DETECTION OF TEMPORAL INTERPOLATION IN VIDEO SEQUENCES

P. Bestagini, S. Battaglia, S. Milani, M. Tagliasacchi, S. Tubaro

Dipartimento di Elettronica ed Informazione, Politecnico di Milano
piazza Leonardo da Vinci 32
20133 Milano, Italy

bestagini/milani/tagliasa/tubaro@elet.polimi.it

ABSTRACT

Nowadays, considering the availability of relatively cheap devices and powerful editing software, video tampering is a relatively easy task. Video sequences can be tampered with by performing, e.g., temporal splicing. However, if the sequences spliced together do not share the same frame rate, they have to be temporally interpolated beforehand. This operation is often made using motion compensated interpolators, which allow to minimize visual artifacts. In this paper we propose a detector of this kind of interpolation. Moreover, the detector is capable of identifying the interpolation factor used, allowing an analyst to uncover the original frame rate of a sequence. This method relies on the analysis of the correlation introduced by the filter adopted by the interpolator. Results show that detection is successful, provided that the number of observed interpolated frames is large enough. Moreover, tests on compressed sequences obtained from television broadcasts validate the method in a real world scenario.

Index Terms— video forensics, interpolation, motion compensation, motion vectors

1. INTRODUCTION

Thanks to the increasing availability of multimedia sharing platforms and user-friendly editing software, it is now easy to collect video sequences and tamper with them. A typical scenario consists in splicing together different sequences in order to make a compilation video. Another common situation is that of taking some objects from a sequence (e.g., some people) and pasting them into another one (e.g., depicting the desired background).

In the literature, many ways to identify such situations have been studied for still images [1, 2, 3, 4]. However, more recently, video forensics is becoming a field of major interest [5]. In the simplest case, image-based techniques are applied on a frame-by-frame basis [6]. More interestingly, the temporal dimension of the video sequences can be exploited by techniques specifically tailored to videos [7, 8].

When several original sequences are used to create a tampered sequence, it is often the case that they were originally acquired at different frame rates. This can be due either to the use of different acquisition devices or to settings used during video capturing. When different sequences of this kind are spliced together in order to create a realistic tampering, the frame rate of the whole sequence must be unified. This means that sequences with different frame rates must be temporally interpolated to obtain the desired frame rate.

Simple methods to achieve this goal involve frame repetition or frame averaging in time. However, such techniques result in visual artifacts (e.g., ghosting), especially in sequences with high motion. A common method to interpolate frames, while minimizing temporal artifacts, is to use motion-compensated interpolation. This technique consists in performing motion estimation before interpolating neighboring frames, so that new frames are obtained by filtering along the motion trajectories identified by motion vectors. Indeed, this method takes explicitly into account the temporal correlation between neighboring frames. However, performing such a filtering leaves characteristic footprints on the sequence itself. These footprints can then be exploited in order to reveal the use of interpolation.

In this work we focus on extracting these footprints from videos, leveraging the principle successfully adopted in the case of image (spatial) resampling. The goal is to design a detector capable of revealing the use of frame interpolation on a set of consecutive frames in a sequence, even when motion-compensation is applied before filtering. Moreover, when interpolation is detected, we estimate the interpolation factor used, thus inferring the original frame rate of a sequence. This detector can be used on a video sequence (or part of it) to verify if its frame rate has been changed along time. Changes in frame rate can then be used as evidence of video tampering.

The rest of the paper is organized as follows. In Section 2 we introduce the video interpolation problem and the used notation. In Section 3 we describe the proposed detector. In Section 4 we show the results obtained on a dataset composed by well known test sequences, which are interpolated adopting different algorithms. In addition, we also experimented with television broadcasts. Finally, in Section 5, we draw some conclusions and present the possible future works.

