EXTRACTING PRNU NOISE FROM H.264 CODED VIDEOS

Enes Altinisik[1], *Kasim Tasdemir* [2], *Husrev Taha Sencar*[1]

[1] TOBB University, Ankara, Turkey
[2] Abdullah Gul University, Kayseri, Turkey

ABSTRACT

Every device equipped with a digital camera has a unique identity. This phenomenon is essentially due to a systematic noise component of an imaging sensor, known as photo-response non-uniformity (PRNU) noise. An imaging sensor inadvertently introduces this noise pattern to all media captured by that imaging sensor. The procedure for extracting PRNU noise has been well studied in the context of photographic images, however, its extension to video has so far been neglected. In this work, considering H.264 coding standard, we describe a procedure to extract sensor fingerprint from non-stabilized videos. The crux of our method is to remove a filtering procedure applied at the decoder to reduce blockiness and to use macroblocks selectively when estimating PRNU noise pattern. Results show that our method has a potential to improve matching performance significantly.

1. INTRODUCTION

It is now well established that photo-response non-uniformity (PRNU) noise associated with an imaging sensor can be used as a fingerprint to reliably identify an imaging sensor. The validity of this modality in source camera attribution is extensively studied and found to be reliable enough to be accepted by courts. The majority of the studies in this field, including the method for estimating and detecting PRNU noise associated with a sensor, has mainly considered photographic images as the application medium.

Most cameras do not save raw sensor data as it is acquired. Therefore, PRNU noise pattern needs to be extracted from the camera output media which is effectively a processed and compressed version of the sensor readings. The processing steps in a digital camera pipeline have important implications on the robustness of camera fingerprint extraction and matching procedures. Not only it may reduce the strength of the PRNU noise due to compression but it may also introduce artifacts that yield spurious similarities between unrelated camera fingerprints.

In the context of image data, since most cameras save images in high-quality JPEG format, compression is not viewed as a major hindrance. To cope with sensor-independent artifacts due to in-camera processing, PRNU noise estimate is processed in two ways: First, by zeroing out the means of its rows and columns to eliminate traces of color interpolation process. And second, by performing Wiener filtering to suppress blockiness effect due to JPEG coding which reveals itself as a periodic pattern in the noise estimate at 8×8 block boundaries.

Videos provide a wealth of data for camera attribution as the number of frames that can be utilized for fingerprint extraction is typically very high. However, this apparent advantage is offset by some severe restrictions. A number of works have already considered the problem of extracting camera fingerprints from videos. These work mainly proposed modifications over the existing procedure tailored for photographic images. To eliminate the blockiness artifact due to video coding, the initial work in this field [1] proposed first cropping out noise components at macroblock boundaries and then applying the filtering procedure to eliminate other periodic components. Focusing on video compression authors in [2] investigated how fingerprint matching accuracy varies with with the type of codec, video quality, and video resolution. They observed the non-linear relation between the quality of a video and the matching statistic. In [3], authors proposed using only one type of frames, so called *I* frames, in a video sequence, and [4] considered deploying an additional MACE filter for more reliable extraction of camera fingerprints. Later work, [5, 6], also considered codecs' ability to compensate for missing data under lossy transmission scenarios by identifying extrapolated block data and excluding them from fingerprint extraction.

Essentially, the operation of a camera during video capturing is quite different than when taking a photo. At the acquisition level, in addition to usual color processing steps, cameras use a different portion of the sensor and offer some form of image stabilization to reduce blur due to camera shake. When done electronically, stabilization may introduce very complex geometric transformations to successive video frames impairing the fingerprint extraction significantly. In terms of encoding, the size of a raw video (sequence of images) is impractically large to store or transfer. Therefore, video coders need to provide a very efficient representation of the content. As compared to photos, this results with a far more compressed media with additional artifacts.

In this work, by focusing on the video coding aspect, we introduce the procedure to extract a camera fingerprint from a video. Our approach is based on two key principles. First, since compression is a major obstacle, we identify and discard heavily compressed blocks and create a new set of frames by stitching together remaining blocks. Second, by intervening in the video decoding process, we remove or miminize effects of the loop filter, the mechanism used by a video codec to suppress hard blockiness effect. Our method and results are based on the operation of H.264 codec which is the most prevalent and popular video compression standard today.

