Geometric Transformation Invariant Copy-Move Forgery Detection in Images

A Project Report Submitted in
Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology

by

Sai Akhil Koditala (14-1-5-009)
Surya Pratap Singh (14-1-5-007)
Raktim Sharma (14-1-5-035)
Prity Singh (14-1-5-081)

Under the Supervision of

Prof. Badal Soni

Computer Science & Engineering Department
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

May, 2018

Declaration

Report Title: **Geometric Transformation Invariant Copy-Move Forgery Detection in Images**
Degree for which the report is submitted: **Bachelor of Technology**

We declare that the presented thesis represents largely our own ideas and work in our own words. Where others ideas or words have been included, we have adequately cited and listed in the reference materials. The report has been prepared without resorting to plagiarism. We have adhered to all principles of academic honesty and integrity. No falsified or fabricated data have been presented in the report. We understand that any violation of the above will cause for disciplinary action by the Institute, including revoking the conferred degree, if conferred, and can also evoke penal action from the sources which have not been properly cited or from whom proper permission has not been taken.

Sai Akhil Koditala
Date: 16^{th} **May 2018** Scholar No. : 14-1-5-009

Surya Pratap Singh
Date: 16^{th} **May 2018** Scholar No. : 14-1-5-007

Raktim Sharma
Date: 16^{th} **May 2018** Scholar No. : 14-1-5-035

Prity Singh
Date: 16^{th} **May 2018** Scholar No. : 14-1-5-081

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

It is certified that the work contained in this thesis entitled **"Geometric Transformation Invariant Copy-Move Forgery Detection in Images"** submitted by

Sai Akhil Koditala (14-1-5-009)
Surya Pratap Singh (14-1-5-007)
Raktim Sharma (14-1-5-035)
Prity Singh (14-1-5-081)

for the Bachelor of Technology Project, 2018 is absolutely based on his own work carried out under my supervision.

Place:
Date:

Prof. Badal Soni
Computer Science & Engineering
National Institute of Technology Silchar

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY SILCHAR

This is to certify that the report titled **"Geometric Transformation Invariant Copy-Move Forgery Detection in Images"** submitted by

Sai Akhil Koditala (14-1-5-009)
Surya Pratap Singh (14-1-5-007)
Raktim Sharma (14-1-5-035)
Prity Singh (14-1-5-081)

for the partial fulfillment of the requirements for award of the degree of Bachelor of Technology in Computer Science and Engineering, embodies the bonafide work done by them during the session 2017-18.

The work presented here has not been submitted elsewhere for award of any degree.

Place:
Date:

Head of Department
Department of Computer Science & Engineering
National Institute of Technology Silchar

Acknowledgement

We would like to express our sincere gratitude to **Prof. Badal Soni**, for providing us with the opportunity to do our research work under his guidance. His emphasis on steady and committed effort has motivated us during the course of the research work. We have immensely benefited from the excellent research environment that he has created and nurtured.

We are also grateful to **Dr.Arup Bhattacharjee**, HOD of the Department of Computer Science & Engineering and **Dr. Dalton Meitei Thounaojam**, B.Tech. Project Coordinator of Department of Computer Science & Engineeing for their guidance and encouragement throughout our research work. Their concern and support have been invaluable to us in the completion of our research work.

We are very much thankful to all the faculty members and staffs of the Department of Computer Science & Engineering for their advice and timely help.

Last but not the least, most importantly we are thankful to our family members and our friends for providing us their unfailing support, understanding and love.

Abstract

In the present world, digital images can be included among the prevalent ways for information sharing. The term information ranges from casual to highly confidential. There is high chance of forging the image with the existing available technology to tamper the images. It is also not so time taking or difficult to forge the image by replacing the important portion of the image with some other part of the same image.As the copied part resides in the same image, its important properties, such as noise, brightness, texture is compatible with rest of the image making its detection very difficult. The mentioned type of forging is generally defined as copy-move forgery. Till date many algorithms had been proposed in this field to detect the copy-move forged regions of the image. The algorithms vary from one to one depending on the way it processes the image pixels (Block based or Key-point based). All the proposed algorithms suffer from the problem of time-consumption.

The purpose of this study is to present an efficient copy-move forgery detection algorithm that which uses ORB (Oriented FAST Rotated BRIEF) for key-point detection as well as description and Trie data structure for the purpose of matching, to reduce the overall computation time complexity of detecting duplicated regions of the image. The proposed algorithm is tested on the data-sets 'MICC-F220' and 'MICC-F2000'. Experimental results on the data sets shows the accuracy and ability to reduce the computational time complexity.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

In the present world, it is obvious to suggest that images play a vital role as a source of information. But with the technologies available, forgery of such precision is possible that is not detectable by the naked eye. The main purpose of forgery analysis is to check whether there is any manipulation in the content of an image. Various algorithms have been proposed to detect image forgeries. Among such algorithms, the copy-move forgery detection is used to detect parts of the image which have been directly copied and pasted onto another part. There are two generic techniques of copy-move forgery detection, namely block-based and keypoint-based. The first block based method for detecting copymove forgery was introduced by Fridrich et al [1]. Keypoint-based techniques is the preferred choice between the two, owing to its various advantages.

We have implemented a copy-move forgery detection system that is invariant to geometric transformations like scaling and rotation, and also compared its performance to similar existing efficient systems.

1.1 Motivation

The principal motivation behind this project is that existing copy-move forgery detection algorithms have a requirement of being computationally efficient. Another

motivation is to use an ORB-based system which has the advantage of being open-source as well, which is invariant to geometric transformation of the forged part of the image.

1.2 Objective

Our main objective is to implement an ORB (Oriented FAST Rotated BRIEF) features-based algorithm which is invariant to geometric transformations of the image. Also, an added objective is to design an algorithm which takes computationally lesser amount of time to perform this task than similar existing algorithms.

1.3 Problem Definition

The main objective of any CMFD method is to accurately determine the forged regions in the original image. The other methods which are implemented for forgery detection are time consuming and also the accuracy is not up to the mark. So, in this project our aim is to design a method that can overcome the problems of accuracy and time complexity. The proposed method is programmed on a 64-bit windows operating system. The datasets used for the testing purpose are MICC-F220 and MICC-F2000.

The report is divided as follows: Chapter 2 provides a brief literature survey of Copy Move Forgery Detection. Chapter 3 consist of proposed algorithm to detect the forgery. Chapter 4 describes experimental results and discussion. Conclusion and future work is given in Chapter 5.

CHAPTER 2

Literature Survey

A number of algorithms have been proposed till date in the field of forgery detection in an image. The following explains various papers that had been considered in designing our algorithm.

2.1 Copy-Move Forgery Detection Pipeline

The copy-move forgery detection technique can be represented as a pipeline. The various steps involved in this pipeline process are:

1. **Pre-processing**: Most of the methods operate on gray scale images. Therefore, the images need to be first converted from RGB to gray scale.

2. **Feature Extraction**: There are two techniques for feature extraction in copy-move forgery detection:

    - **Block Based Approach**: In this method, the input image is divided into fixed size overlapping or non-overlapping blocks. Generally, square blocks are used but some researchers have also used circular blocks. Feature vectors are extracted from each block using several techniques. These features vectors are then sorted using some sorting technique so that the similar features lie in proximity. After that blocks with similar shifting are detected using

the concept of shift vector which are then counted by setting a counter. On the basis of threshold value and the above mentioned principle, the similar blocks are detected.

- **Key-Point Based Approach**: In this method, key points are scanned using techniques which is based on identifying and selecting high-entropy image regions. The features corresponding to these key points are then extracted. On the basis of correlation, similar feature vectors are identified to find altered region present in image. In this category, image is not divided into fixed size blocks.

3. **Matching**: Similar feature vectors are matched as high similarity is an indicator for presence of duplicated regions.

4. **Filtering**: Filtering is done to reduce the possibility of false matches. For example suppressing noise to remove matches between spatially close regions.

5. **Post processing**: The goal of this last step is to preserve matches that exhibits common behavior.

