CHAPTER 3

BLOCK MATCHING METHOD/ COPY-MOVE FORGERY DETECTION

3.1 Preamble

One of the main purposes of image forgery is to conceal an object in the image. Normally the simplest way is to copy another portion of the same image and paste it over the object to be concealed such that it blends with the remaining part of the image and leaves no clues of tampering. To cover up any traces of forgery, retouching may be done at the periphery of the pasted object. An example of this kind is shown in figure 3.1.

Figure 3.1 : A truck in the original (left) is removed by covering with surrounding foliage

The second purpose may be to replicate some object in an image. Then, a copy of the desired object in the image is made and pasted at one or more appropriate locations in the image. Once again, to cover up any traces of forgery, retouching may be done at the periphery of the pasted object. An example of this kind is shown in figure 3.2.

Figure 3.2 : A boat with a yellow-orange hoist in the original (left) is replicated

There can also be a situation when an object from another image is copied and pasted at multiple locations in the forged image.

The Copy-Move forgery described above introduces a correlation between the original image object and the pasted one. This correlation can be used as a basis for a successful detection of this type of forgery. Because of the possibility of retouching and saving of the image in a lossy compressed format like JPEG, the two objects may match only approximately and not exactly.

In this chapter, a strategy that can efficiently detect and localize duplicated regions in an image is developed from the simplest and seemingly obvious techniques.

3.2 Exhaustive Search

The most obvious technique for detecting copied and pasted (duplicated) regions in the same image is to overlay the image and its circularly shifted version and search for closely matching image segments.

If the size of the image is MxN and x_{ij} is the pixel value at the position (i, j), the following differences are examined:

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

……..…..(3.1)

Since 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, it is sufficient to inspect only shifts $1 \leq k \leq M/2$, $1 \leq l \leq N/2$. Hence the computational complexity is reduced by a factor of 4.

For each shift [k,l], the differences $\Delta xij = | xij - xi+k \bmod(M)\, j+l \bmod(N) |$, 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. Also it should be verified if the matching pixels are forming a connected segments of certain minimal size. Even though this approach is effective, it is computationally very costly. For this purpose, in each shift after every pixel pair is compared and thresholded, the whole image is to be eroded and dilated to identify if any segment of minimal size is present. Hence the time complexity is proportional to $(MN)^2$. Thus exhaustive search is viable only for small images and is impractical even for medium-sized images. [10]

3.3 Autocorrelation

The next obvious technique for detecting duplicated regions in an image is to use the autocorrelation because the original and copied segments will introduce peaks in the autocorrelation for the shifts that correspond to the copied-moved segments.

For an image of size MxN, the autocorrelation is defined by the formula:

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

……..….(3.2)

This can be implemented using the Fourier transform because of the fact that:

$r = x * \hat{x}$, where $\hat{x}_{ij} = x_{M+1-i,N+1-j}$, i = 0, …, M–1, j = 0, …, N–1.

……..……...(3.3)

Hence, $r = F^{-1}\{F(x)\,F(\hat{x})\}$, where F denotes the Fourier transform.

As the 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. Hence, the autocorrelation is not computed from the image directly, but from its high-pass filtered version.

Although not computationally intensive, unless the size of the forged area considerably large (almost ¼ the size of the image) false positives are more. [10]

3.4 Block Match

A good method for detecting copy-move forgery is to verify if a set of blocks of pixels in a region of the image matches with another in a different region of the image. That is, the image is divided into *n* non-overlapping blocks, and each block is compared with the remaining ones. But, selecting the size of the block is difficult. If the size of the block is larger than the forged area, an exact match of the blocks does not result. If the size of block is smaller, the forged area may cross the boundaries of adjacent blocks and then also exact matches would not result. If the size of the block size is made very small, the matching process becomes computationally intensive, particularly with large images. Also uniform areas in the original image will be shown as duplicates. This kind of block matching can be termed as non-overlapped block matching.

3.5 Exact Match

A better alternative is to select overlapping blocks. Blocks of size bxb pixels are selected from the top-left corner, moving right and down, to the bottom-right corner one pixel at a time along the image. For each block the pixel values are extracted by columns into a row of a two-dimensional array A with (M–b+1) (N–b+1) rows and b^2 columns. Two identical rows in the matrix A correspond to two identical bxb blocks. To recognize the identical rows easily and quickly, the rows of the matrix A are lexicographically sorted. Matching rows can be easily searched by going through the rows of the ordered matrix A and looking for two successive rows that are identical.

