Application Of Image Blobs And Illumination Component In Image Tamper Detection Techniques

A thesis submitted during 2021 to the University of Hyderabad in partial fulfillment of the award of a Ph.D. degree in School of Computer and Information Sciences

by

Niyishaka Patrick

School of Computer and Information Sciences
University of Hyderabad
P.O. Central University
Hyderabad - 500046
Telangana - India

March - 2021

CERTIFICATE

This is to certify that the thesis entitled **"Application of Image Blobs and Illumination Component in Image Tamper Detection Techniques"** submitted by **Niyishaka Patrick** bearing **Reg. No. 14MCPC21** in partial fulfillment of the requirements for the award of **Doctor of Philosophy** in **Computer Science** is a bonafide work carried out by him under my supervision and guidance.

This thesis is free from plagiarism and has not been submitted previously in part or in full to this or any other University or Institution for the award of any degree or diploma.

The student has the following publications before submission of the thesis for adjudication and has produced evidence for the same in the form of acceptance letter or the reprint in the relevant area of his research:

1. **Niyishaka Patrick**, Chakravarthy Bhagvati. "Copy-Move forgery detection using image blobs and BRISK feature". *Multimedia Tools and Applications*, https://doi.org/10.1007/s11042-020-09225-6. Work reported in this paper appears in **Chapter 4**
2. **Niyishaka Patrick**, Chakravarthy Bhagvati."Image splicing detection technique based on Illumination-Reflectance model and LBP". *Multimedia Tools and Applications*, https://doi.org/10.1007/s11042-020-09707-7. Work reported in this paper appears in **Chapter 6**

and has made the presentations in the following conferences:

1. **Niyishaka Patrick**, Chakravarthy Bhagvati. "Digital Image Forensics Technique for Copy-Move Forgery Detection Using DoG and ORB".2018 International Conference on Computer Vision and Graphics (ICCVG 2018), Warsaw, Poland, September 17-19, 2018, Proceedings, LNCS 11114, pp. 472-483, $10.1007/978-3-030-00692-1\_41$. Work reported in this paper appears in **Chapter 3**

Further, the student has passed the following courses towards fulfillment of coursework requirement for Ph.D.:

Course Code	**Name**	**Credits**	**Pass/Fail**
CS 801	Data Structures and Algorithms	4	Pass
CS 802	Operating Systems and Programming	4	Pass
AI 820	Digital Image Processing	4	Pass
IT 811	Secure Computing	4	Pass

(Prof. Chakravarthy Bhagvati)
Supervisor
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

(Prof. Chakravarthy Bhagvati)
Dean
School of Computer and Information Sciences
University of Hyderabad
Hyderabad – 500 046, India

DECLARATION

I, **Niyishaka Patrick**, hereby declare that this thesis entitled **"Application of Image Blobs and Illumination Component in Image Tamper Detection Techniques"** submitted by me under the guidance and supervision of **Prof. Chakravarthy Bhagvati** is a bonafide research work which is also free from plagiarism. I also declare that it has not been submitted previously in part or in full to this University or any other University or Institution for the award of any degree or diploma. I hereby agree that my thesis can be deposited in Shodganga/INFLIBNET.

A report on plagiarism statistics from the University Library is enclosed.

Date : **Name: Niyishaka Patrick**

Signature of the Student:

Reg. No.: 14MCPC21

Signature of the Supervisor:

Acknowledgements

I take this opportunity to express my gratitude to everybody who helped me directly or indirectly during this journey towards the doctorate degree.

First and foremost I want to thank my Ph.D. supervisor **Prof. Chakravarthy Bhagvati**. I appreciate all his contributions of time, guidance, ideas, knowledge, expertise and the exceedingly helpful assistance during tough time. His guidance and advice helped brighten my vocation and help me to overcome the Ph.D. challenges and obstacles.

I would like to thank the members of my Doctoral Review Committee (DRC) **Prof. Siba Kumar Udgata** and **Dr. Y. V. Subba Rao**. They have contributed immensely to my research work progress and my professional time at the University of Hyderabad.

I am thankful to each and every faculty member of the school of computer and information sciences, who have whole heartedly contributed to whatever I have learnt. I am especially thankful to **Dr. Wilson Naik Bhukya** and **Dr. Anjeneya Swami Kare** from whom I worked as teaching assistant. I sincerely appreciated the assistance, collaboration, support and help extended by the international office. My sincere appreciation and gratitude to Tagore international house for providing a wonderful accomodation.

A major part of my research work was supported by the Government of India through Indian Council for Culture Relations (ICCR). I am

especially grateful for the ICCR for providing me the Ph.D. scholarship, funding my university tuition fees and my stay in India.

I very much appreciated the collaborations, advice, inspirations, motivations, and friendships from the members and colleagues in the Optical Character Recognition (OCR) lab at the School of Computer and Information Sciences. I am thankful to my fellow lab mates (**Rajesh, Raghu, Rakesh, Melinda, Deepti, and Salman**) for working together and sharing chai with me at the student canteen.

Finally, my heartfelt thanks to my lovely family, this journey would not have been possible without your love, support, consideration, guidance, and encouragement.

