Digital Video Forensics

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Weihong Wang

DARTMOUTH COLLEGE

Hanover, New Hampshire

June, 2009

Examining Committee:

(chair) Hany Farid, Ph.D.

Devin Balkcom, Ph.D.

Fabio Pellacini, Ph.D.

Lorenzo Torresani, Ph.D.

Brian W. Pogue, Ph.D.
Dean of Graduate Studies

Abstract

We present new forensic tools that are capable of detecting traces of tampering in digital video without the use of watermarks or specialized hardware. These tools operate under the assumption that video contain naturally occurring properties which are disturbed by tampering, and which can be quantified, measured, and used to expose video fakes. In this context, we offer five new forensic tools (1) Interlaced, (2) De-interlaced, (3) Double Compression, (4) Duplication, and (5) Re-projection, where each technique targets a specific statistical or geometric artifact. Combined, these tools provide a valuable first set of forensic tools for authenticating digital video.

Acknowledgments

First, I would like to thank my advisor, Hany Farid, for bringing me into the image science group. I am grateful for his patient guidance, and for teaching me the value of hard work in research. Thanks as well to the other members of my committee, Devin Balkcom, Fabio Pellacini and Lorenzo Torresani, for their comments and suggestions. I am grateful to my professors: Chris Bailey-Kellogg, Thomas Cormen, Prasad Jayanti, Chris McDonald, Doug McIlroy, Bill McKeeman and Daniel Rockmore, for all that I have learned from them. I'd also like to thank past and current members of the image science group: Siwei Lyu, Alin Popescu, Kimo Johnson, Eric Kee and Jeff Woodward. I am also grateful to the staff and system administrators for keeping everything running smoothly.

I thank my parents and my brother for their love and support all these years. Whenever I encountered difficulties, it was always their love that sustained me.

Contents

Chapter 1

Introduction

With the wide-spread availability of sophisticated and low-cost digital video cameras and the prevalence of video sharing websites such as YouTube, digital videos are playing a more important role in our daily life. Since digital videos can be manipulated, their authenticity cannot be taken for granted. While it is certainly true that tampering with a digital video is more time consuming and challenging than tampering with a single image, increasingly sophisticated digital video editing software is making it easier to tamper with videos.

Of course not every video forgery is equally consequential; the tampering with footage of a popstar may matter less than the alteration of footage of a crime in progress. But the alterability of video undermines our common sense assumptions about its accuracy and reliability as a representation of reality. As digital video editing techniques become more and more sophisticated, it is ever more necessary to develop tools for detecting video forgery.

1.1 Video Forgeries

The movie industry is probably the strongest driving force for improvement of video manipulation technology. With the video editing technology currently available, professionals can easily remove an object from a video sequence, insert an object from a different video source, or even insert an object created by computer graphics software. Certainly, advanced video manipulation technology greatly enriches our visual experience. However, as these techniques become increasingly available to the general public, malicious tampering with

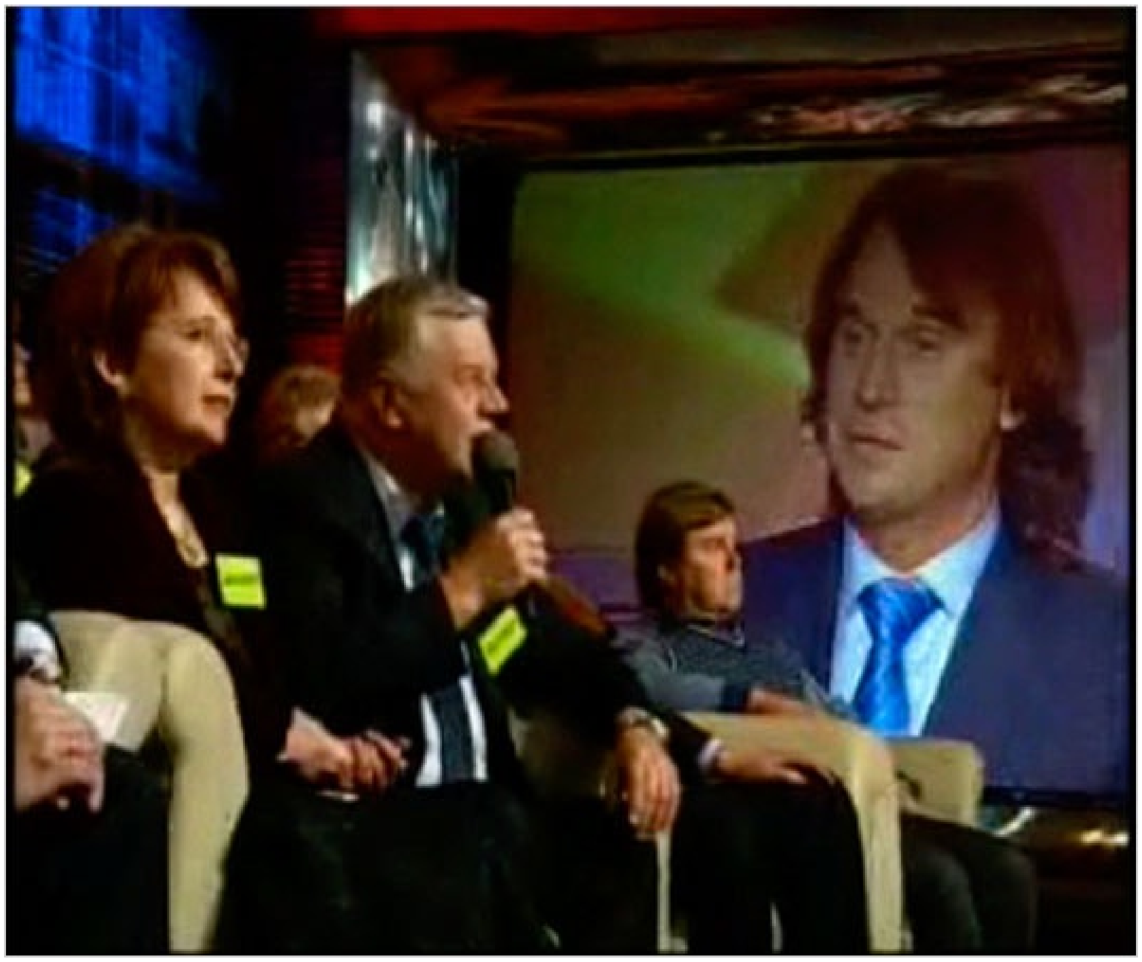


Figure 1.1: Shown is a frame from a Russian talk show in 2007. Mikhail G. Delyagin, a prominent political analyst, was digitally erased from the show though the technicians neglected to erase his legs in one shot.

video recordings is emerging as a serious challenge.

Although tampering with video is relatively hard, in recent years we have begun to encounter video forgeries. Figure 1.1, for example, shows a frame from a Russian talk show in the fall of 2007. In that program, a prominent political analyst named Mikhail G. Delyagin made some tart remarks about Vladimir V. Putin. Later, when the program was broadcast, not only were his remarks cut, he was also digitally removed from the show, though the technicians neglected to erase his legs in one shot.

Growth in video tampering is creating a huge impact on our society. Although currently only a few digital video forgeries have been exposed, such instances are eroding the public trust in video. Therefore, it is urgent for the scientific community to come up with methods for authenticating video recordings.

1.2 Watermarking

One solution to video authentication is digital watermarking. There are several types of watermark. Among them, fragile and semi-fragile watermarks can be used to authenticate videos. Fragile watermarking works by inserting imperceptible information that will be altered if there is any attempt to modify the video. Later, the embedded information can be extracted to verify the authenticity of the video. The semi-fragile watermark works in a similar fashion. The difference is that it is less sensitive to classical user modifications such as compression. The assumption is that these modifications do not affect integrity of the video. The major drawback of the watermarking approach is that a watermark must be inserted at precisely the time of recording, which limits this approach to specially equipped digital cameras.

1.3 Related Work

Since we cannot expect that videos are always recorded with a watermark, we need tools that can detect digital forgeries without the help of watermarks. Researchers in the field of image forensics have been successfully developing tools for image authentication for many years (see [10] for a general survey). These techniques work on the assumption that although digital forgeries may leave no visual clues of having been tampered with, they may alter the underlying statistics of an image. The existing image forensic tools can be roughly categorized into five groups: (1) *Pixel-based techniques* detect anomalies introduced at the pixel level by manipulations such as cloning, resampling, splicing etc. For example, Popescu et al. [42] proposed a computationally efficient algorithm based on principal component analysis (PCA) to detect cloned image regions (also see [12, 31, 36]). (2) *Format-based techniques* leverage statistical correlations introduced by a specific lossy compression scheme. For example, in [34, 43], the authors proposed two techniques to detect images that are compressed twice by JPEG. (3) *Camera-based techniques* utilize artifacts introduced by the camera lens, sensor, or on-chip postprocessing. For example, the authors in [21] described a technique to detect inconsistency of camera response to identify image forgeries. (4) *Physics-based techniques* explicitly model and detect anomalies in the 3D interaction between physical objects, light, and the camera. For example, Johnson et al. [23, 27, 26] proposed a series of tools to detect inconsistencies in 2D and 3D light directions and the light environment. (5) *Geometry-based techniques* analyze the projection geometry of image formation. For example, the authors in [25] proposed using the inconsistency of the estimated principal point across the image as a way to detect image forgeries. While some of these tools may be applicable to video, the unique nature of video warrants its own set of specialized forensic tools.

In recent years, researchers have expanded some of the image forensic tools to incorporate video and proposed several techniques for video authentication that do not rely on watermarks. Kurosawa et al. [29] proposed using the non-uniformity of the dark current of CCD chips for camcorder identification. This technique works under the assumption that the dark current generation rate of some pixels in the CCD may deviate from the average, and that these defective pixels cause a fixed pattern noise that is unique and intrinsic to an individual camera. In [5] Chen et al. extended their image oriented technique in [6] to video and proposed using the photo-response non-uniformity (PRNU) of digital sensors as a way to identify the source digital camcorder. Due to inhomogeneity and impurities in silicon wafers and imperfections introduced by the manufacturing process, pixels in the camera sensor have varying sensitivities to light. This property appears to be constant in time and unique for each imaging sensor. The technique works by estimating the PRNU from a sequence of frames using the Maximum Likelihood Estimator and then detecting the presence of PRNU using normalized cross-correlation. In [39], Mondaini et al. proposed a related technique based on sensor pattern noise. Hsu et al. [20] proposed a technique for locating forged regions in a video using correlation of noise residue at block level. In their method, the noise residual is extracted by using a wavelet denoising filter, after which the correlation of the noise residual is computed for each pair of temporally adjacent blocks. This correlation is later used as a feature by a Bayesian classifier to detect tampering.

1.4 Contributions

While the past few years have seen considerable progress in the area of digital image forensics, less attention has been paid to digital video forensics. In this thesis, we present five forensic tools for detecting tampering in digital videos. These tools can work in the absence of a watermark. The fundamental assumption behind our techniques is that tampering with a digital video may disturb certain underlying properties of the video and these perturbations can be modeled and estimated in order to detect tampering. This is very similar to the approaches used to detect tampering in digital images (e.g., [35, 23, 45, 44, 13, 40, 43]). As an early effort in this field, we propose the following approaches to digital video forensics:

1. **Interlaced.** Most video cameras record video in interlaced mode, which means that even and odd scan lines are recorded at different times. As a result, there is motion between the two sets of scan lines within a frame. Tampering will likely disrupt the expected consistency in the motion within a frame and the motion between frames.

2. **De-interlaced.** Sometimes, interlaced videos are de-interlaced to minimize "combing" artifacts. The de-interlacing procedure introduces correlations among the pixels within a frame and between frames. Tampering, however, is likely to destroy these correlations.

3. **Double MPEG.** When an MPEG video is modified, and re-saved in MPEG format, it is subject to double compression. In this process, two types of artifacts – spatial and/or temporal – will likely be introduced into the resulting video. These artifacts can be quantified and used as evidence of tampering.

4. **Duplication.** Techniques for detecting image duplication have previously been proposed. These techniques, however, are computationally too inefficient to be applicable to a video sequence of even modest length. Therefore, we propose new method for video duplications.

5. **Re-projection.** A simple and popular way to create a bootleg video is to simply record a movie from the theater screen. Such a re-projected video usually introduces

distortion into the intrinsic camera parameters; the distortion to camera skew in particular is evidence of tampering.

Each technique focuses on one specific form of tampering and cannot be applied single-handedly to detect all video forgeries. Taken together, and used in combination, these five tools offer a promising beginning to detecting forgery in digital videos without watermarks. We hope that our methods will inspire the development of many more tools for detecting a wide variety of video forgeries.

Chapter 2

Interlaced

Most video cameras do not simultaneously record the even and odd scan lines of a single frame. 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, $f(x, y, t + 1)$, is recorded at time $t + 1$. If the even and odd fields are simply woven together, as shown in the exaggerated example in Figure 2.1, 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. In this section, we describe a technique for detecting tampering in interlaced video. We show that the motion between the fields of a single frame and across the fields of neighboring frames in an interlaced video should be equal. We propose an efficient way to measure these motions and show how tampering can disturb this relationship. Then, we show the efficacy of our approach on simulated and visually plausible forgeries.

2.1 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 for frame $F(x, y, 1)$ and the odd lines for frame $F(x, y, 2)$, respectively. With the constant

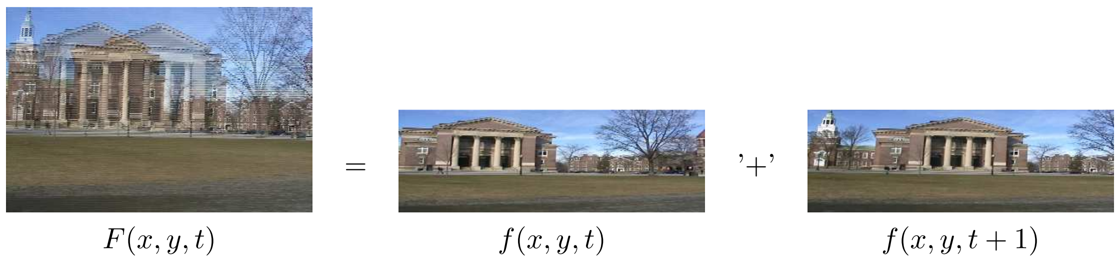


Figure 2.1: One half of the scan lines, $f(x, y, t)$, of a full video frame are recorded at time t, and the other half of the scan lines, $f(x, y, t+1)$ are recorded at time $t+1$. An interlaced frame, $F(x, y, t)$, is created by simply weaving together these two fields.

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 $f(x, y, 3)$. 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 [1].

2.1.1 Motion Estimation

We consider a classic differential framework for motion estimation [19, 1, 48]. 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:

$$f(x, y, t) = f(x + v_x, y + v_y, t - 1), \tag{2.1}$$

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:

$$E(\vec{v}) = \sum_{x,y \in \Omega} \left[f(x, y, t) - f(x + v_x, y + v_y, t - 1) \right]^2, \tag{2.2}$$

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

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:

$$\begin{aligned} E(\vec{v}) &\approx \sum_{x,y\in\Omega} \left[f(x,y,t) - (f(x,y,t) + v_x f_x(x,y,t) + v_y f_y(x,y,t) - f_t(x,y,t))\right]^2 \\ &= \sum_{x,y\in\Omega} \left[f - (f + v_x f_x + v_y f_y - f_t)\right]^2, \end{aligned} \tag{2.3}$$

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:

$$E(\vec{v}) = \sum_{x,y\in\Omega} \left[f_t - v_x f_x - v_y f_y\right]^2 = \sum_{x,y\in\Omega} \left[f_t - \begin{pmatrix} f_x & f_y \end{pmatrix} \begin{pmatrix} v_x \\ v_y \end{pmatrix}\right]^2 = \sum_{x,y\in\Omega} \left[f_t - \vec{f_s}^{\,T} \vec{v}\right]^2. \tag{2.4}$$

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:

$$\frac{dE(\vec{v})}{d\vec{v}} = \sum_{x,y\in\Omega} -2\vec{f_s}\left[f_t - \vec{f_s}^{\,T} \vec{v}\right] \tag{2.5}$$

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

$$\vec{v} = - \begin{pmatrix} \sum_\Omega f_x^2 & \sum_\Omega f_x f_y \\ \sum_\Omega f_x f_y & \sum_\Omega f_y^2 \end{pmatrix}^{-1} \begin{pmatrix} \sum_\Omega f_x f_t \\ \sum_\Omega f_y f_t \end{pmatrix}, \tag{2.6}$$

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.

2.1.2 Spatial/Temporal Differentiation

The spatial/temporal derivatives needed to estimate motion in Equation (2.6) are determined via convolutions [11] as follows:

$$f_x = (\tfrac{1}{2}f_t(x,y,t) + \tfrac{1}{2}f_t(x,y,t-1)) \star d(x) \star p(y) \tag{2.7}$$

$$f_y = (\tfrac{1}{2}f_t(x,y,t) + \tfrac{1}{2}f_t(x,y,t-1)) \star p(x) \star d(y) \tag{2.8}$$

$$f_t = (\tfrac{1}{2}f_t(x,y,t) - \tfrac{1}{2}f_t(x,y,t-1)) \star p(x) \star p(y), \tag{2.9}$$

where the 1-D filters are $d(\cdot) = \begin{bmatrix}0.425287 & 0.0 & -0.425287\end{bmatrix}$ and $p(\cdot) = \begin{bmatrix}0.2298791 & 0.540242 & 0.2298791\end{bmatrix}$. Note that while we employ 2-tap filters for the temporal filtering ($\begin{pmatrix}\frac{1}{2} & \frac{1}{2}\end{pmatrix}$ and $\begin{pmatrix}\frac{1}{2} & -\frac{1}{2}\end{pmatrix}$), 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.

2.1.3 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 (2.6). 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.

2.2 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 $3 \leq t \leq 2T - 3$. 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 2.2 are the estimated inter-field and inter-frame motions for each video

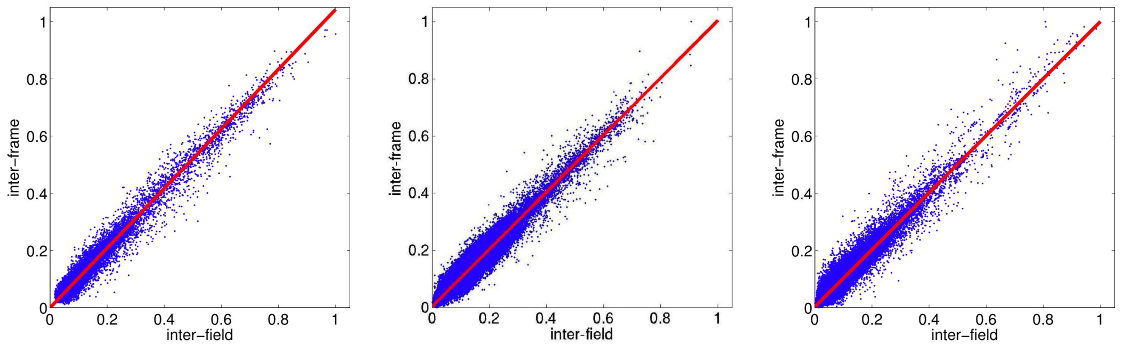


Figure 2.2: Shown in each panel is the normalized inter-field motion versus the normalized inter-frame motion for three different video sequences. The solid line represents a line fit to the underlying data points. We expect the motion ratio to be near 1 in an authentic video sequence and to deviate significantly for a doctored video.

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 2.3 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. Shown in lower portion of Figure 2.3 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.

Since compression will introduce perturbations into the image, 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.

2.2.1 Frame Rate Down-Conversion

This technique can be adapted to detect frame rate down-conversion. 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.

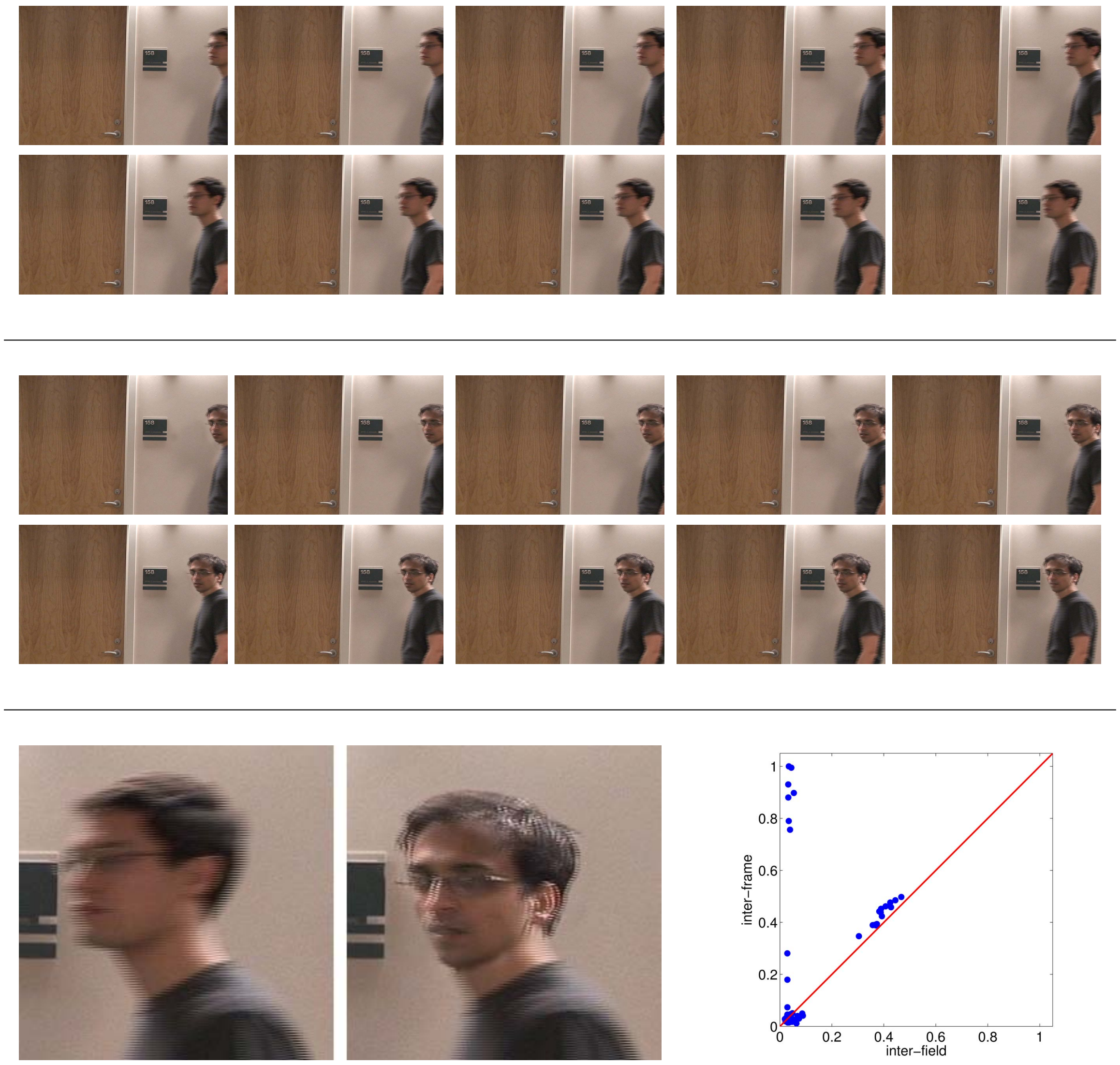


Figure 2.3: Shown in the top two rows are 10 frames of an original interlaced video. Shown in the next two rows is a doctored version of this video sequence, and in the lower-left are enlargements of the last frame of the original and doctored video. The plot shows the inter-field versus the inter-frame 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).

2.3 Discussion

We have presented a technique for detecting tampering in 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. This technique can localize tampering both in time and space. It can also be adapted slightly to detect frame rate down-conversion. Compression artifacts have little effect on the estimation of motion in interlaced video, so this approach is appropriate for a range of interlaced video. The weakness of this method is that it cannot detect manipulations in regions where there is no motion, since in this case the inter-field and inter-frame motion are constantly zero.