The steps in the Exact Match algorithm are as follows:

1. **Grayscale conversion**: If the given image is a colour image, convert it in to grayscale.
2. **Forming overlapped blocks**: Divide the grayscale image is into overlapped blocks of size (bxb).
3. **Extracting features**: For each of the (M-b+1)*(N-b+1) blocks, extract the features. Here, pixel values are the features. Store the extracted features of each block as rows of a matrix.
4. **Lexicographic sorting**: Sort the matrix in ascending order.
5. **Matching process**: Search successive identical rows and get the respective block positions.

3.5.1 Implementation Details

The Exact Match algorithm was implemented using MATLAB software. Eighty images obtained from various sources were used for testing the algorithm. Colour images were converted into grayscale using the formula I = 0.299 R + 0.587 G + 0.114 B. Images were resized to 128x128 pixels. Blocks of size of 8x8 pixels were considered.

Duplicate regions found by matching blocks found in the forged image 3.1 are shown in the figure 3.3.

Figure 3.3 : Copied and Pasted regions in the forged image 3.1

Discontinuities in the detected regions may be due to some retouching operation done after the forgery. But, if the forged image had been saved as JPEG, many of identical blocks would have disappeared because the match would become only approximate and not exact.

Figure 3.4 shows a forgery that could not be detected with a block size of 8x8 as retouching has been done after forgery. However when the block size was reduced to 3x3, some blocks are shown as forged but many are false matches.

Figure 3.4 : Original Image (left), Forged Image (middle), and result of Exact Match Algorithm with a block size of 3x3

It is observed that the Exact Match algorithm can detect plain Copy-Move forgeries but produces many false matches in retouched forgeries.

3.6 Robust Match

The best alternative to detect copy-move forgery is Robust Match where instead of matching the pixel representation of blocks, their robust representations are matched. One of the robust representations is the quantized DCT (Discrete Cosine Transform) coefficients. The advantage of DCT is that the signal energy would be concentrated on the first few coefficients, while most other coefficients are negligibly small. Therefore, the changes in high frequencies, which would occur due to the operations such as noise addition, compression, and retouching, do not affect these first coefficients greatly.

3.6.1 Discrete Cosine Transform (DCT)

Discrete Cosine Transform expresses a finite sequence of data points in termsof a sum of cosine functions oscillating at different frequencies and it is a Fourier-related transform similar to the Discrete Fourier Transform (DFT), but using only real numbers. DCT finds numerous applications in signal processing where small high-frequency components can be discarded. Lossy compression techniques like MP3 and JPEG utilize DCTs. The most common variant of Discrete Cosine Transform are the Type-II and Type-III which are popularly called the DCT and the Inverse DCT respectively.

In image processing, transform coding relies on the premise that pixels in an image exhibit a certain level of correlation with their neighboring pixels. Therefore transformation is defined to map this spatial (correlated) data into transformed (uncorrelated) coefficients and DCT attempts to decorrelate the image data.

DCT represents an image as a sum of sinusoids of varying magnitudes and frequencies. DCT is often used in image processing because of its strong energy compaction property and decorrelation property. Decorrelation is the removal of redundancy between neighboring pixels. This leads to uncorrelated transform coefficients, which can be encoded independently. Energy compaction property is the ability to pack input data into as few coefficients as possible. For a typical image, most of the visually significant information about the image is concentrated in just a few coefficients of the DCT [78].

Two-dimension DCT of an MxN image is given by the equation:

$$c(u,v) = \alpha(u)\,\alpha(v) \sum_{x=0}^{N-1} \sum_{u=0}^{M-1} f(x,y) \cos(\pi(2x+1)u/2N) \cos(\pi(2y+1)v/2M)$$

where $$\alpha(i) = \begin{cases} \sqrt{1/N} & for\ \ i = 0 \\ \sqrt{2/N} & for\ \ i \neq 0 \end{cases}$$

and u = 0, 1, 2, .., N-1 and v= 0, 1, 2, .., M-1……..(3.4)

The values c are called the DCT coefficients of the image.

$$\alpha(u)\,\alpha(v) \cos(\pi(2x+1)u/2N) \cos(\pi(2y+1)v/2M)$$

……..(3.5)

The above equation is called the basis function of the DCT.

