

# TAMPER DETECTION AND SELF-RECOVERY ON GRAYSCALE IMAGES

## GRİ TONLAMALI GÖRÜNTÜLER ÜZERİNDE MÜDAHALE TESPİTİ VE KURTARMA

Hüseyin Bilal MACİT

Mehmet Akif Ersoy University, Tefenni MYO, [hbmact@mehtetakif.edu.tr](mailto:hbmact@mehtetakif.edu.tr)

Orhan GÜNGÖR

Mehmet Akif Ersoy University, Tefenni MYO, [orhangungor@mehtetakif.edu.tr](mailto:orhangungor@mehtetakif.edu.tr)

**ABSTRACT:** Tampering on analogue images is quite difficult. However, digital images can be easily tampered even with free software. Malicious tampering on a digital image, such as a photo, financial document, or official paper image, can lead to irreversible material losses, time losses and similar problems. Therefore, it is important to quickly detect and repair the tampering to the image. In this study, a tamper detection and repair method is presented on grayscale images. The method works with low process complexity. The original image is divided into 2x2 equal blocks and the detail information of each block is written to other blocks. Thus, tamper in one block can be detected and repaired by checking other blocks simply. The proposed method has been tested with different tamper ratios on a test image and the visual and mathematical results obtained are shown.

**Key words:** tamper detection, LSB, image recovery, grayscale

**ÖZET:** Analog görüntüler üzerinde müdahale yapmak oldukça zordur. Ancak dijital görüntülere ücretsiz yazılımlarla bile kolayca müdahale edilebilmektedir. Bir fotoğraf, finansal evrak veya resmi evrak görüntüsü gibi bir dijital görüntü üzerinde yapılan kötü niyetli müdahale, geri dönülemez maddi zararlara, zaman kayıplarına ve buna benzer sorunlara yol açabilir. Bu nedenle görüntüye yapılan müdahalenin hızlı bir şekilde tespit edilmesi ve onarılması önemlidir. Bu çalışmada gri tonlamalı görüntüler üzerinde bir müdahale tespiti ve onarım yöntemi sunulmuştur. Yöntem düşük işlem karmaşıklığı ile çalışmaktadır. Orijinal görüntü 2x2 eşit bloklara bölünmekte ve her bloğun detay bilgileri diğer bloklara yazılmaktadır. Böylece bir blokta gerçekleşen müdahale diğer bloklar kontrol edilerek tespit edilebilmekte ve basitçe onarılabilir. Önerilen yöntem bir test görüntüsü üzerinde farklı müdahale oranları ile test edilmiş ve elde edilen görsel ve matematiksel sonuçlar gösterilmiştir.

**Anahtar sözcükler:** müdahale tespiti, LSB, görüntü onarımı, gri tonlamalı

## INTRODUCTION

There has been a rapid growth in data storing and sharing in last decade (Wang et al., 2015). Billions of images can transfer and store in just a second. This communication speed brings some security problems. People can easily make changes to an image file even using their mobile devices. Making unauthorized change on an image is called tampering. Copy-move is the most common kind of image tampering, where one needs to cover a part of the image in order to add or remove information (Sharma and Abrol, 2013). Image security techniques are used to avoid image tampering, common methods for image security and integrity are digital signing and watermarking (Tien and Shinfeng, 2008). A watermark is divided into three categories; fragile, semi fragile and robust (Yu and Liao, 2001). Robust watermark should not be distorted even if the image is tampered. However, fragile watermark can be corrupted by geometric tampering. For this reason, usually a fragile or semi-fragile watermark is used for tamper detection. (Lin and Chang, 2000). First, a watermark is emitted to the entire image and the image is sent in the transmission medium. The receiving party checks the integrity of the watermark in the image. If any distortion detected in watermark, the image is considered as tampered on transmission line. (Liu et al., 2007). This method is an effective method for detecting tampering, but it cannot restore tampered data. Usually the original image is needed to restore the tampered data. Simply; the tampered image is extracted from the original image. Pixels with absolute values greater than zero are considered as tampered. In this study, an application has been developed to recover tampered region of an image without having any information about original image.

