Bit-Stream-Based Scrambling for Regions of Interest in H.264/AVC Videos With Drift Reduction

Andreas Unterweger[1], Jan De Cock[2] and Andreas Uhl[1]
[1] Department of Computer Sciences, University of Salzburg, Salzburg, Austria
e-mail: aunterweg@cosy.sbg.ac.at, uhl@cosy.sbg.ac.at
[2] ELIS – Multimedia Lab, Ghent University – iMinds, Ledeberg-Ghent, Belgium
e-mail: jan.decock@ugent.be

***Abstract*— We propose a new scrambling approach for regions of interest in compressed H.264/AVC bit streams. By scrambling at bit stream level and applying drift reduction techniques, we reduce the processing time by up to 45% compared to full re-encoding. Depending on the input video quality, our approach induces an overhead between -0.5 and 1.5% (high resolution sequences) and -0.5 and 3% (low resolution sequences), respectively, to reduce the drift outside the regions of interest. The quality degradation in these regions remains small in most cases, and moderate in a worst-case scenario with a high number of small regions of interest.**

***Keywords*— Region of interest, scrambling, H.264, AVC, video, bit stream, post-compression, drift, reduction**

I. Introduction

In surveillance videos, regions of interest (RoI) like people's faces are often scrambled in order to preserve their privacy. The images are not scrambled entirely so that people's actions can still be seen unscrambled by surveillance personnel to determine whether intervention is required. In addition, descrambling allows law enforcement to recover the identities later if necessary. This is not possible with other forms of de-identification like pixelation or blurring (e.g. [13], [6], [11]). An example of a surveillance system with RoI scrambling is depicted in Fig. 1.

In this paper, we assume a video surveillance scenario where a typical surveillance camera (as of 2016) outputs compressed videos in the form of Motion JPEG [9] or H.264/AVC bit streams [20]. We assume that either the camera or another component provides information on the location of the RoI through face detection to the scrambling system, e.g., as in [17] or [18], respectively.

Many approaches for RoI scrambling have been proposed. Most of them either perform scrambling format-independently in the image domain (e.g., [2], [3], [6]) or format-family-dependently in the transform domain (e.g., [12], [10], [16]). However, this is not practical since neither the captured images nor the encoder within the camera can be modified in typical surveillance equipment. The alternative of applying these methods by decoding the video stream received from the camera and re-encoding it is considered very time consuming and therefore often impractical.

Approaches for bit-stream-based RoI scrambling are very sparse. Although solutions for Motion JPEG exist [14], [21], [1], all of the H.264/AVC-focused approaches have severe disadvantages. Note that we only consider RoI scrambling approaches, i.e., those which aim at maintaining the visual information outside of the RoI.

Dufaux et al. [5] describe an approach for MPEG-4 Part 2 which can be extended to H.264/AVC (as well as other DCT-based formats). Although their scrambling algorithm can be performed at bit stream level, their method of preventing drift, i.e., the propagation of scrambled pixels into non-RoI areas, requires tracking block dependencies and selectively re-encoding the video from the first scrambled frame onwards. As explained above, this is impractical due to its high computational complexity.

The approaches of Iqbal et al. [8] and Unterweger et al. [17] require bit streams in which all RoIs are in separate slices or slice groups to prevent spatial drift through the imposed prediction borders. This is a serious restriction since it requires the camera to reliably detect RoI and to create according slices. Furthermore, the authors do not provide a solution for temporal drift, which is an important issue addressed in this paper.

Our work aims at reducing the amount of drift after scambling while significantly reducing the computational complexity required for processing. This way, we contribute an alternative to the infeasible full re-encoding techniques at the cost of a small amount of drift outside the RoI.

This paper is structured as follows: In Section II, we present our scrambling approach. In Section III, we describe how drift is reduced. Finally, we evaluate our proposed approach in terms of processing speed, space overhead and remaining amount of drift in Section IV before concluding the paper.