Niyishaka Patrick

Abstract

This thesis presents several techniques to detect, localize and estimate parameters of two major types of tamerping digital images: *copy-move forgery* and *image splicing forgery*. In the former, a portion of an image is duplicated, sometimes with geometric transformations, and pasted into another part of the same image. In the latter, a part of a second image is pasted into an image. Existing methods in literature generally require a number of parameters that are tuned to various image characteristics for efficient and robust detection of forgeries. Our motivation is to find general techniques that minimize the number of such tunable parameters while overcoming certain other of existing copy-move and image splicing detection methods.

The problem of photograph manipulation has attracted many researchers in the field of *Digital Image Forensics (DIF)* that aims at validating the authenticity of images by identifying the imaging device that captured the image or detecting the traces of forgeries. Image tampering, also called image forgery, is the act of image editing and manipulation for malicious purposes to modify the semantic meaning of the visual message. The availability of powerful image editing tools has greatly simplified the process of malicious manipulation, editing, and creating tampered images even by lay persons.

In this thesis, we propose the use of image blobs in copy-move forgery detection algorithms. Image blobs are the regions that differ in properties such as brightness or color compared to surrounding regions. The goal of blob detection is to identify and mark these regions. As blobs are detected in scale-space, we demonstrate through our experiments

on standard benchmark datasets that they present several advantages over image blocks and segments in copy-move forgery detection.

Image splicing merges the portions of images from different sources into one composite image. This introduces various artifacts such as sharp transitions around the pasted area and abnormal transient at the splicing boundaries; and introduces illumination inconsistencies because the images are taken from different cameras with different lighting conditions. It also produces some resample traces because it is often necessary to downscale or upscale certain portions of an image. The existing splicing forgery detection techniques exploit the tampering artifacts and the visual information present in the image to expose the traces of splicing.

We approach the problem from the perspective of detecting illumination inconsistencies. The idea is to use the $YCbCr$ color space to obtain luminance and chrominance from the input image. To capture illumination changes as discriminant features, we extract illumination component using *Illumination-Reflectance model* of image formation, which considers the intensity at any pixel as the product of the illumination of the scene and the reflectance of the object(s) in the scene. Illumination normally varies slowly across the image as compared to reflectance that can change suddenly at object edges. This difference is the key to split the illumination component from the reflectance component. We use the Local Binary Patterns to describe the spatial arrangement of colors and capture texture information of the image. Then, all extracted features are fed into different machine learning algorithms such as a Support Vector Machine, Linear Discriminant Analysis, etc, to categorize the input image as authentic or forged.

Our results on detecting and localising copy-move forgery regions, presented in Chapters 3 and 4, show that the use of image blobs results in performance comparable to the state-of-the-art while requiring only a few tunable parameters. Image blobs combined with many off-the-shelf image are capable of extracting geometric transformation information

from copy-move forgeries. The results show that the selection of a feature is not as important when using image blobs. Finally, results on image splicing forgeries show that illumination inconsistencies are as useful as image content based features for forgery detection.

Dedicated to my family and friends, without whose support and encouragement, this would not have been possible.

Contents

List of Figures

List of Tables

List of Acronyms

AO Adaptive Oversegmentation
ANNSP Approximate Nearest Neighbor Searching Problem
AKAZE Accelerated-KAZE
BRISK Binary Robust Invariant Scalable Keypoints
BRIEF Binary Robust Independent Elementary Features
DIF Digital Image Forensics
DoG Difference of Gaussians
DWT Discrete Wavelet Transform
LoG Lapracian of Gaussian
CMF Copy Move Forgery
CMFD Copy Move Forgery Detection
LBP Local Binary Patterns
IoU Intersection over Union
ORB Oriented Fast and Rotated BRIEF
DCT Discrete Cosine Transform
FT Fourier Transform
SWT Stationary Wavelet Transform
ED Euclidean Distance
EM Expectation Maximization
CPPN Compositional Pattern-Producing Network
MROGH Multi-support Region Order-based Gradient Histogram
MLDD Multi-Level Dense Descriptor
HFM Hierarchical Feature Matching
HD Hamming Distance
PCET Polar Complex Exponential Transform
DRHFM Discrete Radial Harmonic Fourier Moments
2NN 2 Nearest Neighbors
CNN Convolutional Neural Networks
RANSAC Random Sample Consensus
TPR True Positive Rate
FPR False Positive Rate
SVD Single Value Decomposition
LSH Locality Sensitive Hashing
DRHFM Discrete Radial Harmonic Fourier Moments
SVM Support Vector Machine

FAST Features from Accelerated Segment Test
DyWT Dyadic Wavelet Transform
HOG Histogram of Oriented Gradients
QDCT Quaternion Discrete Cosine Transform
GLRLM Grey Level Run Length Matrix
SCNN Shallow Convolution Neural Network
LSTM Long-Short Term Memory
LPT Log Polar Transform
SIFT Scale Invariant Feature Transform
SURF Speed-Up Robust Feature
GPU Graphics Processing Unit
PCA Principal Component Analysis
DT Decision Tree
LR Logistic Regression
LDA Linear Discriminant Analysis
KNN K-Nearest Neighbors
NB Naive Bayes
SPT Steerable Pyramid Transform
LPF Low Pass Filter

Chapter 1

Introduction

Today's sophisticated computer tools have made it effortless to do image tampering and the question is "can we trust a digital image content anymore"?