## METHOD

The method developed in this study works in three stages. In the first stage the original image  $I$  is self-watermarked and processed image  $P$  is obtained. Processed image  $P$  can be left vulnerable in storage media or communication channel. In the second step, the user who obtains the image assumes that the  $P$  image is a tampered image  $T$  and a

tamper detection algorithm runs on the image  $T$ . In the third stage, if a tampered region detected in  $T$  image, self-recovery algorithm is performed and recovered image  $R$  is obtained.

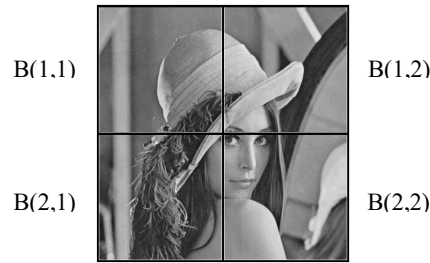
### Image Self-watermarking

Assume that image  $I$  is the original image,  $m$  is the number of horizontal pixels and  $n$  is the number of vertical pixels. The primary goal is to store the information of each pixel in the farthest pixel possible. For this, the original image  $I$  is divided into 4 equal blocks:  $B(1,1)$ ,  $B(1,2)$ ,  $B(2,1)$  and  $B(2,2)$  as shown in figure 1. The size of each block is  $(m/2) \times (n/2)$ . The pseudo code of this process is given below.

```

Blocks = cell(8,8)
count i = 0
for i = 1 → (m/2)+1 STEP m/2
    count i += 1
    count j = 0
    for j = 1 → (n/2)+1 STEP n/2
        count j += 1
        Blocks{count i,count j} = I(i:i+(m/2)-1,j:j+(n/2)-1)
    end
end
end

