Real-Time Detection of Camera Tampering

Evan Ribnick, Stefan Atev, Osama Masoud, Nikolaos Papanikolopoulos, and Richard Voyles
Artificial Intelligence, Robotics, and Vision Laboratory
University of Minnesota, Minneapolis, MN 55455
{ribnick, atev, masoud, npapas, voyles}@cs.umn.edu

Abstract

This paper presents a novel technique for camera tampering detection. It is implemented in real-time and was developed for use in surveillance and security applications. This method identifies camera tampering by detecting large differences between older frames of video and more recent frames. A buffer of incoming video frames is kept and three different measures of image dissimilarity are used to compare the frames. After normalization, a set of conditions is tested to decide if camera tampering has occurred. The effects of adjusting the internal parameters of the algorithm are examined. The performance of this method is shown to be extremely favorable in real-world settings.

1. Introduction

Busy public places such as malls, airports, and public transportation stations often have significantly large numbers of surveillance cameras installed. Simultaneously monitoring this many live camera feeds for an extended period of time is a daunting task, even for the most alert security personnel. It is clear that any automation that is introduced could greatly improve the effectiveness of a surveillance system working under these circumstances. Detection of camera tampering is an important problem in such situations. If the tampering is intentional, it could be indicative of a more profound suspicious activity to which security personnel should be alerted. If unintentional, the tampering should still be noted since it may reduce the surveillance capabilities of the camera.

Detecting camera tampering can be a deceptively easy problem. First, the exact definition of what constitutes camera tampering must be established. For this work, camera tampering is defined as any sustained event which dramatically alters the image seen by the camera. Some examples of camera tampering are a person holding his/her hand in front of the camera, spray painting the lens, or turning the camera so that it points in a different direction. Such an event must be sustained for several seconds in order to be detected. The camera location is an important consideration since there may be events that are harmless and expected but still cause large changes in the image, and these should be distinguished from real tampering events. For example, for cameras mounted in a train station, the system should be insensitive to the motion of trains and large crowds of people in the scene.

The approach used here can be summarized as follows. First, incoming frames of video are stored in buffers. The computation is done in two major stages:

1. Recent frames of video are compared to older frames using several measures of image dissimilarity.
2. Based on these measures, a set of rules is evaluated to decide if camera tampering has occurred.

This approach is based on the idea that, in general, camera tampering will cause the most recent frames of video to be significantly different than the older frames. Since we are only interested in detecting sustained events (as opposed to transient events), it is necessary to compare the recent and old frames in a more robust way than simple frame-to-frame comparisons. This will be discussed in Section 3. The rule base was developed as a result of significant experimentation, and has thresholds which can be adjusted by the user based on the specific environment. The underlying idea here is that if results based on several different measures of image dissimilarity (i.e., image features) are compared, real tampering events can be identified robustly. This is of course conditioned on the fact the image dissimilarity measures are chosen well. The method presented here yields very favorable results when detecting actual camera tampering events, and is insensitive to changes in illumination and other non-tampering events.

2. Related work

Much research has been done regarding detection of tampering with or modification of prerecorded video. For example, [1], [6], and [7] deal with data embedding and watermarking techniques in order to classify video as either authentic or tampered. While these contributions are significant, they are not directly related to this problem, since our goal is to create a system that monitors live video and detects physical camera tampering in real-time.

Another area of research in computer vision in which there has been an immense amount of research is motion estimation. In [8], a factorization method is used in order to estimate structure and motion. In [9], a linear least-squares minimization technique is employed to achieve the same objective. These are two of the fundamental papers in the area. These and similar techniques have potential usefulness in estimating the camera motion for tampering events where a person physically turns the camera to point in a different direction. However, both of these works assume a rigid environment, and thus would be of only limited utility in the current application. Furthermore, in this application, estimating the motion of the camera is not as important as simply recognizing that camera motion has occurred and issuing an alarm accordingly.

In order to detect other types of camera tampering, such as a person covering the lens of the camera with a hand or other object, there are several existing algorithms which might be employed, including much of the work related to scene change and shot detection. For example, the work in [5] focuses on detecting gradual scene changes in movies, such as fade in/fade out, dissolve, and wipe using B-spline interpolation curve fitting. However, this work assumes smooth transitions, which may not be the case with camera tampering. In [2], the authors model each shot as a set of edges in the image, and take a scene/shot change to be anytime when there is a significant change in these edges throughout the image. This may not be the case with camera tampering, since a hand entering the scene and eventually covering the lens would not constitute smooth transition of edges from one shot fading into the edges of another.

