Exposing Digital Forgeries in Interlaced and De-Interlaced Video

Weihong Wang, *Student Member, IEEE,* and Hany Farid, *Member, IEEE*

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 growing number of video surveillance cameras are also giving rise to an enormous amount of video data. The ability to ensure the integrity and authenticity of this data poses considerable challenges. We describe two techniques for detecting traces of tampering in de-interlaced and interlaced video. For de-interlaced video, we quantify the correlations introduced by the camera or software de-interlacing algorithms and show how tampering can disturb these correlations. For interlaced video, we show that the motion between fields of a single frame and across fields of neighboring frames should be equal. We propose an efficient way to measure these motions and show how tampering can disturb this relationship.

I. Introduction

Popular websites such as YouTube have given rise to a proliferation of digital video. Combined with increasingly sophisticated users equipped with cellphones and digital cameras that can record video, the Internet is awash with digital video. When coupled with sophisticated video editing software, we have begun to see an increase in the number and quality of doctored video. While the past few years has seen considerable progress in the area of digital image forensics (e.g., [1], [2], [3], [4], [5], [6], [7]), less attention has been payed to digital video forensics. In one of the only papers in this area, we previously described a technique for detecting double MPEG compression [8]. While this technique can determine if a video was re-saved after its original recording, it cannot explicitly determine if it was tampered with or simply re-saved from within a video editing software after an innocuous viewing or cropping. Here we describe two related techniques that explicitly detect spatially and temporally localized tampering.

Most video cameras do not simultaneously record the even and odd scan lines of a single frame. Instead, one-half of the scan lines are recorded at time t, while the other half are recorded at time $t+1$. In an interlaced video, these scan lines are simply combined to create a full frame. While this approach allows for better temporal sampling, it introduces spatial "combing" artifacts for quickly moving objects. In order to minimize these artifacts, a de-interlaced video will combine the even and odd lines in a more sensible way, usually relying on some form of spatial and temporal interpolation.

Here we describe two techniques for detecting tampering in de-interlaced and interlaced video. For de-interlaced video, we quantify the correlations introduced by the camera or software de-interlacing algorithms and show how tampering can disturb these correlations. For interlaced video, we show that the motion between fields of a single frame and across fields of neighboring frames should be equal. We propose an efficient way to measure these motions and show how tampering can disturb this relationship. Both algorithms are then adapted slightly to detect frame rate conversion that may occur when a video is edited and re-saved from within a video editing software. In all cases, we show the efficacy of our approach on simulated and visually plausible forgeries.

II. De-Interlaced Video

In an interlaced video sequence, a field, $f(x,y,t)$, at time t contains only one-half of the scan lines needed for a full-resolution frame, $F(x,y,t)$. The second half of the scan lines, $f(x,y,t+1)$, are recorded at time $t+1$. An interlaced video, created by simply weaving these fields together, yields artifacts due to motion. A de-interlaced video sequence tries to minimize these artifacts by combining one or more interlaced fields to create a full-resolution video sequence, Figure 1 (see [9] for a general overview and [10], [11], [12], [13] for specific and more advanced approaches.).

There are two basic types of de-interlacing algorithms: field combination and field extension. Given an interlaced video of length T fields, a field combination de-interlacing algorithm yields a video of length $T/2$ frames, where neighboring fields in time are combined into a single frame. A field extension de-interlacing algorithm yields a video of length T, where each field in time is extended into one frame. We will constrain ourselves to field extension algorithms. For notational simplicity, we assume that the odd/even scan lines are inserted into frames $F(x,y,t)$ with odd/even values of t, respectively. Below we describe several field extension de-interlacing algorithms, some of which are commonly found in commercial video cameras. We then describe how to model and estimate the spatial and temporal correlations that result from de-interlacing, and how to exploit these de-interlacing correlations to detect tampering.

W. Wang (whwang@cs.dartmouth.edu) and H. Farid (farid@cs.dartmouth.edu) are with the Department of Computer Science at Dartmouth College, 6211 Sudikoff Lab, Hanover NH 03755.

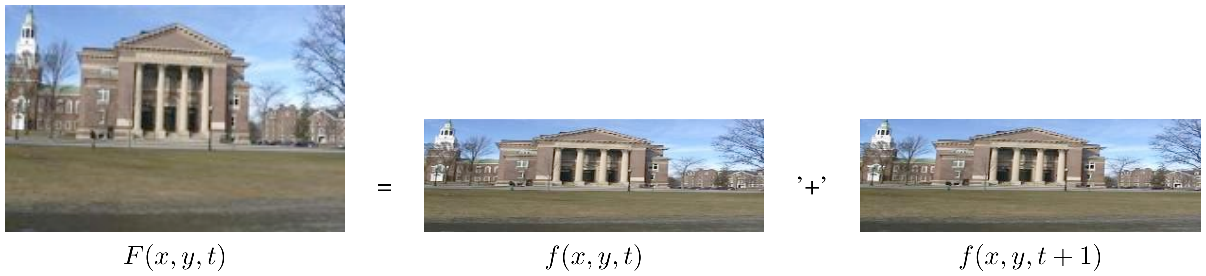

Fig. 1. One half of the scan lines, $f(x, y, t)$, of a full video frame are recorded at time t, and the other half of the scan lines, $f(x, y, t+1)$, are recorded at time $t+1$. A de-interlaced frame, $F(x, y, t)$, is created by combining these two fields to create a full frame.

A. De-Interlacing Algorithms

1) Line Repetition: In this simplest of de-interlacing algorithms, Figure 2(a), the scan lines of each field, $f(x, y, t)$, are duplicated to create a full frame, $F(x, y, t)$:

$$F(x, y, t) \quad = \quad f(x, \lceil y/2 \rceil, t). \tag{1}$$

While easy to implement, the final de-interlaced video suffers from having only one-half the vertical resolution as compared to the horizontal resolution.

2) Field Insertion: In this de-interlacing algorithm, Figure 2(b), neighboring fields, $f(x, y, t)$, are simply combined to create a full frame, $F(x, y, t)$. For odd values of t:

$$F(x, y, t) \quad = \quad \begin{cases} f(x, (y+1)/2, t) & y \bmod 2 = 1 \\ f(x, y/2, t+1) & y \bmod 2 = 0 \end{cases}, \tag{2}$$

and for even values of t:

$$F(x, y, t) \quad = \quad \begin{cases} f(x, y/2, t) & y \bmod 2 = 0 \\ f(x, (y+1)/2, t+1) & y \bmod 2 = 1 \end{cases}. \tag{3}$$

That is, for odd values of t, the odd scan lines of the full frame are composed of the field at time t, and the even scan lines of the full frame are composed of the field at time $t+1$. Similarly, for even values of t, the even scan lines of the full frame are composed of the field at time t, and the odd scan lines of the full frame are composed of the field at time $t+1$.

Unlike the line repetition algorithm, the final de-interlaced video has the full vertical resolution. Significant motion between the fields, however, introduces artifacts into the final video due to the mis-alignment of the fields. This artifact manifests itself with the commonly seen "combing effect", Figure 4.

3) Line Averaging: In this easy to implement and popular technique, Figure 2(c), neighboring scan lines, $f(x, y, t)$, are averaged together to create the necessary scan lines of the full frame, $F(x, y, t)$. For odd values of t:

$$F(x, y, t) \quad = \quad \begin{cases} f(x, (y+1)/2, t) & y \bmod 2 = 1 \\ \frac{1}{2} f(x, y/2, t) + \frac{1}{2} f(x, y/2+1, t) & y \bmod 2 = 0 \end{cases}, \tag{4}$$

and for even values of t:

$$F(x, y, t) \quad = \quad \begin{cases} f(x, y/2, t) & y \bmod 2 = 0 \\ \frac{1}{2} f(x, (y+1)/2-1, t) + \frac{1}{2} f(x, (y+1)/2, t) & y \bmod 2 = 1 \end{cases}. \tag{5}$$

Where necessary, boundary conditions (the first and last scan lines) can be handled by employing line repetition. This algorithm improves on the low vertical resolution of the line repetition algorithm, while avoiding the combing artifacts of the field insertion algorithm.