The main purpose of image tampering is to mislead the viewers or public opinion. History shows that Joseph Nicéphore Niépce took the first photograph made with a camera in 1827 [1, 8, 9], but around 1860 photographs were already being manipulated [8]. Since then, with the digital era, the manipulation of photographs has become simpler and easier by computer tools. Since engineer Steven Sasson [14] invented the first self-contained (portable) digital camera back in 1975, the digital revolution has changed the way how images are created, consumed and perceived. In 1987 the photo editing computer program for personal computers known as Adobe Photoshop was released. Today, it is so popular such that the term "photoshop" is used to call attention to digital image editing and manipulation [46].

The digital era has propelled the digital image to become an important source of information in several areas such as the internet, social media, courts of law, industries, and other several fields. However, today's technology has made effortless to counterfeit both origin and content of digital image and has enabled the digital image to be manipulated and tampered with low cost to deceive the public, as quoted by Hany Farid [74]: "Today's technology allows digital media to be altered and manipulated in ways that were simply impossible twenty years ago". Several tools for digital image editing like Adobe Photoshop, Adobe Illustrator, Gimp, CorelDRAW, etc., [46] can easily be obtained on the market at low cost

or as free and open-source; with such tools, advanced image manipulations have become simple even for non-experts, and the reliability of digital media (photo and video) is doubted.

A deliberate modification of images for malicious motives that include adding, removing, or changing some important features from the image is known as *image tampering* or *image forgery* [119]. Often, this malicious image manipulation is post-processed to hide any conspicuous trace of tampering [108]. Image forgery attempts to mislead the viewers by altering the semantic meaning of the visual message.

According to [23], around 95 million photos are uploaded daily on the social media platform "Instagram"; and since its creation in 2010, more than 40 billion photos have been shared. Around 350 million images per day are uploaded to the social media platform "Facebook" [6, 7]. Now, with the enormous amount of photographs on social media, the questions arise like in [67, 68], how many of the images can we trust on social media? How to tell the authentic from the tampered?

Also, the digital image forgery is being made more sophisticated by the recent advancements in deep learning [93] and computer vision [17]. Computer vision tools like *OpenCV* [16] have introduced the "Seamless cloning" feature that allows the user to seamlessly clone parts of the source image onto a destination image to create a realistic composition in just ten lines of codes in Python or C++. Normally, a photoshop editor would spend a few hours and meticulously alter the brightness on the source image to the brightness of the destination image to fabricate a realistic composition. Another tool called *Faceswap* [91] is used to recognize and swap faces in photos and movies. Faceswap utilizes deep learning to transform the input identity into a target identity while preserving facial expression, brightness and pose.

Several digital image forgery detection algorithms have been proposed. However, to discriminate forged images from original images has become progressively challenging [119]. The need for robust, effective, and efficient image tampering detection techniques is still very significant. Also, image tamper detection techniques pay less attention to image contents (semantic) than to tampering clues (forgery artifacts). Thus, image forgery detection differs from conventional object detection (semantic) [111].

To restore some trust to digital images, the *DIF* field has emerged [119]. DIF aims are:

- To identify the camera that captured the image to validate its authenticity.
- To detect the traces of forgeries.

Generally, DIF categorises digital image forgery detection algorithms into *active* and *passive or blind* approaches [119]. *Active approach* inserts a digital information (watermark or signature) at the source side and it is verified at the destination side. *Passive approach* does not consider any prior information that is inserted beforehand. The active method exhibits a reliable detection accuracy but the precondition of inserting information at the source side is strenuous in some practical applications. The passive detection approach has attracted more attention as most of today's images and videos on social media, web, internet and other fields don't have digital information (watermark or signature) embedded [119].

1.1 Motivation

The most common types of digital tampering include[110, 119]:

- *Copy-move or image region duplication*
- *Image/photo splicing*
- *Image/photo retouching*

Hany Farid quoted: "The field of image forensics, however, has made and will continue to make it harder and more time-consuming (but never impossible) to create a forgery that can not be detected" [74]. The gap between the image tampering techniques and image tampering detection techniques is still vast, and to narrow it was the main drive that directed us in this field .

Numerous image tamper detection methods to combat the Copy Move Forgery (CMF) and image splicing have been proposed [77, 110, 119]. However, it is effortless to make sophisticated tampered images even for amateurs, whereas it requires great effort to make image tampering detection tools even for experts.

To narrow the gap between the image tampering techniques and image tampering detection techniques, the field of DIF has attracted many researchers. Thus, akin to [79], this study was inspired by the considerable need for powerful image forgery detection techniques to assess image authenticity and to ensure the credibility of digital image contents.

We focused on tackling the limitations and drawbacks of the existing image tampering detection techniques that deal with the two common types of digital image tampering known as the CMF, and the *image splicing forgery*.