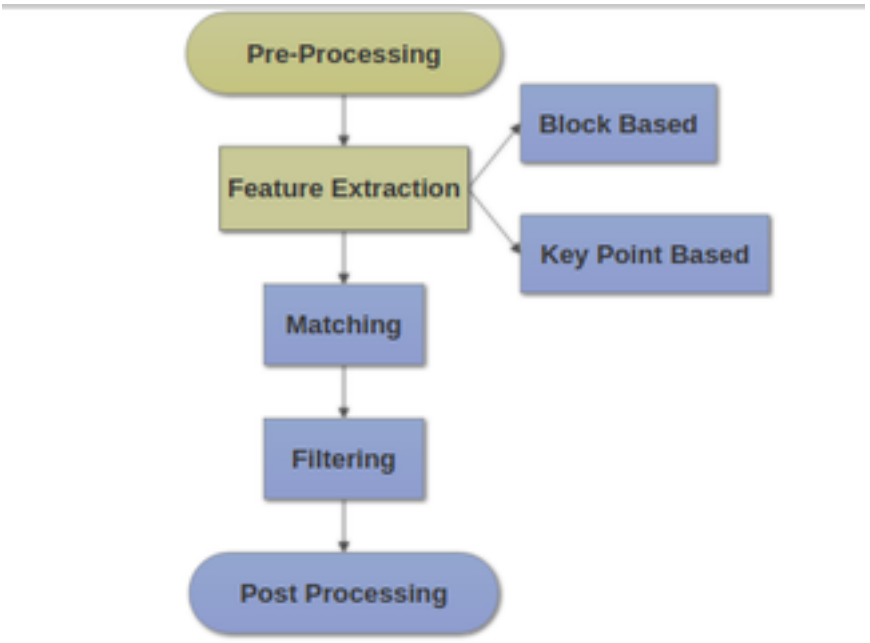

FIGURE 2.1: Pipeline of the process

2.2 Existing methods

Following has the explanation of various papers that had been considered before designing the algorithm :

The first block-based method for detecting copy-move forgery was introduced by **Fridrich et al.**[1]. Discrete Cosine Transformation (DCT) based features have been used. The proposed algorithm had several limitations against operations like blurring, rotation and scaling. Image is initially is divided into overlapping blocks by the sliding window moving by one pixel each time from top left corner to bottom right corner of the image. A matrix is formed with each row holding the pixel values of a particular block. Similar blocks are found using Lexicographical sorting of matrix. This phenomenon is known as exact match method. By calculating DCT coefficients for each block, a robust method had been used. A quality factor parameter named Q-factor is selected for appropriate quantization of DCT coefficients. A matrix with each row holding the DCT coefficients of a particular block is lexicographically sorted. Similar blocks are found with the help of highly occurred shift vectors. A desired threshold is fixed which controls the minimal size of image segment that can be detected.

In 2012, **Sheng et al.**[2] applied the ridgelet transform of divided blocks to extract features and compute Hu moments for these features to produce feature vectors. Euclidean distance of feature vectors is computed for similarity measure. The image is divided into overlapping sub-blocks. Ridgelet transform is performed on every sub-block, getting the result and the Hu-moments of every result to represent every sub-block is computed. The third step is matching, computing the Euclidean distance of features corresponding to each pair of sub-blocks to find similar pairs.

Zimba et al.[3] proposed a copymove forgery algorithm based on discrete wavelet transform (DWT) to extract features from input image to yield a reduction in feature dimensional size. The improvement made here was in space complexity, and also the blocks were not restricted to squares.

Common limitations of block-based methods include direct uses of quantized or frequency based features from divided blocks for matching, which makes the size of feature vectors quite high, and makes the dimension reduction step mandatory. To

overcome these issues, an alternative approach used for the copy-move tampering detection is the key-point based method, as discussed next.

Huang et al.[4], proposed a Scale-invariant feature transform (SIFT) based detection method to identify duplicate regions with scale and rotation, then used the best bin first search (BBF) method to find the nearest neighbors with high probability that return the matched key points (inliers) as a possible duplicate region. To increase the accuracy of detection methods, a nearest neighbor distance ratio (NNDR) is applied for matched key-points.

Amerini et al.[5] have proposed detecting multiple duplicated regions based on SIFT features, and then employed generalized nearest neighbor (G2NN) to improve the similarity match between key-points. The agglomerative hierarchical linkage (AHL) clustering method has been employed to group the similar key-points into the same cluster and merge closest pair of clusters into single cluster to represent the cloned regions. The estimation of affine transformation parameters is computed between duplicated regions.

In the paper published by **Chen et al.**[6], he proposed a different approach of FAST algorithm for the purpose of Image Segmentation using k-means clustering algorithm, that which is a part of Unsupervised machine learning algorithms. One of the important advantages of this algorithm can be said as the stability of the algorithm to segment the images into interest of regions. But, mostly this kind of algorithms are not practiced because of the underlying complexity in implementation and as well understanding the algorithm.

D.G. Lowe[7] presented a method for copy-move forgery detection based on Scale Invariant Features Transform (SIFT) that can extract distinctive invariant features from images which can be used to perform reliable matching between different views of an object or scene in a given image. He cited many advantages of using such an algorithm. There are also some drawbacks that can be mentioned when compared against the existing algorithms. Various advantages can be said as, they are rotation and scaling invariant and it's descriptor is a classic approach, also the "original" inspiration for most of the descriptors proposed later. Drawbacks that are mentioned in the paper include, it is mathematically complicated and computationally heavy. SIFT is based on the histogram of gradients. That is, the gradients of each pixel in the patch

need to be computed and these computations cost time. It is not effective for low powered devices. It is also to be noted that SIFT is patent protected. Given an input image I, SIFT features are detected at different scales using a scale-space representation method and implemented as an image pyramid, We select the interest points as the local extrema and obtain the image pyramid levels by sub-sampling of the image resolution and Gaussian smoothing. These interest points or keypoints are extracted by applying a computable approximation of the Laplacian of Gaussian called Difference of Gaussians (DoG).

$$D(x, y, \sigma) = (GB(x, y, x, k\sigma) - GB(x, y, \sigma)) * I(x, y) = LG(x, y, k\sigma) - LG(x, y, \sigma) \quad (2.1)$$

Where Gaussian blur is at kσ and LG is the convolution of original image I(x,y). We can summarize the method of copy-move forgery detection using SIFT features as follows :

1. SIFT features for an image is calculated and the euclidian distance between the pairs of SIFT keypoints is found.
2. By selecting the appropriate threshold based on minimum distance obtained from above step, best matches are determined.
3. Cluster centres are defined by the best match Keypoints. By using a threshold for the Euclidean Distance and Cluster size the matched clusters are found.
4. Display the clusters if both the clusters have a minimum of two points to decide whether the image is forged or authentic.

Advantages of the method

- Accuracy and precision is very high.
- Robust to post processing operations such as scaling, rotation, noise blurring.

Disadvantages of the method

- Slower than SURF in terms of time.

Xu et al.[8] proposed a SURF (Speed up Robust Feature) method for copy-move forgery detection. The fast and robust SURF keypoints are employed to find the possible duplicate regions in the image.

SURF is a scale invariant interest point or keypoint detector and descriptor. It depends on integral image convolution and the keypoints are found by using a Fast-Hessian Detector that is based on an approximation of the Hessian matrix for a given image point. The Haar wavelet distribution responses are used for orientation assignment, before the keypoint descriptor is formed from the wavelet responses in a certain surrounding of the keypoint. It uses only 64 dimensions thus reduces the time for feature computation and matching and thereby increases robustness simultaneously. In an image, for a given point x = (x_i, y_i), the Hessian matrix H(x,σ) at scale σ can defined as:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{2.2}$$

where $L_{xx}(x,\sigma)$ is the convolution of the Image I at point x with Gaussian second order derivative and similarity for $\frac{\partial^2}{\partial x^2} g(\sigma)$.

$$det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \tag{2.3}$$

Finally, the found maxima of the determinant of the approximated Hessian matrix are interpolated in scale and image space. For calculating descriptor, square region centered at the interest point and having its orientation is selected and divided into 4 X 4 square sub-regions. For each sub-region, the Haar wavelet response in horizontal

direction (d_x) and in vertical direction (d_y) are recorded with a filter size 2s at 5 X 5 regularly spaced sample points. The four-dimensional descriptor is given by:

$$v = (\Sigma d_x, \Sigma d_y, \Sigma |d_x|, \Sigma |d_y|) \tag{2.4}$$

We can summarize the method of copy-move forgery detection using SURF as follows:

1. SURF features for an image is calculated and the Euclidean distance between each pair of SURF keypoints is calculated.
2. By selecting the appropriate threshold based on the minimum distance obtained from above step, best matches are determined.
3. Cluster centres are defined by the best match keypoints. By using a threshold for the Euclidean distance and cluster size the matched clusters are found.
4. Display the clusters if both clusters have a minimum of two points to decide whether the image is forged or authentic.

Advantages of the method

- Very fast compared to SIFT in terms of time.
- Robust to post processing operations such as scaling, rotation, noise blurring.

Disadvantages of the method

- Less precision and accuracy compared to SIFT.

In the paper published by **Rublee et al.**[9] on ORB (Oriented FAST Rotated BRIEF) an efficient alternative to SIFT and SURF, has explained the advantages of ORB in detail. A very trivial point for considering ORB to SIFT and SURF is that

ORB is open source and can be implemented or customized as per the application. Whereas SIFT and SURF are licensed (not Open Source). The theory behind BRIEF and ORB is much easier to understand compared to SIFT and SURF. Computing Hamming distance is lightning fast compared to Euclidean distance. ORB is faster compared to both SIFT and SURF, that which uses the Oriented FAST algorithm for key-point detection and Rotated BRIEF for key-point description, which makes the ORB invariant to Geometric transformations like Scaling and Rotation.

