Detection of Copy-Move Forgery in Digital Images

[a]Jessica Fridrich, [b]David Soukal, and [a]Jan Lukáš

[a]Department of Electrical and Computer Engineering, [b]Department of Computer Science
SUNY Binghamton, Binghamton, NY 13902-6000
{fridrich, dsoukal1, bk89322}@binghamton.edu

Abstract

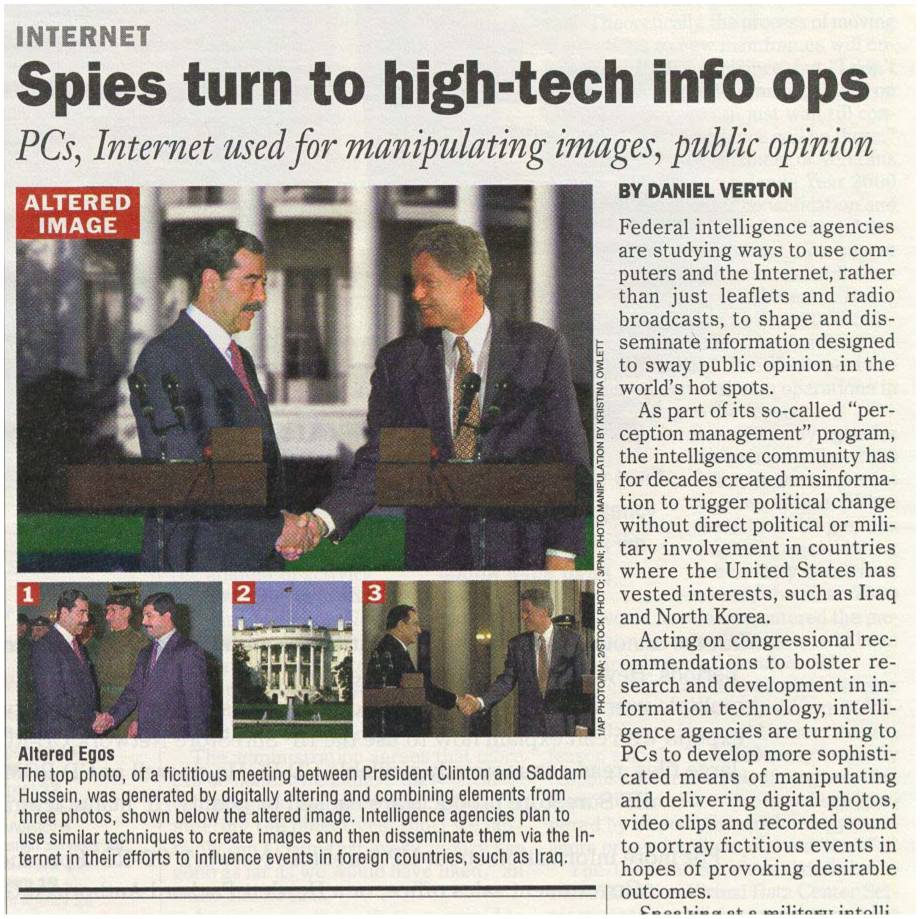
Digital images are easy to manipulate and edit due to availability of powerful image processing and editing software. Nowadays, it is possible to add or remove important features from an image without leaving any obvious traces of tampering. As digital cameras and video cameras replace their analog counterparts, the need for authenticating digital images, validating their content, and detecting forgeries will only increase. Detection of malicious manipulation with digital images (digital forgeries) is the topic of this paper. In particular, we focus on detection of a special type of digital forgery – the copy-move attack in which a part of the image is copied and pasted somewhere else in the image with the intent to cover an important image feature. In this paper, we investigate the problem of detecting the copy-move forgery and describe an efficient and reliable detection method. The method may successfully detect the forged part even when the copied area is enhanced/retouched to merge it with the background and when the forged image is saved in a lossy format, such as JPEG. The performance of the proposed method is demonstrated on several forged images.