Counterattacking this technique would be relatively difficult 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.

Chapter 3

De-Interlaced

In Chapter 2, we described a technique to detect tampering in interlaced video. Interlaced video usually has 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 (see Figure 3.1), usually relying on some form of spatial and temporal interpolation (see [7] for a general overview and [8, 49, 2, 41] for specific and more advanced approaches). In this section, we describe a technique for detecting tampering in de-interlaced video. We quantify the correlations introduced by the camera or software de-interlacing algorithms and show how tampering can disturb these correlations. Then, we show the efficacy of our approach on simulated and visually plausible forgeries.

3.1 De-Interlacing Algorithms

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

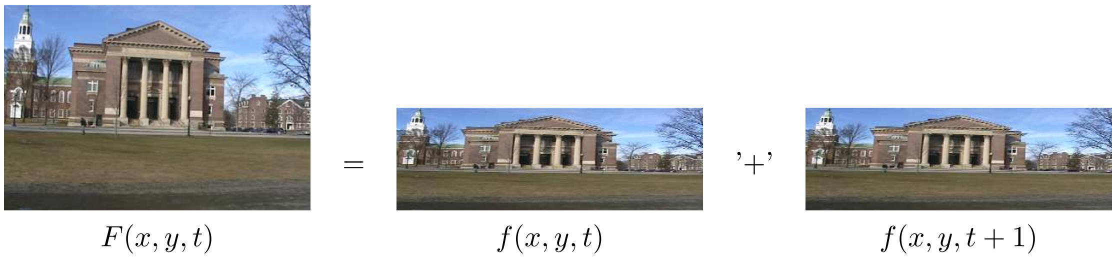


Figure 3.1: One half of the scan lines, $f(x, y, t)$, of a full video frame are recorded at time t, and the other half of the scan lines, $f(x, y, t + 1)$, are recorded at time $t + 1$. A de-interlaced frame, $F(x, y, t)$, is created by combining these two fields to create a full frame.

video cameras.

3.1.1 Line Repetition

In this simplest of de-interlacing algorithms, Figure 3.2(a), the scan lines of each field, $f(x, y, t)$, are duplicated to create a full frame, $F(x, y, t)$:

$$F(x, y, t) = f(x, \lceil y/2 \rceil, t). \tag{3.1}$$

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

3.1.2 Field Insertion

In this de-interlacing algorithm, Figure 3.2(b), neighboring fields, $f(x, y, t)$, are simply combined to create a full frame, $F(x, y, t)$. For odd values of t:

$$F(x, y, t) = \begin{cases} f(x, (y + 1)/2, t) & y \bmod 2 = 1 \\ f(x, y/2, t + 1) & y \bmod 2 = 0 \end{cases}, \tag{3.2}$$

and for even values of t:

$$F(x, y, t) = \begin{cases} f(x, y/2, t) & y \bmod 2 = 0 \\ f(x, (y + 1)/2, t + 1) & y \bmod 2 = 1 \end{cases}. \tag{3.3}$$

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”.

3.1.3 Line Averaging

In this easy to implement and popular technique, Figure 3.2(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:

$$F(x,y,t) = \begin{cases} f(x,(y+1)/2,t) & y \bmod 2 = 1 \\ \frac{1}{2}f(x,y/2,t) + \frac{1}{2}f(x,y/2+1,t) & y \bmod 2 = 0 \end{cases}, \tag{3.4}$$

and for even values of t:

$$F(x,y,t) = \begin{cases} f(x,y/2,t) & y \bmod 2 = 0 \\ \frac{1}{2}f(x,(y+1)/2-1,t) + \frac{1}{2}f(x,(y+1)/2,t) & y \bmod 2 = 1 \end{cases}. \tag{3.5}$$

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.

3.1.4 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:

$$F(x,y,t) = \begin{cases} f(x,(y+1)/2,t) & y \bmod 2 = 1 \\ c_1 f(x,y/2-1,t) + c_2 f(x,y/2,t) + c_3 f(x,y/2+1,t)+ & \\ c_4 f(x,y/2+2,t) + c_5 f(x,y/2-1,t+1)+ & y \bmod 2 = 0 \\ c_6 f(x,y/2,t+1) + c_7 f(x,y/2+1,t+1) & \end{cases} . \tag{3.6}$$

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:

$$F(x,y,t) = \begin{cases} f(x,y/2,t) & y \bmod 2 = 0 \\ c_1 f(x,(y+1)/2-2,t) + c_2 f(x,(y+1)/2-1,t)+ & \\ c_3 f(x,(y+1)/2,t) + c_4 f(x,(y+1)/2+1,t)+ & \\ c_5 f(x,(y+1)/2-1,t+1) + c_6 f(x,(y+1)/2,t+1)+ & y \bmod 2 = 1 \\ c_7 f(x,(y+1)/2+1,t+1) & \end{cases} . \tag{3.7}$$

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.

3.1.5 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.

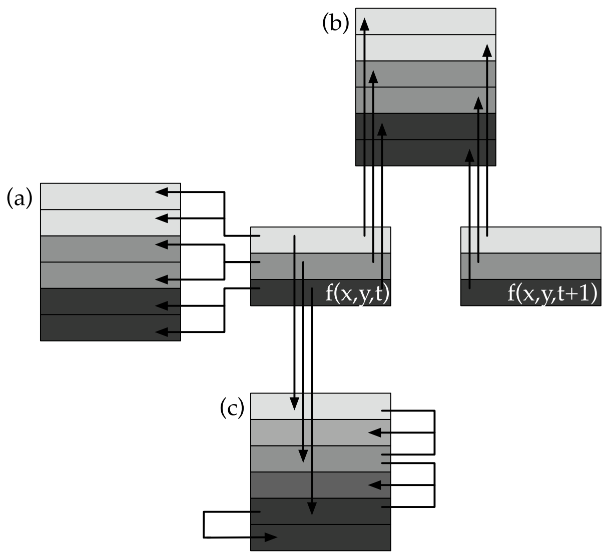


Figure 3.2: Illustrated in this schematic are the (a) line repetition, (b) field insertion and (c) line averaging de-interlacing algorithms. The individual fields, $f(x, y, t)$ and $f(x, y, t+1)$, are illustrated with three scan lines each, and the full de-interlaced frames are illustrated with six scan lines.

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:

$$F(x, y, t) = (1 - \alpha(x, y))F_1(x, y, t) + \alpha(x, y)F_2(x, y, t). \tag{3.8}$$

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

3.1.6 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 3.1.2. 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.

3.2 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 [45], 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 (3.4). In this case:

$$F(x,y,t) = \tfrac{1}{2}F(x,y-1,t) + \tfrac{1}{2}F(x,y+1,t) \tag{3.9}$$

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:

$$F_o(x,y,t) = \sum_{i\in\{-3,-1,1,3\}} \alpha_i F(x,y+i,t) + \sum_{i\in\{-2,0,2\}} \beta_i F(x,y+i,t+1) \; + \; n(x,y), \tag{3.10}$$

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 (3.10) 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:

$$P\{F_o(x,y,t) \in M_1 \mid F_o(x,y,t)\} = \frac{P\{F_o(x,y,t) \mid F_o(x,y,t) \in M_1\} P\{F_o(x,y,t) \in M_1\}}{\sum_{i=1}^{2} P\{F_o(x,y,t) \mid F_o(x,y,t) \in M_i\} P\{F_o(x,y,t) \in M_i\}}, \tag{3.11}$$

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:

$$P\{F_o(x,y,t) \mid F_o(x,y,t) \in M_1\} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{\left(F_o(x,y,t) - \tilde{F}_o(x,y,t)\right)^2}{2\sigma^2} \right], \tag{3.12}$$

where:

$$\tilde{F}_o(x,y,t) = \sum_{i \in \{-3,-1,1,3\}} \alpha_i F(x,y+i,t) + \sum_{i \in \{-2,0,2\}} \beta_i F(x,y+i,t+1). \tag{3.13}$$

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:

$$E(\{\alpha_i, \beta_i\}) = \sum_{x,y} w(x,y) \left(F_o(x,y,t) - \tilde{F}_o(x,y,t)\right)^2, \tag{3.14}$$

where the weights $w(x,y) \equiv P\{F_o(x,y,t) \in M_1 \mid F_o(x,y,t)\}$, Equation (3.11). 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:

$$\begin{aligned} &\sum_{x,y} w(x,y)F(x,y+j,t)F_o(x,y,t) \\ &\qquad = \sum_{i\in\{-3,-1,1,3\}} \alpha_i \left(\sum_{x,y} w(x,y)F(x,y+i,t)F(x,y+j,t) \right) \\ &\qquad\quad + \sum_{i\in\{-2,0,2\}} \beta_i \left(\sum_{x,y} w(x,y)F(x,y+i,t+1)F(x,y+j,t) \right). \end{aligned} \tag{3.15}$$

The derivative with respect to β_j is:

$$\begin{aligned} &\sum_{x,y} w(x,y)F(x,y+j,t+1)F_o(x,y,t) \\ &\qquad = \sum_{i\in\{-3,-1,1,3\}} \alpha_i \left(\sum_{x,y} w(x,y)F(x,y+i,t)F(x,y+j,t+1) \right) \\ &\qquad\quad + \sum_{i\in\{-2,0,2\}} \beta_i \left(\sum_{x,y} w(x,y)F(x,y+i,t+1)F(x,y+j,t+1) \right). \end{aligned} \tag{3.16}$$

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:

$$\sigma^2 = \frac{\sum_{x,y} w(x,y)(F_o(x,y,t) - \tilde{F}_o(x,y,t))^2}{\sum_{x,y} w(x,y)}. \tag{3.17}$$

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

3.3 Results

Shown in the top portion of Figure 3.3 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 (3.11), 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

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

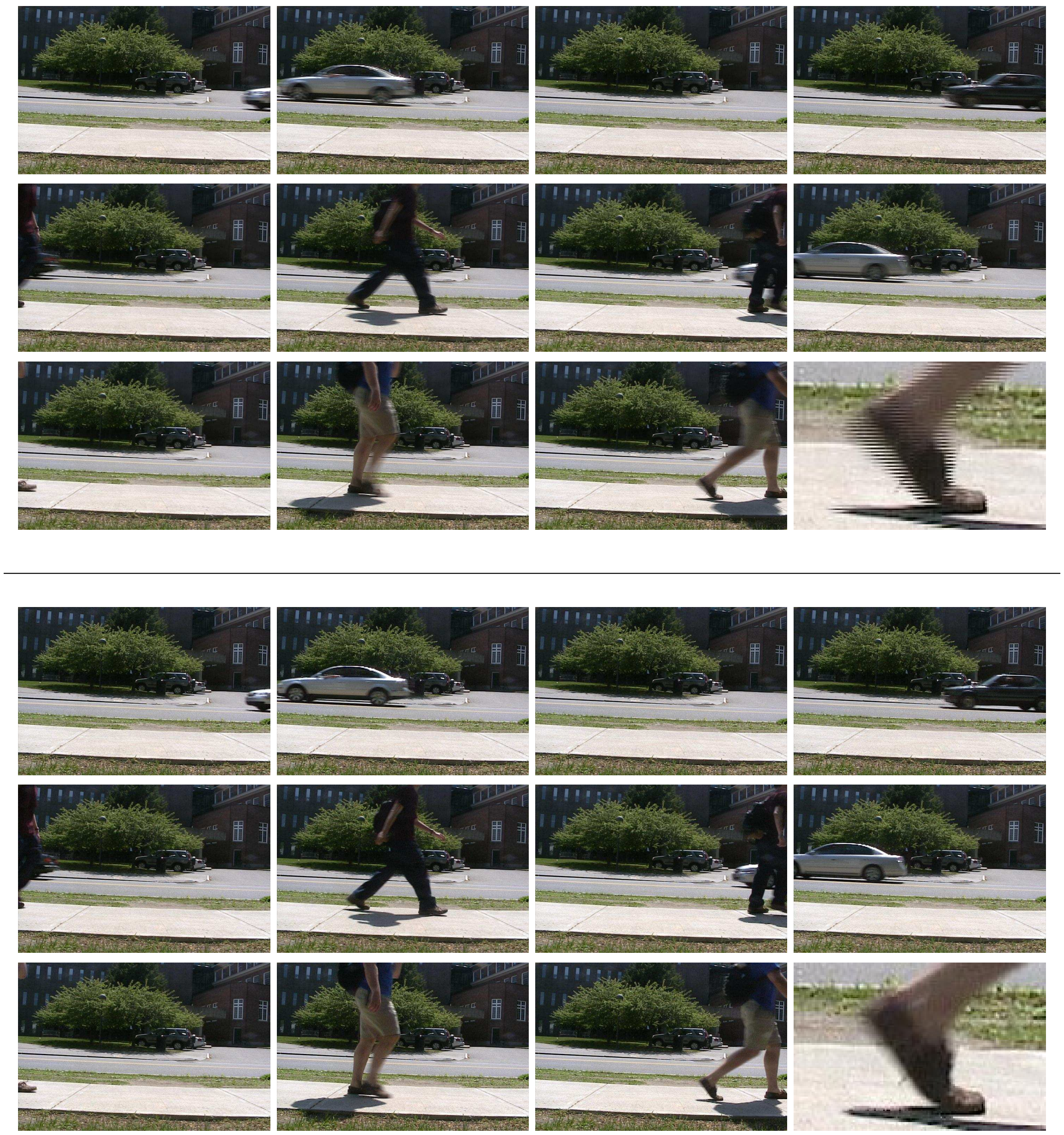


Figure 3.3: Shown in the top portion are eleven frames of a 250-frame long video sequence. Shown in the bottom portion are the same eleven frames de-interlaced using the line average algorithm – notice that the interlacing artifacts are largely reduced. Shown in the lower-right corner is an enlargement of the pedestrian's foot from the last frame.

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 3.1.5, 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 (3.10).

Shown in Figure 3.4 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.

Shown in Figure 3.5 are results for detecting tampering in the video sequence of Figure 3.3. 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 3.5 is the average probability, over all frames, as reported by the EM algorithm for the central tampered region and the surrounding untampered 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 untampered 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 untampered region.

	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	$-0.0003/0.15 \times 10^{-3}$

Figure 3.4: Shown are the actual and estimated (mean/standard deviation) model coefficients for the video sequence of Figure 3.3 that was de-interlaced with the specified algorithms.

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 (3.12) 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

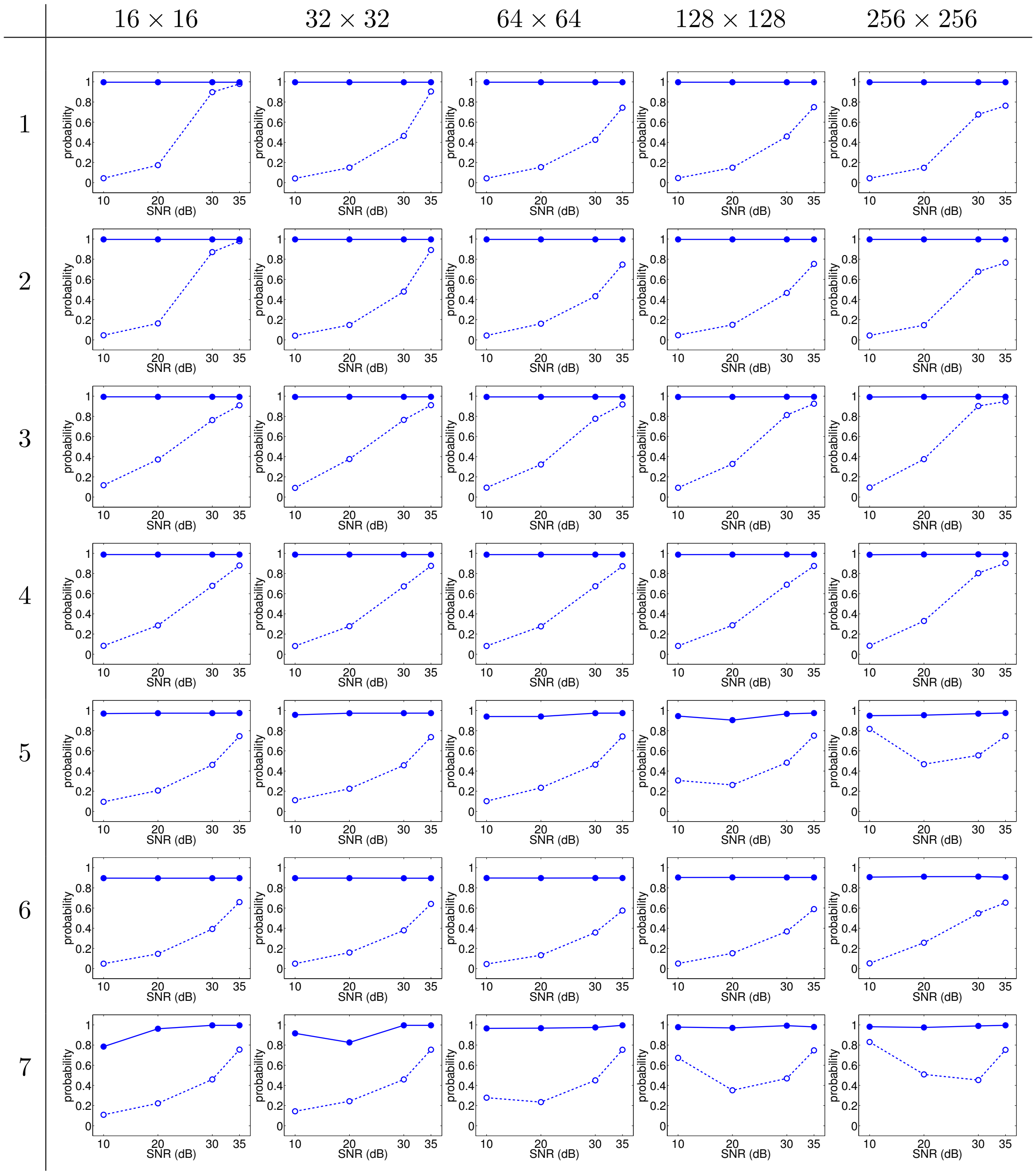


Figure 3.5: Shown in each panel is the probability that regions are consistent with de-interlacing as a function of signal to noise ratio (SNR) – the dashed line/open circle corresponds to tampered regions and the solid line/filled circle corresponds to untampered regions. Each column corresponds to 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.

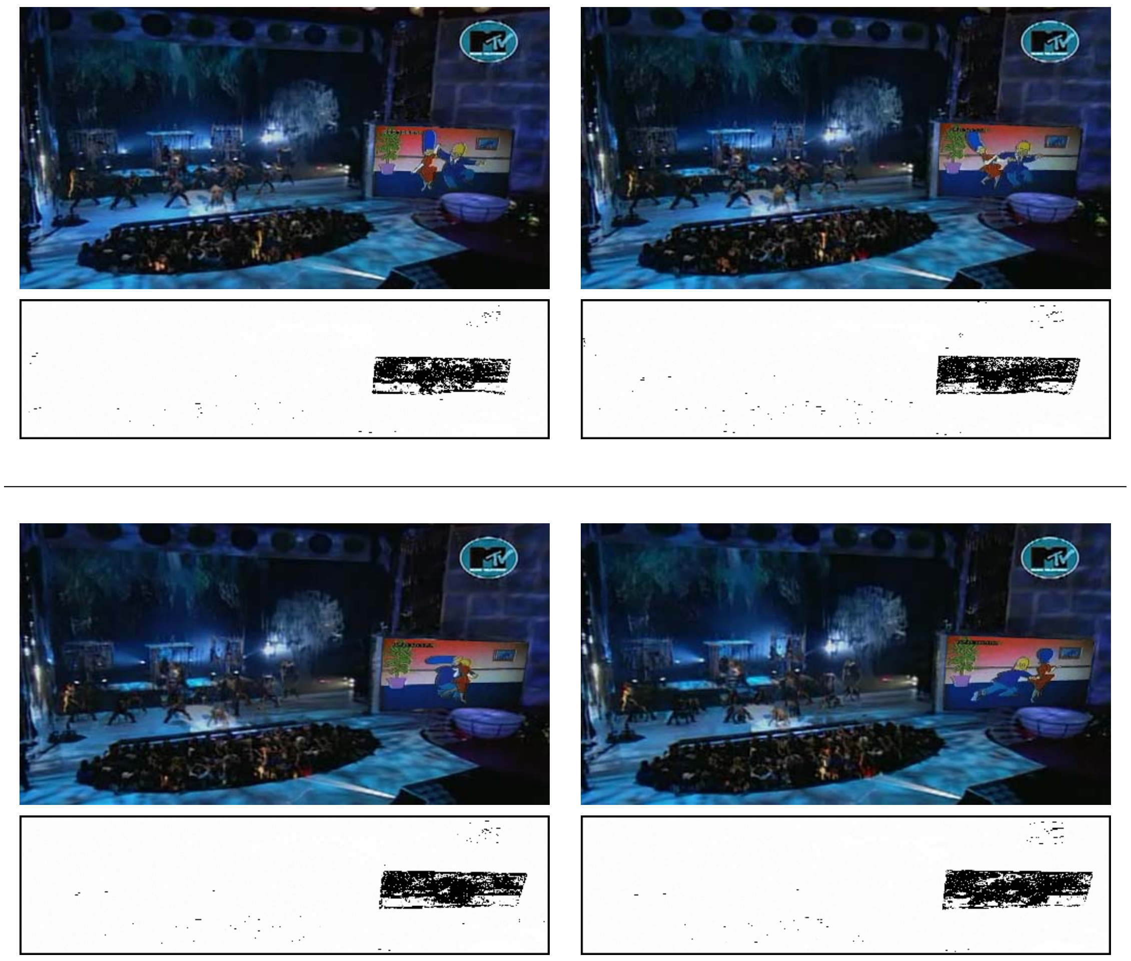


Figure 3.6: Four frames of a Britney Spears concert that have been doctored to include a dancing Marge and Homer Simpson. Shown below are the de-interlacing results that reveal the tampered region.

determine if a region has been tampered with.

Shown in Figure 3.6 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.

3.3.1 Frame Rate Up-Conversion

This technique can be adapted to detect frame rate up-conversion. 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., [30, 3]).

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:

$$F(x,y,t) \;=\; \sum_{i\in\{-1,1\}} \alpha_i F(x,y,t+i), \tag{3.18}$$

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 (3.12). 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).

3.4 Discussion

We have presented a technique for detecting tampering in de-interlaced video. We explicitly model the correlations introduced by de-interlacing algorithms, and show how tampering can destroy these correlations. This technique can localize tampering both in time and in space and can also be slightly adapted to detect frame rate up-conversion that might result from video manipulation. Compression artifacts make it somewhat more difficult to estimate the de-interlacing correlations, so this approach is most appropriate for relatively high quality video.

The model parameters for the untampered region of each frame are consistent throughout the entire sequence. Therefore, we can estimate one set of model parameters for all the frames together instead of for each frame separately. Due to the large amount of data in a video sequence, we need to reduce the computational complexity to a manageable scale. To do this, we can randomly sample pixels from the whole video sequence to provide the input data for the EM algorithm. Then the estimated model parameters are used to calculate the probability map for each frame. This approach can greatly reduce the computational cost because the EM algorithm is only applied once. At the same time, it also improves the accuracy of the estimation of the model parameters because the input data are no longer restricted to one frame.

One way to counterattack the de-interlacing forensic tool would be to recreate the correlations that were destroyed by the tampering. This could be achieved by first doctoring a video, then generating an interlaced video (split the even and odd scan lines), and finally applying a de-interlacing algorithm to generate a new de-interlaced video with intact correlations. If the original video camera is available, this approach requires the forger to employ the same de-interlacing algorithm as that used by the original camera.

Chapter 4

Double MPEG

MPEG is a well established video compression standard. In this chapter, we consider detecting tampering in videos compressed by MPEG. When an MPEG video is modified, and re-saved in MPEG format, it is subject to double compression. In this process, two types of artifacts – spatial and/or temporal – will likely be introduced into the resulting video. We describe how these artifacts can be quantified and estimated in order to detect forgeries. We begin by briefly describing the relevant components of MPEG video compression. Then we propose two techniques, each capable of capturing one type of double compression artifact, and show their efficacy in detecting tampering.

