Forensics of High-Quality JPEG Images with Color Subsampling

Matthias Carnein
University of Münster
Email: matthias.carnein@uni-muenster.de

Pascal Schöttle
Universität Innsbruck
Email: pascal.schoettle@uibk.ac.at

Rainer Böhme
Universität Innsbruck
Email: rainer.boehme@uibk.ac.at

***Abstract*—Detecting prior compression is an essential task in image forensics and can be used to detect forgery in digital images. Many approaches focus on grayscale images and assume compressions with a low quality factor which often leave visible artifacts in the image. In practice, however, color images and high quality compression are much more relevant and widespread. Block convergence has been proposed to estimate the number of JPEG compressions with quality factor 100 for grayscale images and has been shown to produce accurate results [1]. This paper extends block convergence to the more relevant case of color images where chrominance subsampling and color conversion make the estimation more complex. By observing block convergence for macro-blocks over multiple recompressions we are able to produce accurate estimates for color images. Oftentimes block convergence for color images enables similar accuracy and allows to detect more recompressions compared to grayscale images, while maintaining a good distinction between never and once compressed images.**

I. Introduction

Uncovering the compression history of an image is an important part in image forensics. It can serve as an indicator of how often an image has been saved with an image processing software and can be used to analyze whether an image has been tampered with. One of the most popular image formats is JPEG, standardized in 1992 [2]. Various methods have been developed to detect prior JPEG compression. Most of these methods aim to detect JPEG compression for grayscale images and quality factors lower than $Q = 100$. In practice, however, color images are much more widespread and with the decreasing cost of bandwidth and storage, high quality JPEG compression is the predominant choice for natural images.

In 2013, block convergence has been proposed to detect the number of JPEG-100 compressions for grayscale images [1]. It is based on the observation that rounding during repeated JPEG compression and decompression will cause the color values to converge. This allows to calculate a ratio of converged blocks and use it to estimate the number of compressions. This paper extends the concept of block convergence to color images where color space conversion as well as chrominance sub- and upsampling affect the convergence path. The proposed extensions are supported by experiments on a large dataset using the state-of-the-art JPEG library `libjpeg` [3]. Additionally, differences between the most common versions of `libjpeg` and their implications for block convergence are analyzed.

This paper is organized as follows: Section II introduces the concept of block convergence for grayscale images and reviews related work. Section III discusses how block convergence can be extended to the more complex case of color images. Then, Section IV describes the experimental setup and experiments performed. Section V assesses the results and evaluates the performance of the proposed method. Finally, Section VI presents approaches to identify the Discrete Cosine Transformation (DCT) implementation and the sub- and upsampling algorithm. Section VII concludes with a summary of the results and an outlook.

II. Related Work

Block convergence is an approach in digital image forensics proposed by Lai & Böhme [1] in 2013. It can be used for JPEG-100 carbon dating, i.e. estimating the number of times an image has been JPEG compressed with quality factor $Q = 100$.

The authors in [1] define a block as stable after t iterations if its values in $t+1$ equal its values in t. This allows to calculate a ratio of stable blocks:

$$r = (b_{\text{stable}} - b_{\text{flat}})/(b_{\text{total}} - b_{\text{flat}}), \tag{1}$$

where b_{stable} denotes the number of stable blocks and b_{total} denotes the total number of blocks. In addition b_{flat} describes the number of flat blocks. Flat blocks contain only a single value and are always stable in $t = 0$. These blocks are excluded since the amount of flat blocks varies vastly between images and would otherwise distort the ratio [1].

The number of compressions until a block is stable seems to follow a distribution that is widely independent of the image content. This allows to use either natural or randomly generated images in order to obtain a number of critical values. These pre-calculated values can then be compared against observed ratios. The interval that is associated with an observed ratio r is likely to indicate the number of compressions.

A vast amount of the literature in image forensics is dedicated to uncover the compression history of JPEG compressed images, e. g. [4], [5], [6], [7], [8], [9]. From these, only [9] explicitly considers the case of color, while the others either do not mention color at all, or state that "…the generalization to color images is straightforward – each color channel would be independently subjected to the same analysis" [5]. We close this gap in image forensics research by extending the concept of block convergence to JPEG color images.