```



**Figure 1. Original Image  $I$  Divided into Blocks**

The main goal is to hide the maximum possible information of each block to other blocks. To do this, each block hides data to next three blocks. In the data hide stage, the selected block is the source block, and the other three blocks are the destination block. This process is repeated for every block as shown in figure 2. The first block after the source block is called *FirstDestinationBlock*, the second block is called *SecondDestinationBlock*, and the third block is called *ThirdDestinationBlock*. The pixel information of each block is embedded in the same indexed pixel of the next block. Thus, the maximum distance between the pixels is provided. Pseudo code of embedding pixel information is given below.

```

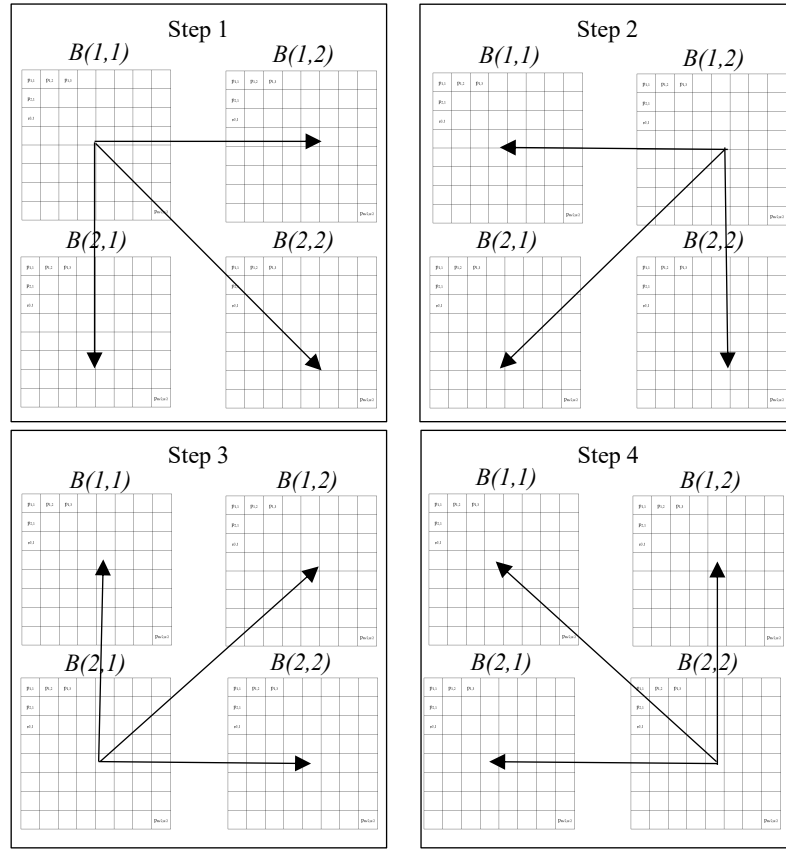
for i = 1 → ( m/2 ) STEP 1
    for j = 1 → ( n/2 ) STEP 1
        SourcePixel = DecimalToBinary ( SourceBlock (i,j), 8 )

        DestinationPixel = DecimalToBinary ( FirstDestinationBlock (i,j), 8 )
        DestinationPixel (6) = SourcePixel (1)
        FirstDestinationBlock (i,j) = DecimalToBinary (DestinationPixel)

        DestinationPixel = DecimalToBinary ( SecondDestinationBlock (i,j), 8 )
        DestinationPixel (7) = SourcePixel (2)
        SecondDestinationBlock (i,j) = DecimalToBinary (DestinationPixel)

        DestinationPixel = DecimalToBinary ( ThirdDestinationBlock (i,j), 8 )
        DestinationPixel (8) = SourcePixel (3)
        ThirdDestinationBlock (i,j) = DecimalToBinary (DestinationPixel)
    end
end
end

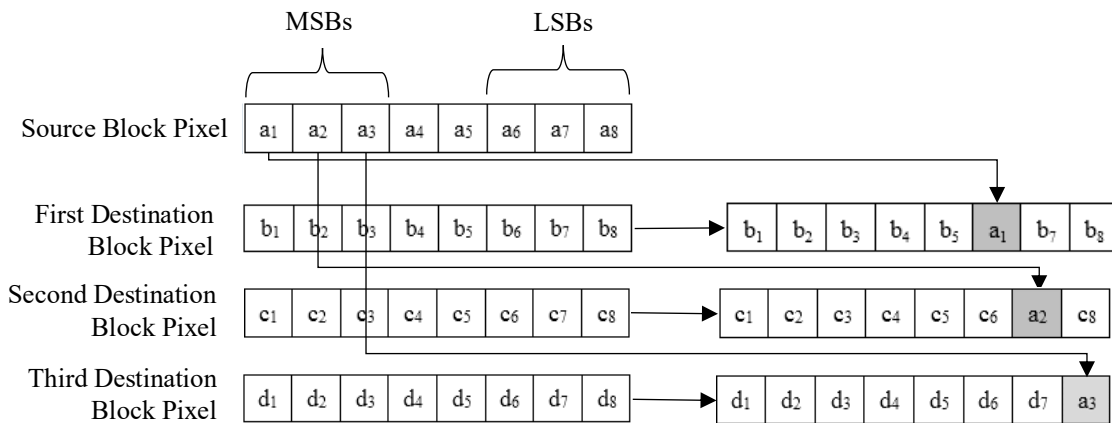
```



**Figure 2. Hiding Block Data to Another Blocks**

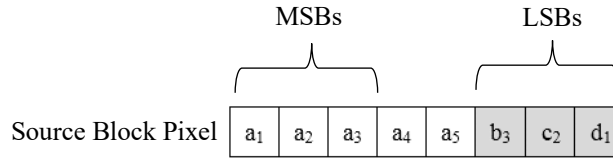
At each step, the selected pixel of the source block is converted to a 8 bits of binary array and transferred to the *SourcePixel* array. The pixels of the target blocks corresponding to the selected pixel are also converted to 8 bits of binary code and transferred to the *DestinationPixel* array respectively. The left side bits of the binary array are called the Most Significant Bits (MSB) and the the right side bits of the binary array are called the Least Significant Bits (LSB). According to these data, three-step embedding process is:

1. First MSB of source pixel *SourcePixel*(1) embeds to *DestinationPixel*(6) which refers to third LSB of the first destination block
2. Second MSB of source pixel *SourcePixel*(2) embeds to *DestinationPixel*(7) which refers to second LSB of the second destination block
3. Third MSB of source pixel *SourcePixel*(3) embeds to *DestinationPixel*(8) which refers to first LSB of the third destination block



**Figure 3. Three Steps Block Pixel Data Embedding Process**

After these processes, the first three MSBs of all pixels of each block are embedded in other blocks' pixels. Source Block Pixel shown in figure 3 has converted to the form as shown in figure 4 after whole algorithm has ran.