4.1 Video Compression

The MPEG video standard (MPEG-1 and MPEG-2) compresses video data by reducing both spatial redundancy within individual frames and temporal redundancy across frames [47]. In this section, we give a brief overview of the MPEG standard.

4.1.1 Coding Sequence

In a MPEG encoded video sequence, there are three types of frames: intra (I), predictive (P) and bi-directionally predictive (B), each offering varying degrees of compression. These frames typically occur in a periodic sequence. A common sequence, for example, is:

$$I_1\ B_2\ B_3\ P_4\ B_5\ B_6\ P_7\ B_8\ B_9\ P_{10}\ B_{11}\ B_{12}\ I_{13}\ B_{14}\ \cdots,$$

where the subscripts are used to denote time. Such an encoding sequence is parameterized by the number of frames in a sequence, N, and the spacing of the P-frames, M. In the above sequence $N = 12$ and $M = 3$. Each N frames is referred to as a group of pictures (GOP).

I-frames are encoded without reference to any other frames in the sequence. P-frames are encoded with respect to the previous I- or P-frame, and offer increased compression over I-frames. B-frames are encoded with respect to the previous and next I- or P-frames and offer the highest degree of compression. In the next three sections,, these encodings are described in more detail.

4.1.2 *I*-frame

I-frames are typically the highest quality frames of a video sequence but afford the least amount of compression. I-frames are encoded using a fairly standard JPEG compression scheme. A color frame (RGB) is first converted into luminance/chrominance space (YUV). The two chrominance channels (UV) are subsampled relative to the luminance channel (Y), typically by a factor of $4:1:1$. Each channel is then partitioned into 8×8 pixel blocks. A macroblock is then created by grouping together four such Y-blocks, one U-block, and one V-block in a 16×16 pixel neighborhood. After applying a discrete cosine transform (DCT) to each block, the resulting coefficients are quantized and run-length and variable-length encoded. The amount of quantization of the DCT coefficients depends on their spatial frequencies (higher frequencies are typically quantized more than lower frequencies). The DC coefficient (the $(0, 0)$ frequency) and the AC coefficients (all other frequencies) are quantized differently. Of our interest is the quantization of the AC coefficients, which is determined by two factors: the quantization table and the quantization scale. The quantization table specifies the quantization for each of 64 DCT frequencies in each YUV channel, and is generally held fixed across the entire video. The quantization scale (a scalar) can vary from frame to frame and from macroblock to macroblock, thus allowing the quantization to adapt to the local image structure. The final quantization for each DCT coefficient is then simply the product of the quantization table and scale.

4.1.3 P-frame

In the encoding of an I-frame, compression is achieved by reducing the spatial redundancies within a single video frame. The encoding of a P-frame is intended to reduce the temporal redundancies across frames, thus affording better compression rates. Consider for example a video sequence in which the motion between frames can be described by a single global translation. In this case, considerable compression can be achieved by encoding the first frame in the sequence and the amount of inter-frame motion (a single vector) for each subsequent frame. The original sequence can then be reconstructed by motion correcting (e.g., warping) the first frame according to the motion vectors. In practice, of course, a single motion vector is not sufficient to accurately capture the motion in most natural video sequences. As such, the motion between a P-frame and its preceding I- or P-frame is estimated for each 16×16 pixel block in the frame. A standard block-matching algorithm is typically employed for motion estimation, Figure 4.1(a). A motion estimated version of frame 2 can then be generated by warping the first frame according to the estimated motion, Figure 4.1(b). The error between this predicted frame, and the actual frame is then computed, Figure 4.1(c). Both the motion vectors and the motion errors are encoded and transmitted (the motion errors are statically encoded using a similar JPEG compression scheme as used for encoding I-frames). With relatively small motion errors, this scheme yields good compression rates. The decoding of a P-frame is then a simple matter of warping the previous frame according to the motion vector and adding the motion errors. By removing temporal redundancies, the P-frames afford better compression than the I-frames, but at a cost of a loss in quality. These frames are of lower quality because of the errors in motion estimation and the subsequent compression of the motion errors.

4.1.4 B-frame

Similar to a P-frame, a B-frame is encoded using motion compensation. Unlike a P-frame, however, a B-frame employs a past, future, or both of its neighboring I- or P-frames for motion estimation. By considering two moments in time, more accurate motion estimation is possible, and in turn better compression rates. The decoding of a B-frame requires that

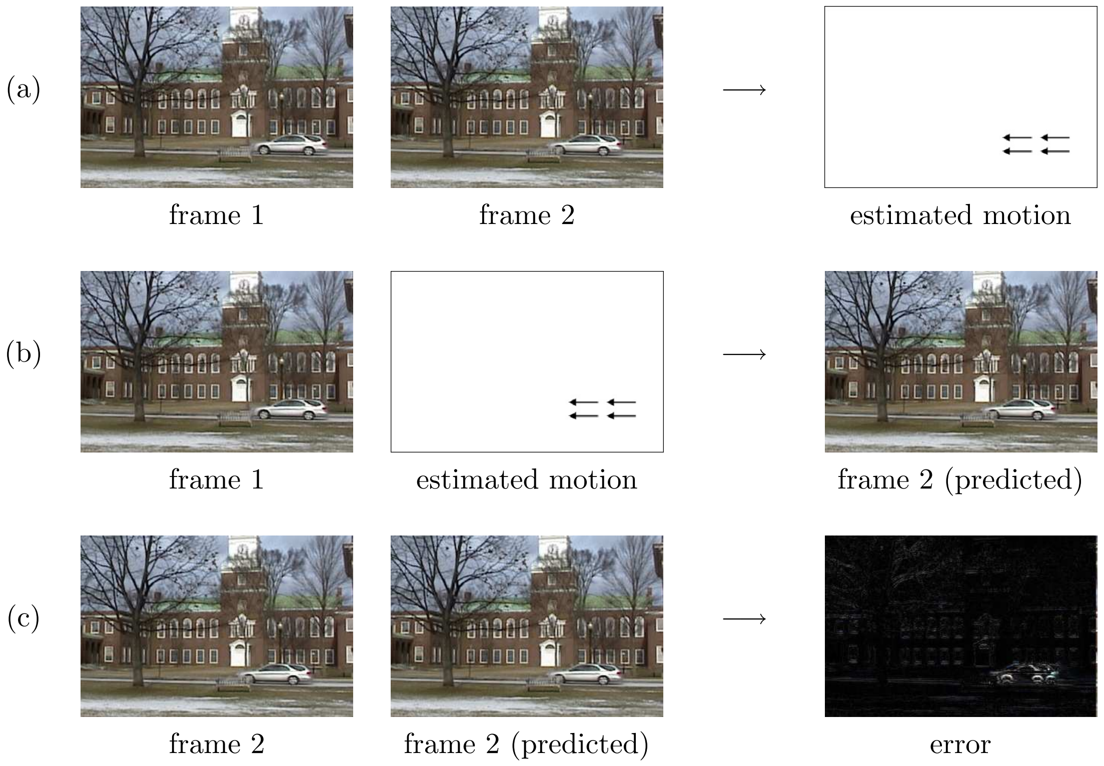


Figure 4.1: Motion estimation is used to encode P- and B-frames of a MPEG video sequence: (a) motion is estimated between a pair of video frames; (b) the first frame is motion compensated to produce a predicted second frame; and (c) the error between the predicted and actual second frame is computed. The motion estimation and errors are encoded as part of a MPEG video sequence.

both frames, upon which motion estimation relied, be transmitted first.

4.2 Spatial

We introduce a technique for detecting if a video frame or part of it was MPEG compressed twice as an I-frame. This manipulation might result from something as simple as recording an MPEG video, editing it, and re-saving it as another MPEG video. This manipulation might also arise from a more sophisticated green-screening in which two videos are composited together. We show that such double compression introduces specific artifacts in the DCT coefficients of the I-frames of an MPEG video. In [43], Popescu et al. have described such double compression artifacts (see also [34]). In [51], we showed that such artifacts can be measured to detect doubly compressed I-frames. However, unlike these earlier tools, this technique can detect localized tampering in regions as small as 16×16 pixels.

4.2.1 Methods

The final quality of an MPEG video is determined by several factors. Among these is the amount of quantization applied to each I-frame. Therefore, when an I-frame is compressed twice with different compression qualities, the DCT coefficients are subjected to two levels of quantization. Recall the final quantization is determined by two factors: the quantization table and the quantization scale. Since video encoders typically employ the default quantization matrix, we assume that the variation in quantization is governed by the quantization scale.

Double Quantization

Consider a DCT coefficient u. In the first compression, the quantized DCT coefficient x is given by:

$$x = \left[\frac{u}{q_1} \right], \tag{4.1}$$

where q_1 (a strictly positive integer) is the first quantization step, and $[\cdot]$ is the rounding function. When the compressed video is decoded to prepare for the second compression, the quantized coefficients are de-quantized back to their original range:

$$y = xq_1. \tag{4.2}$$

Note that the de-quantized coefficient y is a multiple of q_1. In the second compression, the DCT coefficient y is quantized again:

$$z = \left[\frac{y}{q_2} \right], \tag{4.3}$$

where q_2 is the second quantization step and z is the final double quantized DCT coefficient.

To illustrate the effect of double quantization, consider an example where the original DCT coefficients are normally distributed in the range $[-30, 30]$. Shown in Figure 4.2(a) is the distribution of these coefficients after being quantized with $q_1 = 5$, Equation (4.1).

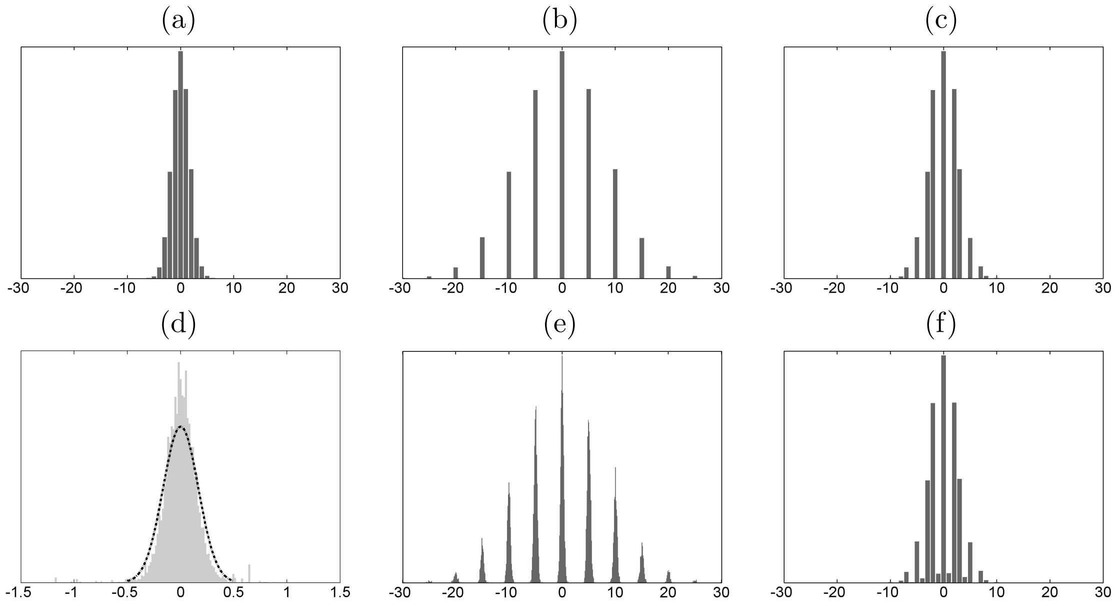


Figure 4.2: Shown are: (a) the distribution of singly quantized coefficients with $q_1 = 5$; (b) the distribution of these coefficients de-quantized; (c) the distribution of doubly quantized coefficients with $q_1 = 5$ followed by $q_2 = 3$ (note the empty bins in this distribution); (d) a magnified view of the central bin in panel (e) – the dashed line is a Gaussian distribution fit to the underlying coefficients; and (e-f) the same distributions shown in panels (b) and (c) but with rounding and truncation introduced after the coefficients are decoded.

Shown in Figure 4.2(b) is the distribution of the de-quantized coefficients, Equation (4.2) (where every coefficient is now a multiple of the first quantization 5). And shown in Figure 4.2(c) is the distribution of doubly quantized coefficients with steps $q_1 = 5$ followed by $q_2 = 3$, Equation (4.3). Because the step size decreases from $q_1 = 5$ to $q_2 = 3$ the coefficients are re-distributed into more bins in the second quantization than in the first quantization. As a result, the distribution of the doubly quantized coefficients contains empty bins (Figure 4.2(c) as compared to Figure 4.2(a)). As described in [34, 43, 18], a similar, although less pronounced, artifact is introduced when the step size increases between quantizations. Since we will be computing double compression artifacts at the level of a single macro-block, we will restrict ourselves to the more pronounced case when $q_1 > q_2$.

Modeling Double Quantization

Equations (4.1)-(4.3) describe the effects of double compression in an idealized setting. In practice, however, when a compressed video is de-quantized, Equation (4.2), and an inverse DCT applied, the resulting pixel values are rounded to the nearest integer and truncated into the range $[0, 255]$. When the forward DCT is then applied, the coefficients will no longer be strict multiples of the first quantization step. Shown in Figure 4.2(d) is an example of this effect, where only a single bin is shown – note that instead of being an impulse at 0, the coefficients approximately follow a normal distribution centered at zero. Superimposed on this distribution is a Gaussian distribution fit to the underlying coefficients. Shown in Figure 4.2(e) is an example of how the rounding and truncation affect the entire distribution (note the contrast to the ideal case shown in panel (b)). After the second compression, the rounding and truncation are propagated into the doubly quantized coefficients. As a result, the previously empty bins are no longer empty, as shown in Figure 4.2(f), as compared to panel (c).

We therefore model the distribution of singly compressed and de-quantized coefficients with a Gaussian distribution:

$$P_{q_1}(y|x) = N(y; xq_1, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-xq_1)^2}{2\sigma^2}}, \tag{4.4}$$

with mean xq_1 and standard deviation σ. This conditional probability describes the distribution of de-quantized coefficients y with respect to x.

The distribution of doubly compressed coefficients is then given by:

$$\begin{aligned} P_{q_1}(z|x) &= \int_{(z-0.5)q_2}^{(z+0.5)q_2} P_{q_1}(y|x)dy \\ &= \int_{(z-0.5)q_2}^{(z+0.5)q_2} N(y; xq_1, \sigma)dy, \end{aligned} \tag{4.5}$$

where the integration bounds mimic the rounding function.

Now, the marginal distribution on the observed doubly compressed coefficients z is given by:

$$\begin{aligned} P_{q_1}(z) &= \sum_x P_{q_1}(x) P_{q_1}(z|x) \\ &= \sum_x P_{q_1}(x) \int_{(z-0.5)q_2}^{(z+0.5)q_2} N(y; xq_1, \sigma)dy. \end{aligned} \tag{4.6}$$

The distribution of $P_{q_1}(z)$ describes the expected distribution of DCT coefficients that result from having been quantized with step size q_1 followed by q_2. Since the second quantization q_2 can be determined directly from the encoded video, this distribution can be used to determine if the observed DCT coefficients are consistent with double compression, where the first compression occurred with quantization q_1.

Note that in our model of $P_{q_1}(z)$, Equation (4.6), the marginal probability $P_{q_1}(x)$ that describes the distribution of the original quantized coefficients is unknown. We next describe how to estimate this unknown distribution.

Let $Z = \{z_1, z_2, \dots, z_n\}$ denote a set of n observations of the DCT coefficients extracted from a single macroblock. Given Z, the distribution $P_{q_1}(x)$ can be estimated using the expectation-maximization (EM) algorithm [9]. The EM algorithm is a two-step iterative algorithm. In the first E-step the distribution of x given each observation z_i is estimated to yield $P_{q_1}(x|z_i)$. In the second M-step the distribution of x is computed by integrating the estimated $P_{q_1}(x|z_i)$ over all possible z_i to yield the desired $P_{q_1}(x)$.

More specifically, in the E-step, we estimate $P_{q_1}(x|z_i)$ using Bayes' rule:

$$P_{q_1}(x|z_i) = \frac{P_{q_1}(x)P_{q_1}(z_i|x)}{P_{q_1}(z_i)}, \tag{4.7}$$

where $P_{q_1}(z_i|x)$ is given by Equation (4.5) and $P_{q_1}(z_i)$ is given by Equation (4.6). Note that this step assumes a known $P_{q_1}(x)$, which can be initialized randomly in the first iteration. In the M-step, $P_{q_1}(x)$ is updated by numerically integrating $P_{q_1}(x|z_i)$ over all possible z_i:

$$P_{q_1}(x) = \frac{1}{n}\sum_{i=1}^{n} P_{q_1}(x|z_i). \tag{4.8}$$

These two steps are iteratively executed until convergence.

Forensics

Our model of double compression described in the previous section can be used to determine if a set of DCT coefficients have been compressed twice with quantization steps of q_1 followed by q_2. Let Z denote the DCT coefficients from a single macroblock whose quantization scale factor is q_2 (the value of q_2 can be extracted from the underlying encoded video).

Let $P(z)$ denote the distribution of Z. This distribution can be compared to the expected distribution $P_{q_1}(z)$, Equation(4.6), that would arise if the coefficients are the result of double quantization by steps q_1 followed by q_2. To measure the difference between the observed $P(z)$ and modeled $P_{q_1}(z)$ distributions we employ a slight variant of the normalized Euclidean distance[1]:

$$D(P(z), P_{q_1}(z)) = \sqrt{\sum_z \frac{(P(z) - P_{q_1}(z))^2}{s^2(z)}}, \tag{4.9}$$

where $s(z)$ is the empirically measured standard deviation of the difference between the probability distributions of coefficients double quantized with steps q_1 followed by q_2 and the corresponding model $P_{q_1}(z)$. Note that the normalized Euclidean distance would have defined $s(z)$ as the standard deviation of $P(z)$, whereas we use the standard deviation of the difference between $P(z)$ and the corresponding model.

This distance is then converted into a probability:

$$P(Z|q_1) = e^{-\alpha D(P(z), P_{q_1}(z))}, \tag{4.10}$$

where the scalar α controls the exponential decay. This probability quantifies the likelihood that the macroblock's coefficients Z were previously quantized by a value of q_1.

In order to determine if a macroblock has been doubly compressed, we consider all possible values of q_1 that are strictly greater than q_2. The maximal value of $P(Z|q_1)$ over all q_1 is taken as the probability that a macroblock has been doubly compressed. This process is repeated for each macroblock, and for each video frame.

Confidence Coefficient

Shown in Figure 4.3 are distributions for (a) an original set of coefficients, and these coefficients (b) singly quantized ($q_1 = 10$) and (c) doubly quantized ($q_1 = 12$ and $q_2 = 10$). As expected, there is a tell-tale empty bin in the doubly quantized distribution. Consider now the distributions in panels (d)-(f). The original distribution in panel (d) has no values

[1]The normalized Euclidean distance is a special case of the Mahalanobis distance with an assumed diagonal covariance matrix.

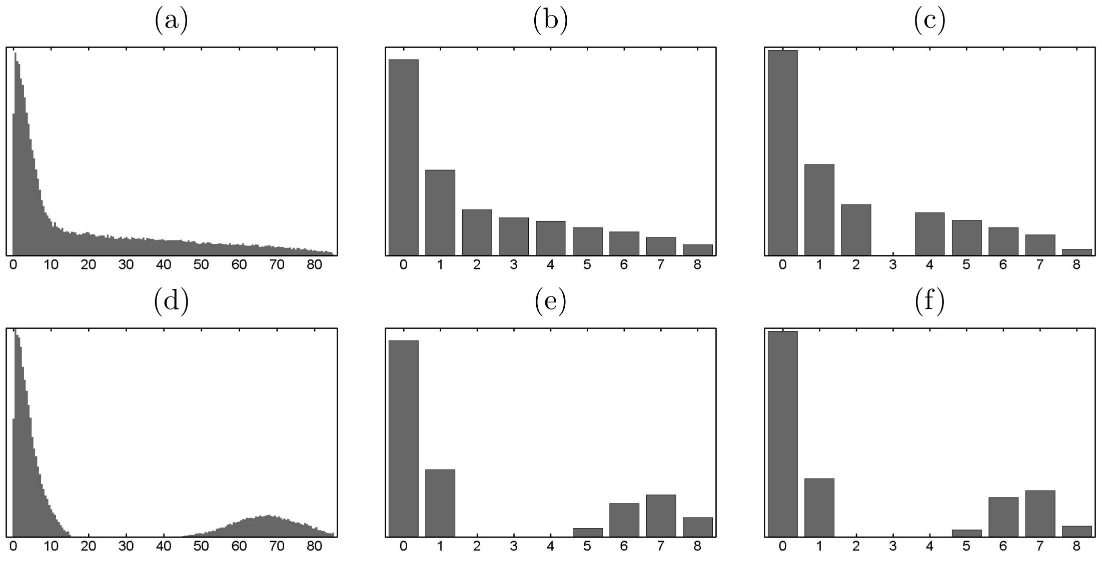


Figure 4.3: Shown in the first row are distributions for (a) an original set of coefficients, and these coefficients (b) singly quantized ($q_1 = 10$) and (c) doubly quantized ($q_1 = 12$ and $q_2 = 10$). Shown in panel (d) is a similar distribution, but where the original coefficients have no data in the range $[15, 45]$. This missing data leads to nearly identical singly (e) and doubly (f) quantized distributions (unlike panels (b) and (c).

in the range $[15, 45]$. As a result, the singly ($q_1 = 10$) and doubly ($q_1 = 12$ and $q_2 = 10$) quantized distributions are nearly identical because the expected empty bin occurs in the region where there is no data. Such a situation will yield a false positive – a macroblock will be classified as doubly compressed when it is not. Since we are considering the distribution of DCT coefficients on a per-macroblock basis, this situation is not uncommon in practice, particularly in largely uniform image regions. We next describe a scheme for avoiding such false positives.

The probability that a set of coefficients Z in a given macroblock have been quantized by quality q_1 prior to its current quantization, Equation (4.10), is scaled by a weighting factor $c(\cdot)$:

$$P_c(Z|q_1) \quad = \quad c(Z, q_1)P(Z|q_1), \tag{4.11}$$

where this weighting factor embodies our confidence that a specific macroblock contains sufficient data. Specifically, this confidence coefficient is given by:

$$c(Z, q_1) = 1 - e^{-\beta \sum_{z \in \Lambda} P(z)/s(z)}, \tag{4.12}$$

where β is a scalar which controls the exponential decay and Λ is an index set which depends on the quantization steps q_1 and q_2. The set Λ is determined by first quantizing synthetically generated data at all pairs of steps q_1 and q_2. For each pair of quantizations, the set Λ consists of all empty bins that result from double quantization, and their immediately adjacent bins. Intuitively, if these bins are all empty, then our confidence in determining double quantization is low (as in Figure 4.3(f)). On the other hand, if these bins are not empty, then our confidence is high (as in Figure 4.3(c)).

4.2.2 Results

We report on three sets of experiments that show the efficacy and limitations of the proposed technique for detecting double quantization. Throughout, we employed an MPEG-2 encoder/decoder developed by the MPEG Software Simulation Group[2]. The encoder affords two quantization modes, linear or non-linear. For simplicity, the linear mode was employed in which the quantization scale is specified as an integer between 1 and 31, and is fixed throughout the entire video sequence. Since we are only interested in the I-frames, the encoder was configured to encode every frame as an I-frame. In each experiment, a video sequence was either compressed (i.e., quantized) once (singly quantized) or twice with different quantization scale factors (doubly quantized).

As described previously, the detection of double quantization is performed on each 16×16 macroblock. As such, the 252 AC coefficients[3] were extracted from the luminance channel of each macroblock[4]. In addition, the quantization scale was extracted from the encoded video. In each experiment, the various parameters are defined as follows: $\sigma = 0.1$ in Equation (4.4); $\alpha = 150$ in Equation (4.10); $\beta = 15$ in Equation (4.12); and a macroblock is classified as doubly quantized when the estimated probability is greater than 0.5.

