air Force (FA8750-06-C-0011), and by the Institute for Security Technology Studies at Dartmouth College under grant 2005-DD-BX-1091 from the Bureau of Justice Assistance and Award Number 2006-CS-001-000001 from the U.S. Department of Homeland Security. Points of view or opinions in this document are those of the author and do not represent the official position or policies of the U.S. Department of Justice, the U.S. Department of Homeland Security, or any other sponsor.

5. REFERENCES

[1] E. D. Castro and C. Morandi. Registration of translated and rotated images using finite fourier transforms. *IEEE Transactions on Pattern Analysis Machine Intelligence*, 9(5):700–703, 1987.

[2] J. Fridrich, D. Soukal, and J. Lukáš. Detection of copy-move forgery in digital images. In *Proceedings of Digital Forensic Research Workshop*, August 2003.

[3] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, New Jersey, 2002.

[4] C. Kuglin and D. Hines. The phase correlation image alignment method. In *IEEE International Conference On Cybernetics and Society*, pages 163–165, New York, September 1975.

[5] A. Popescu and H. Farid. Exposing digital forgeries by detecting duplicated image regions. Technical Report TR2004-515, Department of Computer Science, Dartmouth College, 2004.

[6] W. Wang and H. Farid. Exposing digital forgeries in video by detecting double MPEG compression. In *ACM Multimedia and Security Workshop*, Geneva, Switzerland, 2006.

[7] W. Wang and H. Farid. Exposing digital forgeries in interlaced and de-interlaced video. *IEEE Transactions on Information Forensics and Security*, 2007 (in press).

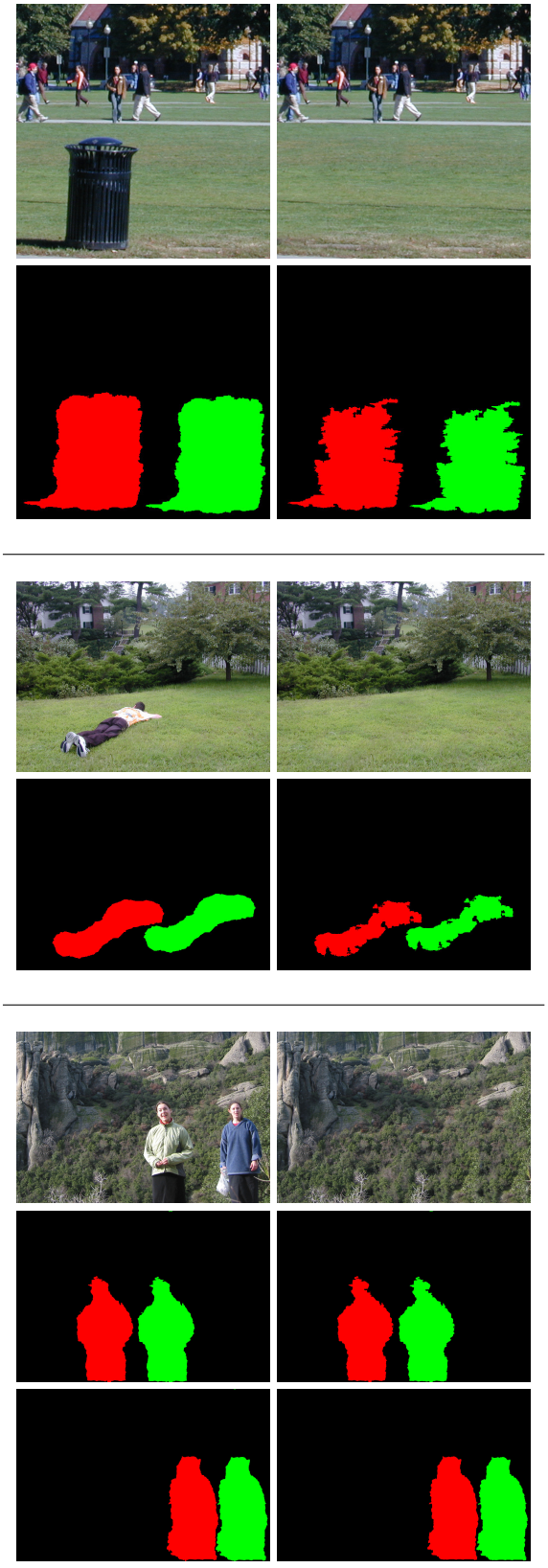

Figure 9: Shown are an original image (left) and the result of tampering by duplication (right), and the results of detecting duplication for two JPEG qualities of 100 (left) and 50 (right). The example in the last row has two different duplications.