1.2 Scope of Work

The main goal of this study is to determine the limitations of the existing *CMFD* and the *image splicing forgery detection techniques*, then provide digital image tamper detection techniques to tackle those limitations. The specifications to achieve our objectives are as follow:

1. Identify the important classifications of image tampering. Chap. 2.

    - Basically, DIF enumerates three categories of digital image forgery known as the *copy-move*, *image/photo splicing*, and *image/photo retouching* [110]. *Copy-move forgery* or *cloning*: an image is tampered by copying a region and pasting it into another part of the same image. This type of image tampering is composed of image regions duplication from the same image as it is shown in Fig. 2.1. *Image splicing* or *image composition*: splices two or more different images to create a doctored (spliced) image [37, 53, 86]. This category of image tampering is composed of several different images that are merged into a single doctored image. Usually, the edges between the spliced regions are visually imperceptible to the naked eye [74]. Fig. 2.2 shows image splicing forgery. *Image retouching or airbrushing*: is done to intensify or to lessen some image features to improve the image appearance (polishing). Image retouching doesn't alter the semantic features [44].

2. Develop detection methods for copy-move and splicing image tampering to tackle the limitations and drawbacks of existing detection methods. Chapters 3, 4, 5, and 6.

    - Detection techniques for CMF are known as aCMFD techniques and they are generally categorized into *block-based*, *keypoint-based*, and *segment-based* approaches [109, 110] . Fig. 2.4 shows the general structure for CMFD.

      Block-based approach: image is split into small blocks (overlapping or non-overlapping). Features are extracted from the blocks and are compared to find which blocks or features that are similar. Block-based techniques are effective for detecting CMF under noise addition and image compression [110].

      The limitations of block-based techniques include the difficulty of finding the appropriate size of the block. The image blocks are proportional to the image pixels. Robust features cannot be extracted from small blocks. The the computational cost is high when using the small blocks. Large blocks cannot be used to detect small tampered regions. Uniform areas (e.g., background) can be detected as duplicates [109, 115].

      Keypoints-based methods: the keypoints are found in the image. Then the regions around the keypoints are described to generate the feature vectors. The feature vectors are analyzed to detect similar regions. Keypoints-based CMFD techniques are effective for detecting copy-move forgery when tampered regions are scaled or rotated.

      The main drawbacks of keypoint-based techniques include the large number of keypoints to match, the large number of false matches, and the need for filtering techniques such as Random Sample Consensus (RANSAC) [71] for reducing of false positives [110].

      Segment-based methods: splits an image into irregular non-overlapping regions (segments or superpixels) [116].

      The main drawback of segment-based methods includes the splitting of a single CMF region into two or more regions (oversegmentation).

Also, uniform areas such as the background areas can be detected as duplicates.

- For image splicing, the regions of the spliced image are obtained from different original images. It is difficult to achieve proper illuminant condition (uniform illumination) for the entire spliced image because different parts of the spliced image are taken from different cameras with different lighting conditions [112]. Abnormal edges are introduced at spliced boundaries [143] and it is frequently inevitable to resize certain portions of an image (resampling images into a new sampling grid) [112].

  Existing methods for detecting image splicing try to detect the abnormal edges (boundary-based) [123, 143], or they detect inconsistencies of the image characteristics (non-uniform illumination, blur estimation, resampling artifacts) [112].

  Most recent techniques consider the image splicing detection problem as a binary classification problem [66, 142]. They consider two phases for the splicing detection task:

  - Phase 1: extracting features and training the classifier (classify image as authentic or tampered).
  - Phase 2: localizing the spliced areas in the tampered image.

  The existing image splicing techniques exhibit some limitations that need to be tackled [107]:

  - The existing methods with high detection accuracy are computationally expensive. Majority of them are based on complex deep learning models. They are expensive to train and require a large amount of data to perform better. They also run on expensive Graphics Processing Unit (GPU).
  - There is no known standard mechanism to localize the spliced areas.

3. In the end, conclusions, recommendations and direction for future work are detailed in Chap. 7.

1.3 Contribution

We proposed the utilization of *image blobs* [90, 100, 101] in CMFD algorithms. Since image blobs are regions detected in scale-space, they present certain advantages over image blocks and image segments in CMFD techniques. We proposed the following three different methods for copy-move forgery detection.

- Copy-move forgery detection using DoG blob detector [101] and ORB features [48].
- Copy-move forgery detection using DoG blob detector and Binary Robust Invariant Scalable Keypoints (BRISK) features [95].
- Geometric transformation parameters estimation from copy-move forgery using image blobs.

We also proposed utilizing *Illumination-Reflectance model* [24, 54, 120] in image splicing to get the illumination component from an image. Assuming that the image splicing perturbs the spatial arrangement (texture information) of color and intensity in the image, we use the Local Binary Patterns (LBP) [66] to extract texture features. To categorize image as tampered or authentic, we used different classifiers such as Support Vector Machine (SVM) [65], and Linear Discriminant Analysis (LDA) [129].