III. Extension to Color Images

Block convergence for color images is more complex than for grayscale images. For grayscale images, the convergence path depends on the block-wise DCT transformation, a possible quantization step, as well as the corresponding inverse operations and the truncation of the spatial domain values to $[0, 255]$. For color images, the convergence path is

Table I. JPEG COMPRESSION STEPS INFLUENCING THE CONVERGENCE PATH

	Color conversion	Subsampling	DCT	Quantization	Rounding	Truncation
Grayscale	–	–	✓	✓	✓	✓
Color (4:4:4)	✓	–	✓	✓	✓	✓
Color	✓	✓	✓	✓	✓	✓

additionally influenced by color space conversions, subsampling and upsampling. Depending on the compression settings, these operations cover image areas of varying size, larger than 8×8, influencing how block convergence needs to be observed. An overview of the compression steps that influence the convergence path is given in Table I.

A. Color Images

Typically, images are represented in the RGB color space. The first step of JPEG compression, the color space conversion, maps colors to the YC_bC_r model by using a linear transformation. Here, the Y channel represents the luminance information and C_b, C_r the color information. Since the human eye is more sensitive to variations in brightness than color, the chrominance values are often stored with a lower resolution in order to increase the compression. This step is called subsampling. During upsampling, these values are then expanded to cover the entire image again.

Now, let us assume that $\boldsymbol{x}_t$ is a matrix of serialized blocks after the t-th compression of an image. Then, in the columns of $\boldsymbol{x}_t$ we have the blocks for the three YC_bC_r channels, i.e. each column contains the intensity values of the YC_bC_r channels of one block. The number of columns corresponds to the number of blocks in the image. With this notation, the remainder of the JPEG compression and decompression can be described by Eq. (2):

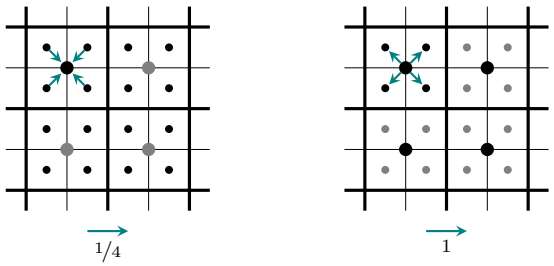
$$\boldsymbol{x}_{t+1} = \operatorname{tr}\left(\left[\boldsymbol{C}_{RGB}\left(\boldsymbol{H}_{\text{up}}\left[\boldsymbol{D}^T\left(\boldsymbol{q}^{-1}\right.\right.\right.\right. \\ \left.\left.\left.\left.\left[\boldsymbol{q}\left(\boldsymbol{D}\left(\boldsymbol{H}_{\text{sub}}\left(\boldsymbol{C}_{YC_bC_r}\boldsymbol{x}_t\right)\right)\right)\right]\right)\right]\right)\right]\right) \quad (2)$$

Here, $[\cdot]$ denotes rounding, $\operatorname{tr}(\cdot)$ truncates to the value range and $\boldsymbol{D}$ is the 2D-DCT matrix. $\boldsymbol{C}_{YC_bC_r}$ and $\boldsymbol{C}_{RGB}$ denote the colour conversion matrices and $\boldsymbol{H}_{\text{up}}$ as well as $\boldsymbol{H}_{\text{sup}}$ the matrices used for the linear filters to sub- and upsample the image. Additionally, $\boldsymbol{q}$ denotes the quantization matrix.

Without subsampling, i. e., subsampling rate 4:4:4, the convergence path depends only on the color conversion, the DCT transformation and a possible quantization step as well as the inverse decompression operations. Assuming baseline JPEG, all of these operations are confined within, at most, an 8×8 block covering an 8×8 image area. Only in this special case, the method of [1] can intuitively be extended to color images. Similar to grayscale images, a block is considered stable if, for all channels, none of its values change with the next compression. We exclude all blocks b_{flat} where at least one channel is flat. Even though this does not necessarily mean that the image area is flat in $t = 0$, these partially flat blocks show abnormal behavior and should be excluded from the analysis.