1. The Need for Detection of Digital Forgeries

The availability of powerful digital image processing programs, such as PhotoShop, makes it relatively easy to create digital forgeries from one or multiple images. An example of a digital forgery is shown in Figure 1. As the newspaper cutout shows, three different photographs were used in creating the composite image: Image of the White House, Bill Clinton, and Saddam Hussein. The White House was rescaled and blurred to create an illusion of an out-of-focus background. Then, Bill Clinton and Saddam were cut off from two different images and pasted on the White House image. Care was taken to bring in the speaker stands with microphones while preserving the correct shadows and lighting. Figure 1 is, in fact, an example of a very realistic-looking forgery.

Another example of digital forgeries was given in the plenary talk by Dr. Tomaso A. Poggio at Electronic Imaging 2003 in Santa Clara. In his talk, Dr. Poggio showed how engineers can learn the lip movements of any person from a short video clip and then digitally manipulate the lips to arbitrarily alter the spoken content. In a nice example, a video segment showing a TV anchor announcing evening news was altered to make the anchor appear singing a popular song instead, while preserving the match between the sound and lip movement.

The fact that one can use sophisticated tools to digitally manipulate images and video to create non-existing situations threatens to diminish the credibility and value of video tapes and images presented as evidence in court independently of the fact whether the video is in a digital or analog form. To tamper an analogue video, one can easily digitize the analog video stream, upload it into a computer, perform the forgeries, and then save the result in the NTSC format on an ordinary videotape. As one can expect, the situation will only get worse as the tools needed to perform the forgeries will move from research labs to commercial software.


ALTERED IMAGE
1
2
3
1/AP PHOTO/INA; 2/STOCK PHOTO; 3/FNI; PHOTO MANIPULATION BY KRISTINA OWLETT

INTERNET

Spies turn to high-tech info ops

PCs, Internet used for manipulating images, public opinion

BY DANIEL VERTON

Federal intelligence agencies are studying ways to use computers and the Internet, rather than just leaflets and radio broadcasts, to shape and disseminate information designed to sway public opinion in the world's hot spots.

As part of its so-called "perception management" program, the intelligence community has for decades created misinformation to trigger political change without direct political or military involvement in countries where the United States has vested interests, such as Iraq and North Korea.

Acting on congressional recommendations to bolster research and development in information technology, intelligence agencies are turning to PCs to develop more sophisticated means of manipulating and delivering digital photos, video clips and recorded sound to portray fictitious events in hopes of provoking desirable outcomes.

Speaking at a military intelli

Altered Egos

The top photo, of a fictitious meeting between President Clinton and Saddam Hussein, was generated by digitally altering and combining elements from three photos, shown below the altered image. Intelligence agencies plan to use similar techniques to create images and then disseminate them via the Internet in their efforts to influence events in foreign countries, such as Iraq.

Figure 1 Example of a digital forgery.

Despite the fact that the need for detection of digital forgeries has been recognized by the research community, very few publications are currently available. Digital watermarks have been proposed as a means for fragile authentication, content authentication, detection of tampering, localization of changes, and recovery of original content [1]. While digital watermarks can provide useful information about the image integrity and its processing history, the watermark must be present in the image before the tampering occurs. This limits their application to controlled environments that include military systems or surveillance cameras. Unless all digital acquisition devices are equipped with a watermarking chip, it will be unlikely that a forgery-in-the-wild will be detectable using a watermark.

It might be possible, but very difficult, to use unintentional camera "fingerprints" related to sensor noise, its color gamut, and/or its dynamic range to discover tampered areas in images. Another possibility for blind forgery detection is to classify textures that occur in natural images using statistical measures and find discrepancies in those statistics between different portions of the image ([2], [3]). At this point, however, it appears that such approaches will produce a large number of missed detections as well as false positives.

In the next section, we introduce one common type of digital forgeries – the copy-move forgery – and show a few examples. Possible approaches to designing a detector are discussed in Section 3. In Section 4, we describe the detection method based on approximate block matching. This approach proved to be by far the most reliable and efficient. The method is tested in the last

Section 5 on a few forgeries. In the same section, we summarize the paper and outline future research directions.