2. VIDEO INTERPOLATION

Let us consider an original video sequence $\mathbf{X}$, whose frames are denoted as $X(t),\ t = 1, 2, ..., T$. Let us now consider the case in which the frame rate of this sequence is changed, scaling the original rate by a factor ω. The resulting interpolated sequence is then $\mathbf{X}^{\omega}$, and its frames are denoted as $X^{\omega}(\omega t)$. If $\omega < 1$ the sequence is upsampled, and new frames are created at non integer values ωt. If $\omega > 1$ the sequence is downsampled, which is typically achieved by upsampling the sequence (if needed) and dropping a given number of frames. The interpolated sequence is composed by some frames belonging to the original one, and, possibly, some frames obtained interpolating the original ones. In the case of downsampling for $\omega = n, n \in \mathbf{N}$, the output sequence can be simply obtained by means of

The project REWIND acknowledges the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission, under FET-Open grant number:268478.

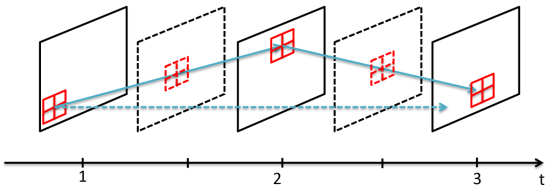

Fig. 1: Trajectory of interpolated pixels across frames when motion compensation is performed (continuous arrow) and when it is not (dashed arrow).

frame dropping.

Let us now consider the upsampling operation. In the simplest case, frames X^{ω} at non-integer positions ωt can be computed by interpolating $2K$ neighboring frames pixel-wise as

$$X^{\omega}_{ij}(\omega t) = \sum_{k=-K}^{K} h_k \cdot X^{0}_{ij}(\omega t + \omega k),$$

where h is the interpolation filter (i.e., a 1-dimensional low-pass filter), i and j denote the spatial pixel position in a frame, and $\mathbf{X}^0$ is the original sequence defined on the support of the interpolated one, such that

$$X^{0}(\omega t) = \begin{cases} X(\omega t) & \text{if } \omega t \text{ is an integer} \\ 0 & \text{otherwise} \end{cases}$$

However, in a more realistic scenario, interpolation among frames is not obtained applying such a simple procedure, but motion is taken into account to avoid annoying temporal artifacts due to moving objects (e.g., ghosting). In order to obtain frames X^{ω} at non-integer positions ωt, neighboring frames $X(t \pm i),\ i = 1, ..., I$, are motion-compensated before being interpolated. Figure 1 shows an example of motion compensated interpolation: when $\omega = 0.5$ dashed frames are the interpolated ones. If motion compensation is performed before interpolation, the pixels filtered to obtain the interpolated ones lie on the trajectory followed by the block and depicted by the solid arrow. On the other hand, if no motion-compensation is performed, the filtered pixels are those along the dashed arrow (all in the same spatial position for each time instant). When motion compensation is performed, the resulting sequence is

$$X^{\omega}_{ij}(\omega t) = \sum_{k=-K}^{K} h_k \cdot X^{0}_{m_{t,i,j} n_{t,i,j}}(\omega t + \omega k),$$

where now the spatial indexes $m_{t,i,j}$ and $n_{t,i,j}$ change in each frame for each pixel position (i, j), in order to follow motion estimated trajectories in time.

3. DETECTOR

When a signal is interpolated as described above, the filter introduces a strong correlation between the samples [9, 10]. In the video case, pixels in the same spatial position (or on the same motion trajectory when motion compensation is considered) but different time instants are correlated. As shown in [11], if we compute an estimation of each sample from its neighbors, we end up with two cases. If the sample we estimated was actually computed from its neighbors, they will be similar. If the sample we estimated was an original one in the sequence, the sample and its estimation will be quite different. Computing the error between our estimation and the interpolated sequence results then in a periodic signal.

As shown in [11], by using a predefined analysis filter h^*, one may compute the prediction error, which adapted to video is given by

$$\begin{aligned} e_{ij}(\omega t) &= X^{\omega}_{ij}(\omega t) - \hat{X}^{\omega}_{ij}(\omega t) \\ &= X^{\omega}_{ij}(\omega t) - \sum_{k=-K}^{K} h^*_k \cdot X^{\omega}_{ij}(\omega t + \omega k),\ h^*_0 = 0. \end{aligned} \tag{1}$$

It is possible to prove that the variance of this prediction error is periodic [12, 13, 14], and its periodicity depends only on the interpolation factor ω, and not on the analysis filter h^*. By exploiting this property, the interpolation factor can be directly inferred by looking at the error periodicity. Figure 2 shows the comparison between the error for an original and an interpolated sequence (top), and the Fourier transform of the errors (bottom). It is clear that the interpolated sequence shows a periodic artifact, easily detectable in the Fourier domain.