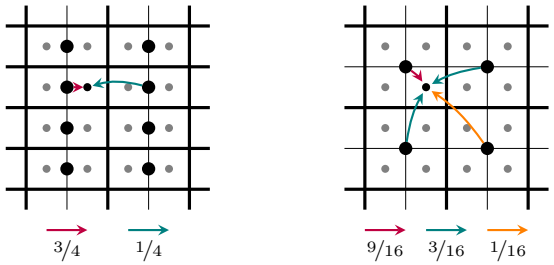
B. Sub- and Upsampling

Various sub- and upsampling algorithms are available. The most intuitive subsampler calculates the average of the source pixel covered by the output pixel. In `libjpeg` this approach is called "simple subsampling" [3]. The area that an output pixel covers depends on the subsampling rate. For example, using the common subsampling rate 4:2:0, only the average of a 2×2 area is stored, as shown in Figure 1(a). The corresponding "simple upsampler" merely replicates the source pixel to cover all output pixels again, as shown in Figure 1(b). This approach is typically referred to as a box filter. Since each chrominance pixel represents a 2×2 area, each transformation performed on a chrominance block now affects a 16×16 macro-block. Then, the number of columns in $\boldsymbol{x}_t$ has to be adapted to the macro-block-size instead of the original block-size and block convergence needs to be analyzed on a macro-block level. This concept can be adapted to other subsampling rates by modifying the macro-block size accordingly.



(a) Simple subsampling (b) Simple upsampling

Fig. 1. Simple sub- and upsampling



(a) Fancy h2v1 upsampling (b) Fancy h2v2 upsampling

Fig. 2. Fancy upsampling in `libjpeg` version 6

C. "Fancy" Upsampling

Until `libjpeg` version 6, two specialized "fancy" upsamplers are available. These algorithms perform a linear interpolation of input pixels. Chrominance pixels are weighted by proximity. This is supposed to produce higher quality results while maintaining a good processing speed [3]. The available upsampling methods handle the two common cases: 1) chrominance information is halved in horizontal and not changed in vertical direction (h2v1); or 2) color information is halved in both, vertical and horizontal, direction (h2v2). For other subsampling rates, `libjpeg` defaults to simple upsampling. For the case of h2v1, the nearest chrominance pixel is weighted with $3/4$ and the more distant pixel with $1/4$, as shown in Figure 2(a). For the case of h2v2, the nearest chrominance pixel is weighted with $9/16$, the two orthogonally adjacent pixels are weighted with $3/16$ and the diagonally adjacent pixel with $1/16$, as shown in Figure 2(b). Both approaches are commonly referred to as triangle filters.

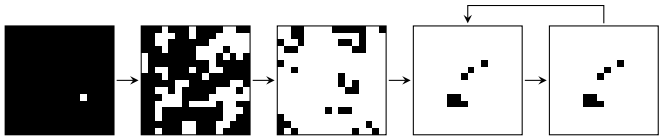


Fig. 3. Difference with successive recompressions and two cyclic final states

Triangle filters influence the values of adjacent pixels even across macro-block boundaries. Hence, we observe *spill-over* effects at the border of macro-blocks due to the chrominance interpolation. As a result, a block cannot be considered stable unless all blocks of an image are stable. One option is to ignore spill-over effects and consider a macro-block stable if, for all channels, none of its values change with the next compression. This leads to longer convergence paths that are not as clear-cut. Some blocks converge fast, while some do not converge at all. Alternatively, we can prevent spill-over effects between blocks by isolating each macro-block and observing its convergence path without the influence of surrounding blocks. This can be done by JPEG-compressing each macro-block individually. The first isolated compression is unlikely to produce many stable blocks due to the change of compression settings, but subsequent compressions produce a distinct convergence path.

However, even when avoiding spill-over effects, we observe that about 15% of all blocks do not become stable within the first 40 iterations. In some cases we even observe cycles: blocks alternate between different states and never become stable. A shortened example of such cyclic behavior is shown in Figure 3 for a 16×16 image. White denotes pixels that did not change with the last compression and black denotes pixels that changed. This enables interesting application scenarios. For example, cyclic blocks could be used to create copy-evident JPEG images, similar to copy-evident printing [10]. By embedding these cyclic "counter blocks" into an image one can observe in which state the blocks are and use this information to determine how often the image has been recompressed. If the cycle lengths of different "counter blocks" do not share a common divisor, i.e. if the lengths are co-prime, the exact number of times the image has been compressed can be determined. Depending on the appearance of the "counter blocks", they can either be embedded into the image or placed as control blocks in a dedicated area. To protect blocks from spill-overs, they should be surrounded by content that is similar at the macro-block boundaries.