4) Vertical Temporal Interpolation: Similar to the line averaging algorithm, the vertical temporal interpolation algorithm combines neighboring scan lines in space (and here, in time), $f(x, y, t)$ and $f(x, y, t+1)$, to create the necessary scan lines of the full frame, $F(x, y, t)$. For odd values of t:

$$F(x, y, t) = \begin{cases} f(x, (y+1)/2, t) & y \bmod 2 = 1 \\ c_1 f(x, y/2-1, t) + c_2 f(x, y/2, t) + c_3 f(x, y/2+1, t) + & \\ c_4 f(x, y/2+2, t) + c_5 f(x, y/2-1, t+1) + & \\ c_6 f(x, y/2, t+1) + c_7 f(x, y/2+1, t+1) & y \bmod 2 = 0 \end{cases}. \tag{6}$$

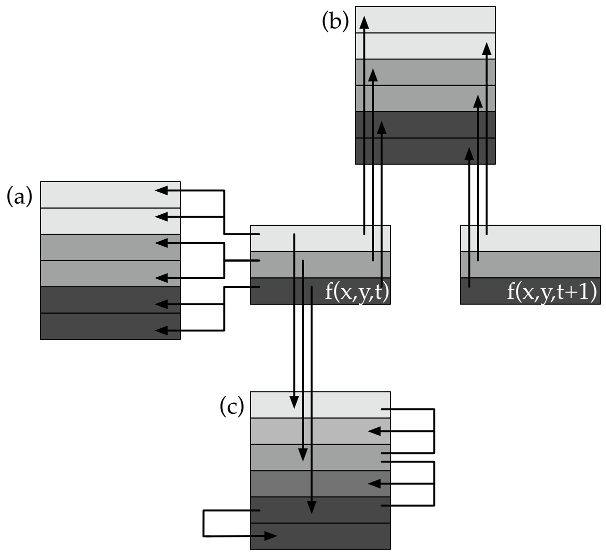

Fig. 2. Illustrated in this schematic are the (a) line repetition, (b) field insertion and (c) line averaging de-interlacing algorithms. The individual fields, $f(x, y, t)$ and $f(x, y, t + 1)$, are illustrated with three scan lines each, and the full de-interlaced frames are illustrated with six scan lines.

While the specific numeric weights c_i may vary, a typical example is $c_1 = 1/18$, $c_2 = 8/18$, $c_3 = 8/18$, $c_4 = 1/18$, $c_5 = -5/18$, $c_6 = 10/18$, and $c_7 = -5/18$. For even values of t:

$$F(x, y, t) = \begin{cases} f(x, y/2, t) & y \bmod 2 = 0 \\ c_1 f(x, (y+1)/2 - 2, t) + c_2 f(x, (y+1)/2 - 1, t) + c_3 f(x, (y+1)/2, t) + & \\ c_4 f(x, (y+1)/2 + 1, t) + c_5 f(x, (y+1)/2 - 1, t+1) + & \\ c_6 f(x, (y+1)/2, t+1) + c_7 f(x, (y+1)/2 + 1, t+1) & y \bmod 2 = 1 \end{cases} . \quad (7)$$

Where necessary, boundary conditions can be handled by employing line repetition. As with the line averaging algorithm, this algorithm improves the vertical temporal resolution. By averaging over a larger spatial and temporal neighborhood, the resolution can be improved beyond line averaging. By incorporating temporal neighborhoods, however, this algorithm is vulnerable to the combing artifacts of the field insertion algorithm.

5) Motion Adaptive: There is a natural tradeoff between de-interlacing algorithms that create a frame from only a single field (e.g., line repetition and line averaging) and those that incorporate two or more fields (e.g., field insertion and vertical temporal interpolation). In the former case, the resulting de-interlaced video suffers from low vertical resolution but contains no combing artifacts due to motion between fields. In the latter case, the de-interlaced video has an optimal vertical resolution when there is no motion between fields, but suffers from combing artifacts when there is motion between fields.

Motion adaptive algorithms incorporate the best of these techniques to improve vertical resolution while minimizing combing artifacts. While specific implementations vary, the basic algorithm creates two de-interlaced sequences, $F_1(x, y, t)$ and $F_2(x, y, t)$, using, for example, field insertion and line repetition. Standard motion estimation algorithms are used to create a "motion map", $\alpha(x, y)$ with values of 1 corresponding to regions with motion and values of 0 for regions with no motion. The final de-interlaced video is then given by:

$$F(x, y, t) \quad = \quad (1 - \alpha(x, y))F_1(x, y, t) + \alpha(x, y)F_2(x, y, t). \quad (8)$$

Although more complicated to implement due to the need for motion estimation, this algorithm affords good vertical resolution with minimal motion artifacts.

6) Motion Compensated: In this approach, standard motion estimation algorithms are employed to estimate the motion between neighboring fields, $f(x, y, t)$ and $f(x, y, t + 1)$. The resulting motion field is used to warp $f(x, y, t + 1)$ in order to undo any motion that occurred between fields. The resulting new field, $f'(x, y, t + 1)$ is then combined with the first field, $f(x, y, t)$ using field insertion, Section II-A.2. The benefit of this approach is that the resulting de-interlaced video is of optimal vertical resolution with no motion artifacts (assuming good motion estimation). The drawback is that of added complexity due to the need for motion estimation and possible artifacts due to poor motion estimates.

B. Spatial/Temporal Correlations in De-Interlaced Video

Given a de-interlaced video generated with any of the above algorithms, we seek to model the spatial and temporal correlations that result from the de-interlacing. The approach we take is similar in spirit to [3], where we modeled color filter array interpolation algorithms.

Consider, for example, the line averaging algorithm in which every other scan line is linearly correlated to its neighboring scan line, Equation (4). In this case:

$$F(x,y,t) \quad = \quad \tfrac{1}{2}F(x,y-1,t) + \tfrac{1}{2}F(x,y+1,t) \tag{9}$$

for either odd or even values of y. The remaining scan lines do not necessarily satisfy this relationship.

Consider for now, only the odd scan lines, $F_o(x,y,t)$, of $F(x,y,t)$ for t even (the formulation for $F_e(x,y,t)$ is identical). If this frame has not been doctored, then we expect that every pixel of $F_o(x,y,t)$ will be correlated to its spatial and temporal neighbors. Regions that violate this relationship indicate tampering. As such, we seek to simultaneously segment $F_o(x,y,t)$ into those pixels that are linearly correlated to their spatial and temporal neighbors and those that are not, and to estimate these correlations. We choose a linear model since it is a good model for most de-interlacing algorithms.

The expectation/maximization (EM) algorithm is employed to solve this simultaneous segmentation and estimation problem. We begin by assuming that each pixel of $F_o(x,y,t)$ belongs to one of two models, M_1 or M_2. Those pixels that belong to M_1 satisfy:

$$F_o(x,y,t) \quad = \quad \sum_{i\in\{-3,-1,1,3\}} \alpha_i F(x,y+i,t) + \sum_{i\in\{-2,0,2\}} \beta_i F(x,y+i,t+1) \; + \; n(x,y), \tag{10}$$

where $n(x,y)$ is independent and identically distributed Gaussian noise. Those pixels that belong to M_2 are considered to be generated by an “outlier” process. Note that Equation (10) embodies all of the de-interlacing algorithms described in the previous sections, except for the motion compensated algorithm (more on this later).

The EM algorithm is a two-step iterative algorithm: (1) in the E-step the probability of each pixel belonging to each model is estimated; and (2) in the M-step the specific form of the correlations between pixels is estimated. More specifically, in the E-step, the probability of each pixel of $F_o(x,y,t)$ belonging to model M_1 is estimated using Bayes’ rule:

$$P\{F_o(x,y,t) \in M_1 \mid F_o(x,y,t)\} \quad = \quad \frac{P\{F_o(x,y,t) \mid F_o(x,y,t) \in M_1\} P\{F_o(x,y,t) \in M_1\}}{\sum_{i=1}^{2} P\{F_o(x,y,t) \mid F_o(x,y,t) \in M_i\} P\{F_o(x,y,t) \in M_i\}}, \tag{11}$$

where the prior probabilities $P\{F_o(x,y,t) \in M_1\}$ and $P\{F_o(x,y,t) \in M_2\}$ are each assumed to be equal to $1/2$. The probability of observing a sample $F_o(x,y,t)$ knowing it was generated from model M_1 is given by:

$$P\{F_o(x,y,t) \mid F_o(x,y,t) \in M_1\} \quad = \quad \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{\left(F_o(x,y,t) - \tilde{F}_o(x,y,t)\right)^2}{2\sigma^2} \right], \tag{12}$$

where:

$$\tilde{F}_o(x,y,t) \quad = \quad \sum_{i\in\{-3,-1,1,3\}} \alpha_i F(x,y+i,t) + \sum_{i\in\{-2,0,2\}} \beta_i F(x,y+i,t+1). \tag{13}$$

The variance, σ^2, of this Gaussian distribution is estimated in the M-step. A uniform distribution is assumed for the probability of observing a sample generated by the outlier model, M_2, i.e., $P\{F_o(x,y,t) \mid F_o(x,y,t) \in M_2\}$ is equal to the inverse of the range of possible values of $F_o(x,y,t)$.