The authors of [10] attempt to detect scene changes by optimizing a color-based shot detection algorithm and discuss with use of color histograms to detect significant changes in the scene. Similarly, [3] uses both histogram-based and pixel-based features to measure image dissimilarity, and also to classify transitions by type. Histogram-based methods in general are well suited for detection of camera tampering, since they offer global measures of image content and are sensitive to most types of significant changes in the image regardless of their nature or cause.

There are, however, some fundamental differences between camera tampering detection and shot detection. By definition, shot changes are expected to occur frequently in a movie. In addition, shot detection algorithms are usually expected to be relatively insensitive to smooth camera rotation. Also, shot/scene changes are often modeled as rapid or transient transitions, and thus image features can be compared by looking only at consecutive frames of video.

On the other hand, camera tampering detection algorithms should be sensitive to any significant camera motion. The number of false detection events should be minimized, since this system will trigger an alarm that attracts the attention of security personnel each time an event is detected (whereas in shot detection it may be prudent to err in the opposite direction). Finally, perhaps the most important difference is that camera tampering is a sustained event (which can be either rapid or smooth), and thus more sophisticated and robust comparisons need to be made between frames of video in order that the module will be insensitive to transient changes which should not be considered tampering. These ideas will be elaborated on in the following section.

3. Approach

The method described here is based on the principle that when camera tampering occurs, the most recent frames of video will be significantly different than the older frames. As live video is received by the program, it is stored in two different buffers. The first buffer is labeled as the short-term pool, and stores frames that are less than 10-50 seconds old (depending on the setting chosen by the user). The second buffer is the long-term pool, which stores frames until they are an additional two minutes old.

Both of these are first in-first out (FIFO) structures, and frames that are evicted from the short-term pool (once they become older than the user-defined time

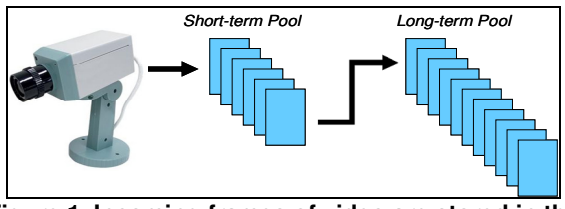

Figure 1. Incoming frames of video are stored in the short-term and long-term pools.

threshold) are inserted into the long-term pool. Frames that are evicted from the long-term pool are no longer stored. An illustration of this structure can be seen in Figure 1. For all of the subsequent analysis, the length of the short-term pool was set to ten seconds (sampled at 3 frames per second), which was found to provide a good balance between timely event detection, computational burden, and insensitivity to transient events. The long-term pool held about sixty video frames equally spaced in time to cover a period of two minutes.

3.1. Comparing the new frames with the old

Each time a new frame is pushed into the short-term pool, the short-term and long-term pools must be compared in order to determine if camera tampering has occurred. The approach presented here uses three different measures of image dissimilarity to compare the frames. However, in order to simplify introduction of the general concepts of the algorithm, before elaborating on the actual measures used, a generic measure of image dissimilarity will be assumed, and the process is described just for this generic measure. Following this description of the procedure, the actual measures of image dissimilarity will be introduced and the process will be specialized to use these.

Using a generic measure of image dissimilarity, every frame in the short-term pool is compared to every frame in the long-term pool. The median of all these measurements is taken and will be called $D_{between}$, since it represents the difference between the two pools. Similarly, the median image difference between all pairs of long-term frames is calculated, and denoted D_{long}.

The median is used to represent the overall image dissimilarity of multiple frames of video because it is a robust statistic (a stable indicator of central tendency). It is not affected by a relatively large fraction of outliers, (i.e., transient events in our application). Consider what happens when tampering occurs. As new frames of video are received, one would expect the median of the intra-short-term pool measurements to be affected as soon as approximately half of the short-term frames are contaminated by the tampering event. If the length of the short-term pool is set to ten seconds, as was done here, then it would take approximately five seconds for the median to be effected by the camera tampering. In this way sustained events are distinguished from transient ones.

The two measurements are combined into a single normalized measure as shown below:

$$D_{norm} = \log\left(D_{between} / D_{long}\right). \quad (1)$$

The normalization above was chosen because it accentuates differences between the short-term and long-term pools, which is desirable in that these differences are assumed to indicate camera tampering. It is reminiscent of the Fisher Linear Discriminant method, where the between-class scatter is normalized by the sum of intra-class scatters in order to maximize the distance between the classes while minimizing the intra-class variance [4]. If the long- and short-term pools are thought of as separate classes, then it is desirable to maximize the between-class scatter when tampering occurs in order to emphasize the event. Furthermore, the long-term pool contains a history of what has happened in the past, and thus normalizing by it yields a measurement that represents the difference between the current and past frames with respect to past (typical) differences.