2. Copy-Move Forgery

Because of the extraordinary difficulty of the problem and its largely unexplored character, the authors believe that the research should start with categorizing forgeries by their mechanism, starting with the simple ones, and analyzing each forgery type separately. In doing so, one will build a diverse Forensic Tool Set (FTS). Even though each tool considered separately may not be reliable enough to provide sufficient evidence for a digital forgery, when the complete set of tools is used, a human expert can fuse the collective evidence and hopefully provide a decisive answer. In this paper, the first step towards building the FTS is taken by identifying one very common class of forgeries, the Copy-Move forgery, and developing efficient algorithms for its detection.

In a Copy-Move forgery, a part of the image itself is copied and pasted into another part of the same image. This is usually performed with the intention to make an object "disappear" from the image by covering it with a segment copied from another part of the image. Textured areas, such as grass, foliage, gravel, or fabric with irregular patterns, are ideal for this purpose because the copied areas will likely blend with the background and the human eye cannot easily discern any suspicious artifacts. Because the copied parts come from the same image, its noise component, color palette, dynamic range, and most other important properties will be *compatible* with the rest of the image and thus will not be detectable using methods that look for incompatibilities in statistical measures in different parts of the image. To make the forgery even harder to detect, one can use the feathered crop or the retouch tool to further mask any traces of the copied-and-moved segments.

Examples of the Copy-Move forgery are given in Figures 2–4. Figure 2 is an obvious forgery that was created solely for testing purposes. In Figure 3, you can see a less obvious forgery in which a truck was covered with a portion of the foliage left of the truck (compare the forged image with its original). It is still not too difficult to identify the forged area visually because the original and copied parts of the foliage bear a suspicious similarity. Figure 4 shows another Copy-Move forgery that is much harder to identify visually. This image has been sent to the authors by a third party who did not disclose the nature or extent of the forgery. We used this image as a real-life test for evaluating our detection tools. A visual inspection of the image did not reveal the presence of anything suspicious.

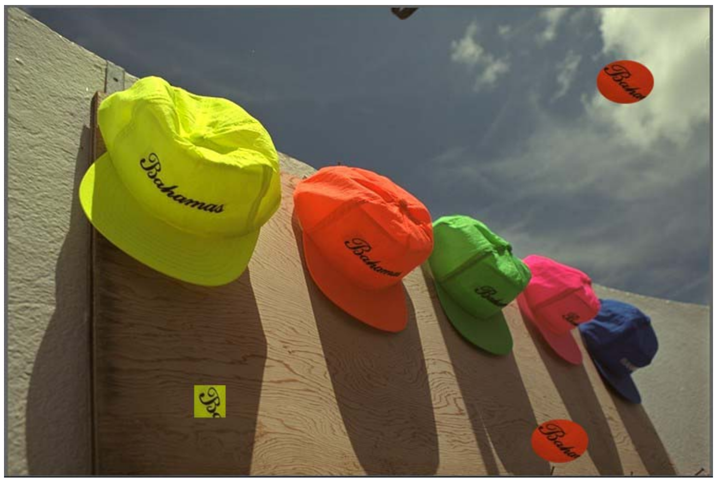
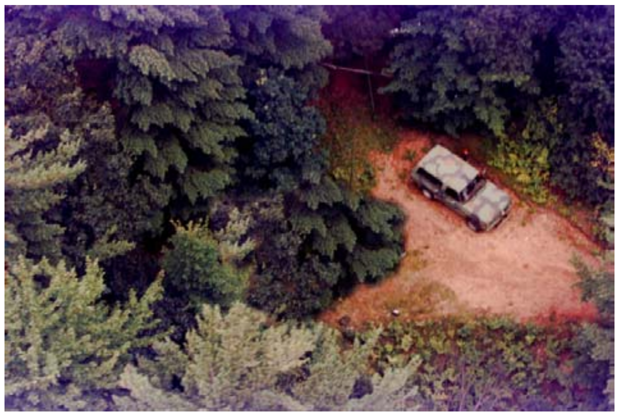


Figure 2 Test image "Hats".