Note that the E-step requires an estimate of the coefficients α_i and β_i, which on the first iteration is chosen randomly. In the M-step, a new estimate of these model parameters is computed using weighted least squares, by minimizing the following quadratic error function:

$$E(\{\alpha_i, \beta_i\}) \quad = \quad \sum_{x,y} w(x,y) \left(F_o(x,y,t) - \tilde{F}_o(x,y,t) \right)^2, \tag{14}$$

where the weights $w(x,y) \equiv P\{F_o(x,y,t) \in M_1 \mid F_o(x,y,t)\}$, Equation (11). This error function is minimized by computing the partial derivative with respect to each α_i and β_i, setting each of the results equal to zero and solving the resulting system of linear equations. The derivative, for example, with respect to α_j is:

$$\begin{aligned} \sum_{x,y} w(x,y) F(x,y+j,t) F_o(x,y,t) \quad &= \quad \sum_{i\in\{-3,-1,1,3\}} \alpha_i \left(\sum_{x,y} w(x,y) F(x,y+i,t) F(x,y+j,t) \right) \\ &\quad + \sum_{i\in\{-2,0,2\}} \beta_i \left(\sum_{x,y} w(x,y) F(x,y+i,t+1) F(x,y+j,t) \right). \end{aligned} \tag{15}$$

The derivative with respect to β_j is:

$$\sum_{x,y} w(x,y)F(x,y+j,t+1)F_o(x,y,t) = \sum_{i\in\{-3,-1,1,3\}} \alpha_i \left(\sum_{x,y} w(x,y)F(x,y+i,t)F(x,y+j,t+1)\right) + \sum_{i\in\{-2,0,2\}} \beta_i \left(\sum_{x,y} w(x,y)F(x,y+i,t+1)F(x,y+j,t+1)\right). \quad (16)$$

Computing all such partial derivatives yields a system of seven linear equations in the seven unknowns α_i and β_i. This system can be solved using standard least-squares estimation. The variance, σ^2, for the Gaussian distribution is also estimated on each M-step, as follows:

$$\sigma^2 = \frac{\sum_{x,y} w(x,y)(F_o(x,y,t) - \tilde{F}_o(x,y,t))^2}{\sum_{x,y} w(x,y)} \quad (17)$$

. The E-step and M-step are iteratively executed until stable estimates of α_i and β_i are obtained.

C. De-Interlacing Results

Shown in the top portion of Figure 7 are eleven frames from a 250-frame long video sequence. Each frame is of size 480×720 pixels. Shown in the bottom portion of this figure are the same eleven frames de-interlaced with the line average algorithm. This video was captured with a Canon Elura digital video camera set to record in interlaced mode. The interlaced video was de-interlaced with our implementation of line repetition, field insertion, line average, vertical temporal integration, motion adaptive, and motion compensated, and the de-interlacing plug-in for VirtualDub [1] which employs the motion adaptive algorithm.

In each case, each frame of the video was analyzed using the EM algorithm. The EM algorithm returns both a probability map denoting which pixels are correlated to their spatial/temporal neighbors, and the coefficients of this correlation. For simplicity, we report on the results for only the odd frames (in all cases, the results for the even frames are comparable). Within an odd frame, only the even scan lines are analyzed (similarly for the odd scan lines on the even frames). In addition, we convert each frame from RGB to grayscale (gray = 0.299R + 0.587G + 0.114B) – all three color channels could easily be analyzed and their results combined via a simple voting scheme.

Shown below is the percentage of pixels classified as belonging to model M_1, that is, as being correlated to their spatial neighbor:

de-interlace	accuracy
line repetition	100%
field insertion	100%
line average	100%
vertical temporal	99.5%
motion adaptive (no-motion | motion)	97.8% | 100%
motion compensation	97.8%
VirtualDub (no-motion | motion)	99.9% | 100%

A pixel is classified as belonging to M_1 if the probability, Equation (11), is greater than 0.90. The small fraction of pixels that are incorrectly classified are scattered across frames, and can thus easily be seen to not be regions of tampering – a spatial median filter would easily remove these pixels, while not interfering with the detection of tampered regions (see below). For de-interlacing by motion adaptive and VirtualDub, the EM algorithm was run separately on regions of the image with and without motion. Regions of motion are determined simply by computing frame differences – pixels with an intensity difference greater than 3 (on a scale of $[0, 255]$ are classified as having undergone motion, with the remaining pixels classified as having no motion. The reason for this distinction is that the motion adaptive algorithm, Section II-A.5, employs different de-interlacing for regions with and without motion. Note that for de-interlacing by motion compensation, only pixels with no motion are analyzed. The reason is that in this de-interlacing algorithm, the de-interlacing of regions with motion does not fit our model, Equation (10).

Shown in Figure 8 are the actual and estimated (mean/standard deviation) model coefficients, α_i and β_i, averaged over the entire video sequence. Note that, in all cases, the mean estimates are very accurate. The standard deviations range from nearly zero (10^{-13}) to relatively small ($10^{-4} - 10^{-3}$). The reason for these differences is two-fold: (1) rounding errors are introduced for the de-interlacing algorithms with non-integer coefficients α_i and β_i, and (2) errors in the motion estimation for the motion-adaptive based algorithms which incorrectly classifies pixels as having motion or no-motion.

[1] VirtualDub is a video capture/processing utility, `http://www.virtualdub.org`.

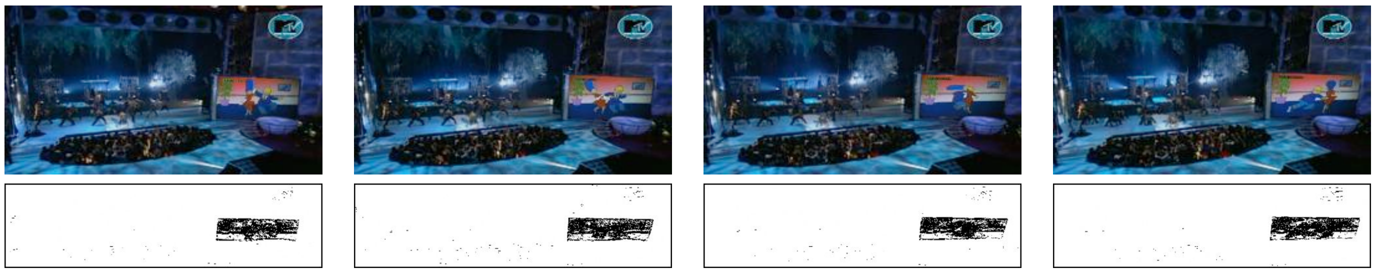

Fig. 3. Four frames of a Britney Spears concert that have been doctored to include a dancing Marge and Homer Simpson. Shown below are the de-interlacing results that reveal the tampered region.

Shown in Figure 9 are results for detecting tampering in the video sequence of Figure 7. We simulated the effects of tampering by adding, into a central square region of varying size, white noise of varying signal to noise ratio (SNR). A unique noise pattern was added to each video frame. The motivation behind this manipulation is that, as would most forms of tampering, the noise destroys the underlying de-interlacing correlations. The tampered regions were of size 256×256, 128×128, 64×64, 32×32 or 16×16 pixels, in each frame of size 480×720. The SNR, 10, 20, 30 or 35 dB, ranges from the highly visible (10 dB) to perceptually invisible (35 dB). Shown in each panel of Figure 9 is the average probability, over all frames, as reported by the EM algorithm for the central tampered region and the surrounding un-tampered region. This probability corresponds to the likelihood that each pixel is correlated to their spatial and temporal neighbors (i.e., is consistent with the output of a de-interlacing algorithm). We expect probabilities significantly less than 1 for the tampered regions, and values close to 1 for the un-tampered region. For the motion adaptive algorithms, we combined the probabilities for the motion and no-motion pixels.