The remainder of this paper is organized as follows. In the next section, we briefly discuss critical steps of video coding workflow from a perspective of how they affect the inherent PRNU noise. Section 3 describes our method of identifying macroblocks that will be used during fingerprint extraction and the removal of the loop filter. Experimental results and our conclusions are given in Sections 4 and 5, respectively.

ISBN 978-90-827970-1-5

2. H.264 VIDEO CODING STANDARD

The H.264 compression reduces temporal (inter-frame) and spatial (intra-frame) correlations by means of predicting frame regions using other visually similar regions. Instead of storing or transferring similar frame regions multiple times, the idea is to transfer a reference region once, then the next time transfer some prediction parameters to the receiver to let it predict the relevant region using the reference region and the prediction parameters.

In H.264, which is a descendent of older block based compression standards MPEG1/2 and H.263, frames are divided into blocks in a way similar to JPEG coding. Those blocks have a size of 16×16 pixels and are called *macroblocks*. During coding, macroblocks can be further divided into sub-macroblocks as small as 4×4 pixels as the basic encoding unit. Figure 1 shows a sample partitioning of a frame into macroblocks and sub-macroblocks.

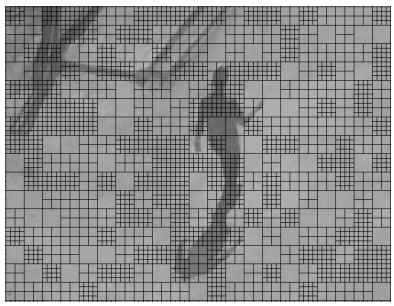

Figure 1: A sample partitioning of a frame into macroblocks and sub-macroblocks obtained using Elecard software tool.

In JPEG, blocks are coded in a self contained manner. That is, they are not predicted from other blocks, and this is a major difference with video coding. In H.264, a macroblock can be predicted from adjacent macroblocks of the same frame which makes the block an *intra-coded* block, or, it can be predicted from macroblocks of previous and/or future frames, which makes it an *inter-coded* block. The difference between the prediction and current macroblock is called *residual*, and the residuals are the data that are compressed and transferred along side the prediction parameters.

In a video sequence, frames can have three different types, namely, I, B, and P frames. An I frame can have only intra-coded blocks. That is, an I frame is temporally independent and self contained. It can be likened to a JPEG image in this sense. However, they are quite different in many aspects including variable block sizes, use of prediction, integer transformation based on DCT, variable quantization, and deployment of a loop filter. The macroblocks in P frame can use previous I or P frame macroblocks to find the best prediction. A B frame is similar to a P frame in its coding. Moreover, it can use future I and P frames when predicting a macroblock.

Similarly, macroblocks can have I, P or B types. An I macroblock is intra-coded, where P macroblock is predicted using a past frame's macroblock and B macroblock is predicted using macroblocks of two different frames. That means a B frame can incorporate I, P or B macroblocks whereas an I frame can only have I type macroblocks. Frames in a video follow a specific sequence of I, P and B frames, *e.g.*, *IBBPBBPBB*, called *group of pictures* (GOP). Each GOP starts with an I frame and includes only one I frame. The same GOP pattern repeats throughout the video.

During video coding, a frame is divided into blocks of various sizes. Prediction block of each macroblock is created using either intra or inter prediction. The difference between prediction block and the mackroblock itself s called *residual macroblock*. A residual macroblock can have negative numbers. A good prediction yields a residual macroblock with small absolute values. Similar to the JPEG compression, the residual macroblock is transformed into frequency domain and quantized. However, instead of DCT, an integer transformation is applied. The transformed and quantized matrix is zig-zag scanned, entropy coded and stored or transferred. By default, the transformation is applied to 4×4 blocks, but depending on the selected coding configuration 8×8 sized blocks can also be used.

From the standpoint of extracting a camera fingerprint, the most critical steps of video coding are the quantization and the loop filtering. These steps are elaborated in the following sections.