If we consider motion compensation, Equation 1 might lead to poor results, since it analyzes pixels along the wrong interpolation trajectories. In order to correct this behavior, we should estimate the interpolation trajectories beforehand. This can be done by performing motion estimation on the interpolated sequence. Even if the estimated motion vectors (MVs) do not coincide with those computed on the original sequence, they are likely to be very similar. In this way, an estimate of $m_{t,i,j}$ and $n_{t,i,j}$ for each frame can be obtained.

The detector we propose can be then summarized in the following steps:

- Compute $\hat{m}_{t,i,j}$ and $\hat{n}_{t,i,j}$ from MVs estimated by performing motion estimation between adjacent frames.
- Compute the prediction error as

  $$e_{ij}(\omega t) = X^{\omega}_{ij}(\omega t) - \sum_{k=-K}^{K} h^*_k \cdot X^{\omega}_{\hat{m}_{t,i,j}\hat{n}_{t,i,j}}(\omega t + \omega k),$$

  using the filter h^* whose only coefficients different from zero are $h^*_{\pm 1} = 0.5$.
- Compute the squared error for each frame as

  $$e(\omega t) = \sum_{ij} |e_{ij}(\omega t)|^2.$$

- Estimate the periodicity of $e(\omega t)$ in the frequency domain by searching for peaks in $|E(f)| = |\mathcal{F}(e(\omega t))|$, where $\mathcal{F}$ indicates the Fourier transform (see Fig. 2).

Once this periodicity is estimated, it can be directly related to an interpolation factor ω using the equation

$$\Delta f = 0.5 - |\omega - 0.5|,$$

where Δf is the position of the first peak in the normalized frequency domain (i.e., the inverse of the period of the error) [11]. This allows an analyst to assess if the sequence has been interpolated and estimate the original frame rate.

However, as noticed in [11], this method might not uniquely identify the used interpolation factor. In particular, when downsampling, or upsampling by a factor less than 1/2 are applied, the detector is subject to aliasing. Indeed, periodic artifacts for sequences resampled with these ω values are coincident with those of other upsampled ones. This fact prevents the detector to estimate the correct interpolation factor in this situation.

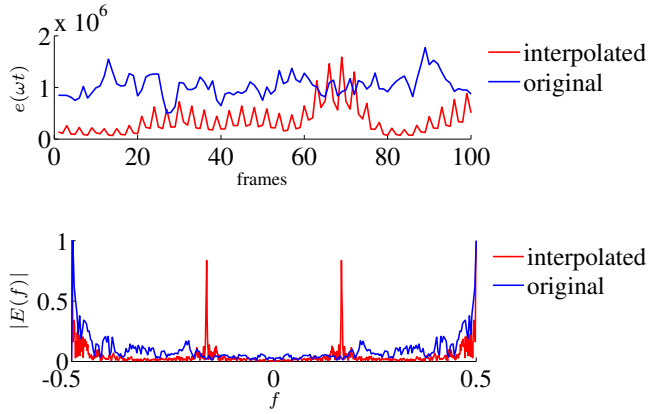

Fig. 2: Comparison of prediction error for the original and an interpolated version (with $\omega = 1/3$ from 30 to 90 frames per second) of the sequence *Foreman*: $e(\omega t)$ for the original (blue) and interpolated (red) sequences (top); $|E(f)|$ for the original (blue) and interpolated (red) sequences (bottom).

Table 1: Relationship between frame rate (in frames per second - FPS) and resampling factor ω when the original sequence is acquired at 30 FPS.