Notice that in general, detection is quite easy for SNR values below 25dB, that is, the tampered region has an average probability significantly less than the un-tampered region. Notice also that the detection gets slightly easier for larger regions, particularly at high SNR values, but that even very small regions (16×16) are still detectable. And finally, note that in the case of de-interlacing by motion adaptive and VirtualDub, the tampered regions are more difficult to detect for large regions with low SNR (rows 5 and 7, last column). The reason for this is the large tampered regions at a low SNR give rise to a significant number of pixels mistakenly classified as having motion or no motion. As a result, the value of σ in Equation (12) increases, which naturally results in a larger probability for the tampered region. Note however, that the final probability is still below the 0.90 threshold used to determine if a region has been tampered with.

Shown in Figure 3 are four frames of a doctored video sequence. The original video was de-interlaced with the vertical temporal algorithm. Each frame of the digitally inserted cartoon sequence was re-sized from its original destroying any de-interlacing correlations. Shown below each frame is the resulting probability map returned by the EM algorithm. The doctored region is clearly visible in each frame.

We naturally expect that various compression algorithms such as MPEG will somewhat disrupt the underlying correlations introduced by de-interlacing algorithms. To test the sensitivity to compression the 250-frame long video sequence described above was de-interlaced with the line repetition and line average algorithms, and then compressed using Adobe Premiere to a target bit rate of 9, 6, and 3 Mbps. Below are the percentage of pixels classified as being correlated to their spatial neighbors.

de-interlace	**accuracy**			
	none	9 Mbps	6 Mbps	3 Mbps
line repetition	100%	97.1%	96.2%	93.2%
line average	100%	97.0%	96.0%	93.5%

Note that even when compressed, the de-interlacing artifacts can still be detected and that the accuracy degrades gracefully with increasing compression.

III. Interlaced Video

Some video sequences forgo de-interlacing altogether and simply weave together the odd and even fields. In the exaggerated example of Figure 4, the motion between times t and $t+1$ leads to a "combing" artifact. The magnitude of this effect depends on the amount of motion between fields. Below we describe how to measure this motion and how to exploit these interlacing artifacts to detect tampering.

A. Temporal Correlations in Interlaced Video

We begin by assuming that the motion is constant across at least three sequential fields. At a typical frame rate of 30 frames/second, this amounts to assuming that the motion is constant for $1/20$ of a second (assuming that the time between fields is $1/60^{th}$ of a second). Consider the fields $f(x,y,1)$, $f(x,y,2)$ and $f(x,y,3)$ corresponding to the odd and even lines

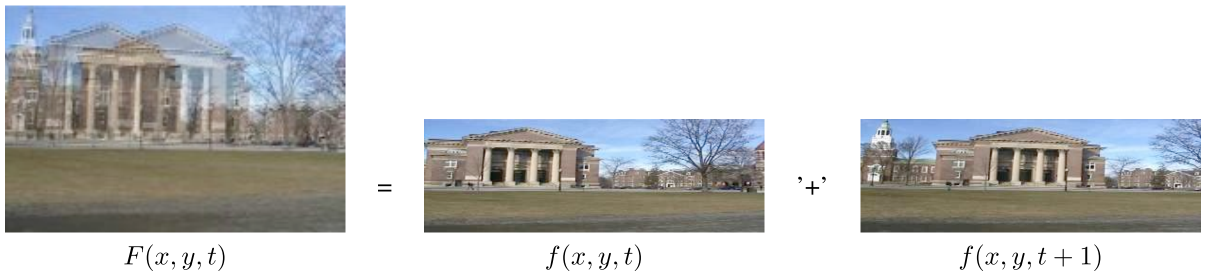

Fig. 4. One half of the scan lines, $f(x,y,t)$, of a full video frame are recorded at time t, and the other half of the scan lines, $f(x,y,t+1)$ are recorded at time $t+1$. An interlaced frame, $F(x,y,t)$, is created by simply weaving together these two fields.

for frame $F(x,y,1)$ and the odd lines for frame $F(x,y,2)$, respectively. With the constant motion assumption, we expect the inter-field motion between $f(x,y,1)$ and $f(x,y,2)$ to be the same as the inter-frame motion between $f(x,y,2)$ and $f(x,y,3)$. And while the overall motion may change over time, this equality should be relatively constant. We will show below how to estimate this motion and how tampering can disturb this temporal pattern. For computational efficiency, we convert each RGB frame to grayscale (gray = 0.299R + 0.587G + 0.114B) – all three color channels could be analyzed and their results averaged [2].

1) Motion Estimation: We consider a classic differential framework for motion estimation [14], [15], [16]. We begin by assuming that the image intensities between fields are conserved (the brightness constancy assumption), and that the motion between fields can locally be modeled with a 2-parameter translation. The following expression embodies these two assumptions:

$$f(x,y,t) \quad = \quad f(x+v_x, y+v_y, t-1), \tag{18}$$

where the motion is $\vec{v} = \begin{pmatrix} v_x & v_y \end{pmatrix}^T$. In order to estimate the motion $\vec{v}$, we define the following quadratic error function to be minimized:

$$E(\vec{v}) \quad = \quad \sum_{x,y\in\Omega} \left[f(x,y,t) - f(x+v_x, y+v_y, t-1)\right]^2, \tag{19}$$

where Ω denotes a spatial neighborhood. Since this error function is non-linear in its unknowns, it cannot be minimized analytically. To simplify the minimization, we approximate this error function using a first-order truncated Taylor series expansion:

$$\begin{aligned} E(\vec{v}) \quad &\approx \quad \sum_{x,y\in\Omega} \left[f(x,y,t) - (f(x,y,t) + v_x f_x(x,y,t) + v_y f_y(x,y,t) - f_t(x,y,t))\right]^2 \\ &= \quad \sum_{x,y\in\Omega} \left[f - (f + v_x f_x + v_y f_y - f_t)\right]^2, \end{aligned} \tag{20}$$

where $f_x(\cdot)$, $f_y(\cdot)$, and $f_t(\cdot)$ are the spatial and temporal derivatives of $f(\cdot)$, and where for notational convenience the spatial/temporal parameters are dropped. This error function reduces to:

$$E(\vec{v}) \quad = \quad \sum_{x,y\in\Omega} \left[f_t - v_x f_x - v_y f_y\right]^2 \quad = \quad \sum_{x,y\in\Omega} \left[f_t - \begin{pmatrix} f_x & f_y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix}\right]^2 \quad = \quad \sum_{x,y\in\Omega} \left[f_t - \vec{f_s}^T \vec{v}\right]^2 \tag{21}$$

Note that this quadratic error function is now linear in its unknowns, $\vec{v}$. This error function can be minimized analytically by differentiating with respect to the unknowns:

$$\frac{dE(\vec{v})}{d\vec{v}} \quad = \quad \sum_{x,y\in\Omega} -2\vec{f_s}\left[f_t - \vec{f_s}^T \vec{v}\right] \tag{22}$$

and setting this result equal to zero, and solving for $\vec{v}$:

$$\vec{v} \quad = \quad -\begin{pmatrix} \sum_\Omega f_x^2 & \sum_\Omega f_x f_y \\ \sum_\Omega f_x f_y & \sum_\Omega f_y^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_\Omega f_x f_t \\ \sum_\Omega f_y f_t \end{pmatrix}, \tag{23}$$

where again recall that the spatial/temporal parameters on the derivatives have been dropped for notational convenience. This solution assumes that the first term, a 2×2 matrix, is invertible. This can usually be guaranteed by integrating over a large enough spatial neighborhood Ω with sufficient image content.

[2]Since the color channels are correlated, we expect little advantage to averaging the motion estimated from each of the three color channels.

2) Spatial/Temporal Differentiation: The spatial/temporal derivatives needed to estimate motion in Equation (23) are determined via convolutions [17] as follows:

$$f_x = (\tfrac{1}{2}f_t(x,y,t) + \tfrac{1}{2}f_t(x,y,t-1)) \star d(x) \star p(y) \tag{24}$$

$$f_y = (\tfrac{1}{2}f_t(x,y,t) + \tfrac{1}{2}f_t(x,y,t-1)) \star p(x) \star d(y) \tag{25}$$

$$f_t = (\tfrac{1}{2}f_t(x,y,t) - \tfrac{1}{2}f_t(x,y,t-1)) \star p(x) \star p(y), \tag{26}$$

where the 1-D filters are $d(\cdot) = [0.425287 \;\; 0.0 \;\; -0.425287]$ and $p(\cdot) = [0.2298791 \;\; 0.540242 \;\; 0.2298791]$. Note that while we employ 2-tap filters for the temporal filtering ($(1/2 \;\; 1/2)$ and $(1/2 \;\; -1/2)$), 3-tap filters are used for the spatial filtering. Despite the asymmetry, these filters yield more accurate motion estimates. To avoid edge artifacts due to the convolution, the derivatives are centrally cropped by removing a 2-pixel border.