D. DCT Scaling

Starting with version 7, `libjpeg` introduces a "fancy" approach for subsampling and replaces the former "fancy" upsampler. This approach is called DCT scaling. The DCT is performed on larger blocks, but higher frequencies of the result are neglected to reduce the color information [11], [12]. For upsampling, a chrominance block is zero-padded to its original size before performing the inverse DCT (IDCT). Taking subsampling rate $4{:}2{:}0$ as an example, the DCT is performed on a 16×16 block but only the lower 8×8 frequencies are stored. For upsampling, the 8×8 block is zero-padded to its original size before applying the IDCT. `libjpeg` only supports DCT for block sizes less or equal to 16×16. For less common subsampling rates that would require larger block sizes, e.g. $4{:}1{:}0$, a form of simple subsampling is used first to bring the block to a supported size.

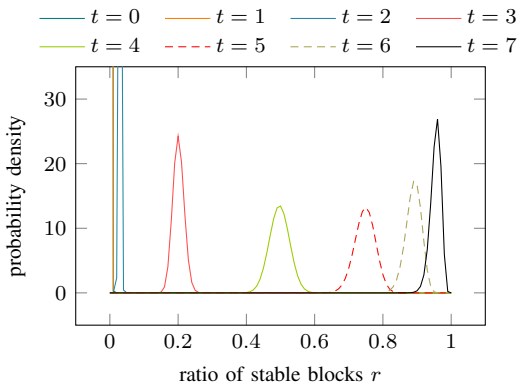


Fig. 4. Fitted beta distributions; no subsampling; `libjpeg 8d`; $Q = 100$

Similar to the simple upsampler, DCT scaling is confined within the size of a macro-block and there are no spill-over effects. Overall, this sub- and upsampling method has a much faster convergence path with low variation in the ratio of stable blocks, similar to simple sub- and upsampling.

E. Carbon Dating for Color Images

A general problem of the convergence paths for color images is that very few blocks become stable within the first compression. This makes it hard to distinguish whether an image was never compressed or has been compressed once. Arguably, this decision is one of the most important questions in carbon dating and image forensics in general. To solve this, we track the development of the ratio of stable blocks r over the next n compressions. This gives us the sequence $r_0, \ldots, r_n$ which we can use as an empirical distribution of r for n compressions. To carbon date images, we fit a theoretical distribution to the observed empirical distribution. Our distribution of choice is the beta distribution with the two shape parameters α and β.

To fit the distribution, we use the maximum-likelihood method to estimate parameters conditional to the number of compressions t. An example for the fitted beta distributions for subsampling rate $4{:}4{:}4$ is shown in Figure 4. We can then estimate t, again by using maximum-likelihood. This means we choose the family of theoretical distributions that has the highest combined probability density for the observed values:

$$\hat{t} = \arg\max_i \left(p_i(r_0) \cdot \ldots \cdot p_{i+n}(r_n) \right), \tag{3}$$

where p_i is the density of the beta distribution fitted for the observed values for $t = i$. Since the beta distribution is undefined for the boundary values 0 and 1 we add a small amount ϵ to 0-values and subtract the same from 1 [13].

To evaluate the choice of the beta distribution, we use Q-Q-plots and compare the fitted distributions against the observed values. Figure 5 exemplarily shows the Q-Q-plot for $t = 3$. Even though we observe some deviation from the theoretical distribution, the beta distribution seems to fit for carbon dating.

In general, the maximum-likelihood estimate works well when there is little overlap between the fitted distributions. However, with overlapping distributions, e.g. for a large t, the