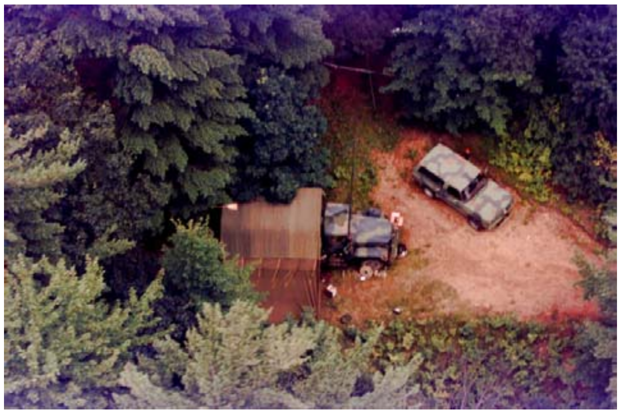


Figure 3 Forged test image “Jeep” (above) and its original version (below).

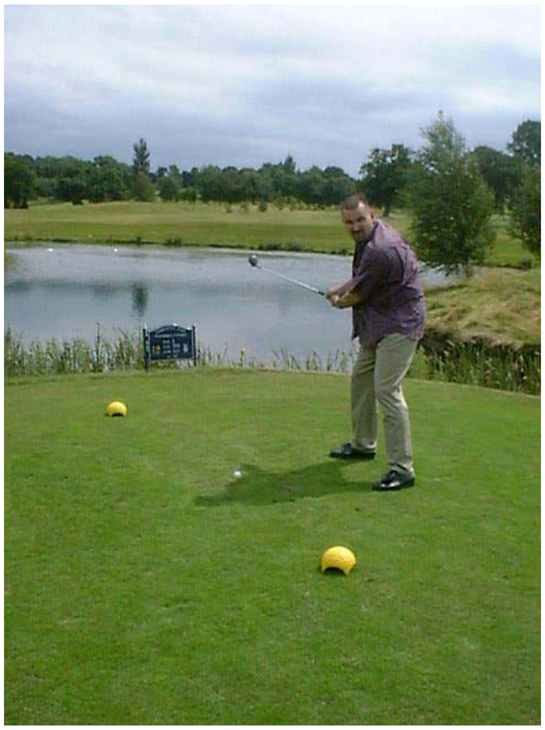


Figure 4 Test image “Golf” with an unknown original.

3. Detection of Copy-Move Forgery

Any Copy-Move forgery introduces a correlation between the original image segment and the pasted one. This correlation can be used as a basis for a successful detection of this type of forgery. Because the forgery will likely be saved in the lossy JPEG format and because of a possible use of the retouch tool or other localized image processing tools, the segments may not match exactly but only approximately. Thus, we can formulate the following requirements for the detection algorithm:

1. The detection algorithm must allow for an approximate match of small image segments
2. It must work in a reasonable time while introducing few false positives (i.e., detecting incorrect matching areas).
3. Another natural assumption that should be accepted is that the forged segment will likely be a connected component rather than a collection of very small patches or individual pixels.

In this section, two algorithms for detection of the Copy-Move forgery are developed – one that uses an exact match for detection and one that is based on an approximate match.

Before describing the best approach based on approximate block matching that produced the best balance between performance and complexity, two other approaches were investigated – Exhaustive search and Autocorrelation.

3.1 Exhaustive search

This is the simplest (in priciple) and most obvious approach. In this method, the image and its circularly shifted version (see Figure 5) are overlaid looking for closely matching image segments. Let us assume that x_{ij} is the pixel value of a grayscale image of size $M\times N$ at the position i, j. In the exhaustive search, the following differences are examined:

$$|x_{ij} - x_{i+k\ mod(M)\ j+l\ mod(N)}|,\ k = 0, 1, \ldots, M-1,\ l = 0, 1, \ldots, N-1 \text{ for all } i \text{ and } j.$$

It is easy to see that comparing x_{ij} with its cyclical shift $[k,l]$ is the same as comparing x_{ij} with its cyclical shift $[k',l']$, where $k'=M-k$ and $l'=N-l$. Thus, it suffices to inspect only those shifts $[k,l]$ with $1\le k \le M/2$, $1\le l \le N/2$, thus cutting the computational complexity by a factor of 4.