[2]`www.mpeg.org/MPEG/video/mssg-free-mpeg-software.html`

[3]63 AC coefficients per each of four 8×8 DCT blocks

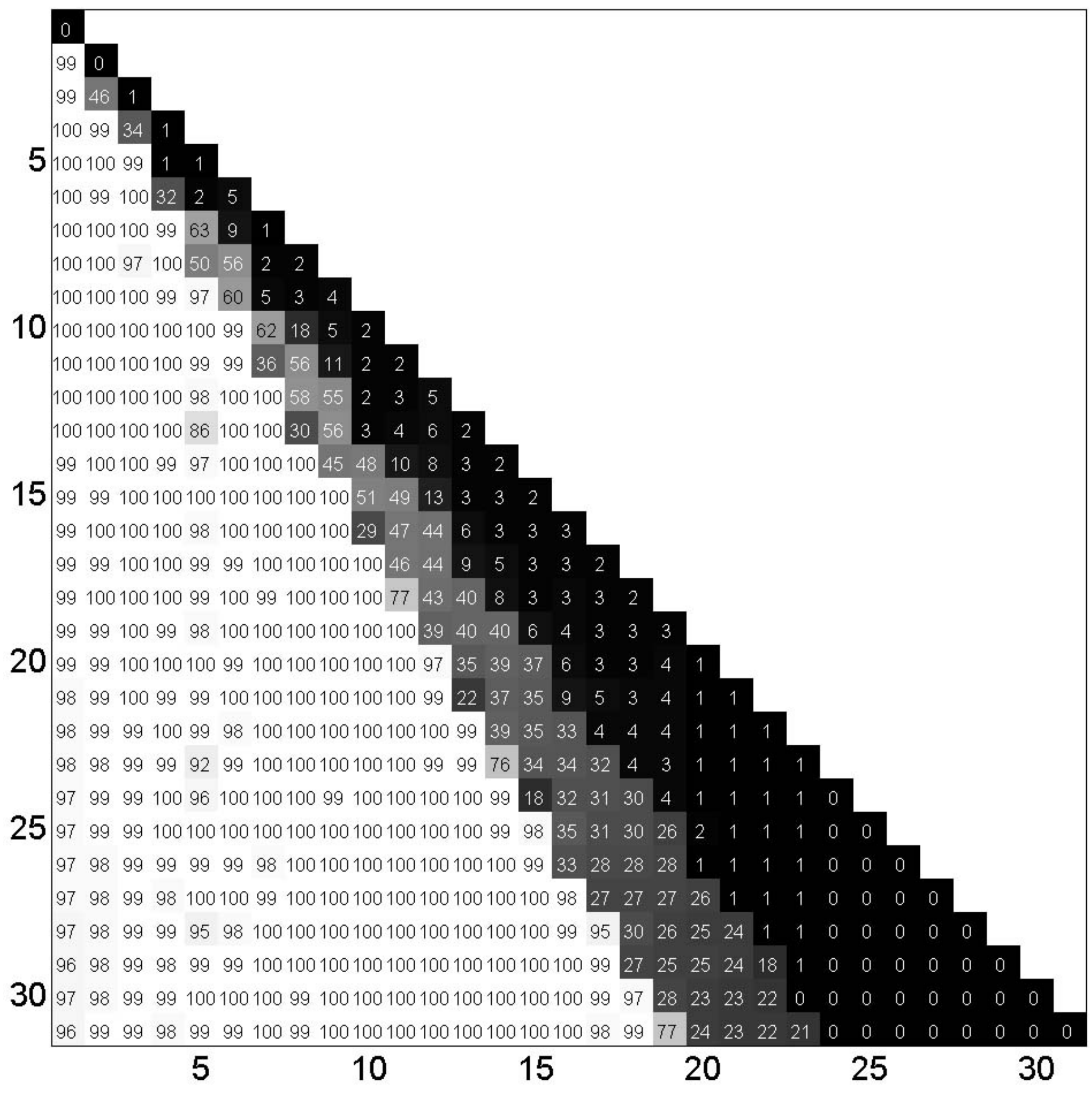
[4]We found little benefit from incorporating the remaining 126 AC coefficients from the chrominance channels.

In the first experiment, a video sequence of length $10,000$ frames was recorded with a SONY-HDR-HC3 digital video camera. The camera was hand-held as we walked aimlessly through the campus. The video was initially captured in DV format at a fixed bitrate of 25 Mbps (this rate was high enough so that its effect could be ignored). The size of each frame is 720×480 pixels. This video was first MPEG compressed with each of the 31 quantization scales. To simulate tampering, the resulting 31 MPEG sequences were then compressed again with each possible quantization scale less than the original scale. This yielded a total of 31 singly compressed (authentic) and 465 doubly compressed (tampered) videos. For each sequence, $135,000$ macroblocks were extracted from 100 frames sampled equally between the first and last frame of the recorded video.

Shown in Figure 4.4 is the percentage of macroblocks classified as doubly quantized, where the vertical and horizontal axes correspond to the first and the second quantization scales respectively. The diagonal entries in this figure correspond to the singly quantized sequences, and the off-diagonal entries correspond to the doubly quantized sequences. A perfect classification would have 0% on the diagonal and 100% on the off-diagonal. For the singly compressed sequences, the mean false positive rate is 1.4% with a standard deviation of 1.4%. That is, on average 1.4% of the $135,000$, or $1,890$, macroblocks in each sequence are mis-classified as doubly quantized. These misclassified macroblocks are typically scattered throughout a frame, and, as we will see below, can typically be removed with a spatial median filter. For the doubly compressed sequences, the detection rate depends on the ratio between the first and the second quantization scale. When the ratio is less than 1.3, the average detection rate is near chance at 2.5% with a standard deviation of 3.1%. When the ratio is between 1.3 and 1.7, the average detection rate is 41.2% with a standard deviation of 24.1%. When the ratio is greater than 1.7, the average detection rate is 99.4% with a standard deviation of 1.3%. The detection accuracy improves with an increasing quantization scale ratio because for these larger ratios the tell-tale empty bins are near the origin where the DCT coefficient values are concentrated.

In the second experiment, a video sequence of length 200 frames and size 1440×1080 pixels was downloaded from Microsoft's WMV HD Content Showcase[5]. In order to remove

[5]`www.microsoft.com/windows/windowsmedia/musicandvideo/hdvideo/contentshowcase.aspx`



					5					10					15					20					25					30	
	0																														
	99	0																													
	99	46	1																												
	100	99	34	1																											
5	100	100	99	1	1																										
	100	99	100	32	2	5																									
	100	100	100	99	63	9	1																								
	100	100	97	100	50	56	2	2																							
	100	100	100	99	97	60	5	3	4																						
10	100	100	100	100	100	99	62	18	5	2																					
	100	100	100	100	99	99	36	56	11	2	2																				
	100	100	100	100	98	100	100	58	55	2	3	5																			
	100	100	100	100	86	100	100	30	56	3	4	6	2																		
	99	100	100	99	97	100	100	100	45	48	10	8	3	2																	
15	99	99	100	100	100	100	100	100	100	51	49	13	3	3	2																
	99	100	100	100	98	100	100	100	100	29	47	44	6	3	3	3															
	99	99	100	100	99	99	100	100	100	100	46	44	9	5	3	3	2														
	99	100	100	100	99	100	99	100	100	100	77	43	40	8	3	3	3	2													
	99	99	100	99	98	100	100	100	100	100	100	39	40	40	6	4	3	3	3												
20	99	99	100	100	100	99	100	100	100	100	100	97	35	39	37	6	3	3	4	1											
	98	99	100	99	99	100	100	100	100	100	100	99	22	37	35	9	5	3	4	1	1										
	98	99	99	100	99	98	100	100	100	100	100	100	99	39	35	33	4	4	4	1	1	1									
	98	98	99	99	92	99	100	100	100	100	100	99	99	76	34	34	32	4	3	1	1	1	1								
	97	99	99	100	96	100	100	100	99	100	100	100	100	99	18	32	31	30	4	1	1	1	1	0							
25	97	99	99	100	100	100	100	100	100	100	100	100	100	99	98	35	31	30	26	2	1	1	1	0	0						
	97	98	99	99	99	99	98	100	100	100	100	100	100	100	99	33	28	28	28	1	1	1	1	0	0	0					
	97	98	99	98	100	100	99	100	100	100	100	100	100	100	100	98	27	27	27	26	1	1	1	0	0	0	0				
	97	98	99	99	95	98	100	100	100	100	100	100	100	100	100	99	95	30	26	25	24	1	1	0	0	0	0	0			
	96	98	99	98	99	99	100	100	100	100	100	100	100	100	100	100	99	27	25	25	24	18	1	0	0	0	0	0	0		
30	97	98	99	99	100	100	100	99	100	100	100	100	100	100	100	100	99	97	28	23	23	22	0	0	0	0	0	0	0	0	
	96	99	99	98	99	99	100	99	100	100	100	100	100	100	100	100	98	99	77	24	23	22	21	0	0	0	0	0	0	0	0

Figure 4.4: Shown is the percentage of macroblocks classified as doubly quantized for singly compressed (diagonal) and doubly compressed (off-diagonal) video sequences. Each entry is also color-coded: 0%=black, 50%=mid-level gray, and 100%=white. The vertical and horizontal axes correspond to the first and the second quantization scales respectively.

any existing compression artifacts, each frame was downsampled by a factor of two and centrally cropped to a size of 720 × 480 pixels. These frames were then singly compressed with a quantization scale of 2, and doubly compressed with a quantization scale of 4 followed by 2. For each frame, a total of 1, 350 macroblocks were extracted and classified as either singly or doubly quantized.

Shown in the first column of Figure 4.5 are five representative frames from this sequence. Shown in the second column are the estimated probabilities for the singly compressed sequence. Each pixel in this probability image corresponds to one macroblock in the original frame. Although a small number of macroblocks have a probability greater than our threshold of 0.5, these macroblocks are scattered throughout the frame and can be removed with a spatial median filter. The macroblocks marked with a cross (red) correspond to those macroblocks for which a probability could not be computed because their AC coefficients were uniformly all zero. Such macroblocks correspond to areas in the image with effectively no intensity variation. Shown in the third column of Figure 4.5 are the estimated probabilities for the doubly compressed sequence. Except for a few scattered macroblocks, nearly all of the macroblocks have a probability close to 1, indicating that these macroblocks were doubly quantized.

In the third experiment, we created a video composite by combining a video of a static background and video of a person recorded in front of a green screen[6], each of length 200 frames. In order to remove any existing compression artifacts in the original videos, the background video, originally of size 1600×1200, was downsampled by a factor of two and centrally cropped to a size of 720×480 pixels. The foreground video was originally of size 1440×1080, and was also downsampled and centrally cropped to a size of 720×480 pixels. The final video composite was created using Adobe Premiere Pro 2.0, and encoded using Premiere's MPEG-2 encoder. The encoder was configured to save each frame as an I-frame. Because the encoder does not allow for direct control of the quantization scale, the compression quality was controlled by adjusting the average encoded bit-rate (which in turn spatially and temporally adjusted the quantization scale to achieve the desired bit-rate). The background video was compressed with a bit-rate of 6 Mbps. The foreground video was composited with the background, and the resulting composition was compressed with a bit-rate of 12 Mbps.

Shown in the first column of Figure 4.6 are five representative frames from this composited sequence. Macroblocks from each frame were extracted from the composited video and classified as either singly or doubly quantized. Shown in the second column of Fig-

[6] `www.timelinegfx.com/freegreenscreen.html`

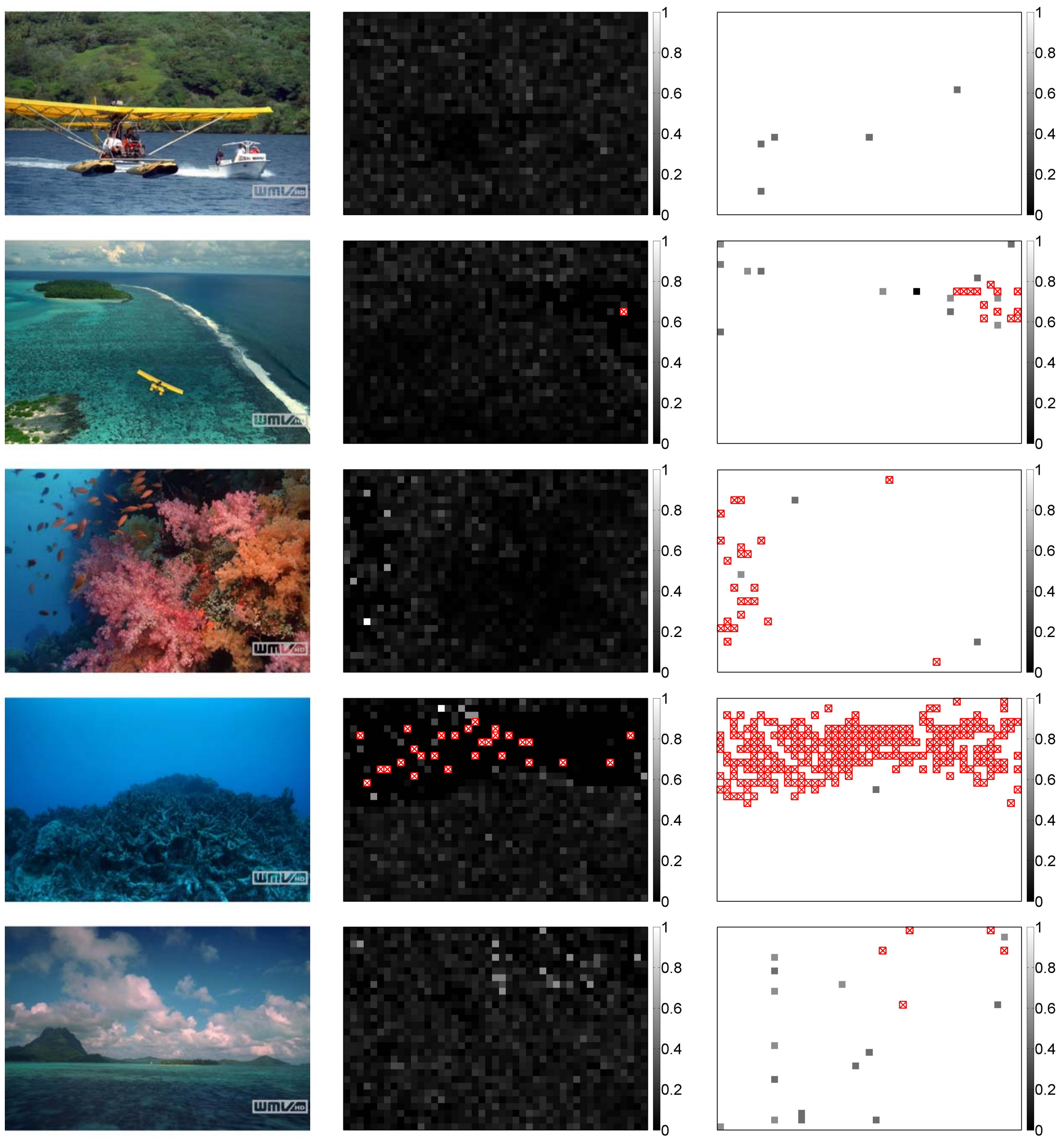


Figure 4.5: Shown in the first column are representative frames from a video sequence. Shown in the second and the third columns are the probability of each macroblock being doubly quantized for a singly and doubly compressed video, respectively. The macroblocks marked with a cross (red) correspond to those macroblocks for which a probability could not be computed due to a lack of sufficient AC coefficients.

ure 4.6 are representative examples of the estimated probabilities. Note that as desired, the background region generally has a high probability since it was compressed twice, while the foreground, compressed only once, has a low probability. Shown in the third column of Figure 4.6, are the results of applying a spatial median filter of size 3×3 macroblocks, and thresholding the filtered probabilities at 0.5.

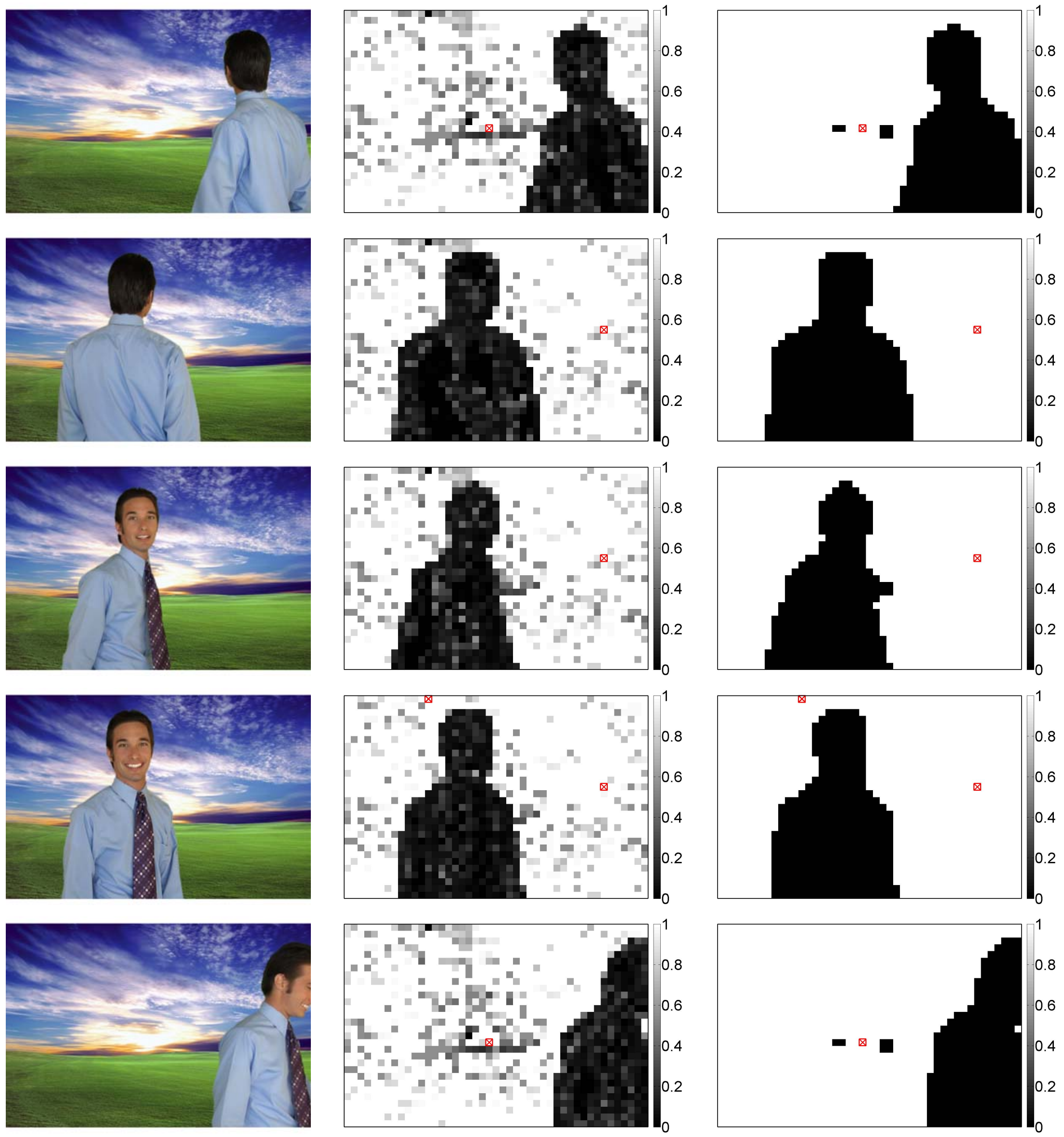


Figure 4.6: Shown in the first column are representative frames from a video sequence created by compositing the person onto a static background. The background was compressed twice, while the foreground was compressed once. Shown in the second column are the probabilities that each macroblock was doubly quantized. Shown in the third column are the results of applying a spatial median filter and threshold.

In summary, our experiments show that we can detect double quantization at the macroblock level (16×16 pixels). When the ratio between the first and the second quantization scale is greater than 1.7, the detection is highly effective. The detection accuracy decreases along with this ratio. At the same time, the number of false positives is generally small and spatially localized making them fairly easy to remove with a spatial median filter.

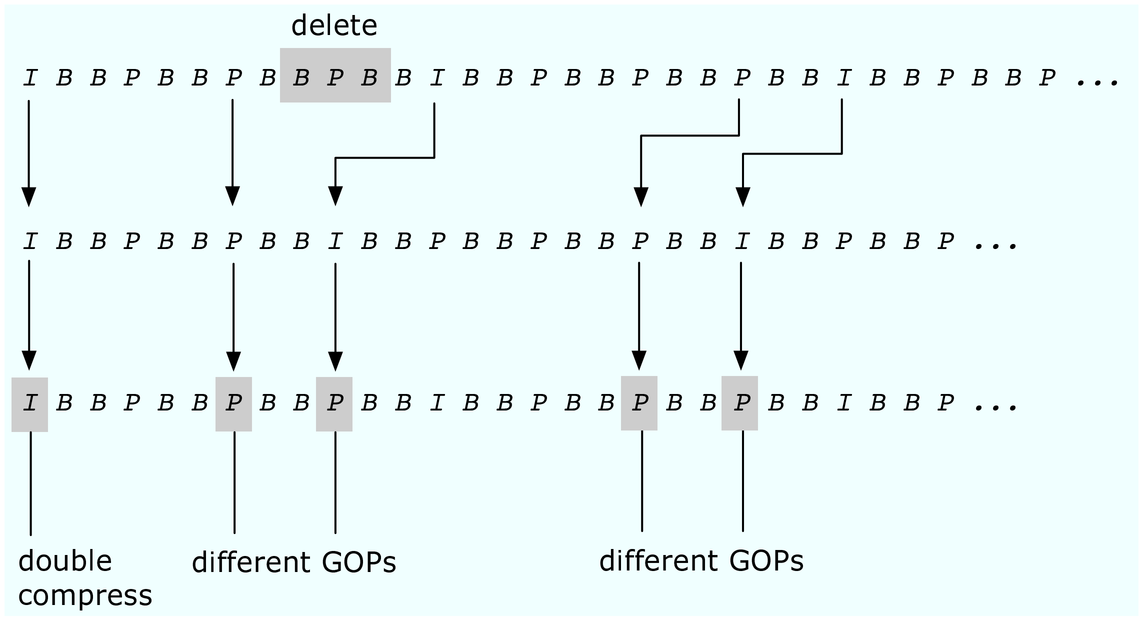

delete
I B B P B B P B B P B B I B B P B B P B B P B B I B B P B B P ...
I B B P B B P B B I B B P B B P B B P B B I B B P B B P ...
I B B P B B P B B P B B I B B P B B P B B P B B I B B P ...
double compress
different GOPs
different GOPs

Figure 4.7: Shown along the top row is an original MPEG encoded sequence. The subsequent rows show the effect of deleting the three frames in the shaded region. Shown in the second row are the re-ordered frames, and in the third row, the re-encoded frames. The I-frame prior to the deletion is subjected to double compression. Some of the frames following the deletion move from one GOP sequence to another. This double MPEG compression gives rise to specific static and temporal statistical patterns that may be used as evidence of tampering.

4.3 Temporal

In addition to spatial artifacts, double compression may also introduce temporal statistical perturbations, whose presence can be used as evidence of tampering. Shown in the top row of Figure 4.7 is a short 31-frame MPEG sequence. Consider the effect of deleting the three frames shown in the shaded region. Shown in the second row are the re-ordered frames, and in the third row are the re-encoded frames after re-saving the spliced video as a MPEG video.

Note that the I-frame prior to the deletion retains its identity and will be re-encoded as an I-frame. In Section 4.2, we showed that such doubly compressed I-frames may introduce artifacts in the DCT coefficients and showed how these artifacts can be quantified and estimated to detect tampering. Note also that the second and third P-frame of the first and second GOP were, in the original sequence, in different GOP sequences. In this section, we will show how this change yields a specific statistical pattern in the distribution of motion errors.

4.3.1 Methods

Recall that the first frame of each group of pictures (GOP) is an I-frame. This frame, which is only statically compressed, effectively corrects for motion estimation errors that accumulate throughout each GOP. Each P-frame within a GOP is, either directly or indirectly encoded with respect to the initial I-frame.