Harris et al.[10] have proposed a corner detection algorithm, that which is found implemented in many applications around. Corner is the intersection of two edges, it represents a point in which the directions of these two edges change. Hence, the gradient of the image (in both directions) have a high variation, which can be used to detect it. Consider a grayscale image I. We are going to sweep a window w(x,y) (with displacements u in the x direction and v in the right direction) I and will calculate the variation of intensity.

$$E(u,v) = \sum_{x,y} w(x,y)[I(x+u, y+v) - I(x,y)]^2 \tag{2.5}$$

Where :

w(x,y) is the window at position (x,y)

I(x,y) is the intensity at (x,y)

I(x+u,y+v) is the intensity at the moved window (x+u,y+v)

Among the features, corners have an upper hand over edges due to the property of rotation invariant. Because corner is a same corner, even if it is rotated. In this algorithm, the main focus will be on detecting a corner Key-point. Harris corner detection algorithm generally sub-divides its computation for Edges, Corners and Flat regions. As the advantages are said above, there are also disadvantages in implementing this algorithm. One of the biggest drawback this algorithm holds is, it fails to detect the corners, if the corner containing part of the original image is scaled. So, to overcome

this drawback we had used ORB algorithm that which uses Oriented FAST key-point detector that which is invariant to scaling.

In the book *Advanced Data Structures* by **Brass**[11], many customized data structures have been explained. Among all such data structures, trie has the nearest quality as a data structure for holding the data that which will be well suited for the purpose of matching the long length strings similar to our purpose of matching the key-point descriptors (we used binary string of 64 length). Trie data structure reduces the computational time complexity of matching n key-points, each of l length binary string, from O(n*n*l) to O((n*$2^{threshold}$)*l). Here, the first mentioned computational time is taken by brute-force method of searching and latter is the computational time taken by the trie data structure. So, this plays a very vital role in matching phase of the detection algorithms as the number of key-points increase. This is because the computational time for brute-force method increases in quadratic fashion whereas trie increases linearly. So, this was the most considered point for implementing the matching phase using trie data structure.

CHAPTER 3

Proposed System

3.1 Problem Description

The goal of the detection algorithm is to detect forged regions in an image with a high accuracy and less time complexity. The proposed algorithm focuses on providing both good accuracy and as well minimizing the time complexity in doing so. The algorithm has been implemented and tested on a 64-bit Linux based Operating System. The data-set used for calculations of accuracy and time complexity are "MICC-F220" and "MICC-F2000". The following chapter gives the detailed explanation of all the data structures and procedure of the algorithm.

3.2 Data Collection

"MICC-F220" and "MICC-F2000" are the data-sets used for testing of the proposed copy-move forgery detection. The data-sets contain 220 and 2000 images respectively. The size of the images used are of 500 x 800 pixels. Both the data-sets contains the images that had undergone geometrical transformations like Scaling and Rotation. Data-sets consists of mixture of forged images (Geometrically transformed) and original images.

3.3 Phases of the Algorithm

3.3.1 Noise Reduction

During the time of capturing images, some noise will be added to the images. So, this added noise must be always removed before implementing the algorithm. This phase of the algorithm plays an important role in detecting the forged regions. Sometimes, it can be even said that none of the forged regions would be detected if the noise reduction is not performed on the image. There are various algorithms for image smoothing, namely average masks, weighted average masks, Statistical masks (Median, Min, Max). In the proposed algorithm GaussianBlur (OpenCV function) is used for smoothing the image (Noise Reduction).

3.3.2 RGB to Gray Scale Conversion

Generally in RGB images, each pixel is made of 24 bits (Red - 8 bits, Blue - 8 bits and Green - 8 bits). whereas pixel of a Gray scale image is made of 8 bits (range of a pixel is from 0 to 255). Calculating the gradient of a pixel of Gray scale image is simple and understandable compared to RGB image. The designed algorithm is implemented on Gray scale images.

3.3.3 Key-Point Detection

A point is said to be a key-point which is considered as a center of the feature. A feature is application dependent. Various examples of the features are Corner, Edges, BLOB (Region of Interest). In the proposed algorithm, Corners are considered as a feature. Here, in this context a key-point denotes a pixel of a corner. In the proposed algorithm ORB (Oriented FAST Rotated BRIEF) algorithm has been used for Key-point detection. ORB is free from licensing restrictions of SIFT and SURF. The ORB algorithm uses 'Oriented FAST (Features from Accelerated Segment Test)' algorithm to detect the scale invariant corner key-points and Harris Corner filter to reject edges and provides a reasonable score. FAST is several times faster than other existing

corner detectors. It is dependent on a appropriate threshold. It identifies a pixel 'p' in the image as an interest point or not, based on the threshold chosen. The detected key-points will be further used in the process of matching and localization.

3.3.4 Key-point Descriptor

The selection of algorithm for key-point description also plays a very important role. ORB uses BRIEF (Binary Robust Independent Elementary Features), A key-point descriptor is basically a binary string of adjustable length (32 bits, 64 bits, 128 bits, etc). BRIEF also provides high recognition rate unless there is large in-plane rotation. The length of the binary string is equal to the number of neighboring pixels that we are considering for corner detection. This means each bit of the binary string is related to a unique neighbor pixel of a key-point. A bit which is set (1) in the binary string denotes that the intensity of the key-point is greater than the intensity of the neighboring pixels that corresponds to the bit. The bit will be unset (0) for the opposite case. So, to make the corner detection rotation invariant, the mean gradient with the neighbor pixels is calculated for 30 orientations (360/12 = 30) of the key-point and the angle with the maximum gradient value is selected and the particular binary description is given to the key-point.

3.3.5 ORB (Oriented FAST Rotated BRIEF)

ORB is basically a fusion of FAST key-point detector and BRIEF descriptor with many modifications to enhance the performance in various situations. It is rotation invariant and resistant to noise, which is two orders of magnitudes faster than SIFT key-point detector and descriptor. The process of ORB starts by detecting FAST points in the image.

1. FAST selects one parameter, an intensity threshold value '**t**'.
2. It selects a pixel **P** in the image which is to be identified as an interest point or not. Let its intensity be I_p.

3. It considers a circle of 16 pixels around the pixel under test(P). Pixel P will be considered as a corner if and only if there exists a set of **n** consecutive pixels in the circle which are all brighter than $I_p + t$, or all darker than $I_p - t$.

FAST does not produce a measure of cornerness. So, it makes use of Harris Corner measure to order the FAST key-points detected. It orders them according to the Harris measure and picks the top desired number (say N) of points. FAST does not produce scale invariant features. To make the detection scale invariant, ORB employs a scale pyramid of the image and produces FAST features at each level of the pyramid. FAST also doesn't compute the orientation. To make detection rotation invariant, it computes the intensity weighted centroid of the patch with located corner at centroid. The orientation is given by the direction of the vector from this corner point to centroid. Now for descriptors, ORB uses binary descriptor based on BRIEF. BRIEF provides a shortcut to find the binary strings directly without finding descriptors. The process of BRIEF goes as follows,

1. It takes smoothened image patch and selects a set of n_d (x,y) location pairs in a unique way.
2. Based on some pixel intensity comparisons are done on these location pairs, we get a n_d-dimensional bitstring.
3. It is done as, for example, let first location pairs be P and Q. If I(P) ¡ I(Q), then its result is 1, else it is 0.

The value of n_d can be 128, 256 or 512 depending on the requirement of the application. The process of matching is carried out on these binary strings of the detected key-points.

3.3.6 Matching the key-points

The whole heart of the designed algorithm lies in this phase. This customized part of matching is used for reducing the computational time of the overall algorithm because in majority of the proposed algorithms matching consumes most of the computation

time. So, minimizing this part of the algorithm will boost the overall performance of the algorithm.

The "Trie" data structure has been implemented to perform the matching process of detected key-points. The used data structure is generally a tree kind of structure which has to be pre-computed before the process starts. The following explains the further process of pre-computation and matching :

3.3.6.1 Pre-Computation and Matching in Trie

The tree constructed after the process of pre-computation will be a binary tree. Every single node of a tree consists an array of pointer data type of size 2 to hold the address of nodes in the next level of the tree and a Point data type which holds NULL for all the non-leaf nodes, whereas, the leaf nodes hold the co-ordinate of a unique key-point. Here each pointer in a node is used to represent a 0 (for the bit which is unset in binary string) and to represent 1 (for the bit which is set in binary string) of a key-point bit string.

The computational process includes considering a binary string of length 5 and then the tree constructed will be of height 5. Let us suppose the binary string be "10011". The process starts from index 0 (first bit from left) and ends at index 4 (right most bit). Let us consider 2 variables ptr and bit-val to explain the algorithm. Initially ptr holds the address of the root of the tree and bit-val contains first bit '1' and checks if the bit-val index of the array of ptr node has been allocated memory (i.e, already defined). If that was not the case then memory would be allocated to the bit-val index of the ptr node and ptr is updated to the address that bit-val index of the array of ptr node holds. This process is recursively computed until bit-val reaches the end of binary string. In the final node, whose address is held by ptr, we will store the co-ordinates of the key-point to which the path (binary string) belongs. Now this way the tree will be constructed using the binary string of the key-points prior to the initialization of matching process.