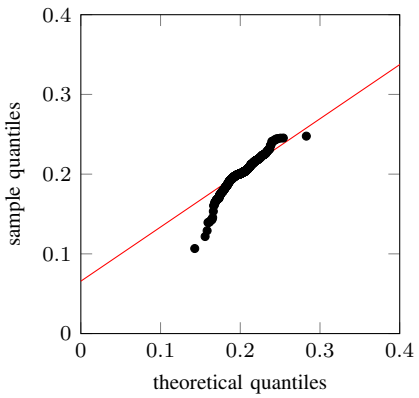


Fig. 5. Q-Q-Plot for the fitted beta distribution and $t = 3$

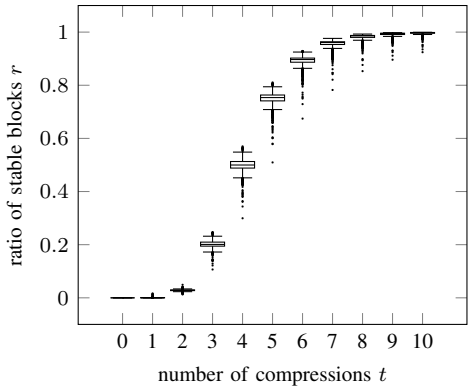


Fig. 6. Boxplots for r dependent on t; no subsampling; `libjpeg 8d`; $Q = 100$

discriminatory power is lower, which makes it hard to estimate the true number of compressions. An alternative approach is to use the sequence $r_0, \ldots, r_n$ as features to train a machine learning algorithm such as a Random Forest [14] or a Support Vector Machine [15] and use their classification for carbon dating. In our experiments we were able to observe slightly superior results when using machine learning. For this reason, we focus on this approach in the following sections.

IV. Experimental Setup

To evaluate the proposed extensions, we benchmark them with a large dataset of 1600 never compressed natural images obtained from the Dresden Image Database [16]. The images are of varying size and acquired with various camera models from different manufacturers. To evaluate the influence of the sub- and upsampling algorithm we compare the common JPEG-libraries `libjpeg 6b` and `libjpeg 8d`. We use `libjpeg 8d` instead of `libjpeg 7` and `libjpeg 9`, as it is more commonly used and offers the same methods for sub- and upsampling. Additionally, the libraries offer three different DCT implementations called "slow", "float" and "fast". In our experiments we focus on the default "slow" implementation but the results can be directly transferred to the "float" method. As described in [1], the convergence path of "fast" differs vastly and produces visible artifacts in the compressed image. We therefore exclude this method from our analysis.

The images are randomly divided into training and testing sets of equal size. For each method, we test its detection accuracy for up to ten compressions. For every training image, we calculate the ratio of stable blocks for the next $n = 10$ compressions and use the resulting features $r_0, \ldots, r_n$ to train a Random Forest with 200 classification trees. The same features are then calculated for the testing set and the classification algorithm is asked to classify the number of compressions. We chose to train a Random Forest since it is a fast and accurate classifier that requires little parameter optimization [14].

V. Empirical Results

When no subsampling is used, the development of the ratio of stable blocks is comparable to the case of grayscale images. Figure 6 shows the boxplots for the distribution of r over multiple compressions. Even though there exist numerous outliers, the probability mass is concentrated. This helps to reduce ambiguity for the detection of prior JPEG compression and provides accurate classification results. A major problem is the little difference of $t = 0$ and $t = 1$. However, we solve this by observing the sequence $r_0, \ldots, r_n$ as discussed in Section III. Table II reports the confusion matrix for up to ten recompressions. Only very few errors can be observed for the first six compressions. For a higher number of compressions, minor misclassification errors appear.

Table II. Confusion matrix for no subsampling and $Q = 100$

Detector output $\hat{t}$	Number of compressions t										
	0	1	2	3	4	5	6	7	8	9	10
0	800	1	0	0	0	0	0	0	0	0	0
1	0	799	2	0	0	0	0	0	0	0	0
2	0	0	798	2	0	0	0	0	0	0	0
3	0	0	0	798	2	1	0	0	0	0	0
4	0	0	0	0	798	4	1	0	0	0	0
5	0	0	0	0	0	795	7	2	0	0	0
6	0	0	0	0	0	0	790	14	2	1	0
7	0	0	0	0	0	0	2	776	30	4	1
8	0	0	0	0	0	0	0	8	746	52	9
9	0	0	0	0	0	0	0	0	18	687	87
10	0	0	0	0	0	0	0	0	4	56	703

To compare the results, we define two ratios for the confusion matrices: First, the *true detection* rate of correctly classified images, seen on the diagonal of the matrix. Second, we define the *off-by-two* ratio as the portion of *all misclassified* images which were misclassified by more than one compression. Note, that these ratios are slightly biased when calculated for the entire confusion matrix, since $t = 10$ is the highest possible estimate in our setup. To avoid this, we exclude $t = 10$ and $t = 9$ for the calculation of these ratios. Without subsampling we observe a true detection rate of 98.61% and an off-by-two ratio of 10.00%. This means that 90% of the the misclassified images are misclassified by only a single compression.