We consider the effect of deleting (or adding) frames from a video sequence, and re-encoding the resulting sequence. As an example, consider the effect of deleting the first six frames of the following sequence:

$$I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B\ I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B$$

The deletion of the first six frames leaves:

$$P\ B\ B\ P\ B\ B\ I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B$$

which, when re-encoded, becomes:

$$I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B\ I\ B\ B\ P\ B$$

Within the first GOP of this sequence, the I-frame and first P-frame are from the first GOP of the original sequence. The second and third P-frames, however, are the I-frame and first P-frame from the second GOP of the original sequence. When this new sequence is re-encoded, we expect a larger motion error between the first and second P-frames, since they originated from different GOPs. Moreover, this increased motion error will be periodic, occurring throughout each of the GOPs following the frame deletion.

This artifact is not unique to a deletion of six frames. Consider, for example, the effect of deleting four frames from the following sequence:

$$I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B\ I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B$$

The deletion of the first four frames leaves:

$$B\ B\ P\ B\ B\ P\ B\ B\ I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B$$

which, when re-encoded, becomes:

$$I\ B\ B\ P\ B\ B\ P\ B\ B\ P\ B\ B\ I\ B\ B\ P\ B\ B\ P\ B$$

Within the first GOP of this sequence, the I-frame and first two P-frames are from the first GOP of the original sequence. The third P-frame, however, originated from the second GOP in the original sequence. As in the above example, we expect a periodic increase in motion error because of this re-location of frames from GOPs.

The reason for this change in motion error is that all of the P-frames within a single GOP are correlated to the initial I-frame. This correlation emerges, in part, because each I-frame is independently JPEG compressed. Because of the motion compensation encoding, these compression artifacts propagate through the P-frames. As a result, each P-frame is correlated to its neighboring P- or I-frame. When frames move from one GOP to another, this correlation is weaker, and hence the motion error increases. To see this more formally consider a simplified 5-frame sequence $F_1\ F_2\ F_3\ F_4\ F_5$ that is encoded as $I_1\ P_2\ P_3\ P_4\ I_5$, where the subscripts denote time. Due to JPEG compression of the I-frame and JPEG compression of the motion error for the P-frames, each of the MPEG frames can be modeled as: $I_1 = F_1 + N_1$, $P_2 = F_2 + N_2$, $P_3 = F_3 + N_3$, $P_4 = F_4 + N_4$, $I_5 = F_5 + N_5$, where N_i is additive noise. Note that, as described above, the noise for I_1 through P_4 will be correlated to each other, but not to that of I_5. The motion error, m_2, for frame P_2 will be:

$$\begin{aligned} m_2 &= P_2 - M(I_1) \\ &= F_2 + N_2 - M(F_1 + N_1) \\ &= F_2 + N_2 - M(F_1) - M(N_1) \\ &= F_2 - M(F_1) + (N_2 - M(N_1)), \end{aligned} \tag{4.13}$$

where $M(\cdot)$ denotes motion compensation. Similarly, the motion errors for frame P_i is

$F_i - M(F_{i-1}) + (N_i - M(N_{i-1}))$. Consider now the deletion of frames that brings frame P_4 into the third position and I_5 into the fourth position. The motion error for the newly encoded P_4 frame will be:

$$\begin{aligned}\hat{m}_4 &= I_5 - M(P_4) \\ &= F_5 + N_5 - M(F_4 + N_4) \\ &= F_5 + N_5 - M(F_4) - M(N_4) \\ &= F_5 - M(F_4) + (N_5 - M(N_4)).\end{aligned} \tag{4.14}$$

For the motion error m_2, the two components of the additive noise term, $(N_2 - M(N_1))$, are correlated and we therefore expect some cancellation of the noise. In contrast, for the motion error $\hat{m}_4$, the two components of the additive noise term, $(N_5 - M(N_4))$, are not correlated leading to a relatively larger motion error as compared to m_2.

This pattern of motion error is relatively easy to detect as the motion error is explicitly encoded as part of the MPEG sequence. Specifically, we extract from the MPEG video stream the motion error and compute, for each P-frame [7], the mean motion error for the entire frame. Periodic spikes in motion error indicate tampering. This periodicity is often fairly obvious, but can also be detected by considering the magnitude of the Fourier transform of the motion errors over time. In the Fourier domain, the periodicity manifests itself with a spike at a particular frequency, depending on the GOP encoding. As will be shown in the following section, this periodic increase in motion error occurs for every number of frame deletions (or insertions) that are not a multiple of the GOP length (12, in these examples).

4.3.2 Results

Shown in the upper portion of Figure 4.8 are ten frames from a 500-frame long video sequence. This video was shot with a Canon Elura digital video camera, and converted from its original AVI format to MPEG with a $IBBPBBPBBPBB$ GOP. We employed a MPEG-1 encoder/decoder written by David Foti – these MatLab routines are based on an

[7]The motion errors of B-frames are not considered here since the bi-directional nature of motion estimation for these frames makes it likely that a B-frame will be correlated to frames in neighboring GOPs.

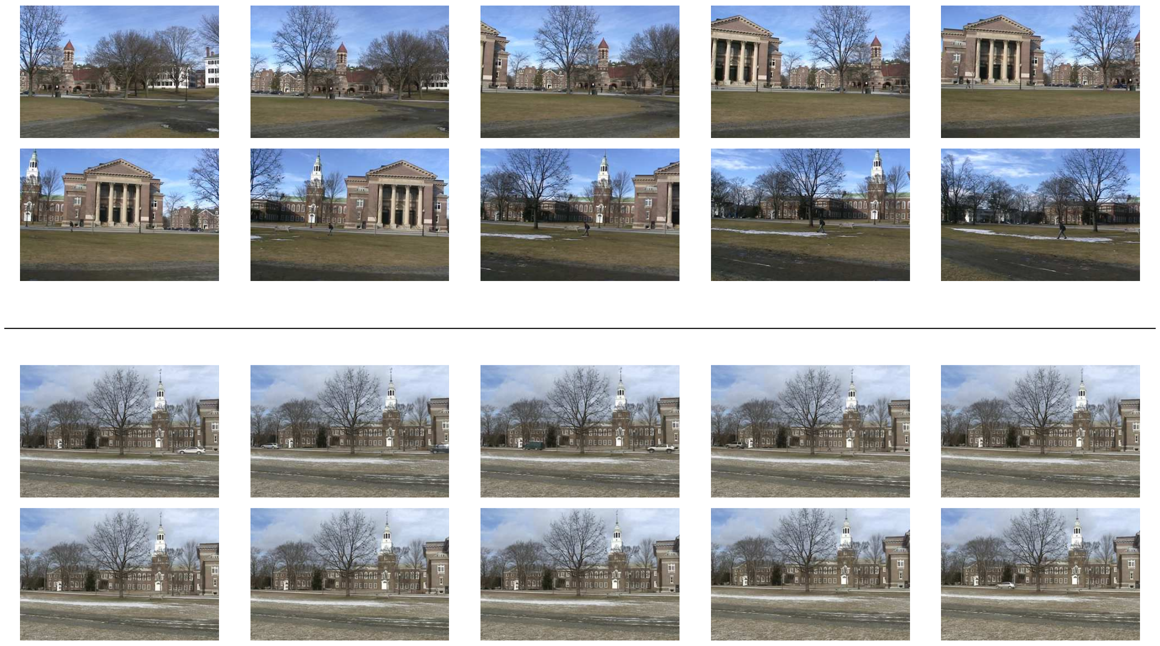


Figure 4.8: Representative frames of two video sequences. Shown are frames 0 to 450 in steps of 50.

encoder/decoder developed at The University of California at Berkeley [37]. The MPEG encoder allows for control over the static compression quality of I-, P- and B-frames and the GOP sequence. These routines were adapted to extract the DCT coefficients and the motion errors.

Shown in the lower part of Figure 4.8 are ten frames from a second 500-frame long video sequence acquired in a similar manner. In the first video sequence, the camera pans across the scene, yielding an overall large global motion. In the second video sequence, the camera is stationary, with relatively small motions caused by passing cars.

A variable number of frames, between 0 and 11, were deleted from the video sequence shown in the upper part of Figure 4.8. The resulting sequence was then re-saved as an MPEG video. The motion error for each P-frame was extracted from the MPEG encoding. Shown in Figure 4.9 is the mean motion error for each P-frame as a function of time (upper panel), and the magnitude of the Fourier transform of this motion error (lower panel). Note that for all non-zero frame deletions, the motion error exhibits a periodic pattern, which manifests itself as peaks in the middle frequency range. Note that the artifacts for frame deletions of 3, 6 and 9 are significantly stronger than others. The reason for this is that for

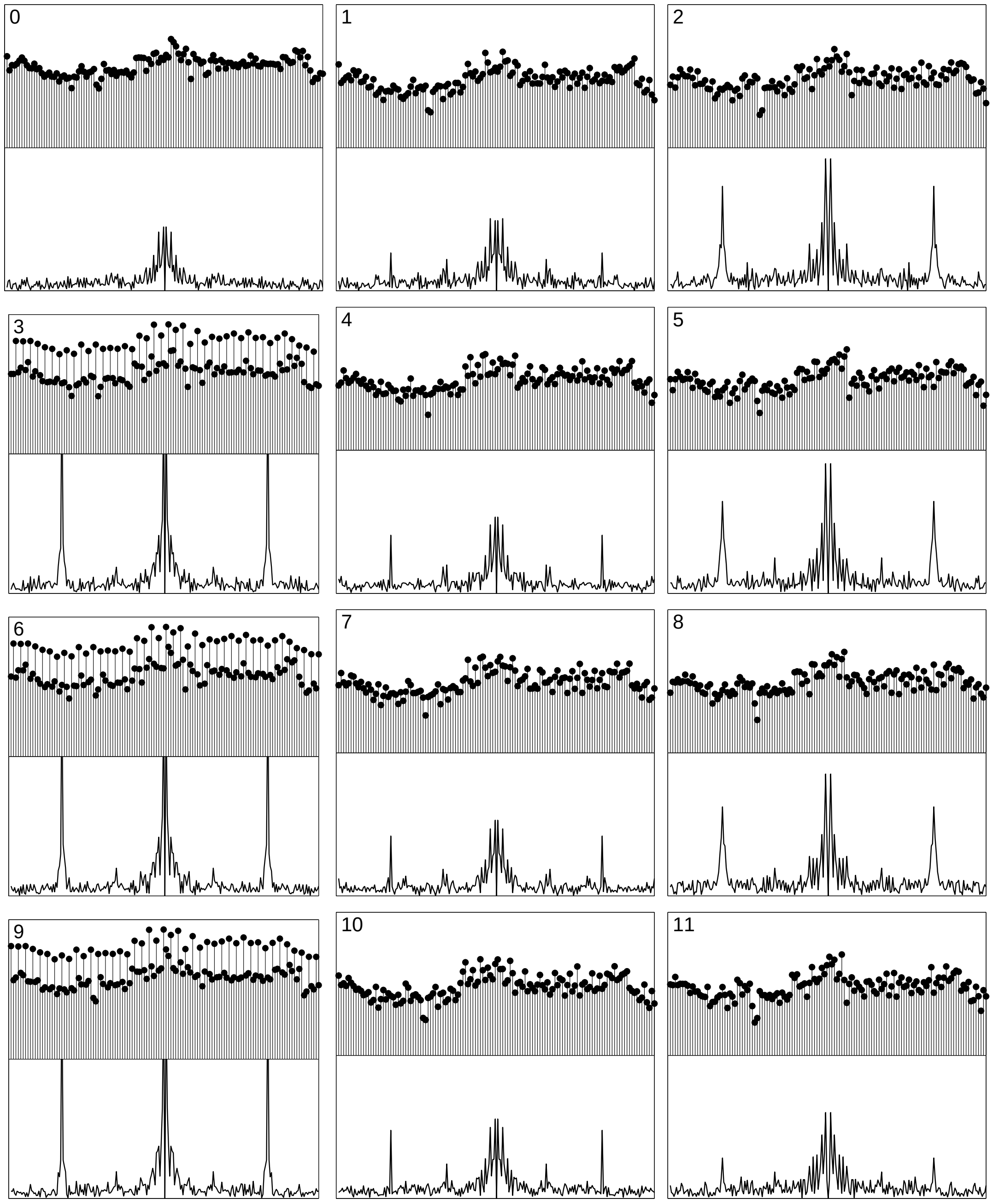


Figure 4.9: Double MPEG detection for the video sequence in the upper portion of Figure 4.8. Shown in each box is the mean motion error over time (upper panel) and the magnitude of its Fourier transform (lower panel). Shown are the results for a variable number of deleted frames, from 0 to 11. Spikes in the Fourier transform indicate double compression.

deletions other than integer multiples of 3, the last two or first two B-frames of a GOP shift into a P-frame. Because of the bi-directional nature of their motion estimation, the noise in these B-frames are correlated to the frames of the GOP in which they are contained, and to the frames in the subsequent GOP. In contrast, for deletions that are a multiple of 3, a

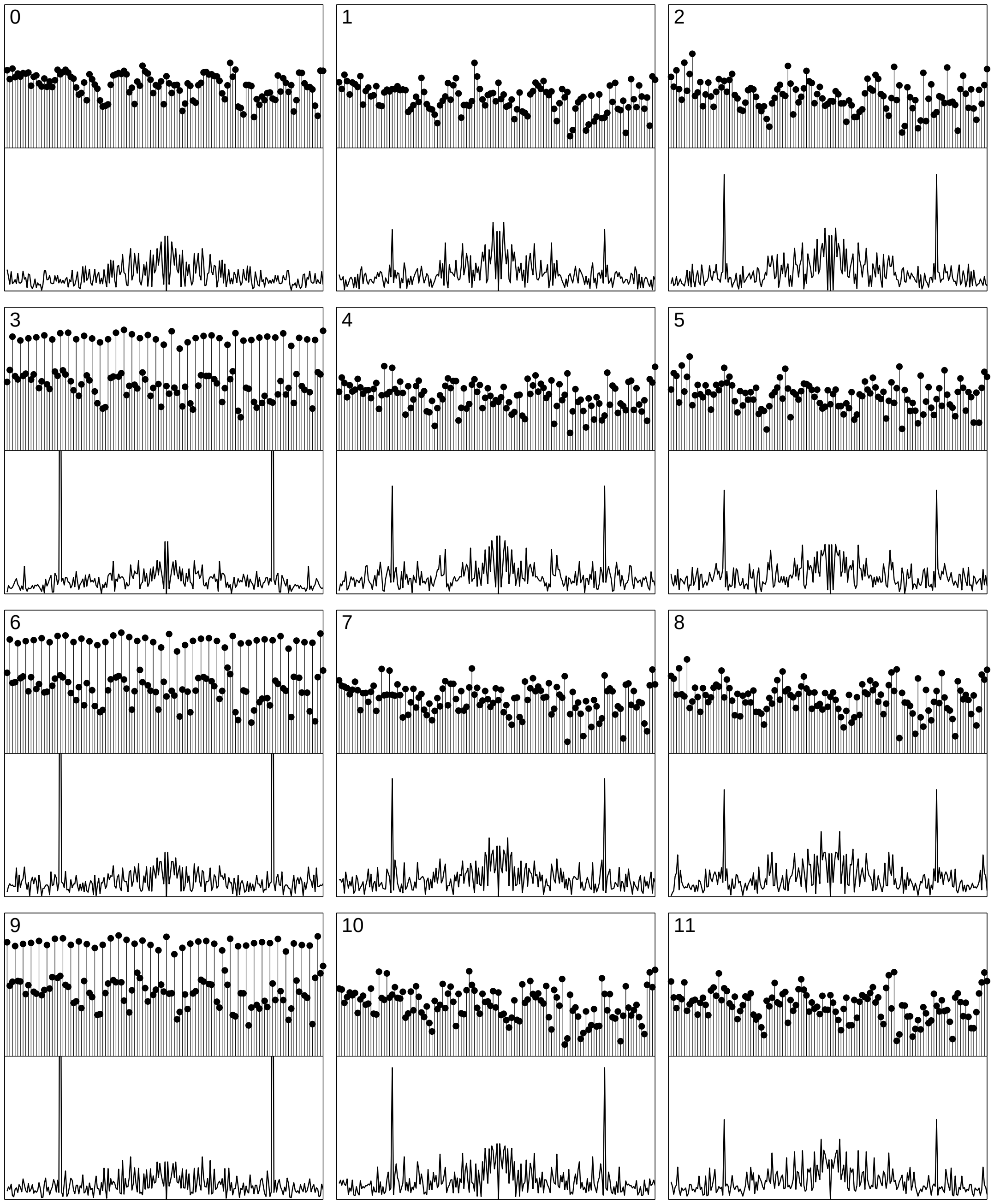


Figure 4.10: Double MPEG detection for the video sequence in the lower portion of Figure 4.8. Shown in each box is the mean motion error over time (upper panel) and the magnitude of its Fourier transform (lower panel). Shown are the results for a variable number of deleted frames, from 0 to 11. Spikes in the Fourier transform indicate double compression.

P- or I-frame from one GOP moves to a P- or I-frame of another GOP. The noise in these frames, unlike the B-frames, are correlated only to the frames in their GOP, Section 4.3.1.

Results for the video sequence shown in the lower part of Figure 4.8 are shown in Figure 4.10. As above, shown is the mean motion error for each P-frame as a function of time (upper panel), and the magnitude of the Fourier transform of this motion error (lower panel). Note that, as in the previous example, the motion error exhibits a periodic pattern for all non-zero frame deletions (or insertions).

4.4 Discussion

We have described two techniques for detecting tampering in MPEG video sequences. Both of these techniques exploit the fact that spatial and temporal artifacts are introduced when a video sequence is subjected to double MPEG compression. Spatially, the I-frames of an MPEG sequence are subjected to double quantization. Temporally, frames that move from one GOP to another, as a result of frame deletion or insertion, give rise to relatively larger motion estimation errors. Since quantization and motion compensation are used in many video compression standards, it may be possible to adapt these two techniques to detect forgeries in other compression formats (e.g. MPEG-4). The spatial technique can detect localized tampering in regions as small as 16×16 pixels. This feature is particularly attractive for detecting the fairly common digital effect of green-screening. The limitation of the spatial technique is that it is only effective when the second compression quality is higher than the first compression quality. The temporal technique is useful to detect tampering that involves frame insertion or deletion. When the camera is stationary and the background is largely static, it is relatively easy to maintain continuity in the doctored sequence. Since only the sequence of the frames is rearranged but individual frames are not modified, such manipulation can be very hard to detect with other techniques. Ideally, we would like to extend this technique to detect spatially localized tampering by averaging motion errors over small blocks instead of entire frames. However, since motion errors are very noisy (partly due to the motion itself) and are not stable enough to be analyzed at a block level, it is not practically feasible to adapt this technique to detect spatially localized tampering. The weakness of the temporal technique is that it cannot detect tampering when the number of inserted or deleted frames is a multiple of the GOP length. We have shown the efficacy of these two techniques on actual video sequences. In both cases, the statistical artifacts are significant, making the detection of tampering in doubly-compressed

MPEG video likely.

A counterattack for the spatial technique is to apply a filter to eliminate the compression artifacts before the final compression. This operation is likely to destroy the spatial artifacts introduced by double MPEG compression. A counterattack for the temporal technique would be quite difficult as it would require the forger to estimate the motion errors and modify them to destroy the temporal artifacts introduced by double MPEG compression.

Chapter 5

Duplication

In the popular movie *Speed*, the characters created a doctored video by simply duplicating a short sequence of frames and substituting it for live footage (Figure 5.1). In so doing, they were able to effectively deceive the person monitoring the surveillance video. This common and relatively simple manipulation can be used to remove people, objects or an entire event from a video sequence. When done carefully, this form of tampering can be very difficult to detect.

Techniques for detecting image duplication have been proposed before [12, 42]. 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 of these techniques on several real video sequences.

5.1 Methods

5.1.1 Frame Duplication

Shown in Figure 5.2 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

Figure 5.1: By duplicating frames on a surveillance video, the characters in the movie *Speed* create a doctored video to conceal activity on the bus.

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 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.

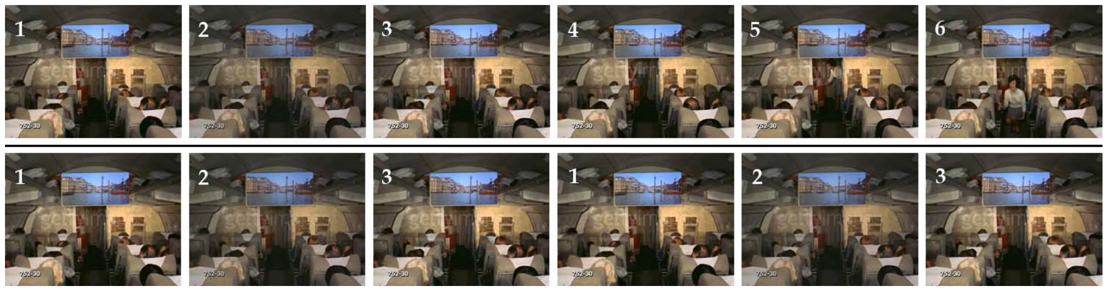


Figure 5.2: Shown in the left column are seven frames of an original video. Shown on the right are the results of duplicating three frames so as to remove the flight attendant from the scene.

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:

$$C(\vec{u},\vec{v}) = \frac{\sum_i (u_i - \mu_u)(v_i - \mu_v)}{\sqrt{\sum_i (u_i - \mu_u)^2}\sqrt{\sum_i (v_i - \mu_v)^2}}, \tag{5.1}$$

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:

$$S_\tau(t) = \{f(x,y,t+\tau) \mid t \in [0, n-1]\}. \tag{5.2}$$

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 5.3 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

```
FRAMEDUP($f(x,y,t)$)
 1  ▷ $f(x,y,t)$: video sequence of length $N$
 2
 3  ▷ $n$: sub-sequence length
 4  ▷ $\gamma_m$: minimum temporal correlation threshold
 5  ▷ $\gamma_t$: temporal correlation threshold
 6  ▷ $\gamma_s$: spatial correlation threshold
 7
 8  for $\tau = 1 : N-(n-1)$
 9      do $S_\tau = \{f(x,y,t+\tau) \mid t \in [0, n-1]\}$
10         build $T_\tau$ ▷ temporal correlation
11
12  for $\tau_1 = 1 : N-(2n-1)$
13      do for $\tau_2 = \tau_1 + n : N-(n-1)$
14             do if $(\min(T_{\tau_1}) > \gamma_m$ & $C(T_{\tau_1}, T_{\tau_2}) > \gamma_t)$
15                  build $B_{\tau_1,k}$ ▷ spatial correlation
16                  build $B_{\tau_2,k}$ ▷ spatial correlation
17                  if ( $C(B_{\tau_1,k}, B_{\tau_2,k}) > \gamma_s$ ) ▷ $\forall k$
18                       do ▷ Frame Duplication at $\tau_1$
```

Figure 5.3: Pseudo-code for detecting frame duplication.

(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.

5.1.2 Region Duplication

In the previous section we showed how to detect duplicated frames in a video sequence. Shown in Figure 5.4 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.