2.1 Quantization

With H.264 coding the transform and quantization steps are designed to minimize computational complexity so that devices using limited-precision integer arithmetic can perform coding. For this purpose instead of using discrete cosine transform (DCT), which is used in JPEG, H.264 uses an integer transform. Moreover, some parts of the integer transform is combined with quantization into a single step as follows. Integer transform uses a scaled integer approximation of a DCT transform matrix. However, unlike a DCT matrix, an integer transformation matrix is not orthogonal. For orthogonalization, it is multiplied by a normalization or a scaling matrix. Later, the transformed matrix is divided by a *quantization step*, Q_{step}. Instead of having a multiplication for scaling and then a division for quantization, H.264 standard offers merging these two operations into one multiplication step which can be very fast when it is realized in hardware.

Therefore, in H.264 standard the actual quantization step size cannot be directly selected by the user. It can be controlled indirectly as a function of a quantization parameter (QP). In practice, user decides on the bit-rate of coded video and the QP is changed accordingly so that the intended rate can be achieved. Therefore, unlike a single quantization table as deployed in JPEG, quantization parameters may change from block to block when coding a frame. Also in contrast to JPEG, a uniform quantizer step is applied to every coefficient in a 4×4 or 8×8 block by default. However, frequency dependent quantization can also be performed for different compression profiles.

2.2 Loop Filtering

Block-wise quantization performed during encoding yields a blockiness effect on the output images, H.264 decoder uses *loop filters* to compensate this effect by applying a low pass filter to smooth the block boundaries. The filter is applied up to 3 pixels from the boundary of 4×4 blocks. The strength of the filter and the number pixels affected from filtering depends on several constraints such as being at the boundary of a macroblock, current QP, and the gradient of image samples across the boundary.

It must be noted that the loop filter is also deployed dur-

ISBN 978-90-827970-1-5 © EURASIP 2018

ing encoding to allow encoder to consider the effect of the loop filter at the decoder. This is because decoder reconstructs each macroblock by adding a residual signal to a reference block identified by the encoder. At the decoder, however, all the previously reconstructed blocks would have been filtered. As a result, to mimic this behavior, encoder needs to perform prediction assuming filtered block data rather than using original macroblocks. The only exception to this is intra-frame prediction where only unfiltered blocks are used.

3. FINGERPRINT EXTRACTION

To be able to extract a camera fingerprint from a video one needs to deal with compression and loop filtering. Since loop filter is deployed at the decoder and is potentially applied to all 4×4 blocks, it must be handled first. In all frames, *I* macroblocks are coded using intra-frame prediction; hence, bypassing the loop filter during decoding will prevent further weakening of the PRNU noise. However for P and B type macroblocks since encoder performs prediction assuming filtered blocks, simply removing the loop filter at the decoder will not yield a correct reconstruction. To compensate for this behavior, decoding process must be modified to reconstruct both a filtered and non-filtered version of each macroblock. Filtered macroblocks must be used for reconstructing future macroblocks, and non-filtered ones need to be used for fingerprint extraction.

Since videos are typically coded at low bit-rates, compression poses the main challenge to camera fingerprint extraction. Two factors help in dealing with irreversible loss of data. First is that encoder determines the quantization parameter of each block independently, and the second is due to the large number of blocks available for fingerprint extraction. Therefore, our main goal is to identify blocks that can be utilized for PRNU noise extraction.

Quantization is essentially performed by dividing transform coefficients by Q_{step} which is determined in terms of the quantization parameter QP that takes values in the range 1 to 50. The relation between the two parameters are such that a value of $QP = 4$ is approximately equivalent $Q_{step} = 1$, and for each 6 higher values of QP, the corresponding values in Q_{step} value doubles. Due to this exponential relation, a linear increase in QP will result with an exponential reduction in the quality of images and the reliability of the PRNU noise.

To determine the relation between the choice of QP and the quality of PRNU noise, we utilized 550 photographic images. These images were captured by the same camera at six different scenes by moving the camera very smoothly. All images had a resolution of 12 megapixels and were originally saved in JPEG lossless format. These images are coded into H.264 videos each time using a different and fixed quantization parameter for all blocks, yielding a total of 50 videos. That is, for each video during H.264 coding QP for all blocks is set to a fixed value, and the GOP size is fixed to 10 frames for all videos. The camera fingerprint was generated independently using a separate set of photos. The PRNU noise estimates extracted from video frames are matched with this known camera fingerprint by computing the the peak to correlation energy (PCE). Figures 2 and 3 display the change in Peak Signal to Noise Ratio (PSNR) and PCE as a function of QP. As it can be seen in these figures, compression with $QP = 24$ or above significantly eliminates the PRNU noise in a video frame.