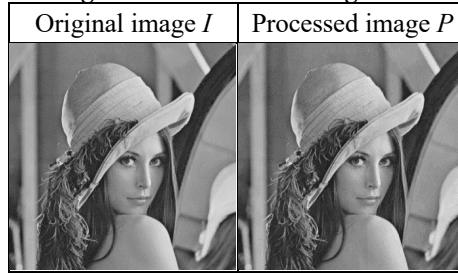


**Figure 4. Result Form of Source Block Pixel**

As is known, in the 8-bit data encoding system; first MSB refers to  $2^8$ , the second MSB refers to  $2^7$  and the third one refers to  $2^6$ . Thus, first three MSBs may have a value up to  $2^8+2^7+2^6=128+64+32=224$ . On the other hand, first LSB which is the 8.bit of array refers to  $2^0$ , the second LSB which is the 7.bit of array refers to  $2^1$  and the third one refers to  $2^2$  which is the 6.bit of array. Thus first three LSBs may have a value up to  $2^2+2^1+2^0=4+2+1=7$ . So three LSBs store maximum  $7/256 = 2.7\%$  of the pixel value and three MSBs store maximum  $224/256 = 87.5\%$  of the pixel value. In this study,  $2.7\%$  of the image has been abandoned to make secure  $87.5\%$  of image data.

In the last step, the processed blocks are combined to create the processed image  $P$ . Table 1 shows the original image  $I$  and the processed image  $P$  as a comparison. There is no difference between  $I$  and  $P$  images that can be easily detected by Human Visual System (HVS).

**Table 1. Original Image  $I$  and Processed Image  $P$  for HVS Comparison**



After embedding process is performed on each block, the image  $P$  may be sent securely in any storage or transmission medium. Image  $P$  can provide self-security.

### Image Tamper Detection

T tampered image is divided into  $2 \times 2$  equal blocks for tamper detection. The method applied for block division is exactly the same as that of the self-watermarking stage. If there is no tamper in the image, three MSBs of the pixels of each block must be truly embedded in the LSBs of other three blocks. First, all three MSBs and three LSBs are correctly compared to detect tampering. A threshold value is used here. The user can assign the threshold value as 1, 2 or 3. If the threshold value is equal to 1, that pixel is considered to be tampered even one of the LSBs of the corresponding pixel is unmatched. If the threshold value is equal to 2, that pixel is considered to be tampered even two of the LSBs of the corresponding pixel is unmatched. If the threshold value is equal to 3, that pixel is considered to be tampered when all of the LSBs of the corresponding pixel is unmatched. The pseudo code used to control of MSBs and LSBs for each pixel is as follows.

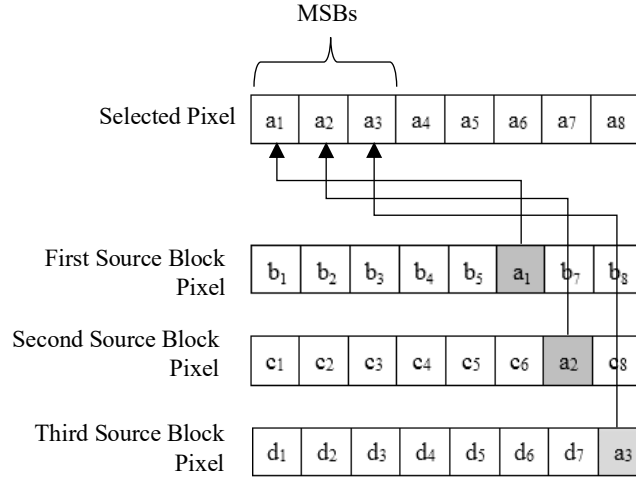
```

source=dec2bin(Blocks{1,1}(i,j),8);
destination_1=dec2bin(Blocks{1,2}(i,j),8);
destination_2=dec2bin(Blocks{2,1}(i,j),8);
destination_3=dec2bin(Blocks{2,2}(i,j),8);
TamperSize =4;
if (destination_1(6)==source(1))
    TamperSize= TamperSize -1;
if (destination_2(7)==source(2))
    TamperSize = TamperSize -1;
if (destination_3(8)==source(3))
    TamperSize = TamperSize -1;

```

## Image Recovery

If the selected number of MSBs and the corresponding LSBs are not the same in the selected pixel, the algorithm considers this pixel to be tampered. In this case, the LSBs of the corresponding pixels are written to the MSBs of the selected pixel, as shown in figure 5 for recovery of the pixel.