3) Inter-Field and Inter-Frame Motion: Recall that we are interested in comparing the inter-field motion between $f(x, y, t)$ and $f(x, y, t + 1)$ and the inter-frame motion between $f(x, y, t + 1)$ and $f(x, y, t + 2)$, for t odd. The inter-field motion is estimated in a two-stage process (the inter-frame motion is computed in the same way).

In the first stage, the motion is estimated globally between $f(x, y, t)$ and $f(x, y, t + 1)$, that is, Ω is the entire image in Equation (23). The second field, $f(x, y, t + 1)$, is then warped according to this estimated motion (a global translation). This stage removes any large-scale motion due to, for example, camera motion. In the second stage, the motion is estimated locally for non-overlapping blocks of size $\Omega = n \times n$ pixels. This local estimation allows us to consider more complex and spatially varying motions, other than global translation. For a given block, the overall motion is then simply the sum of the global and local motion.

The required spatial/temporal derivatives have finite support thus fundamentally limiting the amount of motion that can be estimated. By sub-sampling the images, the motion is typically small enough for our differential motion estimation. In addition, the run-time is sufficiently reduced by operating on sub-sampled images. In both stages, therefore, the motion estimation is done on a sub-sampled version of the original fields: a factor of 16 for the global stage, and a factor of 8 of the local stage, with $\Omega = 13 \times 13$ (corresponding to a neighborhood size of 104×104 at full resolution). While the global estimation, done at a coarser scale, yields a less accurate estimation of motion, it does remove any large-scale motion. The local stage done at a higher scale then refines this estimate. For computational considerations, we do not consider other scales, although this could easily be incorporated. To avoid wrap-around edge artifacts due to the global alignment correction, six pixels are cropped along each horizontal edge and two pixels along each vertical edge prior to estimating motion at $1/8$ resolution.

In order to reduce the errors in motion estimation, the motion across three pairs of fields is averaged together, where the motion between each pair is computed as described above. The inter-frame motion is estimated in the same way, and the norm of the two motions, $\|\vec{v}\| = \sqrt{v_x^2 + v_y^2}$, are compared across the entire video sequence.

B. Interlacing Results

An interlaced video sequence, $F(x, y, t)$, of length T frames is first separated into fields, $f(x, y, t)$ with t odd corresponding to the odd scan lines and t even corresponding to the even scan lines. The inter-field motion is measured as described above for all pairs of fields $f(x, y, t)$ and $f(x, y, t+1)$, for t odd, and the inter-frame motion is measured between all pairs of fields $f(x, y, t + 1)$ and $f(x, y, t + 2)$, for $3 \le t \le 2T - 3$. We expect these two motions to be the same in an authentic interlaced video, and to be disrupted by tampering.

We recorded three videos from a SONY-HDR-HC3 digital video camera. The camera was set to record in interlaced mode at 30 frames/sec. Each frame is 480×720 pixels in size, and the average length of each video sequence is 10 minutes or $18,000$ frames. For the first video sequence, the camera was placed on a tripod and pointed towards a road and sidewalk, as a surveillance camera might be positioned. For the two other sequences, the camera was hand-held.

Shown in Figure 5 are the estimated inter-field and inter-frame motions for each video sequence. Each data point corresponds to the estimate from a single 13×13 block (at $1/8$ resolution). Since we are only interested in the motion ratio, the motions are normalized into the range $[0, 1]$. Also shown in each panel is a line fit to the underlying data (solid line), which in the ideal case would be a unit-slope line with zero intercept. The average motion ratio for each sequence is 0.98, 0.96, and 0.98 with a variance of 0.008, 0.128, and 0.156, respectively.

A frame is classified as manipulated if at least one block in the frame has a motion ratio that is more than 0.2 from unit value. In order to avoid spurious errors, we also insist that at least 3 successive frames are classified as manipulated. For the first video sequence, only one block out of $221,328$ blocks was incorrectly classified as manipulated. For the second and third sequence, two and eight blocks, respectively, were misclassified out of a total of $241,248$ and $224,568$ blocks. After imposing the temporal constraints, no frames were mis-classified.

Shown in the first two rows of Figure 6 are ten frames of an original interlaced video. This video shows a person walking against a stationary background and being filmed with a stationary camera. Shown in the next two rows of this same figure is a doctored version of this video sequence. In this version, a different person's head has been spliced into each frame. Because this person was walking at a slower speed than the original person, the inter-field interlacing is smaller than in the original.

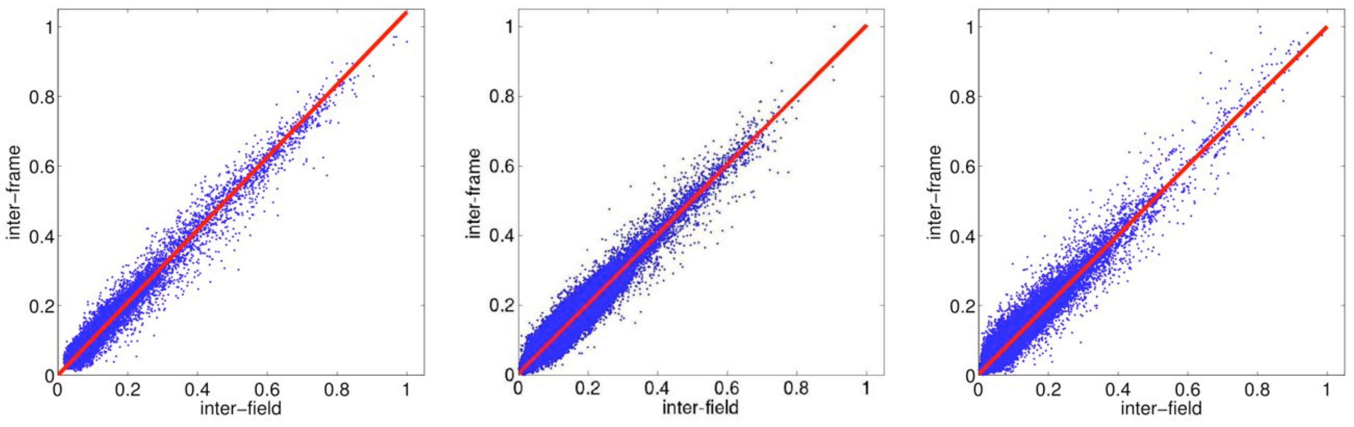

Fig. 5. Shown in each panel is the normalized inter-field motion versus the normalized inter-frame motion for three different video sequences. The solid line represents a line fit to the underlying data points. We expect the motion ratio to be near 1 in an authentic video sequence and to deviate significantly for a doctored video.

Shown in lower portion of Figure 6 are the resulting inter-field and inter-frame motions. The data points that lie significantly away from the unit-slope line correspond to the blocks in which the new head was introduced – occasionally this would occupy two blocks, both of which would deviate from the expected motion ratio. Even though the doctored video looks perceptually smooth, the tampering is easily detected.

As in the previous section, it is important to test the sensitivity of the motion estimation algorithms to compression. The first video sequence described above was compressed using Adobe Premiere to a target bit rate of 9, 6, and 3 Mbps. The same inter-field and inter-frame motions were estimated for each of these video sequences. For the original sequence, only 1 block out of $221,328$ blocks was incorrectly classified as manipulated. After the temporal filtering, no frames out of $18,444$ frames were incorrectly classified. For the three compressed video sequences, 1, 4, and 3 blocks were mis-classified, and 0, 1, and 0 frames were mis-classified. The motion estimation algorithm is largely insensitive to compression artifacts. One reason for this is that we operate on a sub-sampled version of the video (by a factor of $1/16$ and $1/8$) in which many of the compression artifacts are no longer present. In addition, the motion is estimated over a spatial neighborhood so that many of the errors are integrated out.