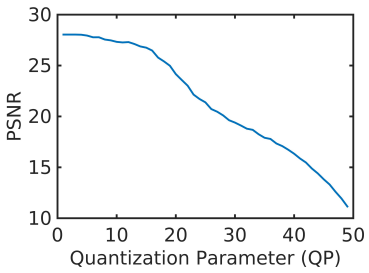

Figure 2: Change in PSNR with respect to QP.

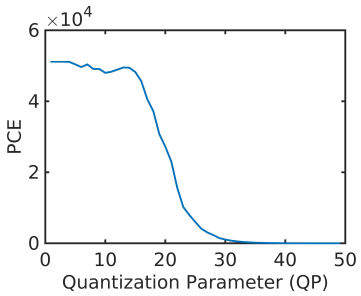

Figure 3: Change in PCE with respect to QP.

The procedure for PRNU noise estimation is known to depend on the underlying content due to assumed noise model. Studies done on still images revealed that image regions with low and very high intensity values as well as high frequency signal content are less favorable for extraction [7–9]. To test how these observation apply to video coding, we cropped out $35,200$ blocks of size 520×390 pixels by dividing each video frame into 64 equal sized blocks. Figure 4 displays how how PCE varies with average block intensity for the case when encoding was performed under the setting $QP = 10$.

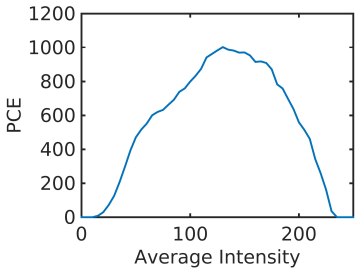

Figure 4: Change in PCE with respect to average intensity computed over blocks of size 520*x*390 pixels under compression with $QP = 10$.

Another factor that affects the PRNU noise estimation is the texture of the underlying image content which can be roughly related to the variance of image blocks. Measurements performed on our test videos show that unless variance of a block is above a value $400 - 500$ there won't be noticeable reductions in the measured PCE values. Further, measuring variances of 16×16 macroblocks in encoded video frames, we found that 90% of them have a variance less than 10. Therefore, we deduce that only macroblocks with abnormally high variances should be excluded from noise estima-

ISBN 978-90-827970-1-5 © EURASIP 2018

tion, for all other macroblocks variance should not be taken as a major determinant of PCE.

Overall, given an H.264 encoded video, macroblocks that should be used for estimating PRNU noise can be identified and sorted on the basis of used quantization parameter and the average macroblock intensity. To more reliably quantify the relation between PCE, QP, and the average block intensity, we used 12 videos compressed at varying QP values $(5, 10, 15, 18, 21, 24, 27, 30, 35, 40, 45, 50)$. Similar to Fig. 4, we extracted $35,200$ blocks from each video, measured their average intensity, and computed the PCE of each block with original camera fingerprint. Figure 5 shows the resulting three-dimensional curve. Although the blocks here are much larger that macroblocks in a video frame, this can guide in identifying the correct macroblocks needed for obtaining a reliable estimate of PRNU noise.

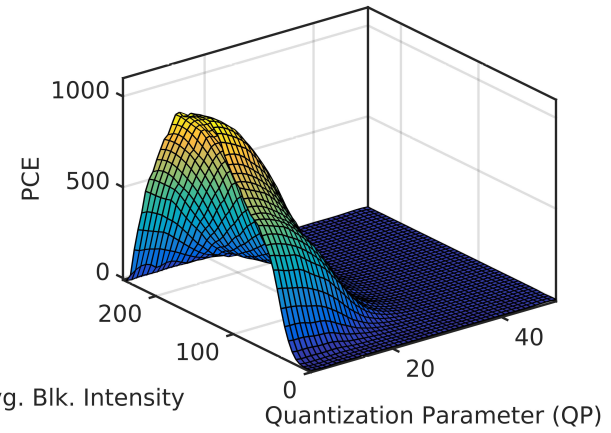

Figure 5: Change in PCE with respect to average intensity computed over blocks of size $520x390$ pixels under compression with varying QP values.