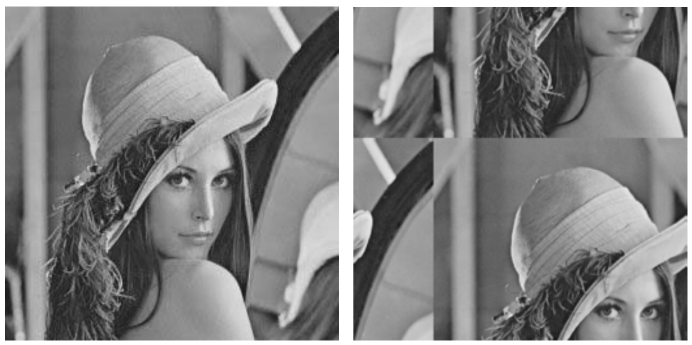


Figure 5 Test image “Lenna” and its circular shift.

For each shift $[k,l]$, the differences $\Delta x_{ij} = |x_{ij} - x_{i+k\ mod(M)\ j+l mod(N)}|$, are calculated and thresholded with a small threshold t. The threshold selection is problematic, because in natural images, a large amount of pixel pairs will produce differences below the threshold t. However, according to our requirements we are only interested in connected segments of certain minimal size. Thus, the thresholded difference Δx_{ij} is further processed using the morphological opening operation (see, for example [ImgProcBook]). The image is first eroded and then dilated with the neighborhood size corresponding to the minimal size of the copy-moved area (in experiments, the 10×10 neighborhood was used). The opening operation successfully removes isolated points.

Although this simple exhaustive search approach is effective, it is also quite computationally expensive. In fact, the computational complexity of the exhaustive search makes it impractical for practical use even for medium-sized images. An estimate of the computational complexity of the algorithm is given below.

During the detection, all possible shifts $[k,l]$ with $1\le k,\ l \le M/2$ need to be inspected. For each shift, every pixel pair must be compared, thresholded, and then the whole image must be eroded and dilated. The comparison and image processing require the order of MN operations for one shift. Thus, the total computational requirements are proportional to $(MN)^2$. For example, the computational requirements for an image that is twice as big are 16 times larger. This makes the exhaustive search a viable option only for small images.

3.2 Autocorrelation

The autocorrelation of the image x of the size $M\times N$ is defined by the formula:

$$r_{k,l} = \sum_{i=1}^{M}\sum_{j=1}^{N} x_{i,j}x_{i+k,j+l},\ i,k = 0,\dots,M-1,\ j,l = 0,\dots,N-1\,.$$

The autocorrelation can be efficiently implemented using the Fourier transform utilizing the fact that $r = x * \hat{x}$, where $\hat{x}_{ij} = x_{M+1-i,N+1-j}$, $i = 0, \dots, M-1$, $j = 0, \dots, N-1$. Thus we have

$$r = \mathsf{F}^{-1}\{\mathsf{F}(x)\ \ \mathsf{F}(\hat{x})\},$$

where F denotes the Fourier transform.

The logic behind the detection based on autocorrelation is that the original and copied segments will introduce peaks in the autocorrelation for the shifts that correspond to the copied-moved segments. However, because natural images contain most of their power in low-frequencies, if the autocorrelation r is computed directly for the image itself, r would have very large peaks at the image corners and their neighborhoods. Thus, we compute the autocorrelation not from the image directly, but from its high-pass filtered version.

Several high-pass filters were tested: Marr edge detector, Laplacian edge detector, Sobel edge detector, and noise extracted using the 3×3 Wiener filter (see, for example, [ImgProcBook]). The best performance was obtained using the 3×3 Marr filter.

Assuming the minimal size of a copied-moved segment is B, the autocorrelation copy-move detection method consists of the following steps:

1. Apply the Marr high-pass filter to the tested image.
2. Compute the autocorrelation r of the filtered image.
3. Remove half of the autocorrelation (Autocorrelation is symmetric.).
4. Set $r = 0$ in the neighborhood of two remaining corners of the entire autocorrelation.
5. Find the maximum of r, identify the shift vector, and examine the shift using the exhaustive method (this is now computationally efficient because we do not have to perform the

exhaustive search for many different shift vectors).