II. Scrambling and Descrambling Approach

This work focuses on RoI scrambling. The detection of the RoI is assumed to be done prior by other algorithms such as [19] or [15], and is therefore out of scope of this paper. For our scrambling approach, which is described in the following paragraphs, all RoI information is considered to be available from prior processing steps.

For all blocks inside a RoI, we perform a three-step bit-stream-level coefficient scrambling as follows: First, we change the signs of all AC coefficients by xor-ing the original sign bits with the output of a one-time pad. Second, we

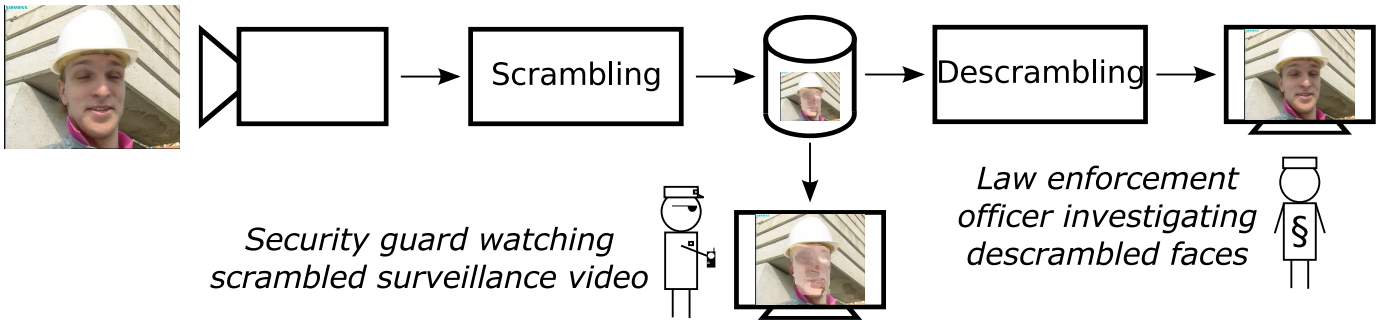

Fig. 1. Surveillance system use case: Images captured by a surveillance camera are scrambled and stored in a database. From there, the scrambled images can either be viewed directly or decrypted and used for legal investigations.

scramble the DC coefficient signs in the same way if the coefficients are stored directly in the bit stream, i.e., if the processed block does not use $16 \cdot 16$ intra prediction, where an additional Hadamard transform is used (which makes it impossible to change the DC coefficient signs at bit stream level). Third, we change the least significant bit of all AC coefficients (whose absolute value is greater than one) in order to additionally scramble blocks at RoI borders (e.g., face-to-background border) with larger coefficients. This can be done by entropy code word replacement at bit stream level. For descrambling, the same three steps are performed, with one caveat that is described in Section V.

We perform scrambling on all blocks which are not fully, but also partially in parts inside of a RoI. This assures that no information at the borders of RoI leaks at the cost of scrambling a small number of non-RoI pixels as well. We limit scrambling to the luminance channel since hardly any identification-related information is contained in the chrominance channels and extending our approach would be trivial.

When applying scrambling, spatial and temporal drift occur due to the difference between the scrambled and the unscrambled pixel values which are used for prediction. As illustrated in Fig. 2 (middle-left), this affects regions around the RoI (e.g., the collar in the top picture or the top-right part of the helmet in the bottom picture), disfiguring them to an extent that they cannot be used anymore for video surveillance and similar applications.

Also note the "position" of the face in the bottom picture which seems to be more similar to the first original frame (top left) than it is to the tenth (bottom left). This is due to the scrambled coefficients in the motion-compensated residuals, which induce an additional error that accumulates over time. Although temporal drift is less of an issue for RoI scrambling than spatial drift, both need to be dealt with to keep the areas outside the RoI viewable.

Thus, it is necessary to reduce drift. One way to do so is full re-encoding, offering perfect compensation (see Fig. 2, middle-right) at high computational complexity. Therefore, we subsequently propose an approach which aims at reducing drift at moderate computational complexity.

III. Drift Reduction

When scrambling macroblocks in the RoI, we have to take into account the dependencies that arise due to spatial and temporal prediction. Simply modifying the residual coefficients in a macroblock will affect the surrounding macroblocks due to (spatial) intra prediction, and due to (temporal) motion-compensated prediction.