IV. Detecting Frame Rate Conversion

Here we describe how the above algorithms can be adapted to detect if the frame rate of a video was altered from its original rate. Consider an original video sequence captured at 30-frames per second that is subsequently manipulated and saved at a lower rate of 25-frames per second. The standard approach to reducing the frame rate is to simply remove the necessary number of frames (5 per second in this example). In so doing, the inter-field and inter-frame motion ratio as described in the previous section will be disturbed. Specifically at the deleted frames, the inter-field motion will be too small relative to the inter-frame motion.

A video sequence of length 1200 frames, originally recorded at 30 frames per second, was converted using VirtualDub to a frame rate of 25 frames per second, yielding a video of length 1000 frames. The inter-field and inter-frames motions were estimated as described. In this frame rate converted video every 5^{th} frame had an average motion ratio of 2.91, while the intervening frames had an average motion ratio of 1.07.

Consider now a video sequence captured at 25-frames per second that is subsequently manipulated and saved at a higher rate of 30-frames per second. This frame rate conversion requires some form of interpolation in order to fill in the extra frames. There are several frame rate conversion algorithms that accomplish this. Frame repetition is the simplest approach where frames from the original sequence are simply repeated to expand the frame rate to the desired rate. When converting from 25 to 30 frames per second, for example, every fifth frame of the original sequence is duplicated. Linear frame rate conversion creates the extra frames by taking linear combinations of neighboring frames in time. More sophisticated motion-based algorithms compensate for the inter-frame motion in a similar way to the motion-based de-interlacing algorithms (e.g., [18], [19]).

Here we consider only the frame repetition and linear conversion techniques. The method we will describe can be adapted, as in the previous section, to be applicable to motion-based algorithms. For both algorithms, some frames in a converted video will be a linear combination of their neighboring frames, while other frames will not. We therefore employ the EM algorithm to simultaneously segment the video into frames that are and are not linear combinations of their neighbors, and determine the linear coefficients of this combination.

Similar to before, the relationship between frames is modeled as:

$$F(x,y,t) \quad = \sum_{i \in \{-1,1\}} \alpha_i F(x,y,t+i), \tag{27}$$

where for simplicity only two temporal neighbors are considered – this could easily be expanded to consider a larger neighborhood. With a few minor changes, the EM algorithm described in the previous section can be adapted to detect frame rate conversion. Note first that the model here operates on entire frames, instead of individual pixels. Second, because the majority of frames will not correlated to their neighbors, we find that it is necessary to use a fixed, and relatively small, σ in Equation (12). And lastly, note that the model coefficients for the first frame in a duplicated pair will be $\alpha_{-1} = 0$ and $\alpha_1 = 1$ while the coefficients for the second duplicated frame will be $\alpha_{-1} = 1$ and $\alpha_1 = 0$. For our purposes, we would like to consider these models as the same. As such, on each EM iteration both the current and the symmetric version of the model coefficients are considered. The model that minimizes the residual error between the frame and the model is adopted.

A video sequence of length 960 frames, originally recorded at 25 frames per second, was converted using VirtualDub to a frame rate of 30 frames per second, yielding a video of length 1200 frames. VirtualDub employs a frame duplication algorithm. The EM algorithm detected that every multiple of 6 frames was a linear combination of their neighboring frames (with an average probability of 0.99), while the remaining frames were not (with an average probability of 0.00).

V. Discussion

We have presented two techniques for detecting tampering in de-interlaced and interlaced video. For de-interlaced video, we explicitly model the correlations introduced by de-interlacing algorithms, and show how tampering can destroy these correlations. For the interlaced video, we measure the inter-field and inter-frame motions which for an authentic video are the same, but for a doctored video may be different. In both cases, we can localize tampering both in time and in space (what part of a video frame was manipulated). We also described how both algorithms could be slightly adapted to detect frame rate conversion that might result after a video was manipulated.

Compression artifacts make it somewhat more difficult to estimate the de-interlacing correlations, so this approach is most appropriate for high- to medium- quality video. Compression artifacts have little effect on the estimation of motion in interlaced video, so this approach is appropriate for a range of interlaced video.

A counter-attack to the de-interlacing forensic tool is to first doctor a video, and then generate an interlaced video (split the even and odd scan lines), and apply a de-interlacing algorithm to generate a new de-interlaced video whose correlations will be intact. If the original video camera is available, then this approach requires the forger to employ the same de-interlacing algorithm as that used by the original camera. A counter-attack for the interlacing forensic tool would be a bit more involved as it would require the forger to locally estimate the inter-frame motions and interlace the doctored video so as to match the inter-field and inter-frame motions.

As with all forensic approaches, these techniques have their strengths and limitations. We expect that these techniques when coupled with our earlier work [8], and future techniques, to begin to make it more difficult to alter digital video.

Acknowledgment

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.

References

[1] J. Lukáš, J. Fridrich, and M. Goljan, "Detecting digital image forgeries using sensor pattern noise," in *Proceedings of the SPIE*, vol. 6072, 2006.
[2] M. Johnson and H. Farid, "Exposing digital forgeries by detecting inconsistencies in lighting," in *ACM Multimedia and Security Workshop*, New York, NY, 2005.
[3] A. Popescu and H. Farid, "Exposing digital forgeries in color filter array interpolated images," *IEEE Transactions on Signal Processing*, vol. 53, no. 10, pp. 3948–3959, 2005.
[4] ——, "Exposing digital forgeries by detecting traces of re-sampling," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 758–767, 2005.
[5] J. Fridrich, D. Soukal, and J. Lukáš, "Detection of copy-move forgery in digital images," in *Proceedings of DFRWS*, 2003.
[6] T. Ng and S. F. Chang, "A model for image splicing," in *IEEE International Conference on Image Processing*, Singapore, October 2004.
[7] A. Popescu and H. Farid, "Statistical tools for digital forensics," in *6th International Workshop on Information Hiding*, Toronto, Cananda, 2004.
[8] W. Wang and H. Farid, "Exposing digital forgeries in video by detecting double MPEG compression," in *ACM Multimedia and Security Workshop*, Geneva, Switzerland, 2006.
[9] G. de Haan and E. Bellers, "Deinterlacing–an overview," in *Proceedings of the IEEE*, vol. 86, no. 9, 1998, pp. 1839–1857.
[10] J. Deame, "Motion compensated de-interlacing: the key to the digital video transition," in *SMPTE 141st Technical Conference*, New York, 1999.
[11] Sugiyama and Nakamura, "A method of de-interlacing with motion compensated interpolation," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 611–616, 1999.
[12] E. B. Bellers and G. de Haan, *De-Interlacing: A Key Technology for Scan Rate Conversion*. Elsevier, 2000, ch. Bayesian Multi-scale Differential Optical Flow.
[13] K. Ouyang, S. L. G. Shen, and M. Gu, "Advanced motion search and adaptation techniques for deinterlacing," in *IEEE International Conference on Multimedia and Expo*, 2005, pp. 374–377.
[14] B. Horn, *Robot Vision*. MIT Press, Cambridge, MA, 1986.

[15] J. Barron, D. Fleet, and S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77, Feb. 1994.

[16] E. Simoncelli, *Handbook of Computer Vision and Applications*. Academic Press, 1999, ch. Bayesian Multi-scale Differential Optical Flow, pp. 397–420.

[17] H. Farid and E. Simoncelli, "Differentiation of discrete multi-dimensional signals," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 496–508, 2004.

[18] R. L. Lagendijk and M. I. Sezan, "Motion-compensated frame rate conversion of motion pictures," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, March 1992, pp. 453–456.

[19] R. Castagno, P. Haavisto, and G. Ramponi, "A method for motion adaptive frame rate up-conversion," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 5, pp. 436–446, 1996.