Similar to the tree construction process the matching process for every key-point is initialized from the top (root) of the tree as well as from the first index of the binary string of the key-point. There will also be an extra added variable mis-hit which will

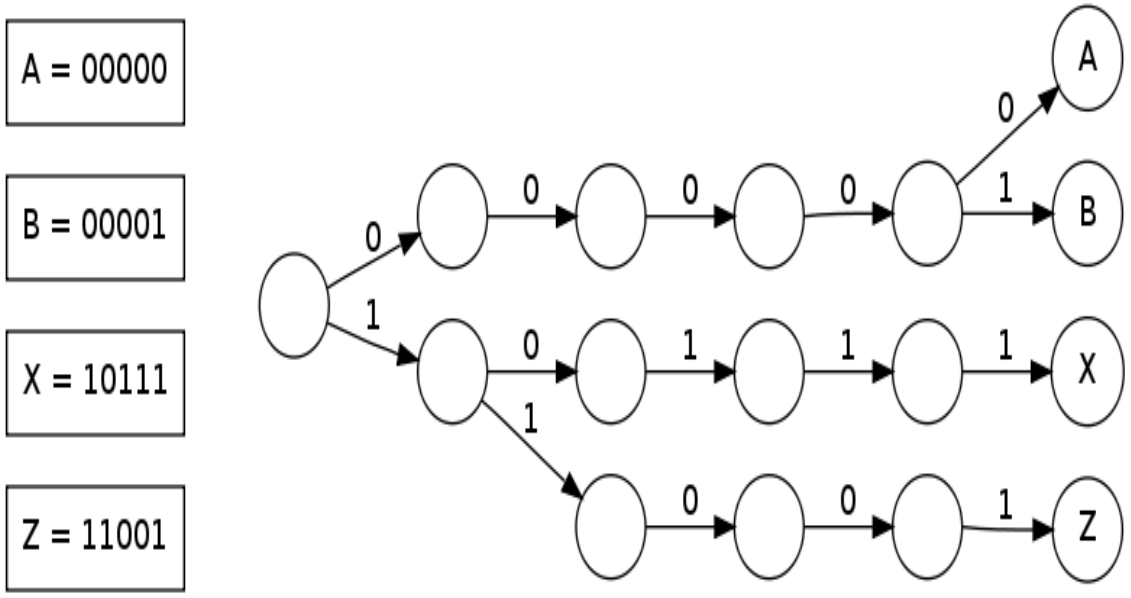

FIGURE 3.1: Image of an arbitrary Trie tree

be initialized to zero and will be incremented for every mishit during the propagation of the tree from top to bottom. Now, the bit-val index of the array of ptr node will be checked if it is initialized. If it is initialized, then it is a hit and with out the increment of the mis-hit we will propagate to the next level. If there is a mishit then we will increment the value of the mis-hit by one and propagate to the next level of tree through other index and check if the mis-hit value is under the threshold (for maximum number of allowed miss hits). If the mis-hit value crosses the threshold at any stage, then the particular recursion is terminated. Every time ptr reaches a leaf node, and if the mis-hit value is less than the previously found, the point will be updated with the point in the leaf node. This is how, the key-point with maximum hamming distance is found if there is any. Below is the structure of the each node in the tree (Trie node).

```
struct trie{
    struct trie* bit[2];
    Point pt;
};
```

3.3.7 Displaying the Matched Key-points

After the process of Matching, we will get a uniquely matched point for all the key-points that have a nearly matched key-point. For all such pairs a line is drawn between the points to show the connection between the copied regions of the image.

3.3.8 Proposed Algorithm

Algorithm 1 Copy-move forgery detection algorithm

```
1: procedure COPY-MOVE(img)
2:     [M,N] ← size of img
3:     M ← Number of rows in Resolution matrix
4:     N ← Number of columns in Resolution matrix
5:     forged ← 0
6:     threshold ← optimal
7:     noOfKeypoints ← optimal
8:     Detector ← ORB::create(noOfKeypoints)
9:     Descriptor ← Detector.ExtractDescriptor()
10:    keyPoints ← Detector.ExtractKeyPoints()
11:    bitset⟨256⟩ bitset[noOfKeypoints]
12:    bitset⟨64⟩ normalizedBitset[noOfKeypoints]
13:    for I=1 to noOfKeypoints do
14:        copy Descriptor.row(I).data to bitset[I]
15:    end for
16:    normalizedBitset ← createNormalizedBitset(bitset)
17:    trieNodeRoot ← NULL
18:    trieNodeRoot ← createNewNode()
19:    for I=1 to noOfKeypoints do
20:        addDescriptorToTrie(normalizedBitset[I])
21:    end for
22:    for I=1 to noOfKeypoints do
23:        matchDescriptor(normalizedBitset[I], cutoff)
24:    end for
25:    drawMatchedKeyPointsOnImage(img)
26:    showImage(img)
27: end procedure
```

Algorithm 2 Adding descriptor to trie

```
1: procedure ADD-DESCRIPTOR-TO-TRIE(Bitset, RootNode)
2:     tempTrieNode ← createNewNode()
3:     for I=0 to Bitset.size() do
4:         copy Descriptor.row(I).data to bitset[I]
5:         if tempTrieNode.f[Bitset[I]] = NULL then
6:             tempTrieNode.f[Bitset[I]] ← createNewNode()
7:         end if
8:         tempTrieNode ← tempTrieNode.f[Bitset[I]]
9:     end for
10: end procedure
```

Algorithm 3 Matching algorithm

```
1: procedure MATCH-DESCRIPTOR(Bitset, RootNode, Mismatch)
2:     if mis > cutoff then
3:         return false
4:     end if
5:     if RootNode.f[Bitset[Index]] = 1 then
6:         if Index+1 = Bitset.size() and Mismatch > 0 then
7:             matchedPoint = node.f[bs[idx]].pt
8:             return true
9:         end if
10:        MATCH-DESCRIPTOR(Index+1, Mismatch, RootNode.f[Bitset[Index]])
11:    end if
12:    if RootNode.f[1-Bitset[Index]] = 1 then
13:        if Index + 1 = Bitset.size() and Mismatch + 1 <= cutoff then
14:            MatchedPoint = RootNode.f[1 − Bitset[Index]].pt
15:            return true
16:        end if
17:    end if
18:    MATCH-DESCRIPTOR(Index+1, Mismatch+1, RootNode.f[1-Bitset[Index]])
19: end procedure
```

CHAPTER 4

Experimental Results and Discussions

In this section, description of different benchmarking databases used for experimentation purposes, performance measures, robustness test of the proposed algorithm, analysis of the experimental results for three different databases and comparison of the proposed approach with existing approaches are given.

4.1 Database

The performance of the proposed system is evaluated on two different publicly available standard datasets. These are the MICC-F220 and the MICC-F2000.

4.2 Performance Measures

Several parameters are used for the purpose of performance measure where we have considered 4 different parameters namely, true positive rate (Recall or Sensitivity) (TPR), false positive rate (FPR), precision and accuracy. They explain various properties of the implemented algorithm in the following ways:

4.2.1 True Positive (TP)

The number of correctly identified positives. Positives are the images for which forgery is detected by the algorithm.

4.2.2 False Positive (FP)

The number of falsely identified positives.

4.2.3 True Negative (TN)

The number of correctly identified negatives. Negatives are the images for which no forgery is detected by the algorithm.

4.2.4 False Negative (FN)

The number of falsely identified negatives.

4.2.5 True Positive Rate (TPR)

Measures the proportion of positives that are correctly identified as such.

$$TPR = \frac{TP}{TP + FN} \tag{4.1}$$

4.2.6 False Positive Rate (FPR)

Measures the proportion of positives that have been falsely identified.

$$FPR = \frac{FP}{FP + TN} \tag{4.2}$$

4.2.7 Precision

Proportion of correct positives among the total matches.

$$Precision = \frac{TP}{TP + FP} \tag{4.3}$$

4.2.8 Accuracy

Proportion of correctly identified positives and correctly identified negatives.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4.4}$$

4.3 Analysis among various performance measures

As in the proposed algorithm we used many parameters like number of key-points, selection of threshold (allowed mishits) which are very sensitive towards the performance of the algorithm. So, to decide on particular values of these parameters, an analysis among the parameters on which performance rely, is required.