When fully re-encoding the sequence, the original sequence is first decoded, and subsequently re-encoded using the same mode and motion information as in the input bitstream. In this case, the RoI scrambling is embedded after the second (encoding) loop. Since the reconstructed frames are available at the encoder loop, the compensation can be done without drift for the non-RoI area. The complexity of such an approach, however, will be high given the two prediction loops. We use this approach as a benchmark.

In the past, reduced-complexity transcoding approaches have been presented that were based on single-loop or open-loop architectures, mainly in the context of transrating [4]. Open-loop techniques can be used to perform modifications to coefficients with minimal complexity, but are unable to compensate for potential error drift. A *single-loop* approach, however, can compensate for the drift by storing the differences between the reconstructed coefficients before and after scrambling. These differences can afterwards be used for intra prediction (IP) and/or motion-compensated prediction (MCP) in dependent blocks.

In this paper, we introduce such a compensation loop in the scrambling framework, resulting in a significant reduction of the drift in the macroblocks which are dependent on the RoI area. In our framework, we make a distinction between three types of macroblocks:

- Macroblocks that are not inside the RoI or dependent on the RoI can be processed open-loop (i.e., by performing an entropy decoding and encoding step only), without further calculations.
- For the RoI macroblocks, the scrambling approach introduced in Section II is performed. Additionally, the differences between the (inverse quantized and transformed) coefficients before and after scrambling are calculated and

Fig. 2. First (top) and eleventh (bottom) frame of the *foreman* sequence (left-most) with QP 27 and $IP*$ GOP structure. The face as RoI is scrambled without drift compensation (middle-left), full compensation (middle-right) and our approach (right-most), respectively. Our approach shows only very little drift outside the RoI and is comparable to full compensation (middle-right).

accumulated, as illustrated in Fig. 3. The accumulation is performed along the direction of the intra prediction, or along the motion-compensated prediction direction. The resulting accumulated values are only stored within the RoI, but are not yet used for compensation. Hence, the coefficients for the RoI blocks are only modified by the scrambling process, with no impact on the bit rate.

- For the macroblocks that are directly dependent on the RoI (either through intra prediction or motion-compensated prediction), single-loop compensation is applied and the differences that were accumulated in the RoI are used to compensate for the error drift. This will lead to a small increase in the bits required to code these blocks, as discussed in Section IV-C. The single-loop compensation loop is illustrated in Fig. 4. By applying compensation, we notice that a significant portion of the drift is eliminated for the non-RoI area.

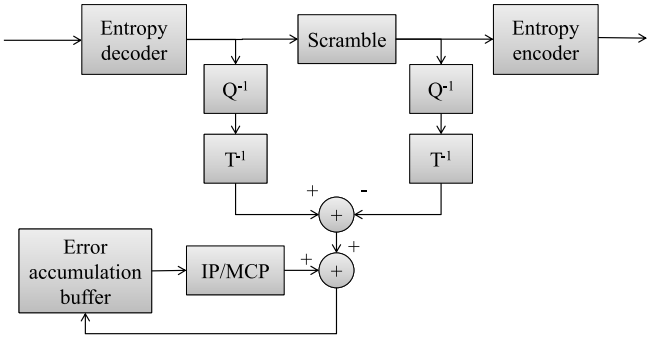

Fig. 3. Schematic transcoder architecture for RoI macroblocks (error accumulation): Differences between the scrambled and unscrambled values are accumulated in an error buffer. $Q(^{-1})$: (inv.) quantization, $T(^{-1})$: (inv.) transform, IP: intra prediction, MCP: motion-compensated prediction.

An overview of the error accumulation and compensation process for the second and third types of macroblocks is given in Fig. 5, for the case of spatial error propagation (intra prediction). The non-colored macroblocks correspond to the first type of macroblocks and can be processed open-loop.