6. If the detected area is larger than B, finish, else repeat Step 5 with the next maximum of r.

Although, this method is simple and does not have a large computational complexity, it often fails to detect the forgery unless the size of the forged area is at least ¼ of linear image dimensions (according to our experiments).

Both the exhaustive search and the autocorrelation method were abandoned in favor of the third approach that worked significantly better and faster than previous approaches.

4. Detection of Copy-Move Forgery by Block Matching

4.1 Exact match

The first algorithm described in this section is for identifying those segments in the image that match exactly. Even though the applicability of this tool is limited, it may still be useful for forensic analysis. It also forms the basis of the robust match detailed in the next section.

In the beginning, the user specifies the minimal size of the segment that should be considered for match. Let us suppose that this segment is a square with $B \times B$ pixels. The square is slid by one pixel along the image from the upper left corner right and down to the lower right corner. For each position of the $B \times B$ block, the pixel values from the block are extracted by columns into a row of a two-dimensional array A with B^2 columns and $(M-B+1)(N-B+1)$ rows. Each row corresponds to one position of the sliding block.

Two identical rows in the matrix A correspond to two identical $B \times B$ blocks. To identify the identical rows, the rows of the matrix A are lexicographically ordered (as $B \times B$ integer tuples). This can be done in $MN\log_2(MN)$ steps. The matching rows are easily searched by going through all MN rows of the ordered matrix A and looking for two consecutive rows that are identical.

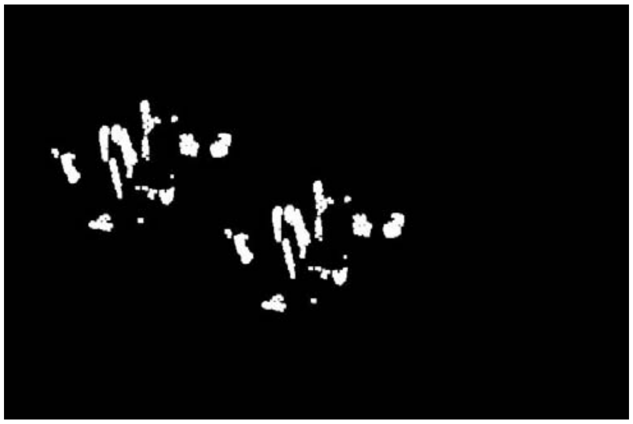


Figure 6 Results of the Block Match Copy-Detection forgery algorithm
(the exact match mode with block size B=4).

The matching blocks found in the BMP image of Jeep (Figure 3) for B=8 are shown in Figure 6. The blocks form an irregular pattern that closely matches the copied-and-moved foliage. The fact that the blocks form several disconnected pieces instead of one connected segment indicates that the person who did the forgery has probably used a retouch tool on the pasted segment to cover the traces of the forgery. Note that if the forged image had been saved as JPEG, vast majority of identical blocks would have disappeared because the match would become only approximate and not exact (compare the detection results with the robust match in Figure 8). This also why the exact match analysis of images from Figures 2 and 4 did not show any exactly matching blocks. In the next section, the algorithm for the robust match is given and its performance evaluated.

4.2 Robust match

The idea for the robust match detection is similar to the exact match except we do not order and match the pixel representation of the blocks but their robust representation that consists of quantized DCT coefficients. The quantization steps are calculated from a user-specified parameter Q. This parameter is equivalent to the quality factor in JPEG compression, i.e., the Q-factor determines the quantization steps for DCT transform coefficients. Because higher values of the Q-factor lead to finer quantization, the blocks must match more closely in order to be identified as similar. Lower values of the Q-factor produce more matching blocks, possibly some false matches.

The detection begins in the same way as in the exact match case. The image is scanned from the upper left corner to the lower right corner while sliding a $B\times B$ block. For each block, the DCT transform is calculated, the DCT coefficients are quantized and stored as one row in the matrix A. The matrix will have $(M-B+1)(N-B+1)$ rows and $B\times B$ columns as for the exact match case.