**Figure 5. Pixel Recovery Process**

After the tamper control and recovery are performed for all pixels of block  $B(1,1)$ , same process runs for every pixel of  $B(1,2)$ ,  $B(2,1)$  and  $B(2,2)$  respectively. After all, recovered image  $R$  is obtained.





## EXPERIMENTAL RESULTS AND CONCLUSIONS

A 512x512 piksel Lena image is used to test proposed method. The most important reason for selecting this image is that it is the most used image in the literature in the field of data hiding. Artificial cutting operations have been made on Lena image with ratios 0.1%, 1%, 10% and 25% and tried to recover image by the method developed. HVS is not a sufficient perception to assess the performance of the method. So, mathematical similarity ratios between processed image  $p$ , original image  $I$ , tampered image  $T$  and recovered image  $R$  are calculated with Mean Squared Error (MSE), Structured Similarity Index (SSIM) and Peak Signal to Noise Ratio (PSNR) measurements. Watermarked images with PSNR values over 28 dB and SSIM values over 0.96 have very accepted perceptual quality (Taha et al., 2018). All the results obtained from Lena test image are shown in tables 2 to 5.





**Table 2. Mathematical Similarity Results of Proposed Method With 0.1% Tampering on Lena Image**

Tamper percentage	Tampered image	Similarity values					
		Original image versus tampered image			Processed image versus tampered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
0.1%		22.0634	34.6941	0.9516	10.9509	37.7363	0.9985
Threshold value	Recovered image	Similarity values					
		Original image versus recovered image			Processed image versus recovered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
1		11.5370	37.5099	0.9517	0.8722	48.7244	0.9986
2		14.4875	36.5209	0.9521	3.6105	42.5551	0.9991
3		22.0634	34.6941	0.9516	10.9509	37.7363	0.9985





**Table 3. Mathematical Similarity Results of Proposed Method With 1% Tampering on Lena Image**

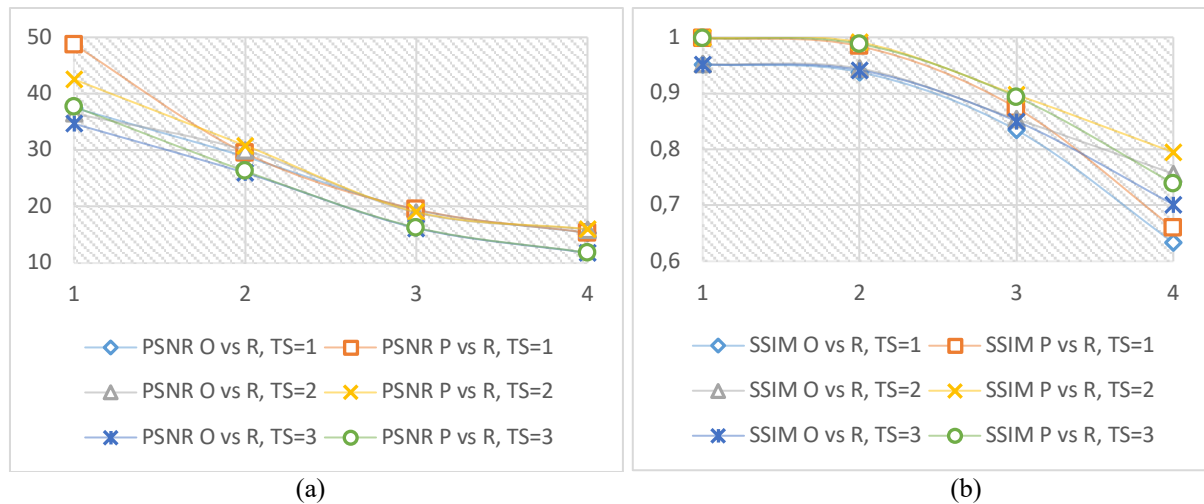
Tamper percentage	Tampered image	Similarity values					
		Original image versus tampered image			Processed image versus tampered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
1%		164.7717	25.9620	0.9416	153.6431	26.2657	0.9883
Threshold value	Recovered image	Similarity values					
		Original image versus recovered image			Processed image versus recovered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
1		84.6923	28.8524	0.9379	73.0577	29.4941	0.9842
2		64.9836	30.0028	0.9443	53.6746	30.8331	0.9911
3		163.6010	25.9929	0.9416	152.4400	26.2998	0.9883