4.3.1 TPR vs Threshold

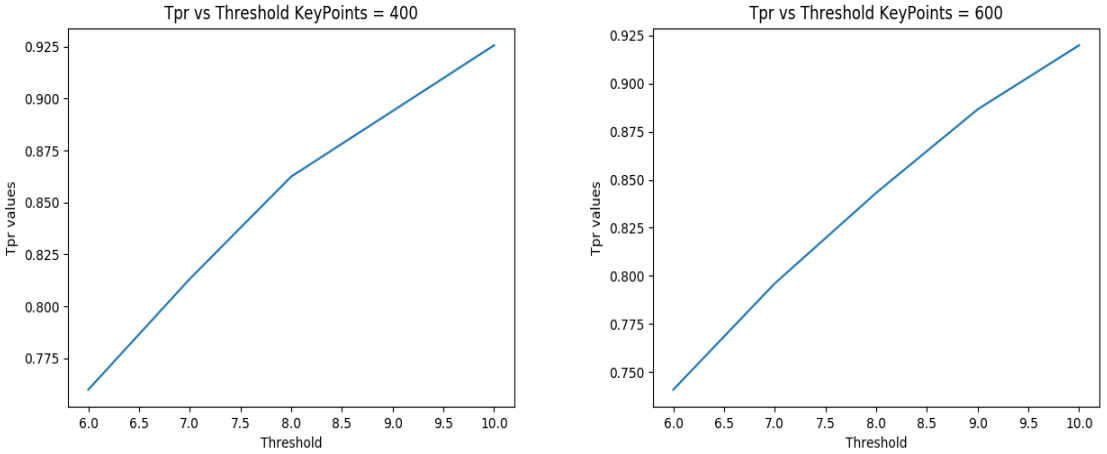

FIGURE 4.1: For 400 and 600 Key-points (TPR vs Threshold)

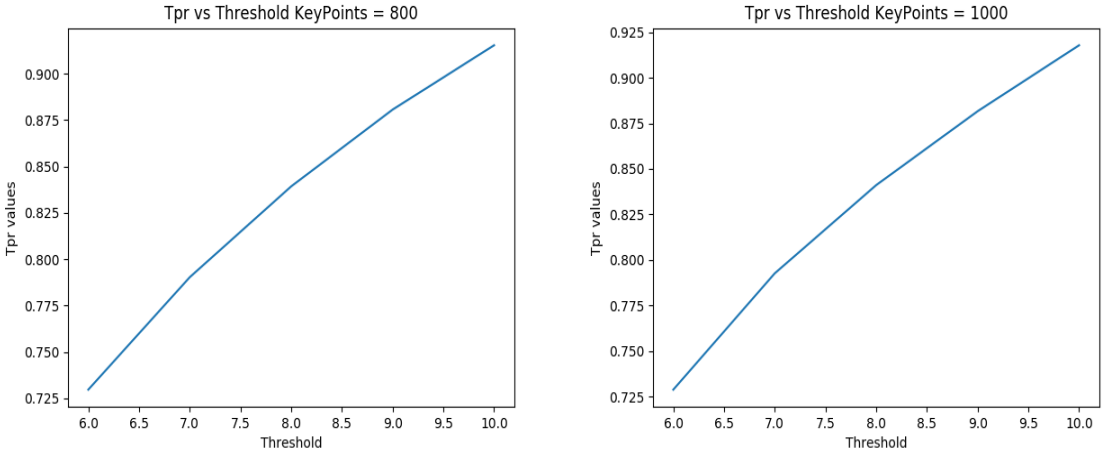

FIGURE 4.2: For 800 and 1000 Key-points (TPR vs Threshold)

From the experimental analysis between TPR and threshold values, it is straightforward that increase in the threshold value would result in increase in the TPR. This is because high threshold allows more number of miss matches among the descriptor bit sets which will result in increase in the number of matches between the detected key-points which would include both true matches and false matches. So, this analysis independently will not help in selecting the desired threshold value. This is because, accuracy of the algorithm is dependent on both TPR and FPR values, which will require a trade-off between the TPR and FPR values.

4.3.2 Accuracy vs Threshold

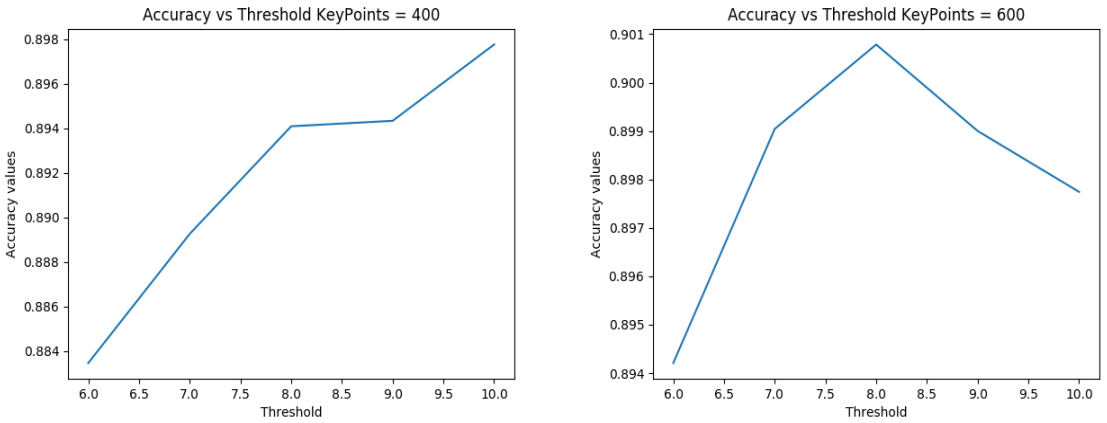

FIGURE 4.3: For 400 and 600 Key-points (Accuracy vs Threshold)

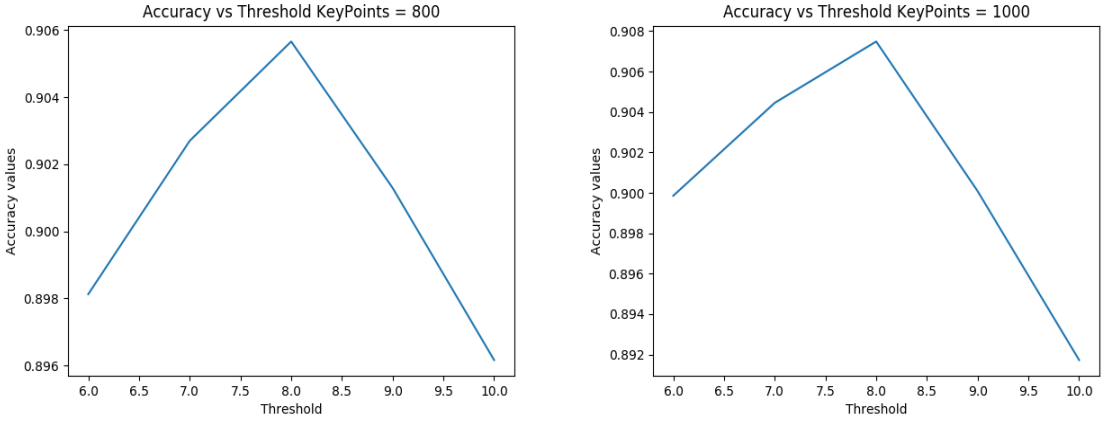

FIGURE 4.4: For 800 and 1000 Key-points (Accuracy vs Threshold)

From the experimental analysis between Accuracy and Threshold values, in the 1st graph (key-points = 400), the accuracy was monotonically increasing along with the threshold, this is because, as there are less number of considered key-points, increase in threshold would allow TPR to increase more, comparatively with the increase in FPR. In contrast 2nd, 3rd and 4th graphs shows different behaviour with 1st graph. Here, the accuracy increases monotonically from 6 to 8 threshold value and with further increase in threshold would result in the decrease of accuracy. This behaviour depicts that, with increase in the number of key-points and threshold would allow FPR to increase more, comparatively with the increase in TPR. So increasing the threshold beyond a particular value will allow false matches and is not beneficial.

4.3.3 TPR vs Key Points

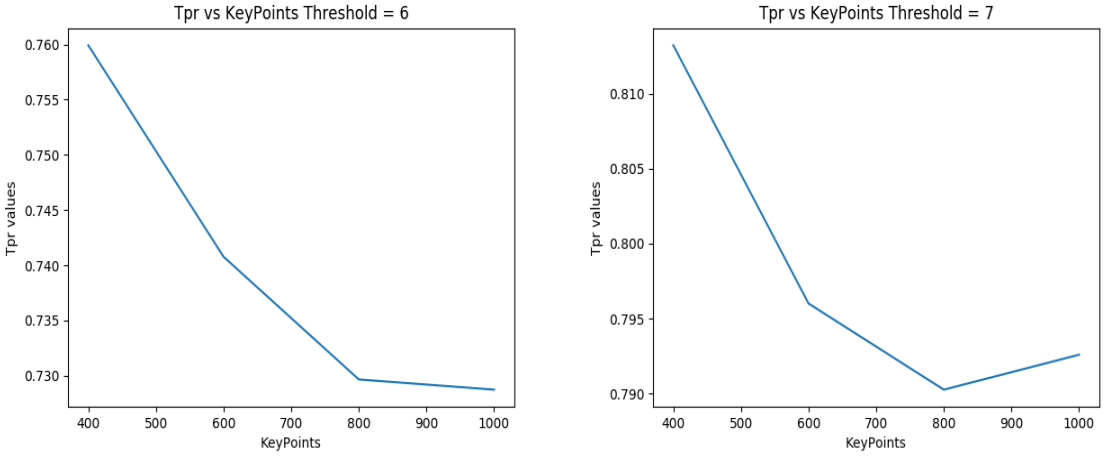

FIGURE 4.5: For 6 and 7 values of Threshold (TPR vs Keypoints)