When using a subsampling rate of $4{:}2{:}0$, carbon dating highly depends on the sub- and upsampling algorithm. For simple sub- and upsampling the influence seems to be small. Blocks typically require one more compression in order to converge. Overall, however, the range of values remains small, enabling accurate estimates. The boxplots for the distribution of r are shown in Figure 7. We can observe a slight increase in accuracy in comparison to the case without subsampling with a true detection rate of 98.75%. The classification for

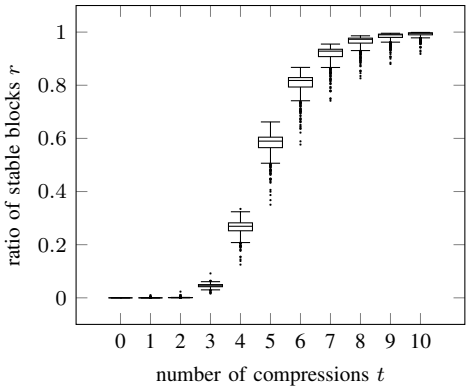


Fig. 7. Boxplots for r dependent on t; subsampling rate 4:2:0; simple sub- and upsampling; `libjpeg 8d`; $Q = 100$

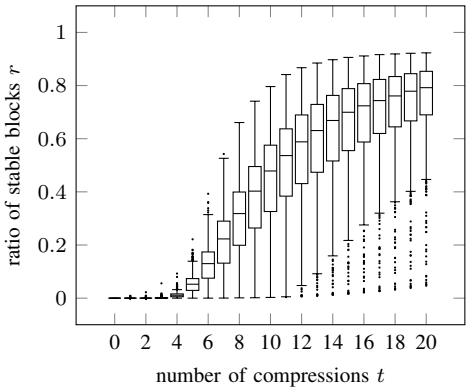


Fig. 8. Boxplots for r dependent on t; subsampling rate 4:2:0; fancy upsampling; `libjpeg 6b`; $Q = 100$

up to six compressions remains very accurate with only few misclassified images. Again, most of the the misclassified images are off by only one compression, leading to an off-by-two ratio of only 4.44%.

Carbon dating for fancy upsampling of `libjpeg 6b`, on the other hand, is considerably more difficult. When ignoring spill-over effects, the spill-overs and linear interpolation of chrominance pixels lead to a large range of values compared to simple upsampling. The observed ratios when ignoring spill-over effects are depicted in Figure 8. Alternatively, we can avoid spill-over effects and compress each macro-block individually. This reduces the range of values slightly as shown in Figure 9 but interrupts the convergence path. In general, both approaches allow us to analyze block convergence. In our experiments we observed slightly better results when ignoring spill-over effects, especially for higher number of compressions. The resulting confusion matrix in Table III shows a slightly higher classification error compared to simple subsampling, that increases with the number of compressions. The true detection rate is reduced to 94.12%, while the off-by-two ratio increases to 15.84%.

Next, we observe the detection performance for DCT scaling of `libjpeg 8d`. As discussed in Section III, the approach affects similar areas as its simple counterpart. The result is a fast convergence path without spill-over effects. Again, this leads to much clearer intervals with little variance allowing accurate estimates. Due to space constraints we refrain from

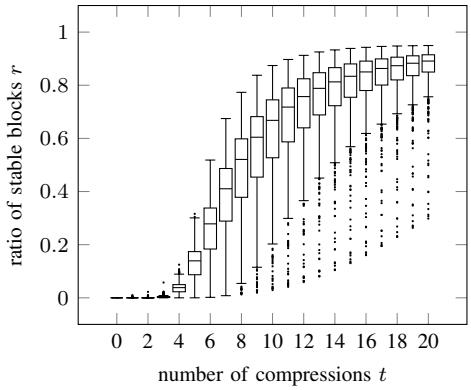


Fig. 9. Boxplots for r dependent on t; without spill-over effects; subsampling rate 4:2:0; fancy upsampling; `libjpeg 6b`; $Q = 100$