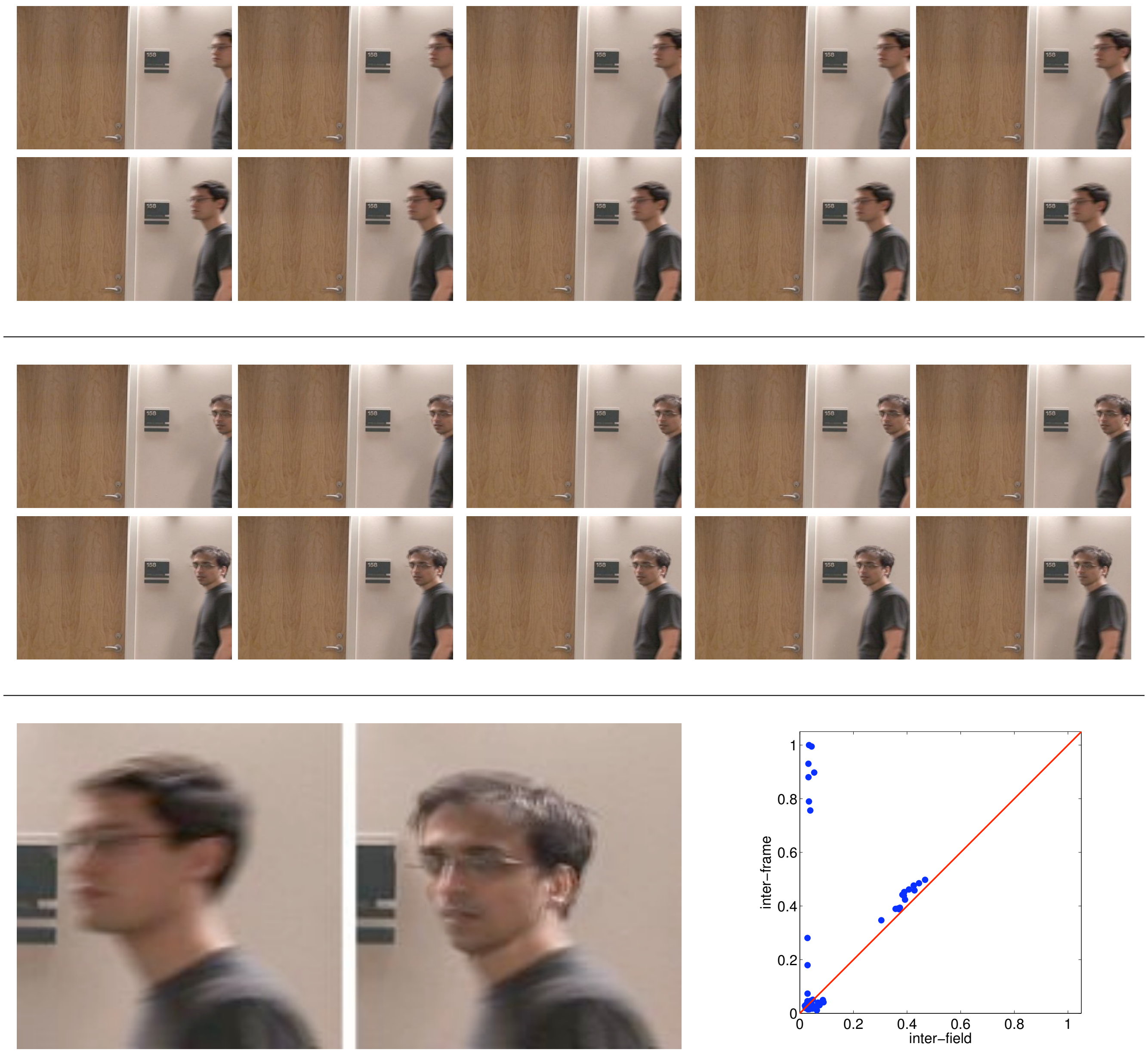

Fig. 6. Shown in the top two rows are 10 frames of an original interlaced video. Shown in the next two rows is a doctored version of this video sequence, and in the lower-left are enlargements of the last frame of the original and doctored video. The plot shows the inter-field versus the inter-frame motion – the data points that lie significantly away from the unit slope line correspond to doctored blocks (the head) while the data points on the line correspond to the original blocks (the body).

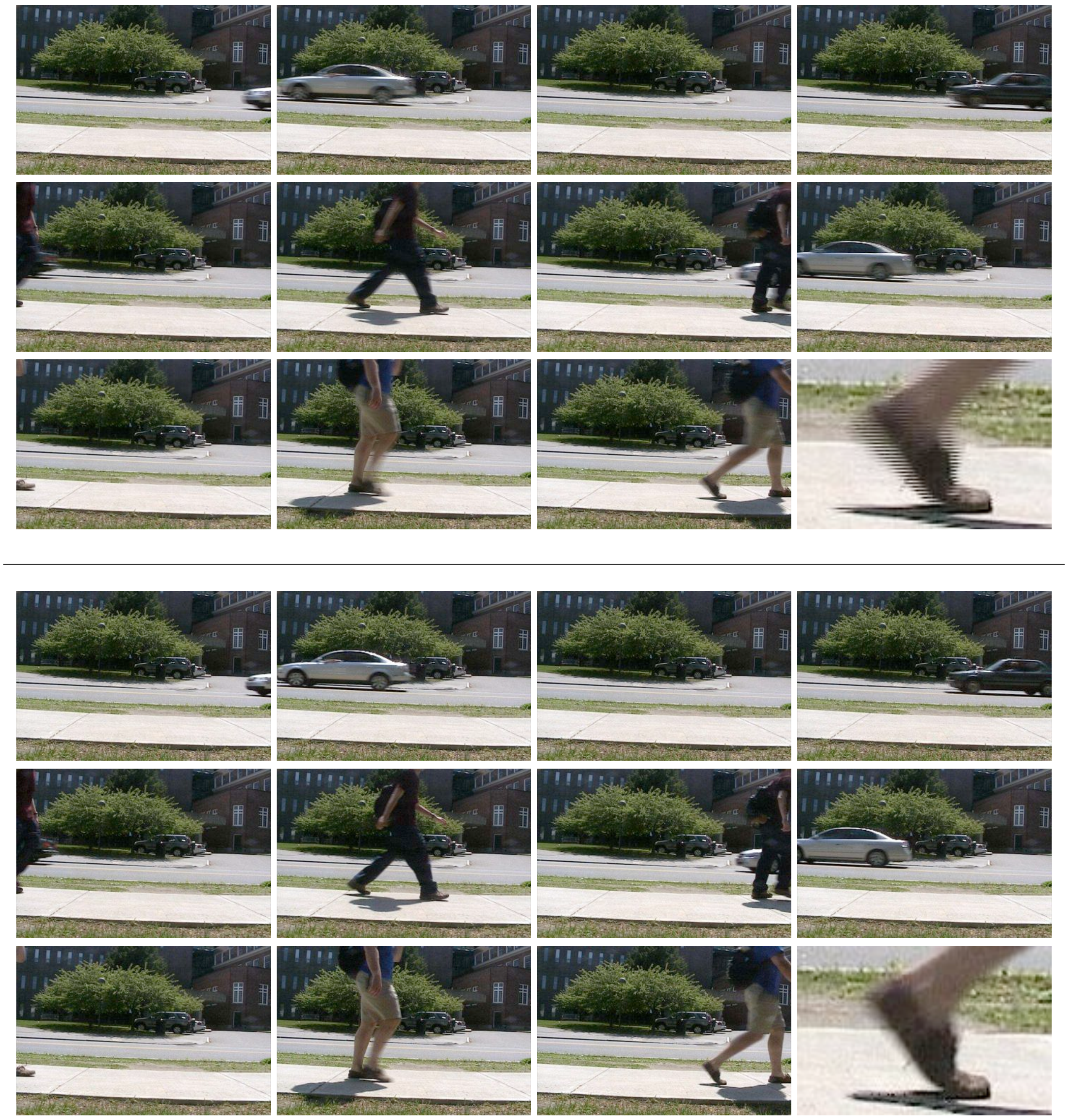

Fig. 7. Shown in the top portion are eleven frames of a 250-frame long video sequence. Shown in the bottom portion are the same eleven frames de-interlaced using the line average algorithm – notice that the interlacing artifacts are largely reduced. Shown in the lower-right corner is an enlargement of the pedestrian's foot from the last frame.