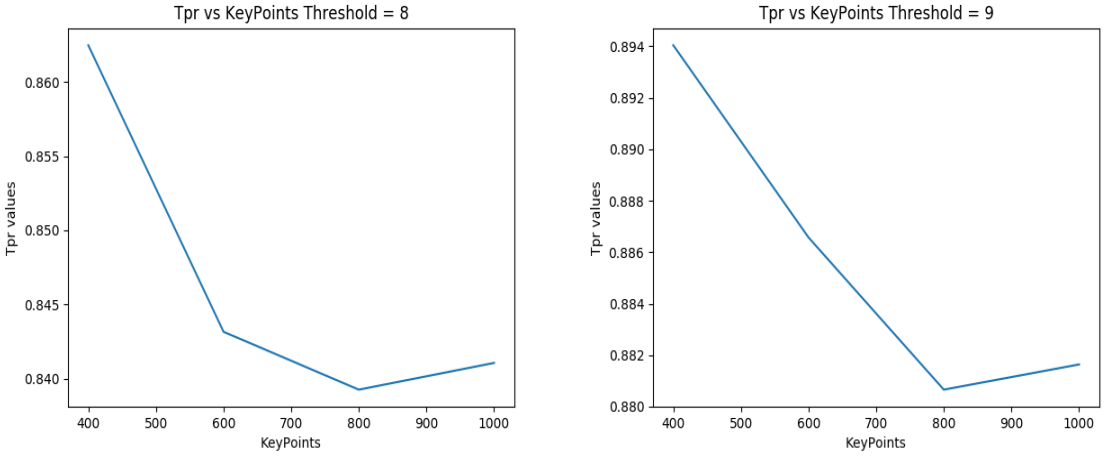

FIGURE 4.6: For 8 and 9 values of Threshold (TPR vs Keypoints)

From the experimental analysis between TPR and number of key points, we find that, the trade-off between number of key-points and TPR will be independent of the threshold value selected. With increase in the number of key points, the value of false negative increases simultaneously with true positive. As TPR is dependent on both the values of true positive and false negative, increase in the value of false negative will effect in the decrease of TPR value.

4.3.4 Accuracy vs Key Points

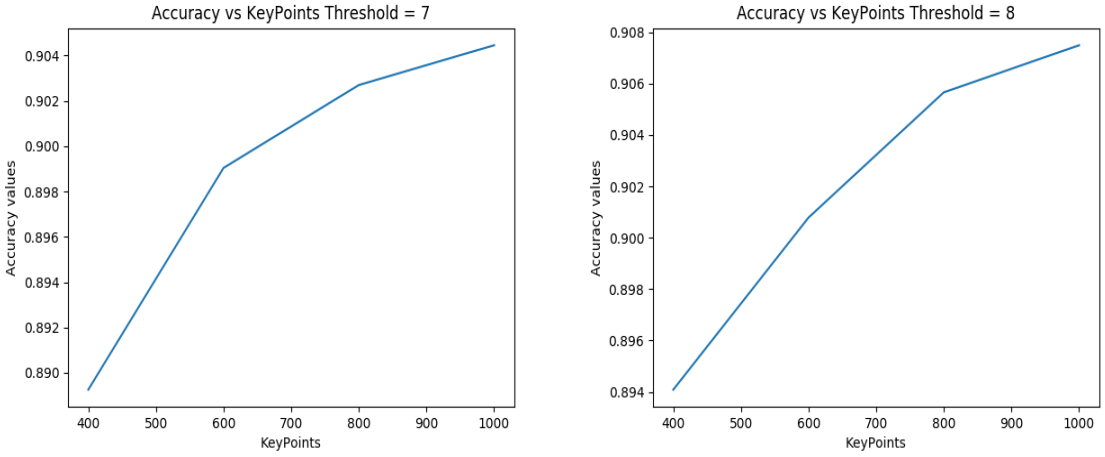

FIGURE 4.7: For 7 and 8 values of Threshold (Accuracy vs Keypoints)

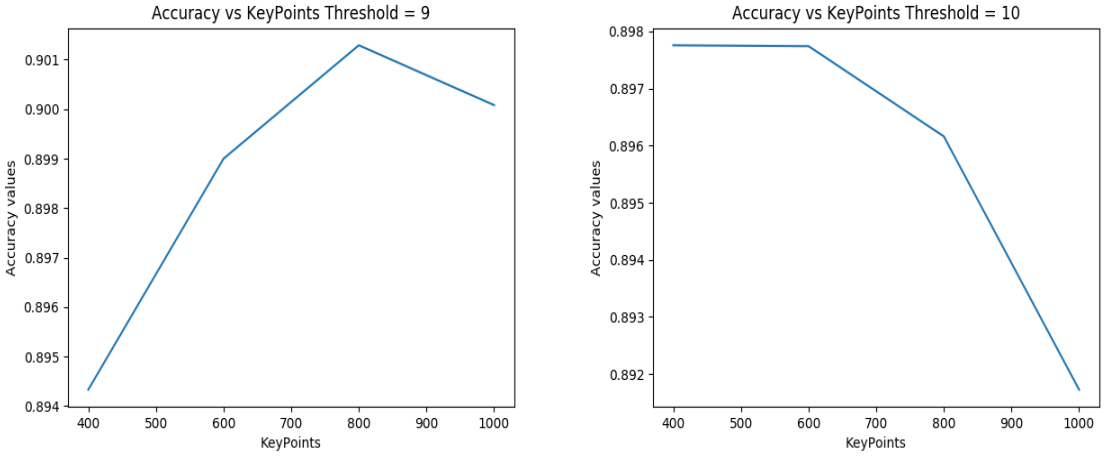

FIGURE 4.8: For 9 and 10 values of Threshold (Accuracy vs Keypoints)

From the experimental analysis between accuracy and number of key-points, We find that up to threshold value of 8, both false negatives and false positives were gradually increasing with increase in number of key points. But after a particular constraint of threshold value (here it is 8) false positive rises rapidly with increase in the number of key points. This is because increase in the threshold allows more number of miss matches between descriptor bitsets, resulting in the decrease of accuracy after threshold value of 8.

4.3.5 Time vs Threshold

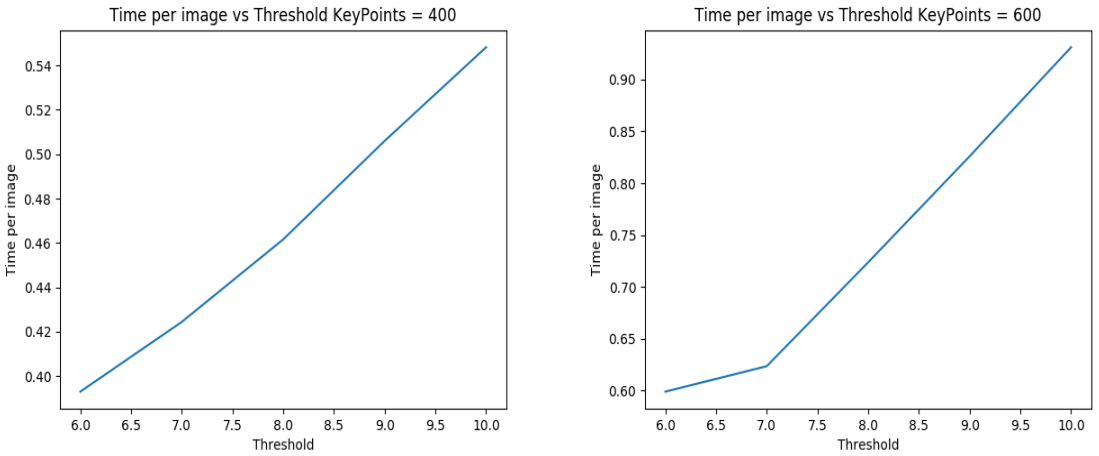

FIGURE 4.9: For 400 and 600 Number of Key Points (Time vs Threshold)

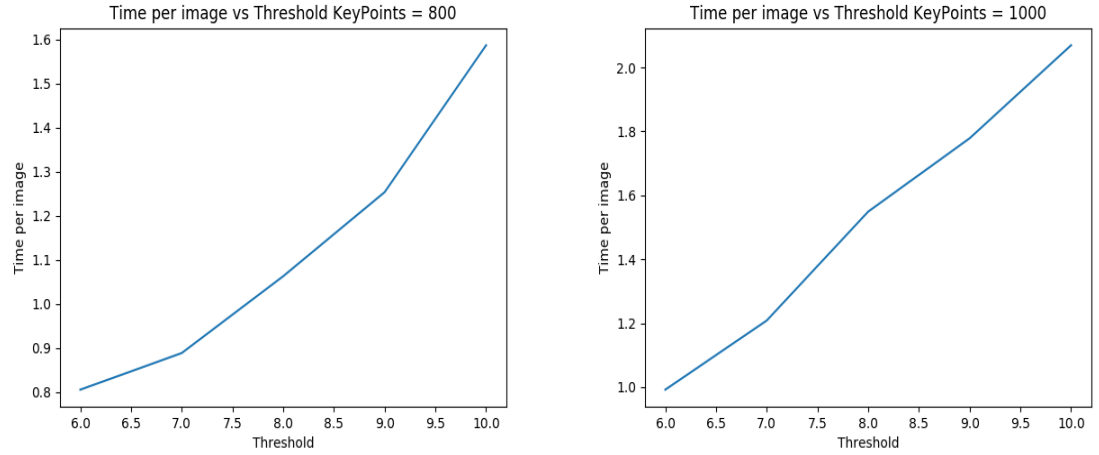

FIGURE 4.10: For 800 and 1000 Number of Key Points (Time vs Threshold)

From the experimental analysis between time and threshold values, it is straightforward to say that, with increase in the threshold value the time taken per image increases exponentially (refer to the complexity of the algorithm, mentioned above). With increase in the threshold value, it allows more number of mismatches between the descriptor bitsets making the way for two paths along the trie built resulting in the exponential relation between the threshold value and the time taken per image.