3.2. Measuring image dissimilarity

In the preceding subsection a generic measure of image dissimilarity was assumed for simplicity of explanation. In reality, the process described above is repeated each time a new frame is sampled using three different image dissimilarity functions. The ideal image dissimilarity measure is one that is insensitive to small camera rotations and translations, but is sensitive to changes in both shape and color content. It should also be illumination invariant, but not completely insensitive to extreme changes in illumination, since extreme changes may represent tampering. Since it would be difficult to define one image dissimilarity measure that exhibited all these characteristics, a variety of measures are used here in order to extract the most useful information. They are described below, and values of these parameters from one of the test videos are shown in Section 4.

3.2.1. Histogram chromaticity difference.

For each image (i.e., frame of video), a normalized RGB histogram is calculated. This is a two-dimensional histogram, where the two axes represent the normalized red and green components of each pixel, respectively (the blue component in normalized RGB representation is uniquely determined by the other two and is therefore ignored). The bin assignment for pixel i with components R_i, G_i, and B_i is calculated according to Equation (2).

To calculate the dissimilarity between two images, the sum of absolute-value differences of their two

histograms is computed. For images I_1 and I_2 with histograms H_1 and H_2, respectively, the image dissimilarity is calculated according to:

$$Bin_R = R_i NumBins / (R_i + G_i + B_{i\,R}) \qquad (2)$$
$$Bin_G = G_i NumBins_G / (R_i + G_i + B_i)$$

$$Difference_{12} = \sum_{i,j} |H_1(i,j) - H_2(i,j)|. \qquad (3)$$

The normalized RGB histogram is fairly insensitive to illumination changes, since the normalized RGB color space is illumination-invariant for Lambertian surfaces, and therefore the measure is sensitive only to color change. A measure of image dissimilarity that is even less sensitive to illumination changes could be formulated, but the video coming from CCTV surveillance cameras is often of poor color quality due to the limited chrominance bandwidth, and the cameras themselves are rarely color-calibrated. In such cases, the computation of more sophisticated photometric invariants based on color is not reliable.

3.2.2. Histogram L1R difference.

A two-dimensional histogram is calculated for each image where, for each pixel, the axes are the sum (L1-norm) and range (R) of the red, green, and blue components of the pixel. The bin assignments for pixel *i* are calculated according to the following equations:

$$Bin_{L1} = (R_i + G_i + B_i) NumBins_{L1} / 3$$
$$Bin_R = (\max(R_i, G_i, B_i) - \min(R_i, G_i, B_i)) NumBins_R. \qquad \textbf{(4)}$$

Again, the dissimilarity between two images is calculated by summing the absolute-values of differences between the histograms of the images.

The L1-norm value for any given pixel is proportional to its intensity, while the range is related to its saturation. This measure is sensitive to changes in illumination, which should be taken into account. Even though some change in illumination should be ignored, more extreme changes should be recognized and considered tampering. It has also been established that the L1-norm is a better similarity measure than the L2-norm [3].

3.2.3. Histogram gradient direction difference.

Each image is first convolved with the 3x3 Sobel kernels. These convolved images are used to estimate the gradient direction $Dir(i,j)$ at each pixel location (i,j), where $Dir(i,j)$ is in the range $[-\pi, \pi]$. Next, a one-dimensional histogram is formed, where the axis represents the gradient direction of a given pixel. The bin assignment for pixel (i,j) is calculated using the following equation:

$$Bin_{GradDir} = 1/\pi \left(Dir(i,j) + \pi/2\right) NumBins_{GradDir}. \qquad (5)$$

The dissimilarity between two images is given by the sum of absolute-vale differences between the two histograms. Since the pixel gradient orientations are insensitive to both change in illumination and slight camera movements, this measure is extremely robust.

3.3. Detecting camera tampering

Each time a new frame is pushed into the short-term pool, the three dissimilarity measures described above are computed. Based on these three values the decision is made as to whether tampering has occurred by evaluating a set of conditions, each of which is given by a set of three thresholds. For each condition, if the three thresholds are exceeded, the condition is evaluated as true. If any of the conditions are true, the decision is made that tampering has occurred. For all the results presented in this paper, the thresholds were tuned for optimal performance based on a set of training videos.