FPS	7.5	10	15	25	36	45	60	90	120	150
ω	4	3	2	6/5	5/6	2/3	1/2	1/3	1/4	1/5

4. EXPERIMENTAL RESULTS

The proposed detector was tested on three sequences (namely, *Foreman*, *Hall*, and *Mobile* at CIF spatial resolution) of 300 frames each. All these sequences are uncompressed and their original frame rate is equal to 30 Frames Per Second (FPS). Four different motion-compensated interpolators were tested (ISTWZCodec [15], Medianet [16], and the two freeware tools, namely MSU and MVTools2). Every sequence was temporally resampled using the factors in Table 1, which also reports the target frame-rate. Table 2 shows the values of ω used for each interpolator, and the probability of correct identification averaged on all the sequences when all the frames are used. The notation "1*" indicates that, instead of the correct resampling factor, the aliased version was found. We do not report results on non-interpolated sequences, since on the pool of 10 uncompressed CIF sequences we tested, the detector always correctly identified the sequences as non-interpolated.

Results for these sequences can be clustered in three classes: i) $\omega \leq 1/2$; ii) $1/2 < \omega < 2$; iii) $\omega \geq 2$. For the first class (upsampling with $\omega \leq 1/2$) the resampling factor is always correctly estimated. For the second class, when the sequence is upsampled or downsampled with ω up to 6/5, the estimated ω is always confused with the aliased version, as expected. For downsampling with integer $\omega \geq 2$ the detector does not work. However this is an expected behavior. When the sequence is downsampled by an integer factor, no interpolation is performed, but the only operation performed is frame dropping. This means that no filtering operation is involved, thus the detector fails.

It is interesting to analyze how the detection accuracy varies when changing the number of available frames. Figure 3 shows this analysis averaged on all the sequences and detectors when we do not disambiguate between aliased interpolation factors. Results are averaged on all the tested sequences, and show that the correct resampling factor can be identified even if only a subset of frames is analyzed. As an example, for sequences interpolated with $\omega = 1/2$ (from 30 to 60 FPS), by analyzing only 56 consecutive frames (i.e., less than 1 second) the detector correctly identifies the resampling factor with probability 1. Moreover, this result does not depend on the portion of the sequence that is analyzed (e.g., beginning, central part, end). This allows an analyst to apply the detector locally on a long sequence, in order to identify possible FPS changes, as it happens if several video sequences are spliced together.

Another aspect that has been investigated is the performance of the detector using a smaller amount of pixels for each frame. Indeed, a possible attack consists in pasting an object from a video into another sequence with different FPS. In this situation, in order to detect the forgery, the detector should work on small spatial windows for each frame. For this purpose we tested the detector on *Foreman* using a square spatial window ranging from 2×2 to 288×288 pixels. Figure 4 shows the accuracy on interpolation detection when different interpolation factors are used, averaged on all the interpolators. We notice that for some interpolation factors, especially for the upsampling case, the detector seems to be very robust to spatial cropping. As an example, when the sequence is interpolated from 30 to 60 FPS, almost any size of the window can be used. At 90 FPS results are more accurate with a window of 200×200 pixels than with the maximum window size. However, this result is compatible with the detector. Indeed, if a big window covers an area such as a flat background where the effect of interpolation might be hardly perceptible, the detector could lead to a wrong result. On the other hand, using a smaller window, these flat areas can be discarded, and the analysis leads to the correct result.

In order to validate the detector in a real world scenario, an

Table 2: ω identification probability for different interpolators at different FPS. "1*" indicates that ω is aliased, while "-" means that the given interpolation factor could not be used or was not tested with the selected interpolator.