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 [28, 4] affords a robust and computationally efficient mechanism

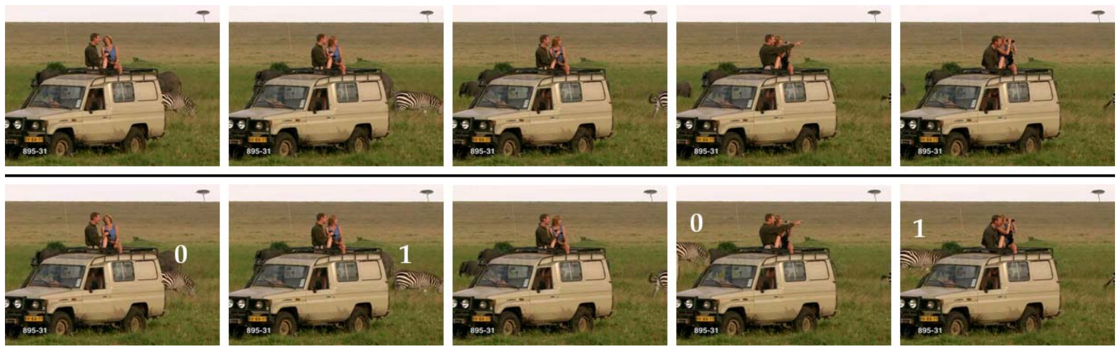


Figure 5.4: Shown in the left column are seven frames of an original video. Shown on the right are the results of region duplication so as to add another zebra into the scene.

for this estimation. We begin by defining the normalized cross power spectrum:

$$P(\omega_x, \omega_y) = \frac{F(\omega_x, \omega_y, \tau_1) F^*(\omega_x, \omega_y, \tau_2)}{\| F(\omega_x, \omega_y, \tau_1) F^*(\omega_x, \omega_y, \tau_2) \|}, \tag{5.3}$$

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 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 (5.1). 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 [14]. 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 5.6 for a detailed algorithmic description.

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

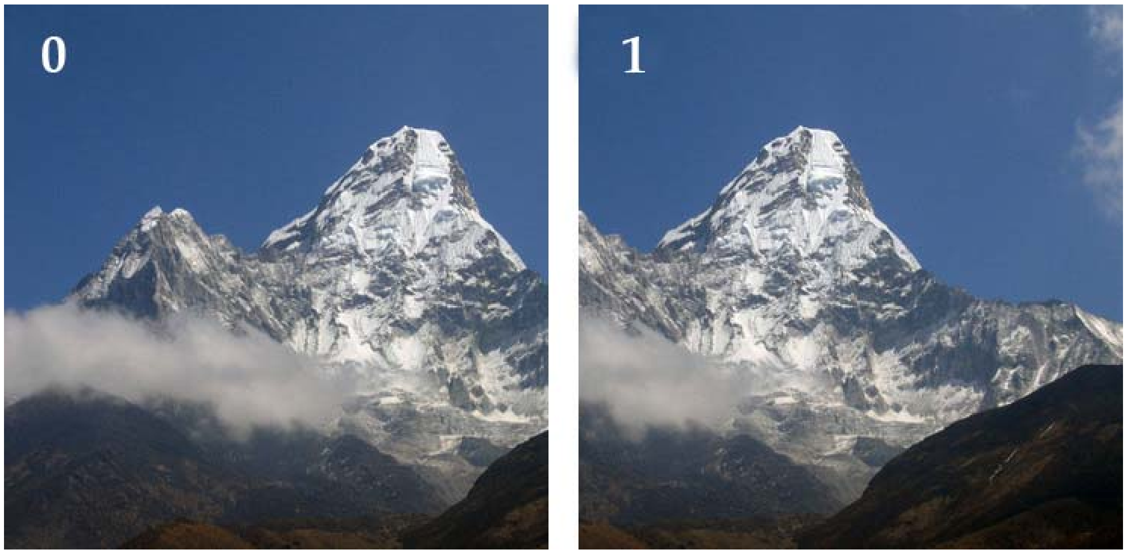


Figure 5.5: Two frames of a a mountain peak as the camera pans from left to right.

duplication from a moving camera where objects simply appear in different spatial locations due to camera motion. Shown in Figure 5.5 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 5.6 for a detailed algorithmic description.

5.2 Results

Shown in Figure 5.7 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.

5.2.1 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

```
REGIONDUP1($f(x,y,\tau_1), f(x,y,\tau_2)$)
$\triangleright$ $f(x,y,\tau_1), f(x,y,\tau_2)$: two video frames

$\triangleright$ $\gamma_p$: phase correlation threshold
$\triangleright$ $\gamma_o$: phase correlation offset threshold
$\triangleright$ $b$: block neighborhood size
$\triangleright$ $\gamma_b$: block correlation threshold
$\triangleright$ $\gamma_a$: minimum area threshold

compute $p(x,y)$ $\triangleright$ phase correlation
for each $(\Delta_x, \Delta_y)$, s.t. $p(\Delta_x, \Delta_y) > \gamma_p$ and
                        $(|\Delta_x| > \gamma_o$ or $|\Delta_y| > \gamma_o)$
    do for each $i, j$
            do $i' = i + \Delta_x$
               $j' = j + \Delta_y$
               $b_1 = f(i : i + b - 1, j : j + b - 1, \tau_1)$
               $b_2 = f(i' : i' + b - 1, j' : j' + b - 1, \tau_2)$
               if ( $C(b_1, b_2) > \gamma_b$ ) $\triangleright$ correlation
                   do mask$(i,j) = 1$
        mask = connected_components(mask)
        if ( area( mask$(x,y)$ ) $> \gamma_a$ )
            do $\triangleright$ Region Duplication at $(x,y)$
```

```
REGIONDUP2($f(x,y,t), \tau_1, \tau_2$)
$\triangleright$ $f(x,y,t)$: video sequence
$\triangleright$ $\tau_1, \tau_2$: two frame indices
$\triangleright$ frame size: $N_x \times N_y$ pixels

$\triangleright$ $s$: motion camera offset threshold

for $\tau = \tau_1 : \tau_2 - 1$
    do compute $p(x,y)$ for $f(x,y,\tau)$ and $f(x,y,\tau+1)$
       $(\delta_x, \delta_y) = \arg\max_{x,y}(p(x,y))$
       $(\Delta_x, \Delta_y) = (\Delta_x, \Delta_y) + (\delta_x, \delta_y)$
if ( $\Delta_x > sN_x$ and $\Delta_y > sN_y$ )
    do REGIONDUP1( $f(x,y,\tau_1), f(x,y,\tau_2)$ )
```

Figure 5.6: Pseudo-code for detecting region duplication from a stationary (REGIONDUP1) and moving camera (REGIONDUP2).

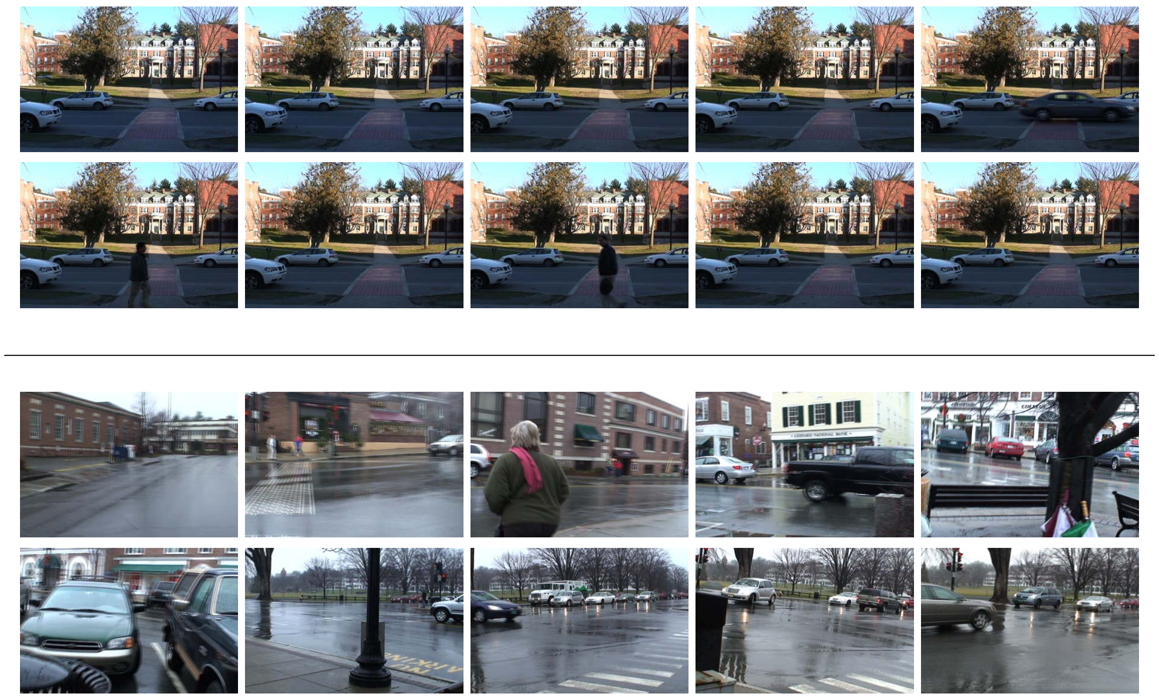


Figure 5.7: Sample frames from two video sequences captured from a stationary camera (top) and hand-held camera (bottom).

frames of a sub-sequence 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 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.

5.2.2 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.

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 5.7).

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

Shown in Figure 5.8 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.

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.

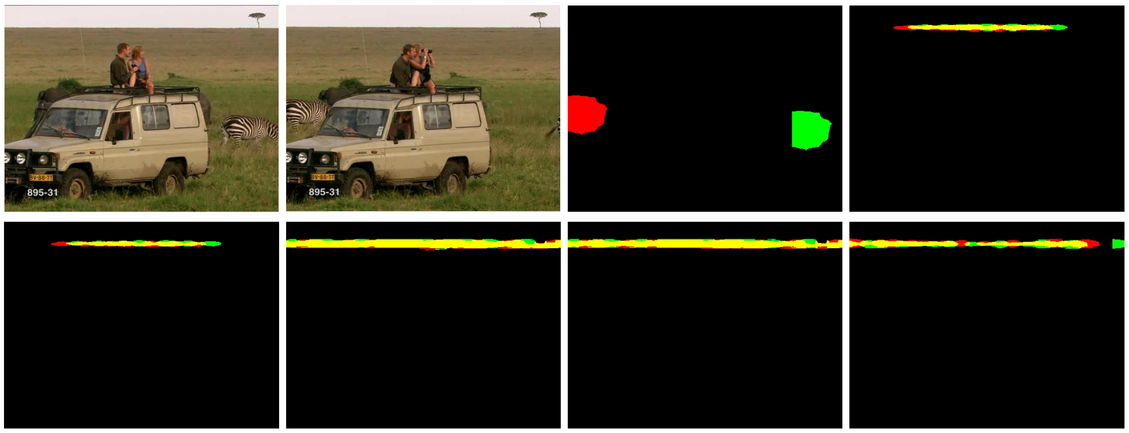


Figure 5.8: Shown in the first panel is one frame of a 800 frame video taken from a stationary camera. Shown in the second panel is the result of region duplication where the zebra from the first panel was copied into this frame. Shown in the subsequent panels are the detection results: in the third panel is the duplication corresponding to the zebra, and in the subsequent panels are false positives corresponding to the largely uniform sky.

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, 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.

5.2.3 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 [42].

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 5.10, and 5.9 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

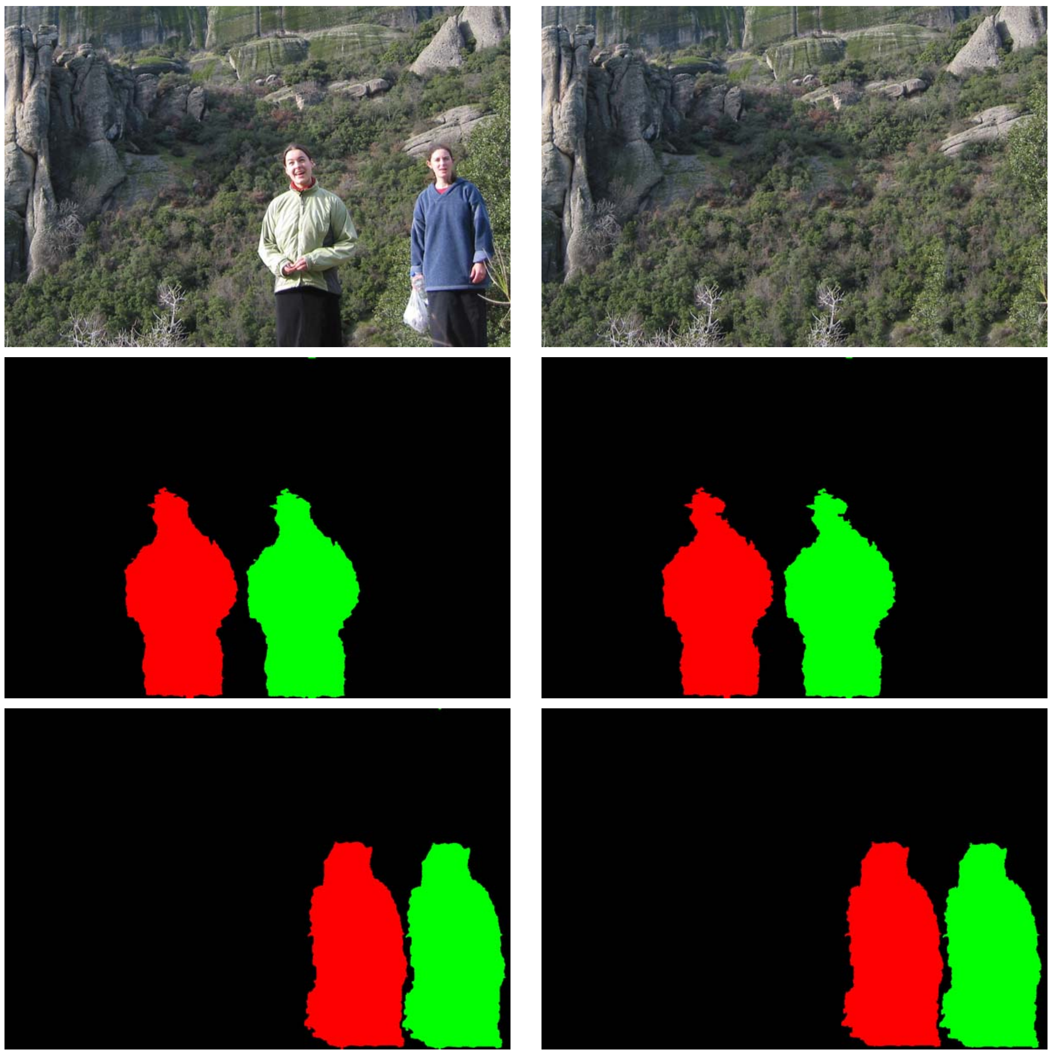
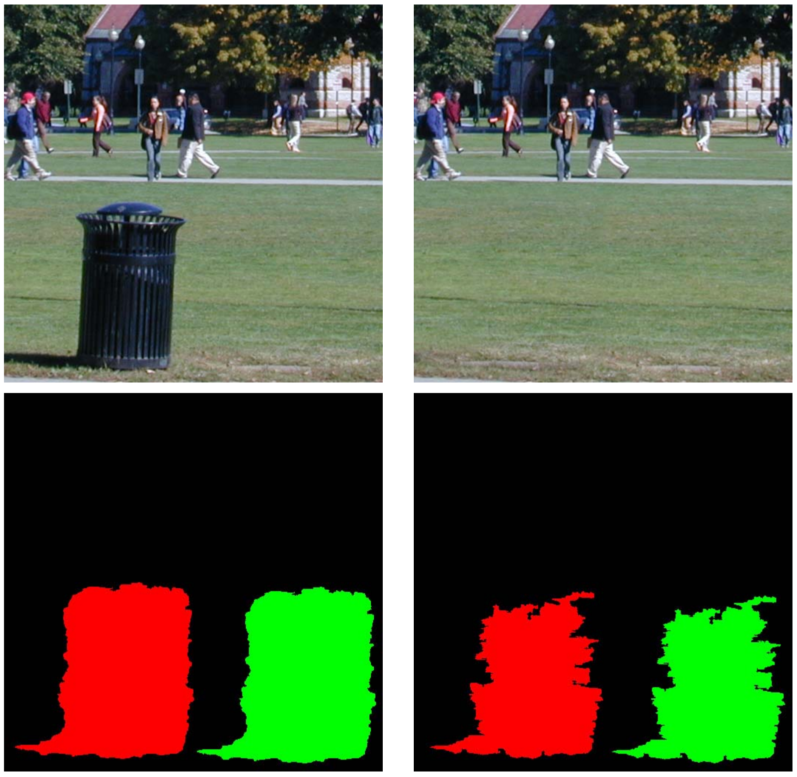


Figure 5.9: Shown are an original image (left) and the result of tampering by duplication (right), and the results of detecting duplication for two JPEG qualities of 100 (left) and 50 (right). This example has two different duplications.



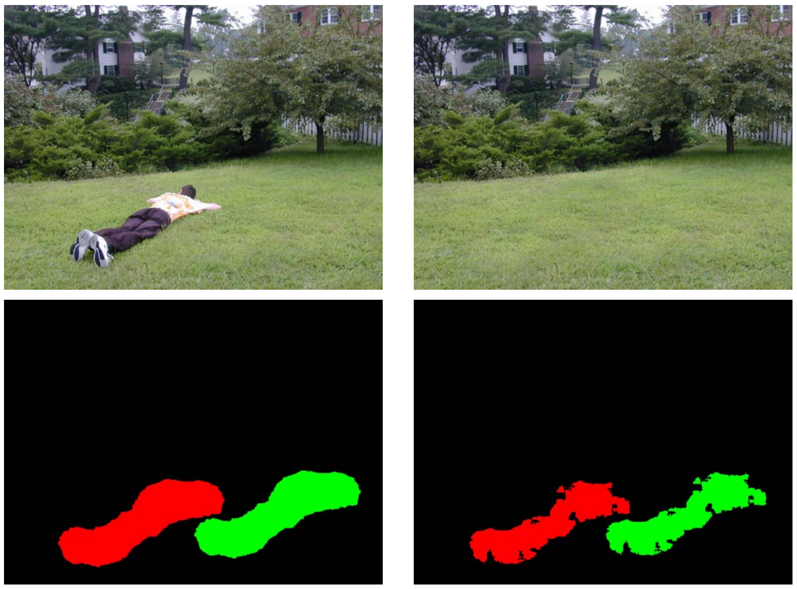


Figure 5.10: Shown are an original image (left) and the result of tampering by duplication (right), and the results of detecting duplication for two JPEG qualities of 100 (left) and 50 (right).

the 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 [42].

5.3 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 duplication of only a portion of one or more frames. In each case, central to the design of the algorithms is 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 suggest 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. The limitation of the frame duplication technique is that it cannot detect frame duplication when there is no motion in the subsequence containing the duplicated frames, since the technique cannot discriminate static frames from duplicated frames. However, if the video is compressed with MPEG, the temporal technique proposed in Chapter 4 might be capable of detecting the duplication in this scenario. The limitation of the region duplication technique is that it is not designed to be robust to geometric transformations.

A counterattack to the technique for detecting frame duplication is to alter the temporal or spatial correlation matrices by making minor variations on the inserted frames. Since each frame is downsampled by a factor of 8 in advance, it can be very hard to make significant changes in the temporal or spatial correlation matrices without introducing visually noticeable artifacts. A counterattack to the technique for detecting region duplication is to scale or rotate the duplicated region, as our technique is not designed to be robust to scaling and rotation attacks.

Chapter 6

Re-Projection

Often only hours after their release, major motion pictures can find their way onto the Internet. A simple and popular way to create such bootleg video is to simply record a movie from the theater screen. Although these videos are certainly not of the same quality as their subsequent DVD releases, increasingly compact and high resolution video recorders are affording better quality video recordings.

We describe how to automatically detect a video that was recorded from a screen. Shown in Fig. 6.1, for example, is a scene from the movie *Live Free Or Die Hard*. Also shown in this figure is the same scene as viewed on a theater screen. Note that due to the angle of the video camera relative to the screen, a perspective distortion has been introduced into this second recording. We show that re-projection can introduce a distortion into the intrinsic camera parameters (namely, the camera skew, which depends on the angle between the horizontal and vertical pixel axes). We leverage previous work on camera calibration to estimate this skew and show the efficacy of this technique to detect re-projected video.

6.1 Methods

We begin by describing the basic imaging geometry from 3-D world to 2-D image coordinates for both arbitrary points (Section 6.1.1) and for points constrained to a planar surface (Section 6.1.2). See [17] for a thorough treatment. We then describe the effect of a re-

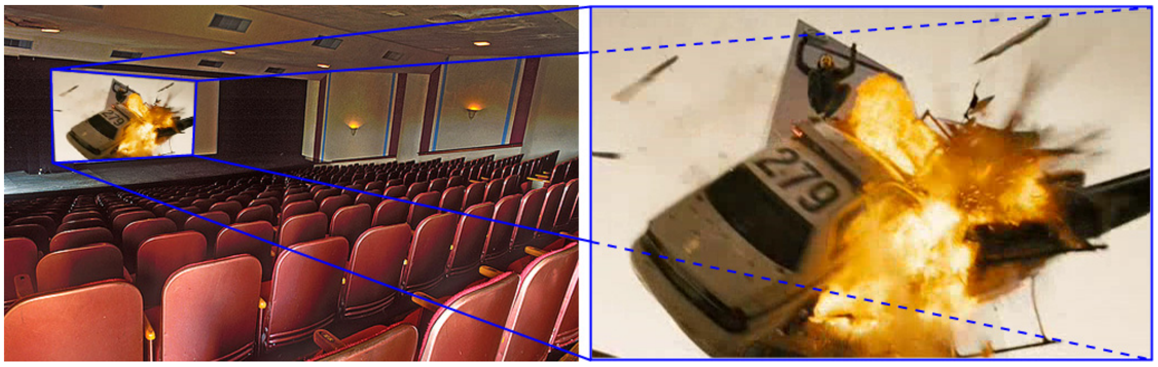


Figure 6.1: Shown on the right is a scene from the movie *Live Free Or Die Hard*, and shown on the left is the same scene as viewed on a movie screen. A recording of the projected movie introduces distortions that can be used to detect re-projected video.

projection: a non-planar projection followed by a planar projection (Section 6.1.3). Such a re-projection would result from, for example, video recording the projection of a movie.

6.1.1 Projective Geometry: non-planar

Under an ideal pinhole camera model, the perspective projection of arbitrary points $\vec{X}$ (homogeneous coordinates) in 3-D world coordinates is given by:

$$\vec{x} = \lambda K M \vec{X}, \tag{6.1}$$

where $\vec{x}$ is the 2-D projected point in homogeneous coordinates, λ is a scale factor, K is the intrinsic matrix, and M is the extrinsic matrix.

The 3×3 intrinsic matrix K embodies the camera's internal parameters:

$$K = \begin{pmatrix} \alpha f & s & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{pmatrix}, \tag{6.2}$$

where f is the focal length, α is the aspect ratio, (c_x, c_y) is the principle point (the projection of the camera center onto the image plane), and s is the skew (the skew depends on the angle, θ, between the horizontal and vertical pixel axes: $s = f\tan(\pi/2-\theta)$). For simplicity, we will assume square pixels ($\alpha = 1$, $s = 0$) and that the principal point is at the origin ($c_x = c_y = 0$) – these are reasonable assumptions for most modern-day cameras. With these assumptions, the intrinsic matrix simplifies to:

$$K = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.3}$$

The 3×4 extrinsic matrix M embodies the transformation from world to camera coordinates:

$$M = \begin{pmatrix} R \mid \vec{t} \end{pmatrix}, \tag{6.4}$$

where R is a 3×3 rotation matrix, and $\vec{t}$ is a 3×1 translation vector.

6.1.2 Projective Geometry: planar

Under an ideal pinhole camera model, the perspective projection of points $\vec{Y}$ constrained to a planar surface in world coordinates is given by:

$$\vec{y} = \lambda K P \vec{Y}, \tag{6.5}$$

where $\vec{y}$ is the 2-D projected point in homogeneous coordinates, and $\vec{Y}$, in the appropriate coordinate system, is specified by 2-D coordinates in homogeneous coordinates. As before, λ is a scale factor, K is the intrinsic matrix, and P is the extrinsic matrix. The intrinsic matrix K takes the same form as in Equation (6.3). The now 3×3 extrinsic matrix P takes the form:

$$P = \begin{pmatrix} \vec{p}_1 \; \vec{p}_2 \; \vec{t} \end{pmatrix}, \tag{6.6}$$

where $\vec{p}_1$, $\vec{p}_2$ and $\vec{p}_1 \times \vec{p}_2$ are the columns of the 3×3 rotation matrix that describes the transformation from world to camera coordinates, and as before, $\vec{t}$ is a 3×1 translation vector.