	line repetition		field insertion		line average	
	actual	estimated	actual	estimated	actual	estimated
α_{-3}	0.0000	$0.0000/0.07 \times 10^{-13}$	0.0000	$0.0000/0.07 \times 10^{-13}$	0.0000	$0.0005/0.06 \times 10^{-3}$
α_{-1}	1.0000	$1.0000/0.21 \times 10^{-13}$	0.0000	$0.0000/0.20 \times 10^{-13}$	0.5000	$0.5000/0.15 \times 10^{-3}$
α_{1}	0.0000	$0.0000/0.23 \times 10^{-13}$	0.0000	$0.0000/0.23 \times 10^{-13}$	0.5000	$0.5001/0.19 \times 10^{-3}$
α_{3}	0.0000	$0.0000/0.08 \times 10^{-13}$	0.0000	$0.0000/0.08 \times 10^{-13}$	0.0000	$0.0006/0.08 \times 10^{-3}$
β_{-2}	0.0000	$0.0000/0.15 \times 10^{-13}$	0.0000	$0.0000/0.15 \times 10^{-13}$	0.0000	$0.0001/0.12 \times 10^{-3}$
β_{0}	0.0000	$0.0000/0.22 \times 10^{-13}$	1.0000	$1.0000/0.19 \times 10^{-13}$	0.0000	$0.0003/0.18 \times 10^{-3}$
β_{2}	0.0000	$0.0000/0.22 \times 10^{-13}$	0.0000	$0.0000/0.22 \times 10^{-13}$	0.0000	$0.0000/0.18 \times 10^{-3}$

	vertical temporal		motion adaptive (no-motion)		motion adaptive (motion)	
	actual	estimated	actual	estimated	actual	estimated
α_{-3}	0.0556	$0.0556/0.39 \times 10^{-4}$	0.0000	$0.0000/0.06 \times 10^{-3}$	0.0000	$0.0000/0.09 \times 10^{-3}$
α_{-1}	0.4444	$0.4444/0.64 \times 10^{-4}$	0.0000	$0.0000/0.25 \times 10^{-3}$	0.5000	$0.5001/0.09 \times 10^{-3}$
α_{1}	0.4444	$0.4444/0.65 \times 10^{-4}$	0.0000	$0.0000/0.25 \times 10^{-3}$	0.5000	$0.5001/0.13 \times 10^{-3}$
α_{3}	0.0556	$0.0556/0.37 \times 10^{-4}$	0.0000	$0.0000/0.06 \times 10^{-3}$	0.0000	$0.0006/0.12 \times 10^{-3}$
β_{-2}	-0.2778	$-0.2778/0.60 \times 10^{-4}$	0.0000	$0.0000/0.15 \times 10^{-3}$	0.0000	$0.0002/0.12 \times 10^{-3}$
β_{0}	0.5556	$0.5556/0.65 \times 10^{-4}$	1.0000	$1.0000/0.31 \times 10^{-3}$	0.0000	$0.0003/0.10 \times 10^{-3}$
β_{2}	-0.2778	$-0.2778/0.64 \times 10^{-4}$	0.0000	$0.0000/0.15 \times 10^{-3}$	0.0000	$0.0004/0.13 \times 10^{-3}$

	motion compensated		VirtualDub (no-motion)		VirtualDub (motion)	
	actual	estimated	actual	estimated	actual	estimated
α_{-3}	0.0000	$0.0000/0.02 \times 10^{-3}$	0.0000	$0.0000/0.76 \times 10^{-13}$	0.0000	$-0.0006/0.11 \times 10^{-3}$
α_{-1}	0.0000	$0.0000/0.08 \times 10^{-3}$	0.0000	$0.0000/0.27 \times 10^{-13}$	0.5000	$0.4998/0.16 \times 10^{-3}$
α_{1}	0.0000	$0.0000/0.09 \times 10^{-3}$	0.0000	$0.0000/0.36 \times 10^{-13}$	0.5000	$0.5000/0.11 \times 10^{-3}$
α_{3}	0.0000	$0.0000/0.02 \times 10^{-3}$	0.0000	$0.0000/0.12 \times 10^{-13}$	0.0000	$-0.0005/0.14 \times 10^{-3}$
β_{-2}	0.0000	$0.0000/0.05 \times 10^{-3}$	0.0000	$0.0000/0.18 \times 10^{-13}$	0.0000	$-0.0002/0.27 \times 10^{-3}$
β_{0}	1.0000	$1.0000/0.11 \times 10^{-3}$	1.0000	$1.0000/0.35 \times 10^{-13}$	0.0000	$-0.0004/0.19 \times 10^{-3}$
β_{2}	0.0000	$0.0000/0.06 \times 10^{-3}$	0.0000	$0.0000/0.30 \times 10^{-13}$	0.0000	$-0.0003/0.15 \times 10^{-3}$

Fig. 8. Shown are the actual and estimated (mean/standard deviation) model coefficients for the video sequence of Figure 7 that was de-interlaced with the specified algorithms.

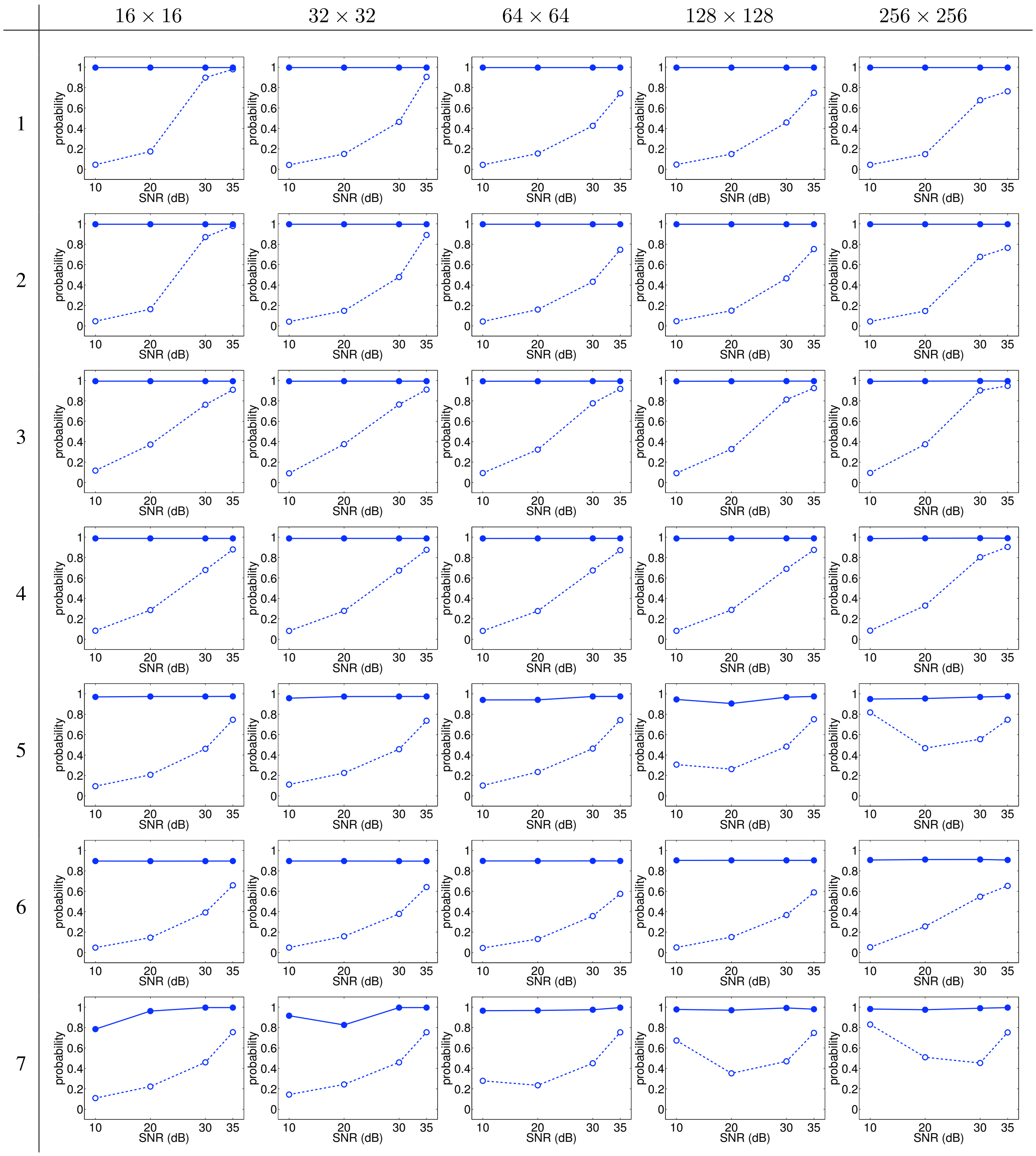

Fig. 9. Shown in each panel is the probability that regions are consistent with de-interlacing as a function of signal to noise ratio (SNR) – the dashed line/open circle corresponds to tampered regions and the solid line/filled circle corresponds to un-tampered regions. Each column corresponds to a different tampered region size (in pixels), and each row corresponds to a different de-interlacing algorithm: (1) line repetition, (2) field insertion, (3) line average, (4) vertical temporal, (5) motion adaptive, (6) motion compensated, and (7) VirtualDub.