1.4 Thesis Organization

The thesis contains 7 chapters. Chap. 2 summarizes the related works from literature. Copy-Move forgery detection methods using block-based, keypoint-based and segment-based techniques are presented in detail so that the context of our work is properly brought out. The major limitations of the existing approaches, the background material on image blobs and the basic ideas on how they may be used for forgery detection are presented in Chap. 2. Chap. 3 deals with the first of our contributions, *viz*, the use of blobs and their efficacy demonstrated through the use of ORB features. Chap. 4 continues with techniques for handling CMF and describes our second contribution, that is, the use of BRISK features which

overcome certain limitations of ORB features when used in conjunction with blobs. Chap. 5 enhances the tamper detection of CMF by extracting the geometric transformation parameters used in making the forged images. A variety of standard off-the-shelf features are used to show that the blob-based approach is inherently capable of such forgery parameter estimations.

Chap. 6 deals with Image-Splicing forgeries where a portion of a second image is pasted into an image. Texture analysis of illumination components, extracted using edge-preserving smoothing filter is shown to work well for detecting image-splicing forgeries. Chap. 7 is a brief summary of our main contributions and presents the main conclusions from our work.

1.5 Summary

An introduction to DIF is given in this chapter. The cases of image tampering that involve the *copy-move forgery* and *image splicing* are discussed. This chapter also talked about image forgery through history and described the gap between the image tampering techniques and image tampering detection techniques. Finally, the need for robust image tampering detection is described, and a brief description of our contribution to the field was introduced.

Chapter 2

Related Work

2.1 Introduction

This chapter covers the different image tampering methods with focus on copy-move and image splicing. Previous research is described and analysed to identify gaps and weaknesses. The chapter serves the content for our research and help in explaining certain algorithmic and design decisions on parameters used in our work. Copy-move forgery is described in Sect. 2.1.1 and image splicing detection methods are described in Sect. 2.2. The details of copy-move forgery and the state-of-the-art of forgery detection methods are discussed in Sect. 2.2. Sect. 2.2.5 discusses the limitations of existing CMFD methods and Sect. 3.2 describes the use of image blobs in CMFD to tackle the limitations of existing methods. Sect. 2.3 discusses the recent image splicing detection methods and their limitations are described in Sect. 2.3.3.

2.1.1 Copy-move forgery

Copy-move forgery or cloning refers to copying a region of an image and pasting it into the same image (see Fig. 2.1 (b) where a white car is copied and pasted into two different regions, whereas a black car is copied and pasted into one region).

2.1.2 Image splicing

Image splicing consists of merging two or more images to produce a tampered (spliced) image [106] as illustrated in Fig. 2.2.

2.1.3 Image/photo retouching

Photo retouching or airbrushing alters image features to improve image appearance (polishing) but the semantic features remain the same (see Fig. 2.3) [44].

Figure 2.1: (a) Authentic image. (b) Copy-move forged image.

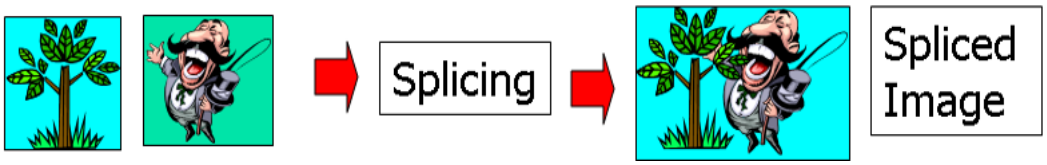


Figure 2.2: Image splicing forgery

Figure 2.3: Image retouching

Our work focuses on copy-move forgery and image splicing because the semantic features are altered in both forgeries.

2.2 Related Work: CMFD methods

A CMF consists of one or more regions copied and pasted into the same image [119]. The motivations behind such falsification comprise highlighting a specific region or object or hiding a specific object or region in the image (steganography). Since both copied and pasted areas are from the same image, their image characteristics like color, noise and contrast will basically be similar. Therefore, this type of forgery is usually unnoticeable by naked eyes. CMF is often combined with geometric transformation and post-processing operations to prevent the detection of manipulated regions. Visible clues like sharp edges introduced by CMF are eliminated by those post-processing operations, whereas geometric transformation operations provide the uniformity between the tampered region and its surrounding regions [119]. [119].

A large number of CMFD methods have been proposed to deal with the CMF [110, 119].

Generally, the workflow of a CMFD technique consists of a *pre-processing stage, feature extraction stage, feature matching stage, and localization stage* [119].

Pre-processing aims to enhance image features or to suppress inadvertent distortions. Image processing techniques such as edge detection, color conversions and dividing the image into a number of blocks are performed in this stage [108].

Feature extraction aims to select information (image characteristics) that can lead to expose the CMF. Common methods for feature extraction on image blocks include 2D-Fourier Transform (FT) [127], Discrete Cosine Transform (DCT) [77], etc. Common methods for feature extraction on whole image include Scale Invariant Feature Transform (SIFT) [135] and BRISK [95].

Feature matching stage aims to find out the resemblance between different features in the image. Also, similar image blocks are found at this stage. Matching techniques of image blocks include Euclidean Distance (ED) and correlation, whereas matching techniques for invariant keypoint features include Hamming Distance (HD) and ED [110].