Table III. CONFUSION MATRIX FOR SUBSAMPLING RATE 4:2:0, FANCY UPSAMPLING, `LIBJPEG 6B` AND $Q = 100$

Detector output $\hat{t}$	Number of compressions t										
	0	1	2	3	4	5	6	7	8	9	10
0	790	3	7	0	1	0	0	0	0	0	0
1	8	791	3	5	0	1	0	0	0	0	0
2	2	5	782	3	4	0	1	0	0	0	0
3	0	1	6	778	4	3	0	0	0	0	0
4	0	0	2	13	775	7	2	0	0	0	0
5	0	0	0	1	15	768	9	2	3	1	1
6	0	0	0	0	0	19	740	30	4	6	3
7	0	0	0	0	1	1	43	699	56	3	2
8	0	0	0	0	0	1	2	60	654	81	10
9	0	0	0	0	0	0	3	5	72	614	104
10	0	0	0	0	0	0	0	4	11	95	680

printing the boxplots. The results are remarkably similar to simple sub- and upsampling as shown in Figure 7 and lead to a true detection rate of 98.72% and a low off-by-two ratio of 2.17%.

Finally, block convergence is not limited to a quality factor of $Q = 100$. Assuming that the same quality factor is used repeatedly and known to the analyst, block convergence can also be applied to lower quality factors. Lower quality factors typically have a shorter convergence path, which makes it hard to detect many compressions since the ratio of stable blocks quickly approaches one. However, few compressions can still be reliably detected. As an example, the detection performance for $Q = 95$ and no subsampling shows almost perfect accuracy for up to four recompressions. For a higher number of compressions more images are misclassified leading to a true detection rate of 88.62% and an off-by-two ratio of 32.60%. An overview of the ratios for all performed experiments is given in Table IV. For comparison, we also report the accuracy of the maximum-likelihood approach which yields competitive results, except for the detection of fancy upsampling.

VI. IDENTIFICATION OF DCT AND SUBSAMPLING IMPLEMENTATION

Until now we assumed that specific implementation choices such as the DCT implementation or the sub- and upsampling algorithm are known to the analyst. In practice, this is hardly applicable. To estimate the DCT implementation, we can extend the approach from [1] and compare the ratio of stable blocks for a number of candidate implementations. While the

Table IV. OVERVIEW OF DETECTION RESULTS

	subsampling		Random Forest		Maximum-likelihood	
Q	rate	algorithm	true detection	off-by-two	true detection	off-by-two
100	4:4:4	–	98.61%	10.00%	88.36%	7.64%
100	4:2:0	simple	98.75%	4.44%	85.42%	6.10%
100	4:2:0	fancy	94.12%	15.84%	41.81%	44.84%
100	4:2:0	fancy (isolated)	90.06%	23.04%	34.71%	60.46%
100	4:2:0	DCT	98.72%	2.17%	93.86%	4.07%
99	4:4:4	–	98.83%	2.38%	90.21%	6.52%
95	4:4:4	–	88.62%	32.60%	64.54%	42.11%
90	4:4:4	–	69.31%	57.15%	29.19%	66.91%
75	4:4:4	–	49.06%	72.25%	23.67%	79.86%
50	4:4:4	–	35.69%	82.83%	33.39%	54.86%

same implementation will continue the convergence path, a different implementation will interrupt it, leading to few stable blocks. Therefore, the candidate that produces the highest ratio of stable blocks is likely to be the true DCT implementation.

Our experiments show that this allows perfect accuracy once blocks start to become stable, e.g. for $t \geq 3$ when no subsampling is used. For lower number of compressions, the DCT implementation appears to have little influence on the ratio and all implementations seem to converge regardless of the originally used DCT implementation. This means that we can use any DCT implementation for low number of compressions and estimate the DCT implementation reliably once it influences the convergence path.

The same approach can also be used to estimate the sub- and upsampling algorithm. The combination of fancy and simple upsampling that produces the highest ratio of stable blocks is likely to indicate the previously used algorithms. Again, our results show almost perfect accuracy once the first blocks start to become stable.