6.1.3 Re-Projection

Consider now the effect of first projecting arbitrary points in 3-D world coordinates into 2-D image coordinates, and then projecting these points a second time. As described in

Section 6.1.1, the first projection is given by:

$$\vec{x} = \lambda_1 K_1 M_1 \vec{X}. \tag{6.7}$$

The second planar projection, Section 6.1.2, is given by:

$$\vec{y} = \lambda_2 K_2 P_2 \vec{x} = \lambda_2 K_2 P_2 \left(\lambda_1 K_1 M_1 \vec{X} \right) = \lambda_2 \lambda_1 K_2 P_2 \left(K_1 M_1 \vec{X} \right). \tag{6.8}$$

We show that the effective projective matrix $K_2 P_2 K_1 M_1$ can be uniquely factored into a product of an intrinsic, K, and extrinsic, M, matrix. We begin by expressing the extrinsic matrix M_1 in terms of its rotation and translation components:

$$K_2 P_2 K_1 M_1 = K_2 P_2 K_1 \left(R_1 \mid \vec{t}_1 \right). \tag{6.9}$$

Multiplying R_1 and $\vec{t}_1$ each by the 3×3 matrix $(K_2 P_2 K_1)$ yields:

$$K_2 P_2 K_1 M_1 \;=\; \left(K_2 P_2 K_1 R_1 \mid K_2 P_2 K_1 \vec{t}_1 \right) \;=\; \left(K_2 P_2 K_1 R_1 \mid \vec{t}\,' \right). \tag{6.10}$$

Consider now the 3×3 matrix $K_2 P_2 K_1 R_1$. Since each of these matrices is non-singular, their product is non-singular. As such, this matrix can be uniquely factored (within a sign), using RQ-factorization, into a product of an upper triangular, U, and orthonormal, O, matrix:

$$K_2 P_2 K_1 M_1 \;=\; \lambda \left(UO \mid \tfrac{1}{\lambda} \vec{t}\,' \right) \;=\; \lambda U \left(O \mid \tfrac{1}{\lambda} U^{-1} \vec{t}\,' \right) \;=\; \lambda K M, \tag{6.11}$$

where $K = U$, $M = (O \mid \frac{1}{\lambda} U^{-1} \vec{t}\,')$, and where λ is chosen so that the $(3,3)$ entry of U has unit value.

Recall that we assumed that the camera skew (the $(1,2)$ entry in the 3×3 intrinsic matrix, Equation (6.2), is zero. We next show that that a re-projection can yield a non-zero skew in the intrinsic matrix K. As such, significant deviations of the skew from zero in the estimated intrinsic matrix can be used as evidence that a video has been re-projected.

Expressing each 3×4 extrinsic matrix M_1 and M in terms of their rotation and translation components yields:

$$\begin{aligned} K_2 P_2 K_1 (R_1 \mid \vec{t}_1) &= \lambda K (R \mid \vec{t}) \\ K_2 P_2 K_1 R_1 &= \lambda K R. \end{aligned} \tag{6.12}$$

Reshuffling[1] a few terms yields:

$$\begin{aligned} K^{-1} K_2 P_2 K_1 R_1 &= \lambda R \\ K^{-1} K_2 P_2 K_1 &= \lambda R R_1^T. \end{aligned} \tag{6.13}$$

Note that the right-hand side of this relationship is an orthogonal matrix – this will be exploited later. On the left-hand side, the left-most matrix is the inverse of the effective intrinsic matrix in Equation (6.2):

$$K^{-1} = \begin{pmatrix} \frac{1}{\alpha f} & -\frac{s}{\alpha f^2} & \frac{s c_y - c_x f}{\alpha f^2} \\ 0 & \frac{1}{f} & -\frac{c_y}{f} \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.14}$$

And the product of the next three matrices is:

$$\begin{aligned} K_2 P_2 K_1 &= \begin{pmatrix} f_2 & 0 & 0 \\ 0 & f_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_{11} & p_{21} & t_1 \\ p_{12} & p_{22} & t_2 \\ p_{13} & p_{23} & t_3 \end{pmatrix} \begin{pmatrix} f_1 & 0 & 0 \\ 0 & f_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} f_1 f_2 p_{11} & f_1 f_2 p_{21} & f_2 t_1 \\ f_1 f_2 p_{12} & f_1 f_2 p_{22} & f_2 t_2 \\ f_1 p_{13} & f_1 p_{23} & t_3 \end{pmatrix} , \\ &= \begin{pmatrix} \vec{q}_1^T \\ \vec{q}_2^T \\ \vec{q}_3^T \end{pmatrix} \end{aligned} \tag{6.15}$$

[1]Since the matrix R_1 is orthonormal $R_1^{-1} = R_1^T$.

where f_2 and f_1 are the focal lengths of the original projections, p_{1i} and p_{2i} correspond to the i^{th} element of $\vec{p}_1$ and $\vec{p}_2$, and t_i corresponds to i^{th} element of $\vec{t}_2$ (the third column of matrix P_2). The product of the four matrices on the left-hand side of Equation (6.13) is then:

$$K^{-1}K_2P_2K_1 = \begin{pmatrix} \left(\frac{1}{\alpha f}\vec{q}_1 - \frac{s}{\alpha f^2}\vec{q}_2 + \frac{sc_y - c_x f}{\alpha f^2}\vec{q}_3\right)^T \\ \left(\frac{1}{f}\vec{q}_2 - \frac{c_y}{f}\vec{q}_3\right)^T \\ \vec{q}_3^T \end{pmatrix}. \tag{6.16}$$

Recall that $K^{-1}K_2P_2K_1 = \lambda R R_1^T$, Equation (6.13), and that R and R_1^T are each orthonormal. Since the product of two orthonormal matrices is orthonormal, $K^{-1}K_2P_2K_1$ is orthogonal (the rows/columns will not be unit length when $\lambda \neq 1$). This orthogonality constrains the above matrix rows as follows:

$$\vec{q}_3^T\left(\frac{1}{f}\vec{q}_2 - \frac{c_y}{f}\vec{q}_3\right) = 0 \tag{6.17}$$

$$\left(\frac{1}{f}\vec{q}_2 - \frac{c_y}{f}\vec{q}_3\right)^T\left(\frac{1}{\alpha f}\vec{q}_1 - \frac{s}{\alpha f^2}\vec{q}_2 + \frac{sc_y - c_x f}{\alpha f^2}\vec{q}_3\right) = 0. \tag{6.18}$$

Solving Equation (6.17) for c_y yields:

$$c_y = \frac{\vec{q}_3^T\vec{q}_2}{\|\vec{q}_3\|^2}. \tag{6.19}$$

Substituting for c_y into Equation (6.18), followed by some simplifications, yields:

$$s = f\frac{\vec{q}_2^T\vec{q}_1\|\vec{q}_3\|^2 - (\vec{q}_3^T\vec{q}_2)(\vec{q}_3^T\vec{q}_1)}{\|\vec{q}_2\|^2\|\vec{q}_3\|^2 - (\vec{q}_3^T\vec{q}_2)^2}. \tag{6.20}$$

Note that the skew, s, is expressed only in terms of the effective focal length f, the pair of intrinsic matrices K_1 and K_2, and the second transformation matrix P_2. We can now see under what conditions $s = 0$.

First, note that the denominator of Equation (6.20) cannot be zero. If $\|\vec{q}_2\|^2\|\vec{q}_3\|^2 - (\vec{q}_3^T\vec{q}_2)^2 = 0$ then, $\vec{q}_2 \propto \vec{q}_3$, in which case $K_2P_2K_1$ is singular, which it cannot be, since each matrix in this product is full rank. And, since $f \neq 0$, the skew is zero only when the numerator of Equation (6.20) is zero:

$$\begin{aligned} \vec{q}_2^T \vec{q}_1 \|\vec{q}_3\|^2 - (\vec{q}_3^T \vec{q}_2)(\vec{q}_3^T \vec{q}_1) &= 0 \\ f_1^2 p_{31} p_{32} - t_1 t_2 + p_{33}^2 t_1 t_2 + p_{31} p_{32} t_3^2 - p_{32} p_{33} t_1 t_3 - p_{31} p_{33} t_2 t_3 &= 0, \end{aligned} \tag{6.21}$$

where p_{3i} is the i^{th} element of $\vec{p}_3 = \vec{p}_1 \times \vec{p}_2$. Although we have yet to geometrically fully characterize the space of coefficients that yields a zero skew, there are a few intuitive cases that can be seen from the above constraint. For example, if the world to camera rotation is strictly about the z-axis, then $p_{31} = p_{32} = 0$ and $p_{33} = 1$, and the skew $s = 0$. This situation arises when the image plane of the second projection is perfectly parallel to the screen being imaged. As another example, if $\vec{t} = \pm f_1 \vec{p}_3$, then the skew $s = 0$. This situations arises when the translation of the second projection is equal to the third column of the rotation matrix scaled by focal length of the first projection – a perhaps somewhat unlikely configuration.

Although there are clearly many situations under which $s = 0$, our simulations suggest that under realistic camera motions, this condition is rarely satisfied. Specifically, we computed the skew, Equation (6.20), from one million randomly generated camera configurations. The relative position of the second camera to the planar projection screen was randomly selected with the rotation in the range $[-45, 45]$ degrees, X and Y translation in the range [-1000,1000], Z translation in the range $[4000, 6000]$, and focal length in the range $[25, 75]$. The average skew was 0.295, and only 48 of the $1,000,000$ configurations had a skew less than 10^{-5} (in a similar simulation, the estimated skew for a single projection is on the order of 10^{-12}).

6.1.4 Camera Skew

From the previous sections, we see that re-projection can cause a non-zero skew in the camera's intrinsic parameters. We review two approaches for estimating camera skew from a video sequence. The first estimates the camera skew from a known planar surface, while the second assumes no known geometry.

Skew estimation I:

Recall that the projection of a planar surface, Equation (6.5), is given by:

$$\vec{y} = \lambda K P \vec{Y} = \lambda H \vec{Y}, \tag{6.22}$$

where $\vec{y}$ is the 2-D projected point in homogeneous coordinates, and $\vec{Y}$, in the appropriate coordinate system, is specified by 2-D coordinates in homogeneous coordinates. The 3×3 matrix H is a non-singular matrix referred to as a homography. Given the above equality, the left- and right-hand sides of this homography satisfy the following:

$$\begin{gathered} \vec{y} \times (H\vec{Y}) = \vec{0} \\ \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} \times \left(\begin{pmatrix} h_{11} & h_{21} & h_{31} \\ h_{12} & h_{22} & h_{32} \\ h_{13} & h_{23} & h_{33} \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} \right) = \vec{0}. \end{gathered} \tag{6.23}$$

Note that due to the equality with zero, the multiplicative scalar λ, Equation (6.22), is factored out. Evaluating the cross product yields:

$$\begin{pmatrix} y_2(h_{13}Y_1 + h_{23}Y_2 + h_{33}Y_3) - y_3(h_{12}Y_1 + h_{22}Y_2 + h_{32}Y_3) \\ y_3(h_{11}Y_1 + h_{21}Y_2 + h_{31}Y_3) - y_1(h_{13}Y_1 + h_{23}Y_2 + h_{33}Y_3) \\ y_1(h_{12}Y_1 + h_{22}Y_2 + h_{32}Y_3) - y_2(h_{11}Y_1 + h_{21}Y_2 + h_{31}Y_3) \end{pmatrix} = \vec{0}. \tag{6.24}$$

This constraint is linear in the unknown elements of the homography h_{ij}. Re-ordering the terms yields the following system of linear equations:

$$\begin{pmatrix} 0 & 0 & 0 & -y_3Y_1 & -y_3Y_2 & -y_3Y_3 & y_2Y_1 & y_2Y_2 & y_2Y_3 \\ y_3Y_1 & y_3Y_2 & y_3Y_3 & 0 & 0 & 0 & -y_1Y_1 & -y_1Y_2 & -y_1Y_3 \\ -y_2Y_1 & -y_2Y_2 & -y_2Y_3 & y_1Y_1 & y_1Y_2 & y_1Y_3 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{21} \\ h_{31} \\ h_{12} \\ h_{22} \\ h_{32} \\ h_{13} \\ h_{23} \\ h_{33} \end{pmatrix} = \vec{0}$$

$$A\vec{h} = \vec{0}. \tag{6.25}$$

A matched set of points $\vec{y}$ and $\vec{Y}$ appear to provide three constraints on the eight unknowns elements of $\vec{h}$ (the homography is defined only up to an unknown scale factor, reducing the unknowns from nine to eight). The rows of the matrix, A, however, are not linearly independent (the third row is a linear combination of the first two rows). As such, this system provides only two constraints in eight unknowns. In order to solve for $\vec{h}$, we require four or more points with known image, $\vec{y}$, and (planar) world, $\vec{Y}$, coordinates that yield eight or more linearly independent constraints. From four or more points, standard least-squares techniques, as described in [17, 24], can be used to solve for $\vec{h}$: the minimal eigenvalue eigenvector of A^TA is the unit vector $\vec{h}$ that minimizes the least-squares error.

We next describe how to estimate the camera skew from the estimated homography H. This approach is a slightly modified version of [52]. Recall that H can be expressed as:

$$H = KP = K\left(\vec{p}_1 \; \vec{p}_2 \mid \vec{t}\right). \tag{6.26}$$

The orthonormality of $\vec{p}_1$ and $\vec{p}_2$, yields the following two constraints:

$$\vec{p}_1^T \vec{p}_2 = 0 \qquad \text{and} \qquad \vec{p}_1^T \vec{p}_1 = \vec{p}_2^T \vec{p}_2, \tag{6.27}$$

which in turn imposes the following constraints on H and K:

$$\begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \end{pmatrix}^T K^{-T} K^{-1} \begin{pmatrix} h_{21} \\ h_{22} \\ h_{23} \end{pmatrix} = 0 \tag{6.28}$$

$$\begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \end{pmatrix}^T K^{-T} K^{-1} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \end{pmatrix} = \begin{pmatrix} h_{21} \\ h_{22} \\ h_{23} \end{pmatrix}^T K^{-T} K^{-1} \begin{pmatrix} h_{21} \\ h_{22} \\ h_{23} \end{pmatrix}. \tag{6.29}$$

For notational ease, denote $B = K^{-T} K^{-1}$, where B is a symmetric matrix parametrized with three degrees of freedom (see Equation (6.33) in Appendix B):

$$B = \begin{pmatrix} b_{11} & b_{12} & 0 \\ b_{12} & b_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.30}$$

Notice that by parametrizing the intrinsic matrix in this way, we have bundled all of the anomalies of a double projection into the estimate of the camera skew. Substituting the matrix B into the constraints of Equations (6.28)-(6.29) yields the following constraints:

$$\begin{pmatrix} h_{11}h_{21} & h_{12}h_{21} + h_{11}h_{22} & h_{12}h_{22} \\ h_{11}^2 - h_{21}^2 & 2\left(h_{11}h_{12} - h_{21}h_{22}\right) & h_{12}^2 - h_{22}^2 \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \\ b_{22} \end{pmatrix} = - \begin{pmatrix} h_{13}h_{23} \\ h_{13}^2 - h_{23}^2 \end{pmatrix}. \tag{6.31}$$

Each image of a planar surface enforces two constraints on the three unknowns b_{ij}. The matrix $B = K^{-T} K^{-1}$ can, therefore, be estimated from two or more views of the same planar surface using standard least-squares estimation. We next show that the desired skew can be determined from the estimated matrix B. The intrinsic matrix is parametrized as:

$$K = \begin{pmatrix} f & s & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.32}$$

Applying the matrix inverse and multiplication yields:

$$B \;=\; K^{-T}K^{-1} \;=\; \begin{pmatrix} \frac{1}{f^2} & -\frac{s}{f^3} & 0 \\ -\frac{s}{f^3} & \frac{s^2+f^2}{f^4} & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.33}$$

from which:

$$\frac{s}{f} \;=\; -\frac{b_{12}}{b_{11}} \tag{6.34}$$

$$s \;=\; -f\frac{b_{12}}{b_{11}}. \tag{6.35}$$

Note that the estimate of the skew, s, is scaled by the focal length, f. Since a camera's skew depends on the focal length, it is desirable to work with the normalized skew, Equation (6.34).

Skew estimation II:

We showed in the previous section how to estimate a camera's skew from two or more views of a planar surface. This approach has the advantage that it affords a closed-form linear solution, but has the disadvantage that it only applies to frames that contain a known planar surface. Here we review a related approach that does not require any known world geometry, but requires a non-linear minimization.

Consider two frames of a video sequence with corresponding image points given by $\vec{u}$ and $\vec{v}$, specified in 2-D homogeneous coordinates. It is well established [17] that these points satisfy the following relationship:

$$\vec{v}^T F \vec{u} = 0, \tag{6.36}$$

where F, the fundamental matrix, is a 3×3 singular matrix (rank(F) = 2). Writing the above relationship in terms of the vector and matrix elements yields:

$$\begin{pmatrix} v_1 & v_2 & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{21} & f_{31} \\ f_{12} & f_{22} & f_{32} \\ f_{13} & f_{23} & f_{33} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ 1 \end{pmatrix} = 0 \tag{6.37}$$

$$u_1 v_1 f_{11} + u_2 v_1 f_{21} + v_1 f_{31} + u_1 v_2 f_{12} + u_2 v_2 f_{22} + v_2 f_{32} + u_1 f_{13} + u_2 f_{23} + f_{33} = 0.$$

Note that this constraint is linear in the elements of the fundamental matrix f_{ij}, leading to the following system of linear equations:

$$\begin{pmatrix} u_1 v_1 & u_2 v_1 & v_1 & u_1 v_2 & u_2 v_2 & v_2 & u_1 & u_2 & 1 \end{pmatrix} \begin{pmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{pmatrix} = 0 \tag{6.38}$$

$$A\vec{f} = \vec{0}.$$

Each pair of matched points $\vec{u}$ and $\vec{v}$ provides one constraint for the eight unknown elements of $\vec{f}$ (the fundamental matrix is defined only up to an unknown scale factor reducing the unknowns from nine to eight). In order to solve for the components of the fundamental matrix, $\vec{f}$, we require eight or more matched pairs of points [32, 16]. Standard least-squares techniques can be used to solve for $\vec{f}$: the minimal eigenvalue eigenvector of $A^T A$ is the unit vector $\vec{f}$ that minimizes the least-squares error.

We next describe how to estimate the camera skew from the estimated fundamental matrix F. We assume that the intrinsic camera matrix K, Equation (6.3), is the same across the views containing the matched image points. The essential matrix E is then defined as:

$$E = K^T F K. \tag{6.39}$$

Since F has rank 2 and the intrinsic matrix is full rank, the essential matrix E has rank 2. In addition, the two non-zero singular values of E are equal [22]. This property will be exploited to estimate the camera skew. Specifically, as described in [38], we establish the following cost function to be minimized in terms of the camera focal length f and skew s:

$$C(f, s) = \sum_{i=1}^{n} \frac{\sigma_{i1} - \sigma_{i2}}{\sigma_{i2}}, \tag{6.40}$$

where σ_{i1} and σ_{i2} are, in descending order, the non-zero singular values of E from n estimated fundamental matrices (each computed from pairs of frames throughout a video sequence), and where K is parametrized as:

$$K = \begin{pmatrix} f & s & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{6.41}$$

Note that since only the relative differences in the singular values of E are considered, the arbitrary scale factor to which E is estimated does not effect the estimation of the skew. As before, by parametrizing the intrinsic matrix in this way, we have bundled all of the anomalies of a double projection into the estimate of the camera skew. The cost function, Equation (6.40), is minimized using a standard derivative-free Nelder-Mead non-linear minimization.

6.2 Results

We report on a set of simulations and sensitivity analysis for each of the skew estimation techniques described in the previous sections. We then show the efficacy of these approaches on a real-video sequence. In each set of simulations we provide the estimation algorithm with the required image coordinates. For the real-video sequence we briefly describe a point tracking algorithm which provides the necessary image coordinates for estimating the camera skew.

6.2.1 Simulation (skew estimation I):

Recall that a minimum of four points with known geometry on a planar surface viewed from a minimum of two views are required to estimate the camera skew. We therefore randomly generated between 4 and 64 points on a planar surface and generated a video sequence of this stationary surface. In all of the simulations, the first projection was specified by the following camera parameters: the planar surface was 2000 units from the camera, between successive frames the rotation about each axis was in the range $[-2.5, 2.5]$ degrees and the translation in each dimension was in the range $[-50, 50]$, and the camera focal length was in the range $[25, 75]$ (but fixed for each sequence). For the second projection, the camera was placed a distance of 5000 units from the first projected image and underwent a motion in the same range as the first camera. We randomly generated $10,000$ such sequences as imaged through a single projection, and $10,000$ sequences as imaged through a double projection (re-projection).

In the first simulation, we tested the sensitivity to additive noise. As described above, 30 frames were generated each containing 4 points on a planar surface. Noise in the range of $[0, 1]$ pixels was added to the final image coordinates. The skew was estimated from 15 pairs of frames, where each frame at time t was paired with a frame at time $t+15$. A sequence was classified as re-projected if one or more of the image pairs yielded an estimated skew greater than 0.1. While this type of voting scheme yields slightly higher false positive rates, it also significantly improves the detection accuracy. In the absence of noise, 0 of the $10,000$ singly projected sequences were classified as re-projected, and 84.9% of the re-projected sequences were correctly classified. With 0.5 pixels of noise, 17.2% of the singly projected sequences were incorrectly classified as re-projected, and 87.4% of the re-projected sequences were correctly classified. Shown in Fig. 6.2(a) are the complete set of results for additive noise in the range of $[0, 1]$ pixels. Note that even with modest amounts of noise, the false positive rate increases to an unacceptable level. We next show how these results can be improved upon.

In this next simulation, we tested the sensitivity to the number of known points on the

(a)

(b)

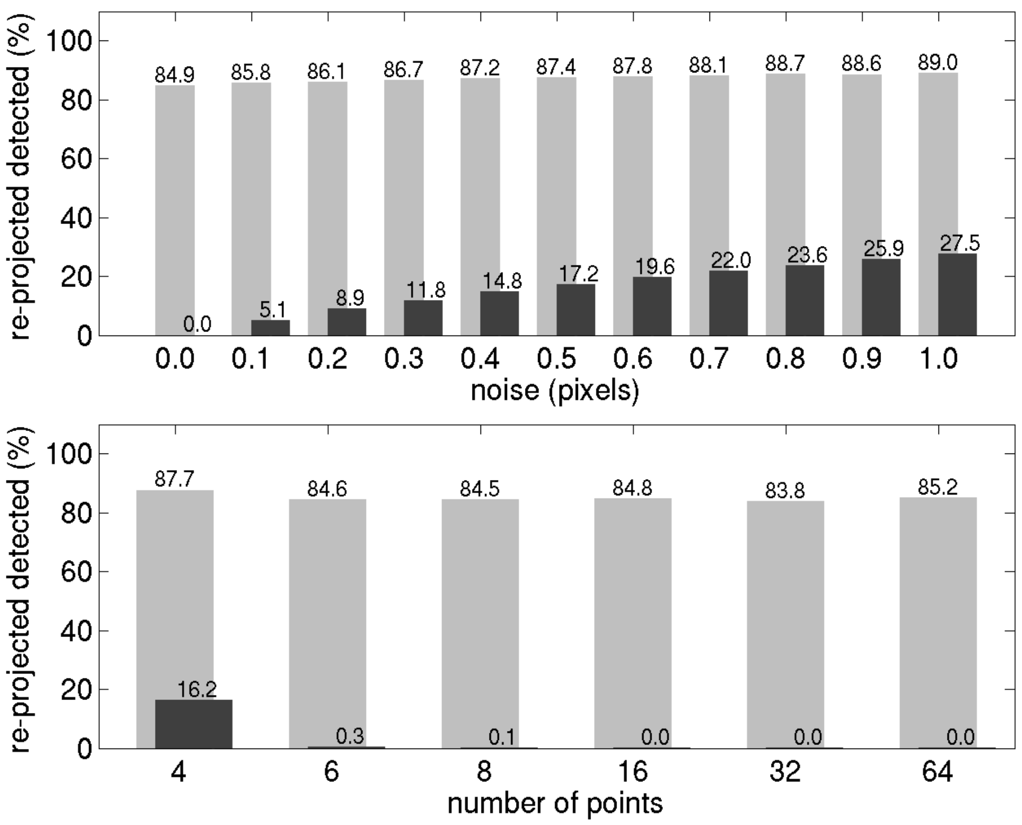


Figure 6.2: Skew Estimation I: Detection accuracy (light gray) and false positives (dark gray) as a function of noise (top) and the number of points (bottom).

planar surface. The noise level was 0.5 pixels, and the number of known points was in the range $[4, 64]$. All other parameters were the same as in the previous simulation. With the minimum of 4 points, 16.2% of the singly projected sequences were incorrectly classified while 87.7% of the re-projected sequences were correctly classified (similar to the results in the previous simulation). With 6 points, only 0.33% of the single projection sequences were incorrectly classified, while the accuracy of the re-projected sequences remained relatively high at 84.6%. Shown in Fig. 6.2(b) are the complete set of results – beyond 8 points, the advantage of more points becomes negligible.