It has to be noted that the drift compensation will not be perfect, due to non-linear operations in the H.264/AVC encoder and decoder loops. For example, the division/shift operations during intra prediction (modes 3-8) and motion compensation, along with clipping at the boundaries of the pixel value range (for 8 bits, between 0 and 255) can lead to rounding errors in the compensation loop. As such, we can still encounter drift errors (albeit to a smaller extent) outside the scrambled RoI.

IV. Evaluation

We implemented our proposed approach as described in sections II and III, and a full re-encoding variant, i.e., a method which decodes the bit stream to the pixel domain, fully compensates the drift therein and re-encodes the pictures with the same parameters, for comparison. In this section, we evaluate our approach in terms of processing time, remaining drift and space overhead.

For the evaluation, we use the *foreman* and *akiyo* sequences with one RoI each as simple test cases as well as the *crew* sequence with up to eleven RoIs as an advanced test case. All three sequences have CIF resolution ($352 \cdot 288$). In addition, we use the *Vidyo1* sequence ($1280 \cdot 720$) with three RoIs and the *Kimono* sequence ($1920 \cdot 1080$) with one RoI to show the impact of higher resolutions. The RoI are the faces of the depicted actors which were segmented manually. As explained above, this information can be easily augmented to be provided

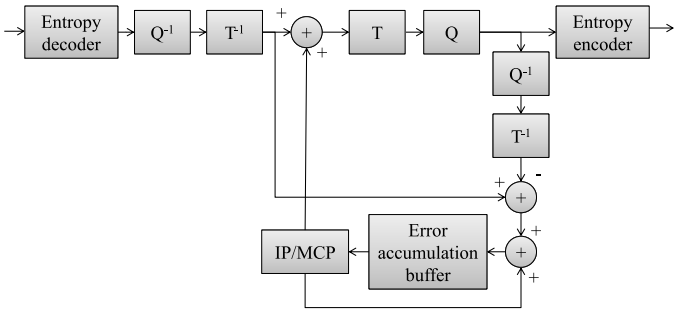

Fig. 4. Schematic transcoder architecture for macroblocks depending on the RoI (error compensation): The accumulated error is corrected approximatively by being considered in the prediction process. The abbreviations are the same as for Fig. 3.

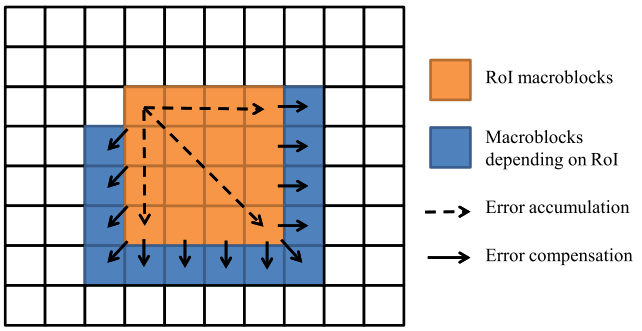

Fig. 5. Overview of error accumulation and compensation for intra prediction. Light (orange) blocks are processed as shown in Fig. 3, dark (blue) blocks as shown in Fig. 4.

by the surveillance system, together with the bit streams.

We used the H.264/AVC reference software (JM) to create Baseline profile bit streams of the sequences listed above. We used default settings and an $IPPP$ GOP structure, i.e., one I frame followed by 3 P frames, repeatedly, to simulate a typical video surveillance camera configuration.

A. Processing Time

To measure the transcoding time, i.e., the time for scrambling and drift reduction, we executed our software implementation three times for cache warming and five times for measuring the time between the entry and exit of the `main` function. The five measurements were averaged; fluctuations were insignificant at around 1%.

Fig. 6 depicts the transcoding time for one sequence per tested resolution and various QP. Our approach (black) is significantly faster than the implemented full re-encoding implementation (grey), saving between 30 and 45% of the execution time, the upper bound being achieved at high resolutions and QP. Transcoding time decreases with increasing QP in both implementations evenly.