VII. CONCLUSION

In this work we propose accurate image forensics for the relevant case of color images and high quality compression. We utilize the concept of block convergence and extend it to color images to reliably detect prior JPEG compressions with high quality factors. This is achieved by observing the convergence behavior for macro-blocks and either ignoring or preventing possible spill-over effects due to chrominance subsampling. Then, we observe the ratio of stable blocks for several recompressions and use a maximum-likelihood estimation or machine learning classification to make accurate estimates on how often an image has been recompressed. Machine learning showed a superior performance in our experiments but the maximum-likelihood approach is straightforward and would provide more accountable results, e.g. in court.

The results show almost perfect accuracy for up to six recompressions if no subsampling is used. If the chrominance information is subsampled, the accuracy depends on the sub- and upsampling algorithm. We examine three different choices available in different versions of the popular JPEG library `libjpeg`. For all methods, the detection accuracy remains strong and the majority of misclassifications are off by only a single compression. This accuracy declines when lower quality factors are used, but block convergence remains applicable.

Future work should derive a better theory for the convergence path and the distribution of stable blocks. Of special interest are the characteristics that influence the length of the convergence path and properties that prevent blocks from converging. Also, we were able to observe cyclic behavior for some blocks which could be used to create copy-evident JPEG images, similar to copy-evident printing, by embedding these "counter blocks" in images. Additionally, the applicability of block convergence analysis when images are recompressed with varying settings should be examined. Furthermore, a more theoretically founded approach to detect the subsampling implementation could be applied, e.g. by estimating linear dependencies between pixel values [17]. Finally, block convergence could be explored for other lossy image and video formats.

REFERENCES

[1] S. Lai and R. Böhme, "Block convergence in repeated transform coding: Jpeg-100 forensics, carbon dating, and tamper detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2013, pp. 3028–3032.

[2] International Telecommunication Union (ITU), "Recommendation T.81: Terminal equipment and protocols for telematic services," 1992.

[3] Independent JPEG Group. libjpeg. [Online]. Available: http://www.ijg.org/

[4] Z. Fan and R. de Queiroz, "Identification of bitmap compression history: Jpeg detection and quantizer estimation," *IEEE Transactions on Image Processing*, vol. 12, no. 2, pp. 230–235, Feb 2003.

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

[6] T.-I. Lin, M.-K. Chang, and Y.-L. Chen, "A passive-blind forgery detection scheme based on content-adaptive quantization table estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 4, pp. 421–434, April 2011.

[7] T. Bianchi and A. Piva, "Image forgery localization via block-grained analysis of JPEG artifacts," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 3, pp. 1003–1017, June 2012.

[8] D. Fu, Y. Q. Shi, and W. Su, "A generalized benford's law for jpeg coefficients and its applications in image forensics," in *SPIE Conference on Security, Steganography, and Watermarking of Multimedia Contents*, E. J. Delp and P. W. Wong, Eds., vol. 6505, 2007.

[9] B. Li, T.-T. Ng, X. Li, S. Tan, and J. Huang, "Statistical model of JPEG noises and its application in quantization step estimation," *IEEE Transactions on Image Processing*, vol. 24, no. 5, pp. 1471–1484, May 2015.

[10] A. Lewis and M. Kuhn, "Towards copy-evident JPEG images," in *Digitale Multimedia-Forensik, 39. Jahrestagung der Gesellschaft für Informatik 2009*, ser. GI-Edition: Lecture Notes in Informatics, vol. P154, 2009, pp. 1582–1591.

[11] R. Dugad and N. Ahuja, "A fast scheme for image size change in the compressed domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 4, pp. 461–474, Apr 2001.

[12] C. L. Salazar and T. D. Tran, "On resizing images in the DCT domain," in *International Conference on Image Processing, 2004. ICIP '04*, vol. 4, Oct 2004, pp. 2797–2800.

[13] M. Smithson and J. Verkuilen, "A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables." *Psychological Methods*, vol. 11, no. 1, pp. 54–71, 3 2006.

[14] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[16] T. Gloe and R. Böhme, "The Dresden image database for benchmarking digital image forensics," *Journal of Digital Forensic Practice*, vol. 3, pp. 150–159, 2010.

[17] M. Kirchner, "Linear row and column predictors for the analysis of resized images," in *Proceedings of the 12th ACM workshop on Multimedia and security*. ACM, 2010, pp. 13–18.