**Table 4. Mathematical Similarity Results of Proposed Method With 10% Tampering on Lena Image**

Tamper percentage	Tampered image	Similarity values					
		Original image versus tampered image			Processed image versus tampered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
10%		1562.3	16.1932	0.8502	1528.9	16.2870	0.8936
Threshold value	Recovered image	Similarity values					
		Original image versus recovered image			Processed image versus recovered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
1		742.9157	19.4214	0.8343	727.6868	19.5114	0.8739
2		831.4489	18.9324	0.8540	806.5203	19.0647	0.8974
3		1560.5	16.1981	0.8502	1527.1	16.2921	0.8936

**Table 5. Mathematical Similarity Results of Proposed Method With 25% Tampering on Lena Image**

Tamper percentage	Tampered image	Similarity values					
		Original image versus tampered image			Processed image versus tampered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
25%		4330.3	11.7656	0.7015	4277.4	11.8190	0.7391
Threshold value	Recovered image	Similarity values					
		Original image versus recovered image			Processed image versus recovered image		
		MSE	PSNR	SSIM	MSE	PSNR	SSIM
1		1905	15.3320	0.6335	1891.7	15.3622	0.6596
2		1658.3	15.9341	0.7564	1623.4	16.0266	0.7948
3		4318.2	11.7778	0.7016	4265	11.8316	0.7393



**Figure 6. (a) Threshold to PSNR Graph on Lena Image (b) Threshold to SSIM Graph on Lena Image**

As it is seen on figure 6, change of threshold value doesn't affect on PSNR and SSIM results. Most effective factor for PSNR and SSIM values is the tamper ratio applied to image. As understood from these two graphs, increase or decrease of threshold value does not affect to the recovery process of image  $T$  to image  $I$ . However, it affects the amount of recovery in the region with high tamper density of the image.

## CONCLUSIONS

Many personal images are shared on the Internet and almost all of these images are vulnerable to tampering. This may seem innocuous. However, there are many negative examples such as people who are on trial for a crime they did not commit, wet signature counterfeiting on bank receipts, checks or other scanned documents and changes in articles. The proposed method can be useful and effective for grayscale image security.

## REFERENCES

- Lin, C.Y., Chang, C., (2000). Semi-fragile watermarking for authenticating JPEG visual content, *SPIE International Conference on Security and Watermarking of Multimedia Contents*, USA.
- Liu, S.H., Yao, H.X., Gao, W., Liu, Y.L., (2007). An image fragile watermark scheme based on chaotic image pattern and pixelpairs, *Applied Mathematics and Computation*, (185)2, 869-882.
- Sharma, D., Abrol, P., (2013). Digital Image Tampering – A Threat to Security Management, *International Journal of Advanced Research in Computer and Communication Engineering*, (2)10, 4120-4123.
- Taha, B., Ngadiran, R., Ehkan, P., Sultan, M.T., (2018). Image Tamper Detection and Recovery Using Lifting Scheme Based Fragile Watermarking, *Journal of Theoretical and Applied Information Technology*, (96)8, 2307-2316.
- Tien, Y., Shinfeng, D., (2008). Dual watermark for image tamper detection and recovery, *Pattern Recognition*, 3497-3506.
- Yu, G., Liao, H., (2001). Mean quantization based fragile watermarking for image authentication, *Optical Engineering*, 40(7), 1396-1408.
- Wang, H.C., Chen, W.M., Yi-Lee, P., (2015). Image Tamper Detection and Recovery Based on Dilation and Chaotic Mixing, *Computer Science and Information Technology*, 3(4), 127-132.