For an 8x8 matrix, the 64 basis functions are shown in the Figure 3.5. Horizontal frequencies increase from left to right, and vertical frequencies increase from top to bottom. The constant-valued basis function at the upper-left i.e. the first transform coefficient, is the average of the sample sequence. In the literature, this value is referred to as the DC Coefficient. All other transform coefficients are called the AC Coefficients.

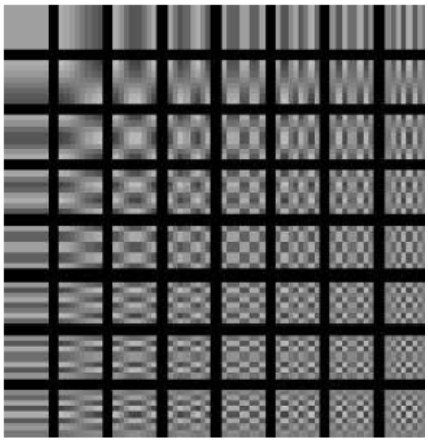

Figure 3.5 : DCT Basis Function

For the detection of forgery, good features of image blocks are needed to compare and match. Most of the visually significant information is concentrated in just a few coefficients of the DCT. Hence DCT coefficients of the blocks are used as features of blocks. To get better quality features, the DCT coefficients are quantized.

3.6.2 Quantization and Quality Factor (Q)

Human eye is good at seeing small differences in brightness over a relatively large area, but not so good at distinguishing the exact strength of a high frequency brightness variation [79]. This allows one to greatly reduce the amount of information in the high frequency components. This is done by dividing each component in the frequency domain by a constant for that component, and then rounding to the nearest integer. This rounding operation is the Quantization. Quantization enables the user to decide quality levels (Quality Factor), which determines the quantization steps for DCT transform coefficients. Quality Factor ranges from 1 to 100, where 1 gives the poorest image quality and highest compression, while 100 gives the best quality and lowest compression. For higher quality i.e. quality level greater than 50 (less compression, higher image quality), the standard quantization matrix is multiplied by ((100-quality level)/50). For quality level less than 50 (more compression, lower image quality), the standard quantization matrix is multiplied by (50/quality level).

The standard quantization matrix is the JPEG quantization matrix with quality level of 50 (Q50) and is shown below [78]:

$$Q_{50} = \begin{bmatrix} 16 & 11 & 10 & 16 & 26 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

Forgery detection needs good quality rather than good compression. Hence quality level greater than 50 is used. For example, for quality level 75 (Q75) the standard quantization matrix is multiplied by ((100-quality level)/50) = ((100-75)/50) = 0.5 and then rounding to the nearest integer. Q75 is given below :

$$Q_{75} = \begin{bmatrix} 8 & 6 & 5 & 8 & 13 & 20 & 26 & 31 \\ 6 & 6 & 7 & 10 & 13 & 29 & 30 & 28 \\ 7 & 7 & 8 & 12 & 20 & 29 & 35 & 28 \\ 7 & 9 & 11 & 15 & 26 & 44 & 40 & 31 \\ 9 & 11 & 19 & 28 & 34 & 55 & 52 & 39 \\ 12 & 18 & 28 & 34 & 55 & 52 & 57 & 46 \\ 25 & 32 & 39 & 44 & 52 & 61 & 60 & 51 \\ 36 & 46 & 48 & 49 & 56 & 50 & 52 & 50 \end{bmatrix}$$

For a better match, block size is taken as 8x8. To quantize the DCT coefficients of 8x8 block, standard JPEG quantization matrix was used. For 16x16 block, quantization matrix is given in equation below [10]:

$$Q_{16} = \begin{pmatrix} Q'_8 & 2.5q_{18}I \\ 2.5q_{81}I & 2.5q_{88}I \end{pmatrix}$$

……..(3.6)

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

and q_{ij} is the standard JPEG quantization matrix with quality factor Q and I is an 8x8 unit matrix.

During forgery detection, the quantization steps are calculated from a user-specified parameter Q which determines the quantization steps for DCT transform coefficients. Higher the value of the Q-factor finer will be the quantization and blocks matching will be more accurate. Lower the value of the Q coarser will be quantization. This may result in more matching blocks and some of them may be false matches also.