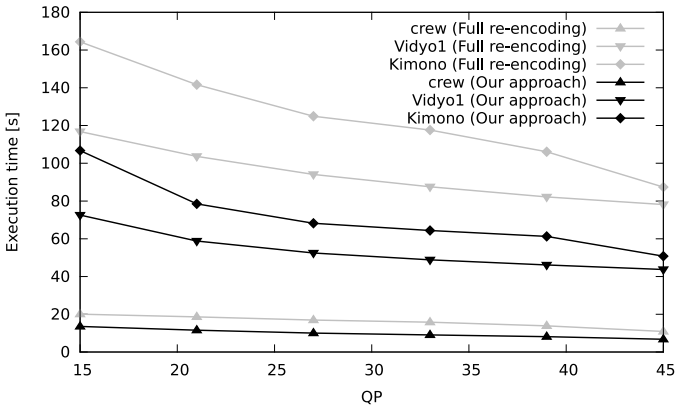

Fig. 6. Processing time of our approach (black) compared to full re-encoding (grey) for different QP: Our approach is between 30 and 45% faster.

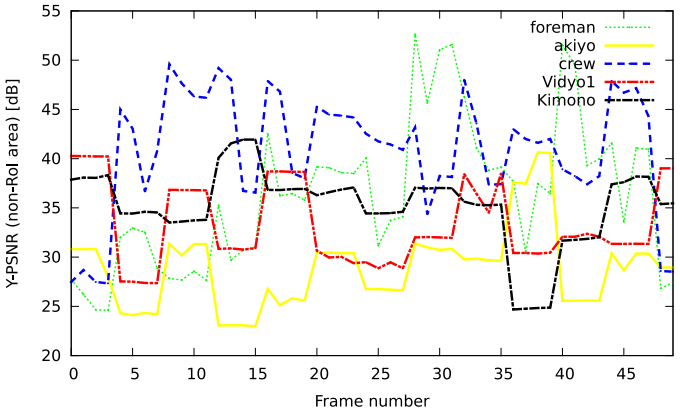

Fig. 7. Per-frame Y-PSNR outside the RoI for different sequences with QP 27 (first 50 frames): Spatial resolution has little to no effect on the quality degradation.

B. Remaining Drift

To quantify the remaining drift and its impact on video quality, we measured the Y-PSNR of all pixels outside the RoI with the respective unscrambled (compressed) sequence as reference. In the full re-encoding case, there would be no drift, i.e., a Y-PSNR of $+\infty$. We only depict the value of the first 50 frames for the sake of visualization. Fig. 7 illustrates the Y-PSNR values for all frames of the tested sequences with QP 27.

There are practically no differences between the CIF and high resolution sequences. The amount of drift largely depends on the amount of movement and changes at GOP borders, most notably in the first few GOPs of the *Vidyo1* sequence (dash-dot-dotted line).

Used as a worst-case scenario, the *crew* sequence (dashed) with up to eleven RoI performs better than most of the other sequences in the first 50 frames, but shows some drops in quality around frame 109 (not depicted). Note, however, that the perceived quality is still relatively very high, i.e., the amount of drift is low and spatially limited at the minimum PSNR value at frame 109, as illustrated in Fig. 8 (bottom

Fig. 8. Left: Frames 46 (top) and 109 (bottom) of the *crew* sequence with QP 27; right: Encoded with our proposed approach with examples of average drift (28 dB, top) and worst-case drift (14 dB, bottom), respectively, for a high number of scrambled RoIs.

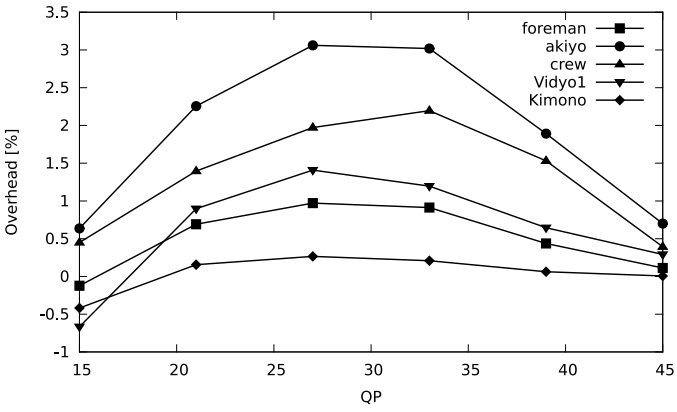

Fig. 9. Bit rate increase for different sequences and QPs: The space overhead increases with the number of RoIs and peaks at medium-quality QPs.

right). For the average case like in frame 45 (Fig. 8, top right), there is quasi no drift at all.