The rows of A are lexicographically sorted as before. The remainder of the procedure, however, is different. Because quantized values of DCT coefficients for each block are now being compared instead of the pixel representation, the algorithm might find too many matching blocks (false matches). Thus, the algorithm also looks at the mutual positions of each matching block pair and outputs a specific block pair only if there are many other matching pairs in the same mutual position (they have the same shift vector). Towards this goal, if two consecutive rows of the sorted matrix A are found, the algorithm stores the positions of the matching blocks in a separate list (for example, the coordinates of the upper left pixel of a block can be taken as its position) and increments a shift-vector counter C. Formally, let (i_1, i_2) and (j_1, j_2) be the positions of the two matching blocks. The shift vector s between the two matching blocks is calculated as

$$s = (s_1, s_2) = (i_1 - j_1,\ i_2 - j_2).$$

Because the shift vectors $-s$ and s correspond to the same shift, the shift vectors s are normalized, if necessary, by multiplying by -1 so that $s_1 \geq 0$. For each matching pair of blocks, we increment the normalized shift vector counter C by one:

$$C(s_1, s_2) = C(s_1, s_2) + 1 .$$

The shift vectors are calculated and the counter C incremented for each pair of consecutive matching rows in the sorted matrix A. The shift vector C is initialized to zero before the algorithm starts. At the end of the matching process, the counter C indicates the frequencies with which different normalized shift vectors occur. Then the algorithm finds all normalized shift vectors $s^{(1)}$, $s^{(2)}$, …, $s^{(K)}$, whose occurrence exceeds a user-specified threshold T: $C(s^{(r)}) > T$ for all $r = 1, …, K$. For all normalized shift vectors, the matching blocks that contributed to that specific shift vector are colored with the same color and thus identified as segments that might have been copied and moved.

The value of the threshold T is related to the size of the smallest segment that can be identified by the algorithm. Larger values may cause the algorithm to miss some not-so-closely matching blocks, while too small a value of *T* may introduce too many false matches. We repeat that the Q-factor controls the sensitivity of the algorithm to the degree of matching between blocks, while the block size *B* and threshold *T* control the minimal size of the segment that can be detected.

For the robust match, we have decided to use a larger block size, *B*=16, to prevent too many false matches (larger blocks have larger variability in DCT coefficients). However, this larger block size means that a 16×16 quantization matrix must be used instead of simply using the standard quantization matrix of JPEG. We have found out from experiments that all AC DCT coefficients for 16×16 blocks are on average 2.5 times larger than for 8×8 blocks and the DC term is twice as big. Thus, the quantization matrix (for the Q-factor *Q*) that is used for quantizing the DCT coefficients in each 16×16 block has the following form

$$Q_{16} = \begin{pmatrix} Q'_8 & 2.5q_{18}I \\ 2.5q_{81}I & 2.5q_{88}I \end{pmatrix}, \text{where } Q'_8 = \begin{pmatrix} 2q_{00} & 2.5q_{12} & \dots & 2.5q_{18} \\ 2.5q_{21} & 2.5q_{22} & \dots & 2.5q_{28} \\ \dots & \dots & \dots & \dots \\ 2.5q_{81} & 2.5q_{82} & \dots & 2.5q_{88} \end{pmatrix}$$

and q_{ij} is the standard JPEG quantization matrix with quality factor *Q* and *I* is an 8×8 unit matrix (all elements equal to 1). We acknowledge that this form is rather ad hoc, but because the matrix gave very good performance in practical tests and because small changes to the matrix influence the results very little, we did not investigate the selection of the quantization matrix further.

Note regarding color images: In both Exact and Robust Match, if the analyzed image is a color image, it is first converted to a grayscale image using the standard formula $I = 0.299\ R + 0.587\ G + 0.114\ B$, before proceeding with further analysis.

5. Robust Match Performance

We have implemented the detection algorithm in C and tested on all three forged images. The output from the detection algorithm are two images. In the first image, the matching blocks are all colored with the same tint. In Figure 7 below, a yellow tint was used for rendering the copied-moved segments. The second output image shows the copied-moved blocks on a black background. Different colors correspond to different shift vectors. Note that the algorithm likely falsely identified two matching segments in the sky. It is to be expected that flat, uniform areas, such as the sky, may lead to false matches. Human interpretation is obviously necessary to interpret the output of any Copy-Move detection algorithm.