FPS	7.5	10	15	25	36	45	60	90	120	150
ISTWZCodec	-	-	-	-	-	-	1	1	1	-
MSU	-	-	-	-	-	-	1	1	1	-
Medianet	0	0	0	1*	1*	1*	1	1	1	1
MVTools2	0	0	0	1*	1*	1*	1	1	1	-

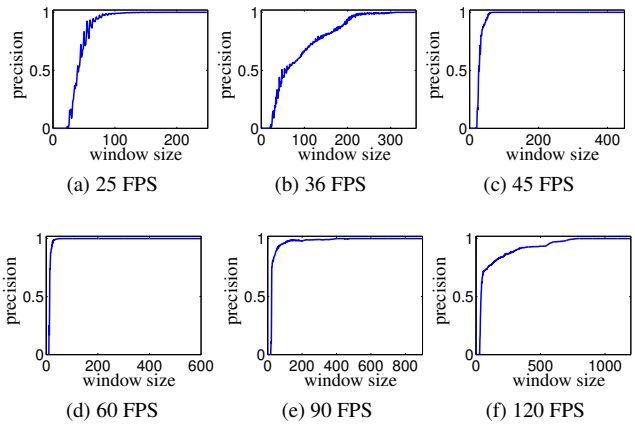

Fig. 3: Precision for different temporal window size (number of frames).

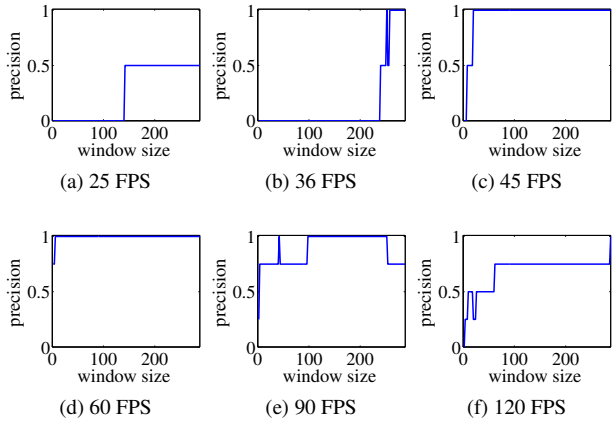

Fig. 4: Precision for different spatial window size (in pixels).

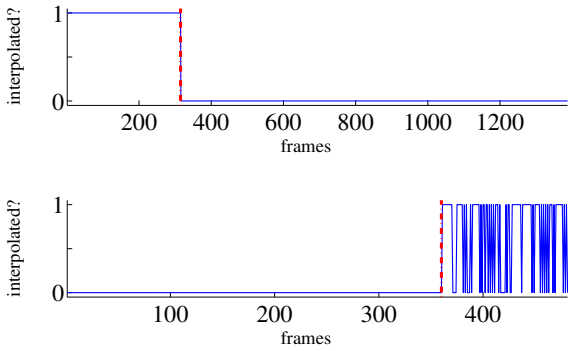

Fig. 5: Binary array for interpolation detector for H.264 (top) and MPEG-2 (bottom) sequences. The red dashed line shows the change point from interpolated to non-interpolated (or vice versa according to the analyzed sequence). The value 1 or 0 indicates that the algorithm has detected interpolation or not.

analysis on temporal windows has been applied to two sequences from television broadcasts. One was obtained directly from a DVB-T stream (encoded with MPEG-2), and the other from an online stream (encoded with MPEG-4 part10 H.264/AVC). Both sequences show sport events, and contain a part that is not interpolated (part of the match), and a part that is interpolated (part of a replay). The ground truth for the interpolation factor was extracted by comparing the length of the interpolated scene with the original one (found analyzing the overall stream). Figure 5 shows a binary array whose values are 0 or 1 according to frames being identified as interpolated or not. The value 1 means that the correct interpolation factor has been identified for the window centered at that frame. The value 0 is associated to frames not belonging to the interpolated part of the sequence. We notice that in the first sequence (top) the detector perfectly identifies which part of the sequence has been interpolated with the correct interpolation factor. In the second sequence (bottom), some interpolated frames are classified as non-interpolated. However it is still possible to discriminate which is the original part of the sequence. The decreased accuracy on the second sequence is probably due to the fact that the used MPEG-2 compression is more aggressive than H.264/AVC, which is used for the other sequence. This hides some interpolation footprints.