4.3.6 Time vs Key Points

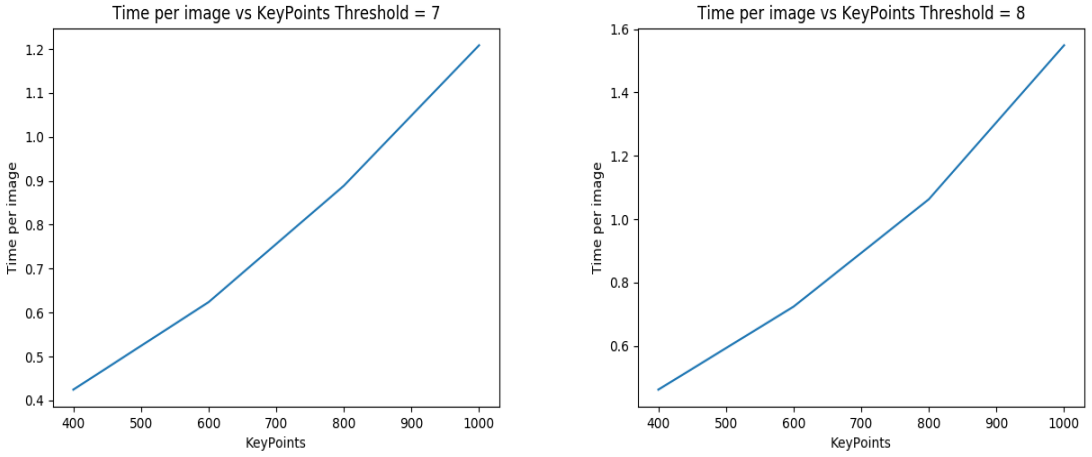

FIGURE 4.11: For 7 and 8 values of Threshold (Time vs Keypoints)

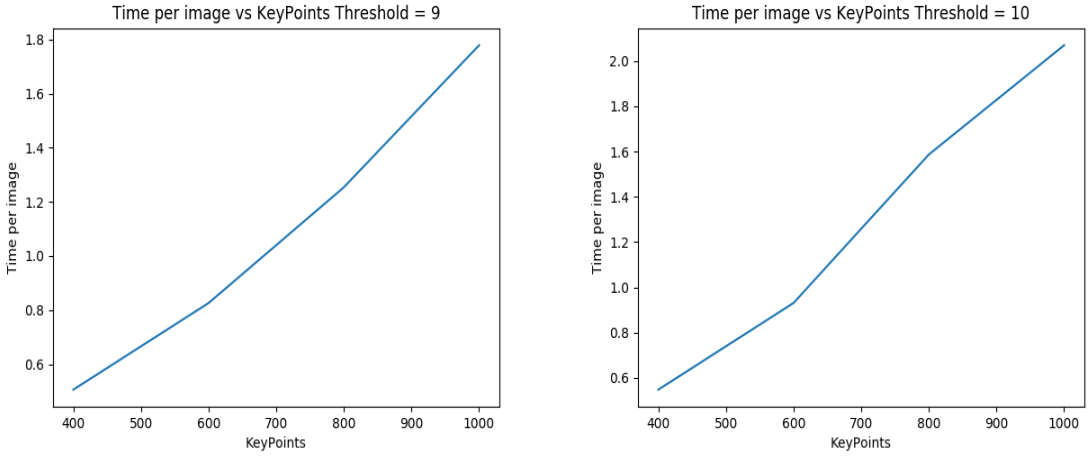

FIGURE 4.12: For 9 and 10 values of Threshold (Time vs Keypoints)

From the experimental analysis between time taken per image and number of key points considered, it is straightforward to say that, with increase in the number of key points the time taken per image increases linearly (refer to the complexity of the algorithm, mentioned above). With increase in the number of key points, we have to consider more number of descriptor bitsets for building the trie as well as for matching using the built tree. Both together increases the time taken per image for matching in the resultant built trie.

4.4 Analysis using MICC-F220

This dataset is composed of 220 images in which 110 are tampered and 110 are originals. Figures 4.13 - 4.20 show the detection results of MICC-F220 images in the presence of different geometric transformations. These results show that the proposed algorithm is robust against rotation and scaling as well as combined transformations.

4.5 Analysis using MICC-F2000

This dataset is composed of 2000 images in which 700 are tampered and 1300 are originals. Images of this dataset are of higher resolution and more complex than MICC-F2000. Figures 4.21 - 4.28 show the detection results of MICC-F2000 images in the presence of different geometric transformations. These results show that the proposed algorithm is robust against rotation and scaling as well as combined transformations.

4.6 Performance Comparison of Proposed System with Existing Methods

Performance comparison of proposed method with other existing techniques is given in Table 4.1 and Table 4.2 below.

TABLE 4.1: Performance comparison of proposed method with existing methods for MICC-F220 dataset

Method	FPR (in %)	TPR (in %)	Time (in sec)
Fridrich et al.[1]	84	89	294.69
Popescu et al.[12]	86	87	70.97
Xu et al.[8]	3.6	73.8	2.85
Li et al.[13]	8.8	91.6	14.45
Yang et al.[14]	9	95.9	10.20
Amerini et al.[5]	8	100	4.94
Proposed System	8.18	92.7	0.92

FIGURE 4.13: Only copy-move forgery (from Dataset MICC-F220)

FIGURE 4.14: Copy-move forgery with scaling (from Dataset MICC-F220)

FIGURE 4.15: Copy-move forgery with rotation (from Dataset MICC-F220)

FIGURE 4.16: Copy-move forgery with rotation and scaling (from Dataset MICC-F220)

FIGURE 4.17: Only copy-move forgery (from Dataset MICC-F220)

FIGURE 4.18: Copy-move forgery with scaling (from Dataset MICC-F220)

FIGURE 4.19: Copy-move forgery with rotation (from Dataset MICC-F220)

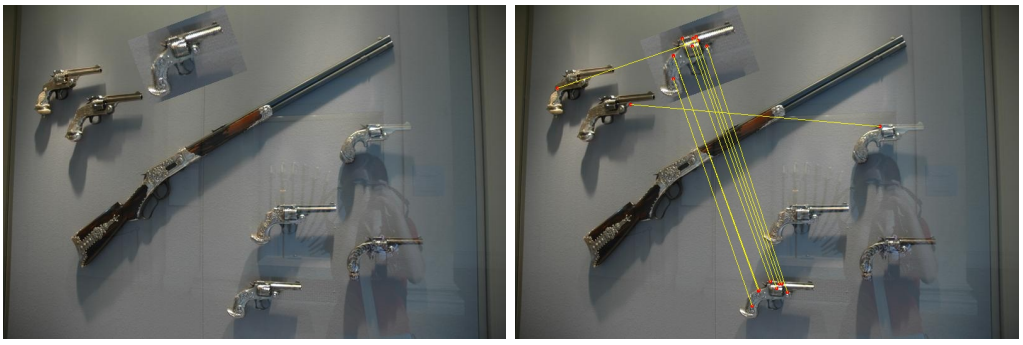

FIGURE 4.20: Copy-move forgery with rotation and scaling (from Dataset MICC-F220)

FIGURE 4.21: Only copy-move forgery (from Dataset MICC-F2000)

FIGURE 4.22: Copy-move forgery with scaling (from Dataset MICC-F2000)

FIGURE 4.23: Copy-move forgery with rotation (from Dataset MICC-F2000)

FIGURE 4.24: Copy-move forgery with rotation and scaling (from Dataset MICC-F2000)

FIGURE 4.25: Only copy-move forgery (from Dataset MICC-F2000)

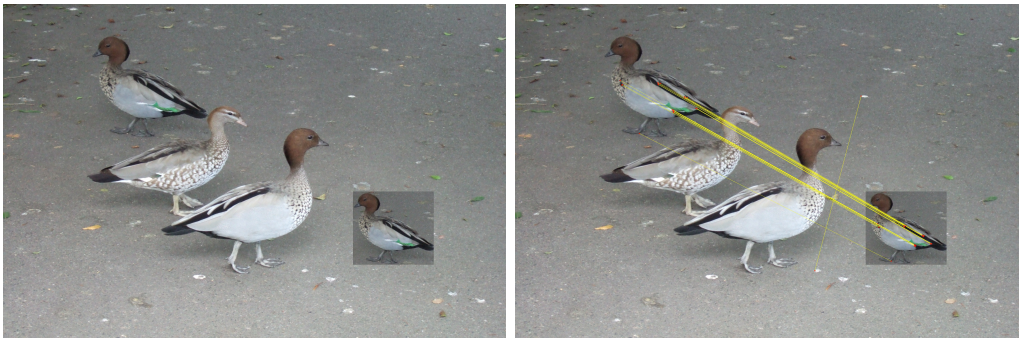

FIGURE 4.26: Copy-move forgery with (-ve) scaling (from Dataset MICC-F2000)