C. Space Overhead

Finally, we determine the increase in file size caused by our approach due to the drift reduction. Fig. 9 illustrates this increase with respect to the original, i.e., unscrambled compressed, file size.

The bit rate increase depends on the QP. High and low QP induce little increase or even decrease, while medium QP increase the bit rate by about 1-3% for the CIF resolution sequences and significantly less for the high resolution sequences. The exact increase depends on the sequence and the number of RoIs. In general, increasing the resolution decreases the overhead.

The number of additional bits required for drift compensation decreases with increasing QP since high QP induce large quality degradations regardless of the presence of drift. Similarly, low QP yield a high number of non-zero coefficients in the transformed intra and inter prediction residuals, making the amount of bits required for drift reduction relatively small in comparison. For medium QP, the number of additional bits required for drift reduction is about the same, but the number of non-zero coefficients is smaller, therefore leading to a relative increase in bit rate.

Note that the bit rate increase of the *akiyo* sequence can be considered a worst-case scenario since there is practically no movement in the sequence outside the RoI. This allows coding this area with a small amount of bits, making every change due to drift reduction relatively large in comparison.

V. Future Work

Two main aspects remain future work. First, the approach proposed in this paper can be combined with the approach of Unterweger et al. [17] which eliminates spatial, but not temporal drift through the use of slice groups. Since our approach reduces temporal drift, a combination of the two approaches would allow for nearly drift-free bit-stream-based RoI scrambling. This would come at the cost of additional space overhead due to using slice groups [17].

Second, descrambling the sequences scrambled with our proposed approach restores the RoI, but introduces additional drift outside the RoI due to the mismatch between the original prediction values and our drift-compensated ones. Although it is possible to copy the non-RoI areas (which are unscrambled) from the scrambled video, this does not allow for perfect reconstruction due to the remaining small amount of drift. Thus, a method to signal the RoI (as proposed, e.g., in [7]) as well as to compress and signal the difference between the original non-RoI areas and their counterparts with drift has to be devised so that the original video can be fully restored. Note that this may not be necessary for many use cases since, typically, only the RoI need to be fully restored, which is already the case with our approach.

VI. Conclusion

We proposed a region of interest scrambling approach for H.264/AVC bit streams. Despite being significantly faster than full re-encoding, it keeps the amount of drift outside the regions of interest at acceptable levels. The remaining amount of drift in all of the tested sequences is relatively small, apart from the *crew* sequence which exhibits some spatially limited drift in a small number of frames due to the high number of small regions of interest and intra-prediction-related dependencies. The bit rate overhead of our proposed approach is small (1.5% for high resolution sequences and 3% for low resolution sequences, tops) and depends on the quality and motion characteristics of the sequence to be scrambled. Our proposed approach is therefore a viable alternative to full re-encoding without the drawback of high computational complexity.

VII. Acknowledgments

This work is supported by FFG Bridge project 832082.

References

[1] S. Auer, A. Bliem, D. Engel, A. Uhl, and A. Unterweger. Bitstream-Based JPEG Encryption in Real-time. *International Journal of Digital Crime and Forensics*, 5(3):1–14, 2013.

[2] T. E. Boult. PICO: Privacy through invertible cryptographic obscuration. In *IEEE/NFS Workshop on Computer Vision for Interactive and Intelligent Environments*, pages 27–38, Lexington, KY, USA, Nov. 2005.

[3] P. Carrillo, H. Kalva, and S. Magliveras. Compression Independent Reversible Encryption for Privacy in Video Surveillance. *EURASIP Journal on Information Security*, 2009:1–13, Jan. 2009.