Finally, the tampered regions in the forged image are localized and visualized. For image blocks, the region of the matching blocks are colored or mapped for the visualization. The lines between every matching pairs are used to visualize the matching invariant keypoint features [108].

Usually, CMFD methods are categorized into three approaches: *block-based*, *keypoint-based* and *segment-based* [109]. In block-based approach, image is divided into small overlapping or non- overlapping blocks, the blocks or block features are matched against each other to determine blocks or features that are similar [134].

In keypoint-based approach, keypoints are detected in the regions/areas with high entropy in the image without any image partitioning. Then, each keypoint is described by a feature vector. Finally, the feature vectors are compared against each other to determine which feature vectors are similar [36].

Some recent techniques have adopted image segments (segment-based) as alternatives to image blocks in CMFD [38, 109, 116]. Image is divided into non-overlapping regions called segments or superpixels [41], features are extracted in each segment, and features from different segments are matched to find out which ones are similar. Fig. 2.4 shows the general structure for CMFD and Tab. 2.1 shows recent CMFD methods.

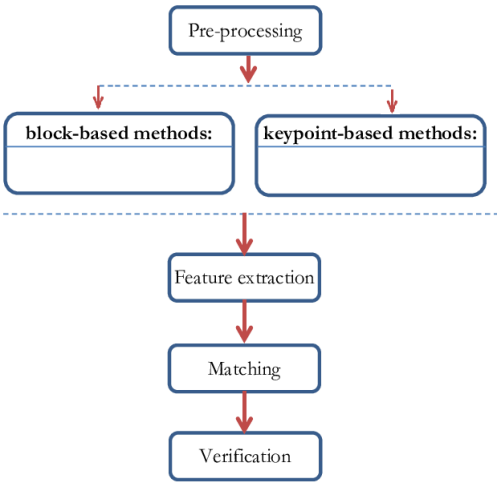


Figure 2.4: Flowchart for CMFD

Table 2.1: Summary of recent CMFD methods

Approach	Method	Author
Block-based	DWT and DCT	Hayat et al. [77](2017)
	Discrete Cosine Transform	Alkawaz et al. [34] (2018)
	Color-segmentation with SWT and SVD	Dixit et al.[59] (2017)
	Circular Harmonic Transforms and PatchMatch	Cozzolino et al.[57] (2015)
	Color information and its histograms	Zhou et al.[145](2016)
	First order moment with SWT	Mahmood et al.[103] (2017)
	Polar representation	Fadl et al.[127](2017)
	Dense descriptor and feature matching	Bi et al. [45](2016)
	Polar transform and ANN	Emam et al.[62] (2016)
	Modified PatchMatch algorithm	Cozzolino et al.[56](2014)
	DRHFMs and 2NN test	Zhong et al. [144](2017)
	Trigonometric transforms and deep learning	Faten et al. [69](2020)
Keypoint-based	Agglomerative hierarchical clustering	Amerini et al.[36](2011)
	J-Linkage clustering algorithm	Amerini et al.[35](2013)
	SIFT and symmetry-based matching	Warif et al.[135] (2017)
	Maximally stable detector and best-matching	Li et al.[96](2016)
	Harris detector and MROGH	Yu et al.[140](2016)
	KAZE feature detector	Yang et al.[139](2017)
	Optimized J-Linkage	Jin et al.[85](2017)
	Automatic matching with SIFT features	Mahdi et al.[102](2020)
	Reduced Local Binary Pattern histogram with scale-invariant features	Jun et al.[87](2020)
Segment-based	Superpixel segmentation and the Helmert transformation	Hui et al.[80](2019)
	Feature matching with adaptive over-segmentation	Pun et al.[116](2015)

2.2.1 Block-based approach

lock-based CMFD methods usually split an image into square or circular blocks (overlapping or non-overlapping) for analysis. Then, they extract important features from each block. The extracted features are matched to find corresponding image blocks. Various image block descriptors include the 2D-FT [127], DCT [77], Log Polar Transform (LPT) [127], Discrete Wavelet Transform (DWT) [77], geometric moment [103], Single Value Decomposition (SVD) [59], histogram [145], and Principal Component Analysis (PCA) [66], etc. After the corresponding image blocks are found, these image blocks constitute the manipulation of CMF conducted in the image. Though the block-based methods are effective in CMFD, often they are not robust to post-processing (compression, noise addition, etc.) operations [134].

In block-based, often an image of size $W \times H$ is split into small blocks (fixed-size and overlapping) of $wb \times hb$ pixels by sliding the block one pixel at a time

resulting in N_b blocks.

$$N_b = (W - wb + 1) * (H - hb + 1) \tag{2.1}$$

However, N_b is proportional to the number of pixels in the image, and it is challenging to set the appropriate size of wb and hb [109, 134].

- N_b is proportional to W and H. It increases as image size increases.
- Small blocks don't give robust features.
- The computational cost is high when using the small blocks.
- Large blocks cannot be used to detect the presence of small tampered areas.
- Uniform areas (e.g., background) can be detected as duplicates.