4. EXPERIMENTAL RESULTS

We provide results on how the removal of loop filter and the use of select macroblocks for noise estimation improve the quality of estimated PRNU noise in comparison to direct application of the conventional procedure [7] to video frames. In our tests, we use videos converted from the same 550 photographic images at various compression levels. In all cases, we computed the PCE between the obtained PRNU noise estimates and the actual camera fingerprint obtained independently using another set of images following the standard procedure.

4.1 Impact of Removing Loop Filter

To determine performance impact of loop filtering, we consider three test cases. The first case forms the baseline where PRNU noise extraction procedure completely disregards the loop filter. In the second case, the video is encoded as in the first case but rather than using the H.264 decoder we used a modified version that removes the loop filter in the uncompressed video frames. Lastly, we completely eliminated application of loop filter both at the encoder and decoder, thereby leaving quantization related data loss as the main culprit for a reduced performance. This last case will yield the best achievable performance and will help us evaluate effectiveness of our method.

Tables 1 and 2 provide corresponding results for the tests where average PCE values for decoded I, P, and B frames are calculated separately for two videos compressed at different bitrates. In both cases, I frames yield the best noise estimate, followed by P and B frames. This is because I frames are compressed less than P and B frames and B frames are compressed the most. More critically, results show that removal of loop filter at the decoder improves PCE values by $30-60\%$ depending on the frame type. However, at the same time, removing the loop filter at the decoder is not as effective for P and B frames as not using the loop filter at all. (Since I frames are subject to intra-frame prediction, removal of loop filter at the decoder is sufficient.) We believe this difference arises because when encoder performs prediction assuming decoder will perform loop filtering, resulting residuals include more high frequency content making them more susceptible to compression induced data loss. Hence, a more accurate reconstruction of PRNU noise pattern is possible when loop filtering is not deployed.

Table 1: Influence of Removing Loop Filter On a Video Compressed at 50 mbits/sec

Loop Filtering Test Cases	**I** frames	**P** frames	**B** frames
Encoding and decoding with loop filtering	14165	3440	693
Filter removed at decoder	18168	5397	1116
No filtering at encoder or decoder	18168	5501	1175

Table 2: Influence of Removing Loop Filter On a Video Compressed at 8 mbits/sec

Loop Filtering Test Cases	**I** frames	**P** frames	**B** frames
Encoding and decoding with loop filtering	117	10	1.92
Filter removed at decoder	230	17	3.09
No filtering at encoder or decoder	230	17.2	3.1

To ensure removal of loop filter does not introduce a systematic artifact we also examined its influence on false positive matches. For this purpose we considered 23 camera fingerprints and matched them with noise estimates obtained from both the filter removed and non-removed videos. The difference in resulting PCE values for all cameras were negligible. Hence, we assert that presence of loop filtering does not contribute to false-positive matches.

4.2 Splicing Macroblocks

Given it is possible to sort macroblocks in order of decreasing reliability, we determine how much improvement can be obtained by leaving out macroblocks that won't contribute to PRNU noise estimation significantly. For this, our method first removes the loop filter in the video and proceeds to extract PRNU noise from full frames. However, rather than matching the resulting noise estimates directly with the camera fingerprint, we utilize the relation observed in Fig. 5 to

ISBN 978-90-827970-1-5 © EURASIP 2018

re-arrange macroblocks with respect to their QP values and macroblock intensities by splicing together new frames of PRNU noise. In the newly generated frames, it is necessary for macroblocks to preserve their position in their original frames to prevent any geometric distorion. Essentially, to maximize PCE, we fill up new frames by select PRNU noise blocks from other frames. Resulting spliced-up frames are then matched with the camera fingerprint.

To test the overall improvement, we created an H.264 video using 500 of the available photographic images with 5 mbps bitrate at 20 frames/sec. We must note here that since each frame has 12 megapixels, coding at the bitrate of 5 mbps in fact corresponds to a low-quality video. Figure 6(a), shows the PCE values obtained by matching the PRNU noise extracted from each frame with the camera fingerprint. It can be seen that PCE values vary in the range $0-110$ with only seven frames yielding a PCE value above 60, the typical threshold value used to make a match decision. Similarly, Fig. 6(b) shows computed PCE values when effects of loop filter are removed at the decoder. We see that the range for PCE values is extended to span $0-230$ with 33 frames yielding a PCE value above 60.