[4] J. De Cock, S. Notebaert, P. Lambert, and R. V. de Walle. Requantization transcoding for H.264/AVC video coding. *Signal Processing: Image Communication*, 25(4):235–254, 2010.

[5] F. Dufaux and T. Ebrahimi. Scrambling for privacy protection in video surveillance systems. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(8):1168–1174, 2008.

[6] F. Dufaux and T. Ebrahimi. A framework for the validation of privacy protection solutions in video surveillance. In *Proceedings of the IEEE International Conference on Multimedia & Expo, ICME '10*, pages 66–71, Singapore, July 2010. IEEE.

[7] D. Engel, A. Uhl, and A. Unterweger. Region of Interest Signalling for Encrypted JPEG Images. In *IH&MMSec'13: Proceedings of the 1st ACM Workshop on Information Hiding and Multimedia Security*, pages 165–174. ACM, June 2013.

[8] R. Iqbal, S. Shahabuddin, and S. Shirmohammadi. Compressed-domain spatial adaptation resilient perceptual encryption of live H.264 video. In *10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA 2010)*, pages 472–475, 2010.

[9] ITU-T T.81. Digital compression and coding of continuous-tone still images — requirements and guidelines, Sept. 1992. Also published as ISO/IEC IS 10918-1.

[10] Y. Kim, S. Jin, and Y. Ro. Scalable Security and Conditional Access Control for Multiple Regions of Interest in Scalable Video Coding. In Y. Shi, H.-J. Kim, and S. Katzenbeisser, editors, *International Workshop on Digital Watermarking 2007 (IWDW 2007)*, volume 5041, pages 71–86. Springer Berlin / Heidelberg, 2008.

[11] P. Korshunov and T. Ebrahimi. Towards Optimal Distortion-Based Visual Privacy Filters. In *21st IEEE International Conference on Image Processing (ICIP 2014)*, Paris, France, Oct. 2014. IEEE.

[12] Q. Meibing, C. Xiaorui, J. Jianguo, and Z. Shu. Face Protection of H.264 Video Based on Detecting and Tracking. In *8th International Conference on Electronic Measurement and Instruments 2007 (ICEMI'07)*, pages 2–172–2–177, Xi'an, China, Aug. 2007.

[13] E. Newton, L. Sweeney, and B. Malin. Preserving privacy by de-identifying face images. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):232–243, Feb 2005.

[14] X. Niu, C. Zhou, J. Ding, and B. Yang. JPEG Encryption with File Size Preservation. In *International Conference on Intelligent Information Hiding and Multimedia Signal Processing 2008 (IIHMSP '08)*, pages 308–311, Aug. 2008.

[15] Y. Sun, X. Wang, and X. Tang. Deep Convolutional Network Cascade for Facial Point Detection. In *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '13, pages 3476–3483, Washington, DC, USA, 2013. IEEE Computer Society.

[16] L. Tong, F. Dai, Y. Zhang, and J. Li. Restricted H.264/AVC video coding for privacy region scrambling. In *2010 17th IEEE International Conference on Image Processing (ICIP)*, pages 2089–2092, Sept. 2010.

[17] A. Unterweger and A. Uhl. Slice groups for post-compression region of interest encryption in H.264/AVC and its scalable extension. *Signal Processing: Image Communication*, 29(10):1158–1170, November 2014.

[18] A. Unterweger, K. Van Ryckegem, D. Engel, and A. Uhl. Building a Post-Compression Region-of-Interest Encryption Framework for Existing Video Surveillance Systems – Challenges, obstacles and practical concerns. *Multimedia Systems*, 2015. accepted.

[19] P. Viola and M. Jones. Robust Real-time Object detection. In *International Journal of Computer Vision*, volume 57, pages 137–154, 2001.

[20] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC video coding standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):560–576, July 2003.

[21] B. Yang, C.-Q. Zhou, C. Busch, and X.-M. Niu. Transparent and perceptually enhanced JPEG image encryption. In *16th International Conference on Digital Signal Processing*, pages 1–6, July 2009.