Figure 7 shows the output of our detection routine applied to Figure 4. Although visual inspection did not reveal any suspicious artifacts, the algorithm identified several three large segments in the grass that were copied and pasted perhaps to cover up some object on the grass. Knowing the program output, we were able to visually confirm a suspicious match in the otherwise "random looking" grass texture.

Figure 8 shows the result of robust match for "Hats" (Figure 2). Because the elliptic area on the orange hat has been copied to two other locations, three different shift vectors have been correctly found by the program (green, blue, and red colors). Note that the red color is almost completely covered by the green color (upper right ellipse) and the blue ellipse (bottom right). One green ellipse is covered by the blue ellipse in the center of the image. The fourth color, yellow, identifies the second copied and moved square segment.

The result of the robust match for the "Jeep" image (Figure 3) is shown in Figure 9. The copied-moved foliage is captured much more accurately than for the exact match (c.f., Figure 6).

Acknowledgements

The work on this paper was supported by Air Force Research Laboratory, Air Force Material Command, USAF, under a research grant number F30602-02-2-0093. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U. S. Government.

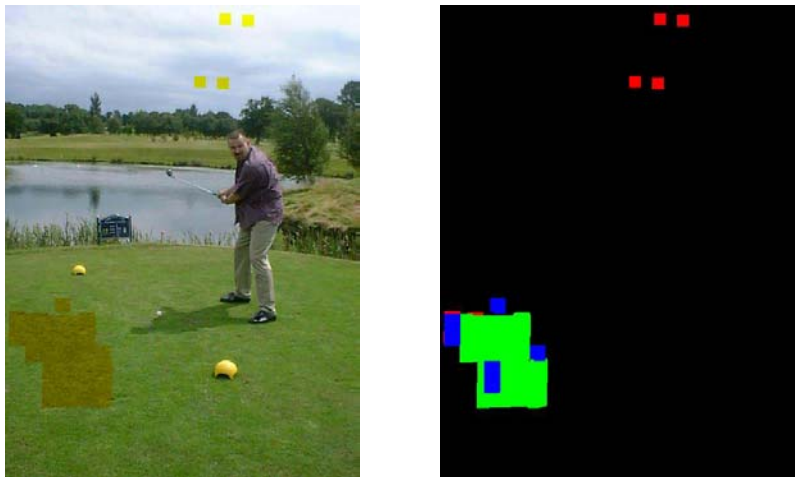


Figure 7

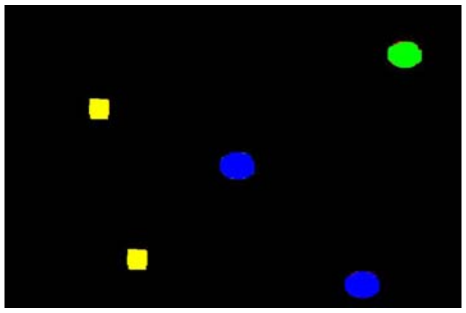


Figure 8

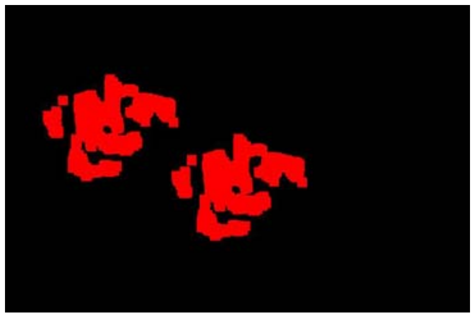


Figure 9

References

[1] J. Fridrich, "Methods for "Methods for Tamper Detection in Digital Images", *Proc. ACM Workshop on Multimedia and Security*, Orlando, FL, October 30–31, 1999, pp. 19–23.

[2] S. Saic, J. Flusser, B. Zitová, and J. Lukáš, "Methods for Detection of Additional Manipulations with Digital Images", *Research Report*, *Project RN19992001003 "Detection of Deliberate Changes in Digital Images",* ÚTIA AV ČR, Prague, December 1999 (partially in Czech).

[3] J. Lukáš, "Digital Image Authentication", *Workshop of Czech Technical University 2001*, Prague, Czech Republic, February 2001.

[4] Img. Processing book