Alkawaz et al.(2018)[34] studied the effects of different block sizes (4×4 and 8×8) to investigate the effect of block size in CMFD. Their method uses DCT coefficients obtained from various block sizes. Tab. 2.2 reports the performance results on ten selected images from CoMoFoD standard dataset [132]. They concluded that the performance is influenced by the different sizes of tampered region, distance between two tampered regions and the threshold value.

Table 2.2: The performance of different block sizes

Images	block size	Precision(%)	Recall(%)
Image I	$4X4$	63.52	97.89
	$8X8$	100	97.53
Image II	$4X4$	42.55	95.54
	$8X8$	95.19	96.25
Image III	$4X4$	27.44	99.70
	$8X8$	49.80	100
Image IV	$4X4$	19.30	98.07
	$8X8$	87.62	99.48
Image V	$4X4$	23.02	92.19
	$8X8$	63.32	88.47
Image VI	$4X4$	3.52	99.35
	$8X8$	62.53	97.30
Image VII	$4X4$	10.37	99.62
	$8X8$	59.51	99.86
Image VIII	$4X4$	59.25	98.68
	$8X8$	9.25	98.68
Image IX	$4X4$	15.85	95.57
	$8X8$	41.49	93.37
Image X	$4X4$	1.73	98.60
	$8X8$	26.58	94.85

Dixit et al.(2017)[59] presented a CMFD technique that utilizes the Stationary Wavelet Transform (SWT) and SVD. Their method uses SWT to produce sab-bands and LL (low frequency) sub-band is split into blocks (overlapping). Features are extracted from blocks using SVD. Finally, they use ED to measure similarities between features.

Zhou et al.(2016)[145] proposed a CMFD technique that uses color-related information and its histograms. Their method splits the image into blocks (fixed size overlapping). They clustered the search space based on color distribution. Blocks from the tampered regions reside within the same cluster because CMF regions have similar color distributions. For each image block, the algorithm extracts the color features like color moments, texture histogram and fuzzy color. These extracted features form a 50-dimensional feature vector which is fed to a Compositional Pattern-Producing Network (CPPN). Similar feature vectors indicate the forged regions. For images distorted with $33dB$ noise signal, their method gives accuracy higher than 83.6%.

Mahmood et al.(2018)[103] proposed an efficient CMFD technique that uses the translation invariant SWT. SWT divides the image in sub-bands, then overlapping blocks are computed from low frequency sub-band. First order moment from blocks are matched to indicate the forgery. The accuracy is 97.023% for block size 8×8. For block size 4×4 the accuracy is 96.05%.

Fadl et al.(2017)[127] proposed a robust CMFD using polar coordinate system. Their technique enhances the block matching by utilization of polar representation as features for each block. The frequency of each block is the main feature. Their experiment results on Columbia dataset [106] show the precision rate of 99.9% for CMF without post-processing operations . For post-processed CMF, the precision varies from 85.02% to 99.5%.

Bi et al.(2016)[45] presented a CMFD technique that uses a Multi-Level Dense Descriptor (MLDD) and a Hierarchical Feature Matching (HFM). The MLDD is used to extract feature (color texture and invariant moment) and HFM method detects tampered regions using extracted features. Finally, CMF regions are highlighted using morphological operations. The experiments carried out on image manipulation dataset [55] show a precision of 87.20% and 89.68% recall.

Emam et al.(2016)[62] presented an efficient CMFD technique that divides an image into overlapping circular blocks. For each block, a Polar Complex Exponential Transform (PCET) is computed. The PCET kernels represent each block. Similar blocks are identified using the Locality Sensitive Hashing (LSH) with the Approximate Nearest Neighbor Searching Problem (ANNSP). To make their method more robust, they used morphological operations small holes and the wrong similar blocks (false matches).

Zhong et al.(2017)[144] presented a block-based efficient method for CMFD that uses circular block. Firstly, forged image is pre-processed by a Gaussian filter to reduce the additive white noise, then overlapping circular blocks are computed. The Discrete Radial Harmonic Fourier Moments (DRHFM) is computed to extract features and a 2 Nearest Neighbors (2NN) test is used to search similar feature vectors. Correlation coefficients and ED are used to reduce the false positives. Finally, they used morphologic operation to remove the isolated pixels. For an image of size $W \times H$, the number of circular blocks (overlapping) CiB is:

$$CiB = (W - 2r) * (H - 2r) \tag{2.2}$$

r is the radius of the circular block.

Faten et al.(2020)[69] proposed a method that uses deep learning for CMFD. Their technique uses a Convolutional Neural Networks (CNN) which has convolution layers, pooling layer and a fully-connected layer. Their method uses a dense layer to detect (classify) image as original or forged.

2.2.2 Keypoint-based approach

Keypoint-based CMFD methods detect feature (keypoint and its descriptor) in the image without splitting the image. A descriptor is generated within the region around the keypoint. Descriptors are matched to find similar regions in the image. Common feature detection algorithms include Harris corner detector [75] , SIFT [135], Speed-Up Robust Feature (SURF) [124], ORB [48], BRISK [95], and Accelerated-KAZE (AKAZE) [128].