Another important point to consider is that when an object in an image is copied to another region, a number of blocks are duplicated and the distance between every original and duplicated block pair would be the same. Therefore decision of forgery can be made only if there are more than a certain number of duplicated image blocks with the same distance and these blocks are connected to each other [80]. Hence, a matching pair is considered if there are many matching pairs in the same mutual positions, that is, if they have the same shift vector. This will avoid many false matches. For this purpose, if two consecutive rows of the sorted matrix A are found matching, the positions of these matching blocks are stored in a separate list and a counter C is incremented.

If (i_1, i_2) and (j_1, j_2) are the positions of two matching blocks, the shift vector between the two matching blocks is calculated ass = $(s_1, s_2) = (i_1 - j_1, i_2 - j_2)$ where $i_1 > j_1$ and $i_2 > j_2$. At the end, the counter C indicates the frequencies with which different shift vectors occur. Then the shift vectors s^1, s^2, …, s^K, whose occurrence exceeds a user-specified threshold T that is $C(s^r) > T$ for r = 1, …, K, are determined. For all such shift vectors, the matching blocks that contributed are identified as segments that might have been copied and moved. The value of threshold T is also important. If it is large, some matching regions may be missed. If it is small, it may lead to many false matches.

The steps in the Exact Match algorithm are as follows:

1. **Grayscale conversion**: If the given image is a colour image, convert it in to grayscale.
2. **Forming blocks**: Divide the grayscale image is into overlapped blocks of size (bxb).
3. **Extracting features**: For each of the (M-b+1)*(N-b+1) blocks, compute the DCT coefficients as the features. Quantize the resulting block by dividing the DCT coefficients block element-wise, by the appropriate quantization matrix, with quality factor Q and round each resultant element.
4. Store the quantized coefficients for the blocks as the rows of a matrix. Also store the co-ordinates of the top-left corner pixel (x,y) as the location of the block. Then the matrix will have (M-b+1)*(N-b+1) rows and (bxb)+2 columns. The last two columns are the block location coordinates x and y.
5. **Lexicographic sorting**: Sort the matrix in ascending order.
6. **Matching process**: Search for consecutive identical rows and if there is a match store the positions of both the blocks in a separate list.
7. **Compute the shift vectors:** Compute shift vectors for every pair of matching blocks using equation: $s = (s_1, s_2) = |(i_1 - j_1, i_2 - j_2)|$ and set the value of a shift vector counter $c(s_1,s_2)$ to 1. If the shift vector value is a repetition, increment its counter value by 1.
8. **Mark the Blocks**: Find all shift vectors whose counter values exceed a user specified threshold value T. Mark all these blocks with some colour to indicate duplication. If there are no such shift vectors, declare that the image is not tampered.

3.6.3 Implementation details

The Robust Match algorithm was implemented using MATLAB software. Eighty images obtained from various sources were used for testing the algorithm. Colour images were converted into grayscale using the formula I = 0.299 R + 0.587 G + 0.114 B. Images were resized to 128x128 pixels. Blocks of size of 8x8 pixels were considered.

Results of forgery detection of the image shown in the Figure 3.1 using Exact Match and Robust Match methods are shown in Figure 3.6 for comparison. The results are better in the Robust Match method.

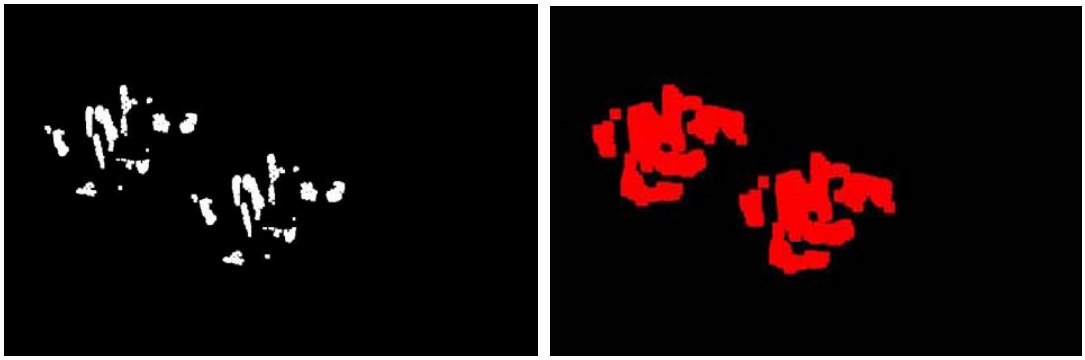

Figure 3.6 : Results of forgery detection of image in the figure 3.1: Exact Match (left) and Robust Match (Right)

Result of forgery detection using the exact match technique on the image in figure 3.2 is shown in the figure 3.7.