4. Results

This algorithm was tested on an extensive set of test videos totaling over 19 hours in duration. The videos were filmed in a variety of different environments, including various lighting conditions. They contained a number of different camera tampering and non-tampering events. In order to test the robustness of the proposed method, the test videos contained events that changed the scene significantly but are not considered camera tampering. The videos contained such events as large crowds of people entering/exiting the scene, very large objects (i.e., trains, buses) passing through the scene, and extreme changes in illumination, both sudden and gradual. Figure 2 shows some snapshots from the test videos, some with tampering.

Figure 3 shows the values of the three normalized image dissimilarity measures for one of the test videos. Note that these measures are responsive to camera tampering, as shown by their increased values during tampering events. Figure 4 shows normalized histograms of the three dissimilarity measures for both frames of video with and without tampering. The histogram values were normalized by the total number of frames in the class (tampering and non-tampering

frames were normalized separately) in order to equalize the overwhelming number of frames without tampering as compared to those with tampering. From this figure, it can clearly be seen that for all three image features, there is good separation between the tampering and non-tampering frames. This is also seen in Figure 5, which shows the scatter plot of the three feature values. Even though there is some overlap between the two classes, most tampering events are detected successfully since each tampering event is a set of the red tampering points in the figure, and only one of them needs to be in the decision region in order for the event to be detected.

Finally, the results for various different values of thresholds are shown in Table 1. The column on the left indicates the threshold values as a percentage of the optimal values. For each set of thresholds the number of missed events, number of false alarms, percent alarms true (PAT), and percent events detected (PED) are shown. Notice that as the threshold values increase, the number of false alarms decreases, but the number of undetected tampering events increases. When the thresholds are optimal (the row labeled 100% in the table), the tradeoff between the number of false alarms and the number of missed tampering events is optimized (roughly speaking, we consider a missed alarm to be three times as important as a false one). At the optimal threshold values, there were five false alarms and one tampering event that was not detected. However, both the events that were not detected and those that were detected erroneously were borderline tampering events. For example, four of the five false alarms were due to extreme changes in illumination (a light being flipped off), where the entire image becomes dark for several frames until the camera aperture size is adjusted. Events such as large crowds entering the scene, severe changes in illumination due to dark passing clouds, and trains and buses passing through the scene were successfully classified as non-tampering.

Figure 2. Some frames from the test videos. The last three snapshots include camera tampering.

Table 1. Results for different threshold values, given as percentages of the optimal values.

Thresholds	False Alarms*	Missed Events*	PAT	PED
75%	23	0	46.8%	100%
80%	14	0	58.8%	100%
85%	12	0	62.5%	100%
90%	10	0	66.7%	100%
95%	7	0	74.1%	100%
100%	5	1	79.2%	95%
105%	4	3	81.0%	85%
110%	4	3	81.0%	85%
115%	4	4	80.0%	80%
120%	4	4	80.0%	80%
125%	4	4	80.0%	80%
130%	4	5	79.0%	75%
135%	4	6	77.8%	70%
140%	4	6	77.8%	70%
145%	4	7	76.5%	65%
150%	4	7	76.5%	65%
* Total 20 real tampering events				

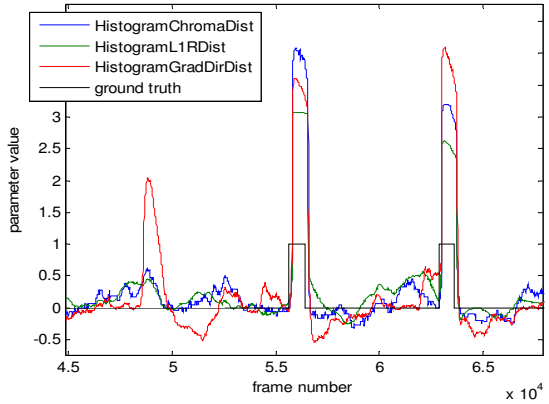

Figure 3. Values of the three normalized image dissimilarity measures. The black line indicates when tampering occurs.

5. Conclusion

We have presented a technique for detection of camera tampering that can be implemented in real-time. The method is based on robust comparisons of recent and older frames of video using three different measures of image dissimilarity. Optimal threshold values were used in order to decide whether tampering has occurred. This method is insensitive to changes in illumination, small camera movements, and transient non-tampering events. Experimental results have been presented which demonstrate a high detection rate while at the same time issuing a very small number of false alarms.

6. Acknowledgements

We would like to thank the Department of Homeland Security, the ITS Institute at the University of Minnesota, and the National Science Foundation (through grant IIS-0219863) for their generous support of this research.