5. CONCLUSIONS

In this paper we presented a method to assess if a video sequence has been temporally interpolated, allowing also to estimate the original frame-rate. This detector is based on solutions adopted to detect spatial resampling in the case of still images. However it has been adapted to video, exploiting the temporal correlation between frames and allowing it to be used when motion compensation is applied before interpolation. The method achieves promising results, even when it is used on a subset of the frames. Moreover, it proves to be valid even on small spatial windows, which allows the detector to be used as a possible tool for copy and paste forgery attacks. Results on slightly compressed sequences from television broadcasts validate the detector even on a real world scenario. However, some open questions could still be answered. Indeed, an in depth analysis on the effect of compression on the resampling detector is under research. Moreover, possible anti-forensics methods could be studied to disguise this detector, hiding the footprints left by resampling.

6. REFERENCES

[1] H. Farid, "Exposing digital forgeries in scientific images," in *MM&Sec '06: Proceedings of the 8th workshop on Multimedia and security*, New York, NY, USA, 2006, pp. 29–36, ACM.

[2] T. Bianchi, A. De Rosa, and A. Piva, "Improved DCT coefficient analysis for forgery localization in JPEG images," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, may 2011, pp. 2444–2447.

[3] S. Milani, M. Tagliasacchi, and S. Tubaro, "Discriminating multiple jpeg compression using first digit features," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012, pp. 2253 –2256.

[4] M. Barni and A. Costanzo, "Dealing with uncertainty in image forensics: A fuzzy approach," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012, pp. 1753 –1756.

[5] S. Milani, M. Fontani, P. Bestagini, M. Barni, A. Piva, M. Tagliasacchi, and S. Tubaro, "An overview on video forensics," *APSIPA Transactions on Signal and Information Processing*, vol. 1, pp. e2, 2012.

[6] S. Milani, P. Bestagini, M. Tagliasacchi, and S. Tubaro, "Multiple compression detection for video sequences," in *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on*, sept. 2012, pp. 112 –117.

[7] P. Bestagini, A. Allam, S. Milani, M. Tagliasacchi, and S. Tubaro, "Video codec identification," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, march 2012, pp. 2257 –2260.

[8] M. Visentini-Scarzanella and P.L. Dragotti, "Video jitter analysis for automatic bootleg detection," in *Multimedia Signal Processing (MMSP), 2012 IEEE 14th International Workshop on*, sept. 2012, pp. 101 –106.

[9] A.C. Popescu and H. Farid, "Exposing digital forgeries by detecting traces of resampling," *Signal Processing, IEEE Transactions on*, vol. 53, no. 2, pp. 758 – 767, feb. 2005.

[10] D. Vazquez-Padin and P. Comesana, "ML estimation of the resampling factor," in *Information Forensics and Security (WIFS), 2012 IEEE International Workshop on*, dec 2012.

[11] M. Kirchner, "Fast and reliable resampling detection by spectral analysis of fixed linear predictor residue," in *Proceedings of the 10th ACM workshop on Multimedia and security*, New York, NY, USA, 2008, MM&Sec '08, pp. 11–20, ACM.

[12] A.C. Gallagher, "Detection of linear and cubic interpolation in jpeg compressed images," in *Computer and Robot Vision, 2005. Proceedings. The 2nd Canadian Conference on*, may 2005, pp. 65–72.

[13] B. Mahdian and S. Saic, "On periodic properties of interpolation and their application to image authentication," in *Proceedings of the Third International Symposium on Information Assurance and Security*, Washington, DC, USA, 2007, IAS '07, pp. 439–446, IEEE Computer Society.

[14] D. Vazquez-Padin and F. Perez-Gonzalez, "Prefilter design for forensic resampling estimation," in *Information Forensics and Security (WIFS), 2011 IEEE International Workshop on*, dec 2011, pp. 1–6.

[15] J. Ascenso, C. Brites, and F. Pereira, "Improving frame interpolation with spatial motion smoothing for pixel domain distributed video coding," in *5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Services*, 2005.

[16] L. Piccarreta, A. Sarti, and S. Tubaro, "An efficient video rendering system for real-time adaptive playout based on physical motion field estimation," in *Proc. 13th European Signal Processing Conference*, Antalya, September 2005, pp. 607–610.