Figure 3.7 : Forgery detected in the image of figure 3.2 – Copied and pasted regions are coloured in pink

Figure 3.8 shows a forgery that could not be detected by the Exact Match technique.

Figure 3.8 : Forgery detected by the Robust Match Method

3.7 Modified Robust Match Technique

This method is same as the method discussed in section 3.6 in which quantized coefficients of DCT of blocks are used as block features. Here the length of the feature of the blocks is reduced which reduces the execution time without affecting the quality of the result. Reduction in the length of the feature ector is possible, because, when the quantization of DCT coefficients of the blocks is done, there are many long run zeros. These are high frequency DCT coefficients [78], which do not contribute to the quality of the image and hence can be omitted. The quantized DCT coefficients are read in zigzag order, as shown in Figure 3.10 and only quantized low frequency coefficients are taken.

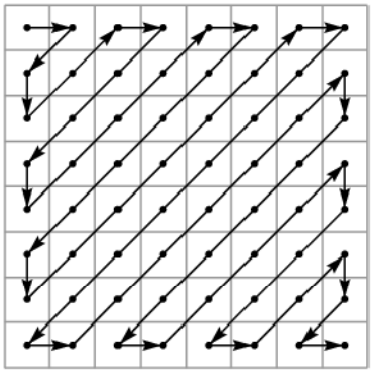

Figure 3.9 : Zigzag order for considering the DCT coefficients

The algorithm for this modified Robust Match Technique is the same but for the step 4 in which selected the features are read in zigzag order as shown and first n features are considered. 120 values are considered for a block size of 16x16 and 45 values are considered for a block size of 8x8 which are stored in the matrix for further processing.

This algorithm is equally efficient as Robust Match algorithm as for as the detection of forgery is considered but the execution is reduced. The average reduction in execution time is observed to be about 24%.

3.8 Performance Measures

The performance evaluation of a forgery detection algorithm can be done at two levels: at image level, where the focus is on the ability to detect if there is a forgery and at pixel level, where the accuracy of detecting the tampered regions [81]. In this work only image level evaluation is done.

At Image level, the important parameters are:

T_P-True Positive - the number of correctly detected forged images,

F_P - False Positive - the number of images that have been falsely detected as forged, and

F_N-False Negative - the number of falsely missed forged images.

The following metrics are used to analyze the performance of the algorithm:

Precision: Probability that a detected forgery is truly a forgery, computed as:

$$P = \frac{T_P}{(T_P + F_P)} \qquad \text{.....(3.7)}$$

Recall: Probability that a forged image is detected, computed as:

$$R = \frac{T_P}{(T_P + F_N)} \qquad \text{.....(3.8)}$$

This is also called True Positive Rate.

Score: This combines both Precision and Recall in a single value. It is computed as:

$$F = 2 * \frac{P * R}{(P + R)} \qquad \text{.....(3.9)}$$

In this work, a total of 220 images are considered out of which 150 were forged 70 were not forged. The performance parameters are given in the Table 3.1.

Method	T_P	F_P	F_N	Precision	Recall	Score
Exact Match	50	0	100	100.00	33.33	49.62
Robust Match	110	20	40	84.61	73.33	78.56
Modified Robust Match	110	20	40	84.61	73.33	78.56

Table 3.1 : Performance of the Block Matching forgery detection algorithms

3.9 Summary

Copy-Move forgery is a widely committed forgery. In this chapter, the concept of block based forgery detection algorithms which are efficient in detecting Copy-Move forgery was developed. Analysis and implementation details of two methods namely Exact Match and Robust Match were presented. Robust Match method yields a better result than the exact match. An improvement over the Robust Match is made by reducing the number of features which results in the reduction of execution time without compromising its ability to detect forgery.