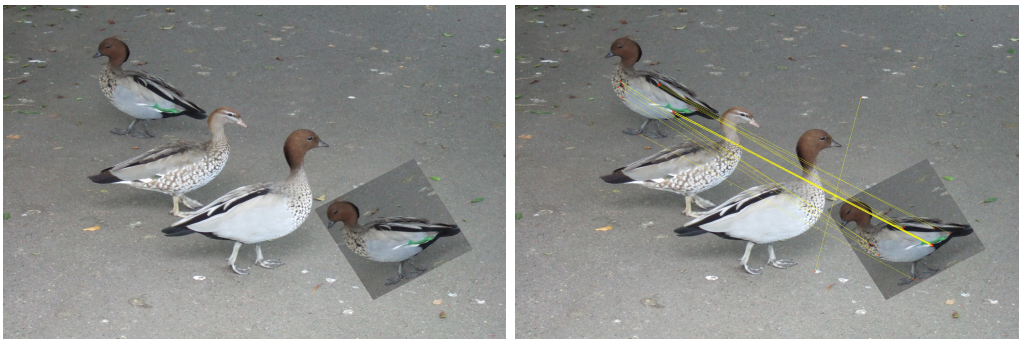

FIGURE 4.27: Copy-move forgery with rotation (from Dataset MICC-F2000)

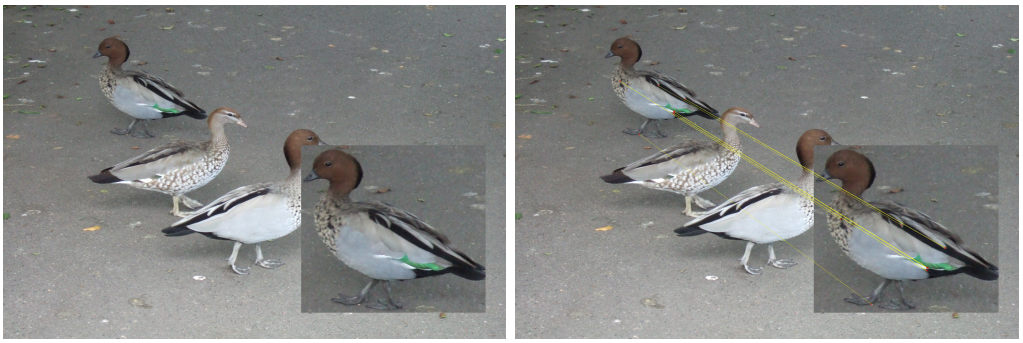

FIGURE 4.28: Copy-move forgery with (+ve) scaling (from Dataset MICC-F2000)

TABLE 4.2: Performance comparison of proposed method with existing methods for MICC-F2000 dataset

Method	FPR (in %)	TPR (in %)	Time (in sec)
Li et al.[13]	11.8	91.6	24.45
Yang et al.[14]	12	92.8	25.20
Zhong et al.[15]	14.8	93.8	22.40
Amerini et al.[5]	11.6	93.4	20.94
Amerini et al.[16]	9.2	94.9	19.20
Proposed System	9.33	92	1.73

4.7 Comparison of Matching algorithm

In Table 4.3 and Table 4.4 we have compared the trie-based matching algorithm with other methods of matching.

TABLE 4.3: Computational time for images of Dataset MICC-F220 (For 1000 Number of Key Points and Threshold value = 8)

Database (MICC-F220)	Image Size	Time for Brute Force match (in sec)	Time for 'Trie' based match (in sec)
Micc-f220-1.jpg	600*800	2.6115	0.5212
Micc-f220-2.jpg	532*800	2.7937	0.8622
Micc-f220-3.jpg	532*800	2.5523	0.9074
Micc-f220-4.jpg	599*800	2.5818	1.1580
Micc-f220-5.jpg	532*800	2.6421	0.8645
Micc-f220-6.jpg	532*800	2.5156	0.9124

TABLE 4.4: Computational time for images of Dataset MICC-F2000 (For 1000 Number of Key Points and Threshold value = 8)

Database (MICC-F2000)	Image Size	Time for Brute Force match (in sec)	Time for 'Trie' based match (in sec)
Micc-f2000-1.jpg	1536*2048	2.7060	1.0361
Micc-f2000-2.jpg	1536*2048	2.7936	1.0612
Micc-f2000-3.jpg	1536*2048	2.6550	1.0321
Micc-f2000-4.jpg	1536*2048	2.7561	1.0780
Micc-f2000-5.jpg	1536*2048	2.7658	1.0524
Micc-f2000-6.jpg	1536*2048	2.6425	1.0412

CHAPTER 5

Conclusion and Future Work

From the above experimental results on Datasets MICC-F220,

- Average computational time for SIFT based Algorithm [5] = 4.94
- Average computational time for Trie based match = 0.92

From the above experimental results on Datasets MICC-F2000,

- Average computational time for SIFT based Algorithm [5] = 20.94
- Average computational time for Trie based match = 1.72

The ratio of computational time between SIFT based match and Trie based match is 5.37 for MICC-F220 and 12.17 for MICC-F2000.

It can be observed from the experimental results, that the Trie based matching with ORB method is faster compared to SIFT based method. The above results were given when 1000 key-points were considered. In case of large images, it is obvious that even number of key-points are also increased. But, the computational time for Brute-force method increases in quadratic fashion, whereas, Trie based match increases in linear fashion.

As for making the algorithm feasible for a 64-bit Linux-based operating system, we considered only 64-bit length binary string as key-point descriptor. As we have used only 64 bit descriptor, there will be some false matches along with the true matches. So, by considering 64-bit bit descriptor we will work on improving the accuracy by minimizing the false matches and increasing the true matches. We also consider the fact that threshold also plays a vital role in providing good accuracy. So simultaneously we will be working on making the threshold adjustable with respect to the size of the image, number of detected key-points, etc.

References

[1] Fridrich, Soukal, and Lukas, *Detection of Copy-Move Forgery in Digital Images*. Cleveland, OH, US: Proceedings of the Digital Forensic Research Workshop, 2003.

[2] Sheng, Gao, Cao, and Fan, *Robust algorithm for detection of copy-move forgery in digital images*. Berlin, Germany: Springer, 2012.

[3] Zimba and Xingming, *DWT-PCA(EVD) based copy-move image forgery detection*. Dig. Content Technologies, 2011.

[4] Huang, Guo, and Zhang, *Detection of copy-move forgery in digital images using SIFT algorithm*. Wuhan, China: Pacific-Asia Workshop on Computational Intelligence and Industrial Application, 2008.

[5] Amerini, Ballan, Caldelli, D. Bimbo, and Serra, *A SIFT-based forensic method for copymove attack detection and transformation recovery*. Texas, US: IEEE Transactions on Information Forensics and Security, 2011.

[6] Chen, Chen, and Chien, *FAST image segmentation based on k-means clustering with histograms in HSV color space*. Cairns, Australia: IEEE 10th Workshop on Multimedia Signal Processing, 2008.

[7] D. G. Lowe, *Distinctive image features from scale-invariant keypoints*. International Journal of Computer Vision, 2004.

[8] X. Bo, W. Junwen, L. Guangjie, and D. Yuewei, *Image Copy-move Forgery Detection Based on SURF*. Shanghai, China: International Conference on Multimedia Information Networking and Security, 2010.

[9] Rublee, Rabaud, Konolige, and Bradski, *ORB: An efficient alternative to SIFT or SURF*. Barcelona, Spain: Computer Vision (ICCV), 2011 IEEE International Conference, 2011.

[10] C. Harris and M. Stephens, *A combined corner and edge detector*. Plessey Research Roke Manor, United Kingdom: The Plessey Company, 1988.

[11] Brass, *Advanced Data Structures*. Cambridge University Press, 2008.

[12] A. Popescu and H. Farid, *Exposing digital forgeries by detecting traces of resampling*. IEEE Transactions on Signal Processing, 2005.

[13] J. Li, X. Li, B. Yang, and X. Sun, *Segmentation-based image copymove forgery detection scheme*. IEEE Transactions on Information Forensics and Security, 2015.

[14] B. Yang, X. Sun, H. Guo, Z. Xia, and X. Chen, *A copy-move forgery detection method based on CMFD-SIFT*. Multimedia Tools and Applications, 2017.

[15] J. Zhong, Y. Gan, J. Young, L. Huang, and P. Lin, *A new block-based method for copy move forgery detection under image geometric transforms*. Multimedia Tools and Applications, 2017.

[16] I. Amerini, L. Ballan, R. Caldelli, A. Bimbo, L. Tongo, and G. Serra, *Copy-move forgery detection and localization by means of robust clustering with J-linkage*. Signal Processing: Image Communication, 2013.