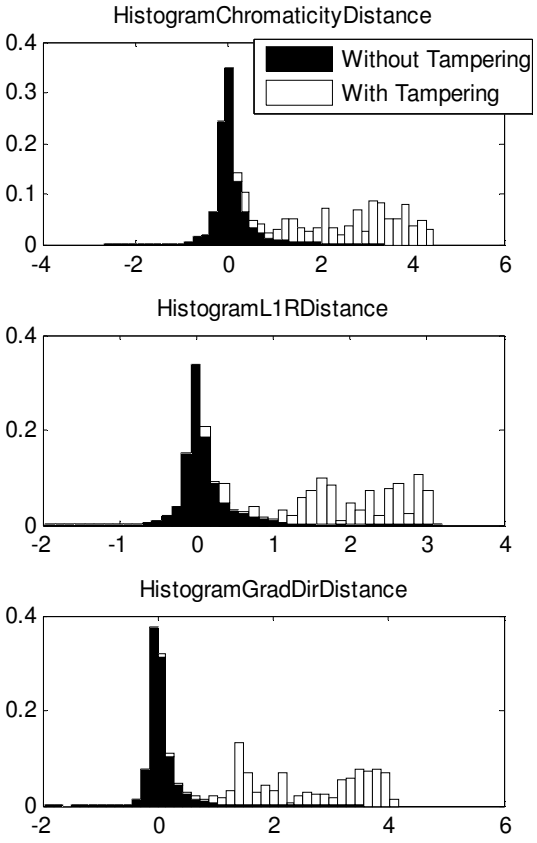

Figure 4. Normalized histograms of image dissimilarity values with and without tampering.

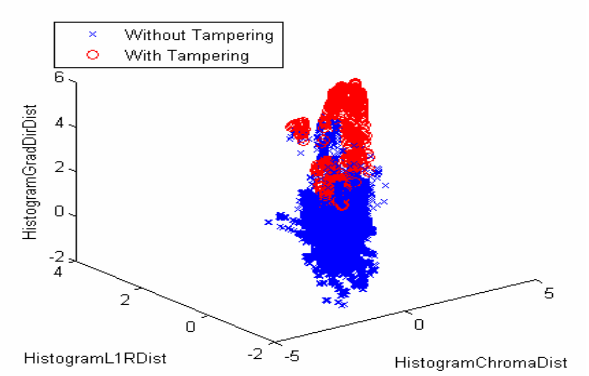

Figure 5. Scatter-plot of image dissimilarity values for frames with and without tampering.

References

[1] Fridrich, J., "Image Watermarking for Tamper Detection", *Proc. of ICIP 1998*, vol. 2, pp. 404-408, 1998.

[2] Heng, W.J. and Ngan, K.N., "Integrated Shot Boundary Detection Using Object-Based Technique", *Proc. of the Int'l Conf. on Image Processing 1999*, vol. 3, pp. 289-293, October 1999.

[3] Huang, C.-L. and Liao, B.-Y., "A Robust Scene-Change Detection Method for Video Segmentation", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, pp. 1281-1288, December 2001.

[4] Muller, K.L. et al. "An Introduction to Kernel-Based Learning Algorithms", *IEEE Trans. on Neural Networks*, vol. 12, no. 2, pp. 181-201, March 2001.

[5] Nam, J. and Tewfik, A.H., "Detection of Gradual Transitions in Video Sequences Using B-Spline Interpolation", *IEEE Trans. on Multimedia,* vol. 7, no. 4, pp. 667-679, August 2005.

[6] Roberts, D.K., "Security Camera Video Authentication", *Digital Signal Processing Workshop 2002 and the Signal Processing Education Workshop 2002,* pp. 125-130, October 2002.

[7] Swanson, M.D. and Tewfik, A.H., "Multimedia Data-Embedding and Watermarking Technologies", *Proc. of IEEE*, vol. 86, no .6, pp. 1064-1087, June 1998.

[8] Tomasi, C. and Kanade, T., "Shape and Motion from Image Streams Under Orthography: a Factorization Method", *IJCP,* vol. 9, no. 2, pp. 137-154, 1992.

[9] Weng, J. et al. "Optimal Motion and Structure Estimation", *IEEE Trans. on PAMI,* vol. 15, no. 9, pp. 864-884, September 1993.

[10] Zhao, W. et al. "Improving Color Based Video Shot Detection", *IEEE Int'l Conference on Multimedia Computing Systems,* vol.2, pp.752-756, June 1999.