In summary, from 6 points, with 0.5 pixels noise, in 30 frames, re-projected video can be detected with 85% accuracy, and with 0.3% false positives.

6.2.2 Simulation (skew estimation II):

Recall that a minimum of eight points viewed from a minimum of two views are required to estimate the camera skew. We therefore generated between 8 and 128 points with arbitrary geometry and generated a video sequence of this stationary cloud of points. In all of the simulations, the first and second projection were generated as described in the previous section. We randomly generated $10,000$ sequences as imaged through a single projection, and $10,000$ sequences as imaged through a double projection (re-projection). As before, a sequence was classified as re-projected if the estimated skew was greater than 0.1.

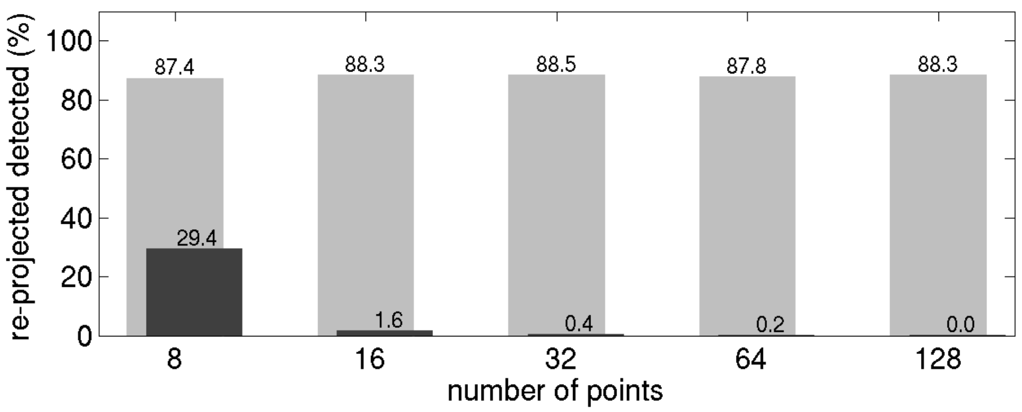
In the first simulation with the minimum of 8 points, 2 frames, and with no noise, 0 of the $10,000$ singly projected sequences were classified as re-projected, and 88.9% of the re-projected sequences were correctly classified. With even modest amounts of noise, however, this minimum configuration yields unacceptably high false positives. We find that the estimation accuracy is more robust to noise when the skew is estimated from multiple frames (i.e., multiple fundamental matrices in Equation (6.40)). In the remaining simulations, we estimated the skew from 5 fundamental matrices, where each frame t is paired with the frame at time $t + 15$.

In the second simulation, the number of points were in the range $[8, 128]$, with 0.5 pixels of additive noise, and 5 fundamental matrices. With the minimum of 8 points the false positive rate is 29.4%, while with 32 points, the false positive rate falls to 0.4%. In each case, the detection accuracy is approximately 88%. Shown in Fig. 6.3(a) are the complete set of results for varying number of points.

In the third simulation, the number of points was 32, with 0.5 pixels of noise, and with the number of fundamental matrices (i.e., pairs of frames) in the range $[1, 20]$. As shown in Fig. 6.3(b), increasing the number of fundamental matrices reduces the false positives while the detection accuracy remains approximately the same.

In summary, from 32 points, with 0.5 pixels noise, in 5 fundamental matrices, re-projected video can be detected with 88% accuracy, and with 0.4% false positives. This is similar to the accuracy for the skew estimation from points on a planar surface. The advantage here, however, is that this approach does not require known geometry of points

(a)



(b)

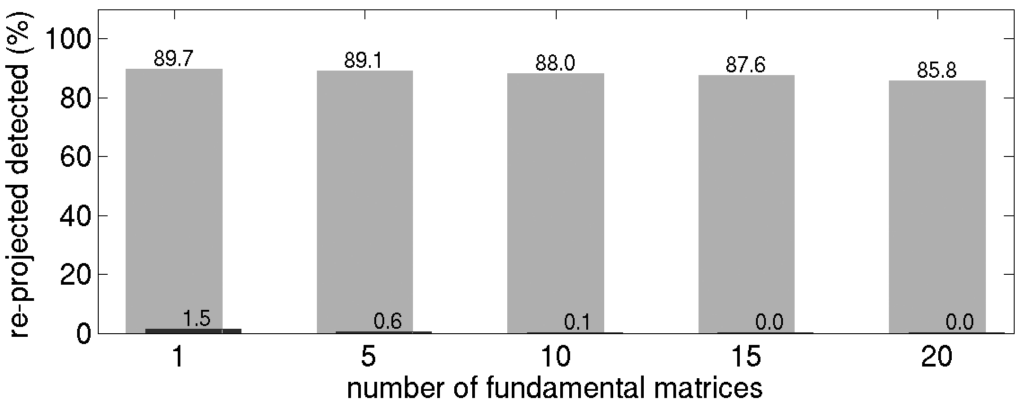


Figure 6.3: Skew Estimation II: Detection accuracy (light gray) and false positives (dark gray) as a function of the number of points (top) and the number fundamental matrices (bottom).

on a planar surface.

6.2.3 Real Video

Shown in Fig. 6.4 are three frames of a 42 frame segment from the movie *Live Free Or Die Hard*. These frames were digitized at a resolution of 720 $\times$ 304 pixels. Superimposed on each frame are 64 features tracked across all frames. We employed the KLT feature tracker [33, 50, 46] which automatically selects features using a Harris detector, and tracks these points across time using standard optical flow techniques. We manually removed any features with clearly incorrect tracking, and any points not on the buildings or street (the estimation of a fundamental matrix requires points with a rigid body geometry). These tracked features were then used to estimate the skew (method II). The 42 frames were grouped into 21 pairs from which the skew was estimated (each frame at time t was paired with the frame at time $t + 21$). The estimated skew was 0.029, well below the threshold of

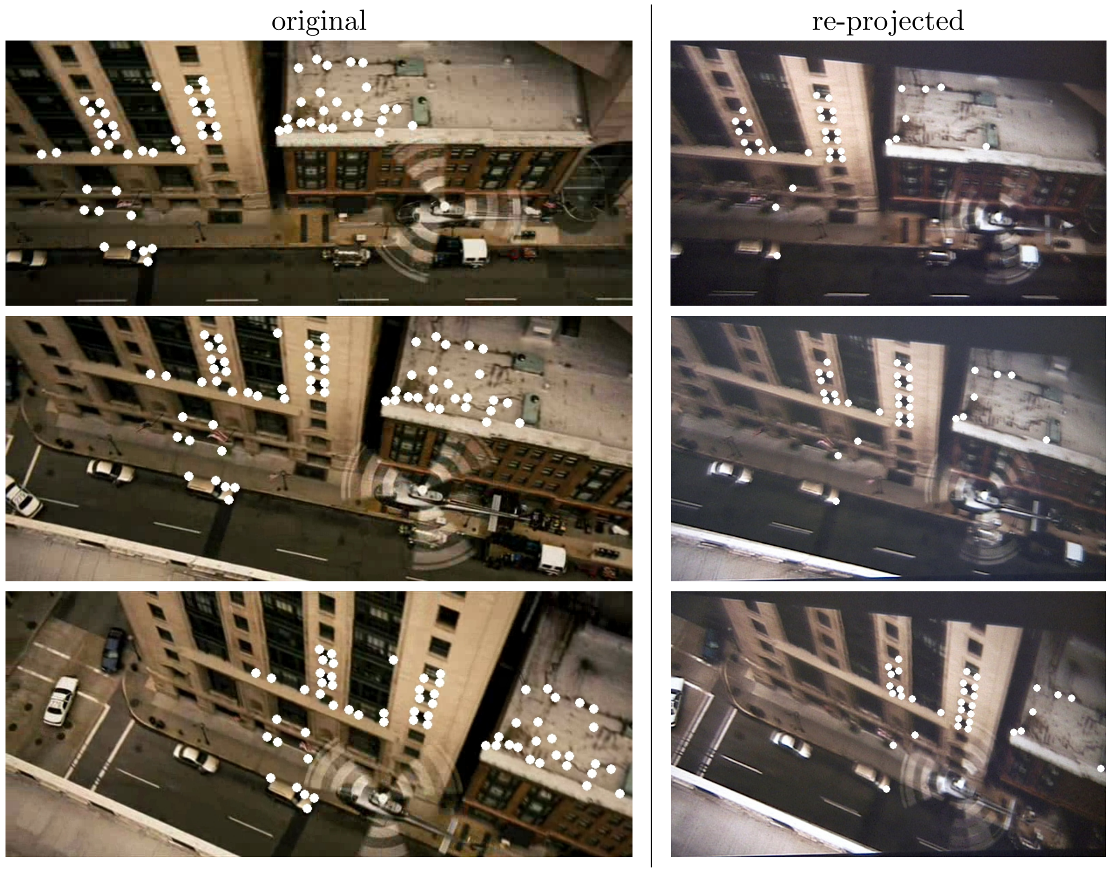


Figure 6.4: Shown are the first, middle, and last frame of a 42-frame segment of the movie *Live Free or Die Hard*. On the left is the original digitized video, and on the right is the re-projected video. The white dots denote the tracked features used to estimate the camera skew using method II.

0.1.

This 42-frame segment was then displayed on a 20 inch LCD computer monitor with 1600×1200 pixel resolution, and recorded with a Canon Elura video camera at a resolution of 640×480. As above, features were tracked in this video segment, from which the skew was estimated. The estimated skew was 0.25, an order of magnitude larger than the skew from the authentic video and well above our threshold of 0.1.

6.3 Discussion

We have described how to detect a re-projected video that was recorded directly from a projection on a movie or television screen. We have shown that such a re-projection introduces a skew into the camera's intrinsic parameters, which does not normally occur with authentic video. The camera skew can be estimated from two or more video frames from either four or more points on a planar surface, or eight or more arbitrary points. In addition, from four or more points with known geometry in general configuration, the camera skew can be estimated from only a single image. This technique can be adapted to detect if a single image has been re-photographed (an oft-cited technique to circumvent some forensic image analysis, e.g., [45]). The weakness of this method is that it cannot detect the re-projected video when the principal axes of the two cameras coincide, since in this case the skew of the effective intrinsic matrix is zero.

A counterattack for this technique would be relatively complex as it would require the pirate to estimate the homography introduced by the re-projection and then apply an inverse transform to each frame. To estimate the homography, the pirate needs to know the intrinsic parameters of the second camera and the relative position between the screen and the camera.

Chapter 7

Discussion

We are living in an age in which we are exposed to a large amount of digital video. Technological advances in video tampering erode our trust in the reliability of video as an accurate representation of reality. Therefore, it is urgent for the scientific community to come up with methods for authenticating video recordings.

Watermarking is one way to authenticate video. But the drawback of this approach is that it requires that a watermark is inserted precisely at the time of recording, which limits its application to specially equipped cameras. The digital video forensic approach, however, can function in the absense of a watermark. Digital video forensic techniques all assume that tampering may disturb certain underlying properties of the video and that these perturbations can be modeled and estimated in order to detect tampering.

Videos are usually recorded in interlaced mode, that is, the even and odd scan lines of a single frame are recorded at different times. Motion in a video leads to a "combing" artifact in the frames. 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. In Chapter 2 and Chapter 3, we described two related techniques to detect tampering in these two types of videos respectively. These two techniques can also be adapted slightly to detect frame rate conversion.

For the technique targeting interlaced video, described in Chapter 2, we assume the motion is constant within a very short period of time. With this assumption, we expect the inter-field motion to be the same as the inter-frame motion for a given frame. While the overall motion may change over time, this equality should be relatively constant. Since manipulation is likely to destroy this temporal correlation, we can rely on this pattern to detect video forgeries. We described how to estimate the inter-field and inter-frame motion, and demonstrated in the experiments that the violation of the equality between these two motions can be used as evidence of tampering. This technique can identify the region that has been tampered with. Since the technique operates on down-sampled images, the motion estimation is relatively robust to compression artifacts, which makes the technique applicable to interlaced videos of varying quality. The drawback of this method is that it fails to detect manipulations in static regions, since in these regions the inter-field and inter-frame motion are constantly zero.

For the technique targeting de-interlaced video, described in Chapter 3, we assume that in a de-interlaced video the pixels are linearly correlated to their spatial and temporal neighbors due to the interpolation and that tampering is likely to undermine these correlations. We described an EM algorithm to simultaneously estimate the linear correlation parameters and the probability of each pixel being authentic. We also demonstrated the efficacy of the technique on simulated and visually plausible forgeries. Similar to the method for interlaced video, this technique can also identify the region that has been tampered with. The drawback of this technique is that heavy compression is likely to destroy the linear correlations among the pixels, which limits the application of this technique to relatively high quality videos.

Piecing together two video sequences with different frame rates one after the other requires matching the two frame rates by either down-conversion or up-conversion of the frame rate of one of the sequences. Down-conversion usually involves periodically removing a certain number of frames. This undermines the assumption of constant motion between frames. So we can slightly adapt the technique for interlaced video to detect frame rate down-conversion. Up-conversion usually involves periodically inserting extra frames created by some form of interpolation of the neighboring frames. This is essentially similar to the de-interlace process. So a slightly adapted version of the EM algorithm that we proposed for the de-interlaced video can be applied to detect frame rate up-conversion.

MPEG (MPEG-1/2) is a well established video compression standard. When an MPEG video is modified, and re-saved in MPEG format, it is subject to double compression. In Chapter 4, we showed that two types of artifacts – spatial and/or temporal artifacts – are likely to be introduced into the resulting video. We described two techniques targeting each of the two types of artifacts respectively to detect forgeries in MPEG videos. The spatial technique is capable of detecting localized tampering in regions as small as 16×16 pixels. This feature is particularly attractive for detecting the fairly common digital effect of green-screening. This technique can also be adapted to detect doubly JPEG compressed images. The drawback of the this technique is that it is only effective when the doctored video is compressed with a quality higher than the original. The temporal technique is useful to detect manipulation that involves frame insertion or deletion that can be fairly easy to perform yet leave no visual clue when the camera is stationary and the background is largely static. Since only the sequence of the frames is rearranged but individual frames are not modified, such manipulation can be very hard to detect with other techniques. The limitation of the temporal technique is that it cannot detect tampering when the number of frames inserted or deleted is a multiple of the GOP length.

Repeating a subsequence in a video can be used to conceal unwanted footage. When carefully done, such duplication can be very difficult to detect. Although researchers have previously proposed techniques [12, 42] to detect duplication in images, these techniques are computationally too inefficient to be applicable to a video sequence of even modest length. In Chapter 5, we described two computationally efficient techniques for detecting duplication, one targeting duplicated frames and the other targeting duplicated regions across frames. Results from both real and simulated tampering suggest that these algorithms are relatively robust to compression and can be applied to detect duplications in both high- and low-quality video. The limitation of the frame duplication technique is that it cannot discriminate static frames from duplicated frames, and therefore cannot detect forgeries when there is no motion in the sequence. The limitation of the region duplication technique is that it cannot detect geometrically transformed duplicated regions.

A simple and popular way to create a bootleg video is to simply record a movie from the theater screen. Although re-projected videos are not forgeries, they can be detected in a way similar to the way we detect forgeries. This is because re-projection may introduce distortion (the skew) into the intrinsic camera parameters. In Chapter 6, we described two methods for estimating the skew; significant deviations of the skew from zero can be used as evidence that a video has been re-projected. The weakness of this technique is that it cannot detect re-projection when the principal axes of the two cameras coincide, since in this case the skew of the effective intrinsic matrix is zero.

Digital video forensics is in its very early stages. Of the many kinds of perturbations that tampering can introduce into a video, we only explored a few. The more developed field of image forensics can provide a source of new ideas for future video forensic tools. For example, Johnson et al. [23, 27, 26] proposed a series of tools to detect lighting inconsistencies in images. These tools can potentially be extended to videos. Watermark-based video authentication techniques only detect forgeries in videos with watermarks, which is the minority of digital videos. The video forensic tools offered here work on all videos, whether or not they contain watermarks. We hope that our work will inspire the development of many more tools for detecting a wide variety of video forgeries and we hope that a combination of these tools can help restore at least some trust in digital videos. We also hope that the development of the field of digital video forensics can contribute to a better understanding of the properties of digital videos.

Bibliography

[1] J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, Feb. 1994.

[2] E. B. Bellers and G. de Haan. *De-Interlacing: A Key Technology for Scan Rate Conversion*, chapter Bayesian Multi-scale Differential Optical Flow. Elsevier, 2000.

[3] 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*, 6(5):436–446, 1996.

[4] E. De 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.

[5] Mo Chen, Jessica Fridrich, Miroslav Goljan, and Jan Lukáš. Source digital camcorder identification using sensor photo response non-uniformity. *Security, Steganography, and Watermarking of Multimedia Contents IX. Proceedings of the SPIE*, 6505:65051G, 2007.

[6] Mo Chen, Jessica Fridrich, Jan Lukáš, and Miroslav Goljan. Imaging sensor noise as digital X-ray for revealing forgeries. In *9th International Workshop on Information Hiding*, Saint Malo, France, 2007.

[7] G. de Haan and E.B. Bellers. Deinterlacing–an overview. *Proceedings of the IEEE*, 86(9):1839–1857, 1998.

[8] Jed Deame. Motion compensated de-interlacing: the key to the digital video transition. In *SMPTE 141st Technical Conference*, New York, 1999.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 99(1):1–38, 1977.

[10] H. Farid. A survey of image forgery detection. *IEEE Signal Processing Magazine*, 2(26):16–25, 2009.

[11] H. Farid and E.P. Simoncelli. Differentiation of discrete multi-dimensional signals. *IEEE Transactions on Image Processing*, 13(4):496–508, 2004.

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

[13] Jessica Fridrich, David Soukal, and Jan Lukáš. Detection of copy-move forgery in digital images. In *Proceedings of DFRWS*, 2003.

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

[15] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *European Conference on Computer Vision*, pages 579–587, 1992.

[16] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, second edition, 2004.

[18] J. He, Z. Lin, L. Wang, and X. Tang. Detecting doctored JPEG images via DCT coefficient analysis. In *European Conference on Computer Vision*, Graz, Austria, 2006.

[19] B.K.P. Horn. *Robot Vision.* MIT Press, Cambridge, MA, 1986.

[20] Chih-Chung Hsu, Tzu-Yi Hung, Chia-Wen Lin, and Chiou-Ting Hsu. Video forgery detection using correlation of noise residue. In *IEEE International Workshop on Multimedia Signal Processing*, Cairns, Australia, 2008.

[21] Y.-F. Hsu and S.-F. Chang. Image splicing detection using camera response function consistency and automatic segmentation. In *International Conference on Multimedia and Expo*, Beijing, China, 2007.

[22] T. S. Huang and O. D. Faugeras. Some properties of the E matrix in two-view motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1310–1312, 1989.

[23] M.K. Johnson and H. Farid. Exposing digital forgeries by detecting inconsistencies in lighting. In *ACM Multimedia and Security Workshop*, New York, NY, 2005.

[24] M.K. Johnson and H. Farid. Metric measurements on a plane from a single image. Technical Report TR2006-579, Department of Computer Science, Dartmouth College, 2006.

[25] M.K. Johnson and H. Farid. Detecting photographic composites of people. In *6th International Workshop on Digital Watermarking*, Guangzhou, China, 2007.

[26] M.K. Johnson and H. Farid. Exposing digital forgeries in complex lighting environments. *IEEE Transactions on Information Forensics and Security*, 3(2):450–461, 2007.

[27] M.K. Johnson and H. Farid. Exposing digital forgeries through specular highlights on the eye. In *9th International Workshop on Information Hiding*, Saint Malo, France, 2007.

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

[29] Kenji Kurosawa, Kenro Kuroki, and Naoki Saitoh. CCD fingerprint method - identification of a video camera from videotaped images. In *IEEE International Conference on Image Processing*, Kobe, Japan, 1999.

[30] 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*, pages 453–456, March 1992.

[31] Guohui Li, Qiong Wu, Dan Tu, and Shaojie Sun. A sorted neighborhood approach for detecting duplicated regions in image forgeries based on DWT and SVD. In *IEEE International Conference on Multimedia and Expo*, pages 1750–1753, Beijing, China, 2007.

[32] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(10):133–135, 1981.

[33] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[34] Jan Lukáš and Jessica Fridrich. Estimation of primary quantization matrix in double compressed jpeg images. In *Proceedings of DFRWS*, Cleveland, OH, 2003.

[35] Jan Lukáš, Jessica Fridrich, and Miroslav Goljan. Detecting digital image forgeries using sensor pattern noise. In *Proceedings of the SPIE*, volume 6072, 2006.

[36] Weiqi Luo, Jiwu Huang, and Guoping Qiu. Robust detection of region-duplication forgery in digital image. In *International Conference on Pattern Recognition*, pages 746–749, Washington, D.C., 2006.

[37] K. Mayer-Patel, B.C. Smith, and L.A. Rowe. The Berkeley software MPEG-1 video decoder. In *ACM International Conference on Multimedia*, New York, NY, 2005.

[38] P. R. S. Mendonça and R. Cipolla. A simple technique for self-calibration. In *Computer Vision and Pattern Recognition*, pages 500–505, Fort Collins, Colorado, 1999.

[39] N. Mondaini, Roberto Caldelli, Alessandro Piva, Mauro Barni, and Vito Cappellini. Detection of malevolent changes in digital video for forensic applications. *Security, Steganography, and Watermarking of Multimedia Contents IX. Proceedings of the SPIE*, 6505:65050T, 2007.

[40] T.T. Ng and S. F. Chang. A model for image splicing. In *IEEE International Conference on Image Processing*, Singapore, October 2004.

[41] K. Ouyang, S. Li G. Shen, and M. Gu. Advanced motion search and adaptation techniques for deinterlacing. In *IEEE International Conference on Multimedia and Expo*, pages 374–377, 2005.

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

[43] A.C. Popescu and H. Farid. Statistical tools for digital forensics. In *6th International Workshop on Information Hiding*, Toronto, Cananda, 2004.

[44] A.C. Popescu and H. Farid. Exposing digital forgeries by detecting traces of re-sampling. *IEEE Transactions on Signal Processing*, 53(2):758–767, 2005.

[45] A.C. Popescu and H. Farid. Exposing digital forgeries in color filter array interpolated images. *IEEE Transactions on Signal Processing*, 53(10):3948–3959, 2005.

[46] Jianbo Shi and Carlo Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[47] T. Sikora. MPEG-1 and MPEG-2 digital video coding standards. In R.K. Jurgen, editor, *Digital Consumer Electronics Handbook*. McGraw-Hill, 1997.

[48] E.P. Simoncelli. Bayesian multi-scale differential optical flow. In B. Jahne, H. Haussecker, and P. Geissler, editors, *Handbook of Computer Vision and Applications*, pages 397–420. Academic Press, 1999.

[49] Sugiyama and Nakamura. A method of de-interlacing with motion compensated interpolation. *IEEE Transactions on Consumer Electronics*, 45(3):611–616, 1999.

[50] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical Report CMU-CS-91-132, Carnegie Mellon University, 1991.

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

[52] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.