Finally, we spliced together 50 frames by choosing best blocks in accordance with the relation shown in Fig. 5 from all decoded video frames. Figure 7 shows resulting PCE values associated with spliced frames. As expected ordering of PCE values exhibit a decreasing characteristic with the best matching frame having a value above 470 and with 45 frames yielding a PCE above 60. This result asserts that when applied to video fingerprint verification, our method will make more reliable decisions. In a similar manner, our method will enable creation of a more reliable camera fingerprint from a video.

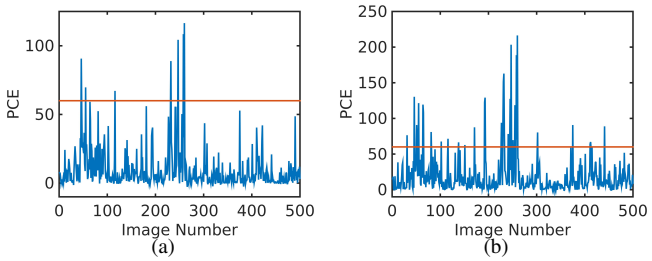

Figure 6: PCE values corresponding to 500 frames of a (a) loop filtered video and (b) loop filter removed video.

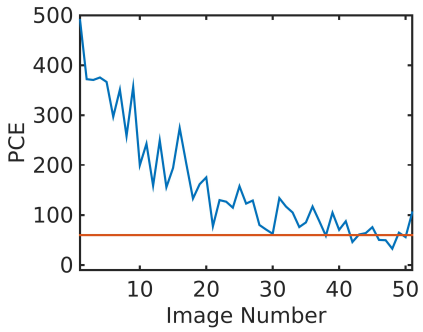

Figure 7: PCE values corresponding to 50 frames generated by macroblock splicing from a loop filter removed video.

5. CONCLUSIONS

Extraction of a camera fingerprint from a video involves many challenges. The change in the sensor's acquisition behavior, the necessity for stabilization, and the need for very effective compression are at the core of these challenges. Therefore, the procedure for camera fingerprint extraction from videos have to be designed with this focus. In this work, we address the problem of extracting PRNU noise from videos compressed using H.264 coding standard. The method we described considering the video coding aspect needs to be further enhanced to address the acquisition and stabilization aspects.

6. ACKNOWLEDGEMENT

This work is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) grant 116E273.

REFERENCES

[1] M. Chen, J. Fridrich, M. Goljan, and J. Lukáš, "Source digital camcorder identification using sensor photo response non-uniformity," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505. International Society for Optics and Photonics, 2007, p. 65051G.

[2] W. Van Houten and Z. Geradts, "Source video camera identification for multiply compressed videos originating from youtube," *Digital Investigation*, vol. 6, no. 1-2, pp. 48–60, 2009.

[3] W.-H. Chuang, H. Su, and M. Wu, "Exploring compression effects for improved source camera identification using strongly compressed video," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE, 2011, pp. 1953–1956.

[4] D.-K. Hyun, C.-H. Choi, and H.-K. Lee, "Camcorder identification for heavily compressed low resolution videos," in *Computer Science and Convergence*. Springer, 2012, pp. 695–701.

[5] A. Pande, S. Chen, P. Mohapatra, and G. Pande, "Architecture for blocking detection in wireless video source authentication," in *VLSI Design and 2014 13th International Conference on Embedded Systems, 2014 27th International Conference on*. IEEE, 2014, pp. 294–299.

[6] S. Chen, A. Pande, K. Zeng, and P. Mohapatra, "Live video forensics: Source identification in lossy wireless networks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 28–39, 2015.

[7] M. Chen, J. Fridrich, and M. Goljan, "Digital imaging sensor identification (further study)," in *Security, Steganography, and Watermarking of Multimedia Contents IX*, vol. 6505. International Society for Optics and Photonics, 2007, p. 65050P.

[8] J. Lukas, J. Fridrich, and M. Goljan, "Digital camera identification from sensor pattern noise," *IEEE Transactions on Information Forensics and Security*, vol. 1, no. 2, pp. 205–214, 2006.

[9] S. McCloskey, "Confidence weighting for sensor fingerprinting," in *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Computer Society Conference on*. IEEE, 2008, pp. 1–6.

ISBN 978-90-827970-1-5 © EURASIP 2018