Keypoint-based CMFD techniques are effective for forgeries that include the geometric transformation (scaling and rotation) [108]. Their limitations consist of the large number of keypoints to match, matching between neighboring keypoints produce a lot of false matches and the need for filtering techniques such as RANSAC to reduce the false positives [110]. Also, many detectors are designed to work on grayscale images where a remarkable color information is disregarded [134].

Jun et al.(2020)[87] proposed a CMFD that uses a LBP histogram and SIFT features. Their technique starts by extracting SIFT features in the image. Then, LBP values are obtained from the local regions around the features and descriptors are generated. Finally, a matching process is done and false matches are removed using RANSAC.

Debbarma et al.(2014)[58] analyzed the keypoints-based CMFD using SIFT and SURF algorithms. Their technique extracts interest points in image using SIFT then SURF. The feature vector size is 64 for SURF and 128 in case of SIFT. To find similar keypoints, a matching operation is performed using Euclidean distance. Finally, false positives are reduced using an agglomerative hier-

archical clustering. Experiments performed on the MICC-F220 dataset [36] show the True Positive Rate (TPR) of 91% for SIFT and 88.18% for SURF. False Positive Rate (FPR) results are 6.36% for SIFT and 5.45% for SURF.

A keypoint-based CMFD technique that uses SIFT, ORB, SVM and Expectation Maximization (EM) was proposed by Rajdeep et al. (2016)[89]. Features are extracted in images using SIFT and ORB. Then, SVM and EM are utilized to detect (classify) an image as original or forged using the extracted features. Their experimental results show accuracy of 92.5% for ORB+EM and 90.5 for SIFT+EM. The CMFD tests are performed on the MICC-F600 dataset [36].
ORB is a robust local feature detector and keypoint descriptor (see Fig. 2.5(b)) [64]. It is composed of an oriented keypoint detector called FAST and a rotated binary feature descriptor called Binary Robust Independent Elementary Features (BRIEF). ORB is recommended for real-time applications because it is faster than SIFT [101] and SURF [42]. In FAST, a pixel Px is a keypoint if there is a set of $\eta = 12$ connected pixels in the Bresenham circle of 16 pixels which are either lighter than $I(px) + t$ or darker than $I(px) - t$. I is intensity and $\mathbf{t}$ is a threshold. Fig. 2.5(a) shows a Bresenham circle of 16 pixels with the radius of 3 units. A high speed test is done to avoid a large number of non-corners, this test

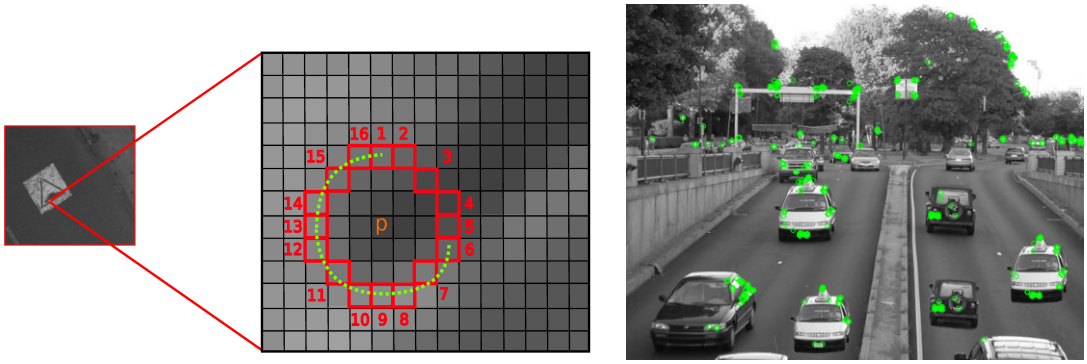


Figure 2.5: (a) FAST . (b) ORB.

examines only the four pixels at 1, 9, 5 and 13. ORB uses the following modification for an orientation compensation mechanism since FAST doesn't compute the

orientation. Given image patch $I(x, y)$, the moments of a patch are defined as:

$$M_{pq} = \sum_x \sum_y x^p y^q I(x, y) \tag{2.3}$$

With the above moments a centroid is found, the "center of mass" C of the patch as:

$$C = \frac{M_{01}}{M_{00}}, \frac{M_{10}}{M_{00}} \tag{2.4}$$

The orientation θ of the patch is given by :

$$\theta = tan^{-1} \frac{M_{01}}{M_{00}} \tag{2.5}$$

BRIEF is a binary feature descriptor such as LBP [27], it only performs simple binary comparison tests. To generate a binary descriptor, intensity between two pixel positions located around the detected interest points are compared. This enables to obtain a feature descriptor at very low computational cost. BRIEF takes a smoothed image region and selects a set of $n(x, y)$ location pairs in that region. Then the pixel intensity comparisons are done on these location pairs. Considering a smoothed image region P, a binary test τ is given by:

$$\tau(P; x, y) := \begin{cases} 1 & : \text{P(x)} < \text{P(y)} \\ 0 & : \text{P(x)} \geq \text{P(y)} \end{cases} \tag{2.6}$$

Where $P(x)$ is intensity of P at point x and $P(y)$ is intensity of P at point y. BRIEF uses a bitstring to describe an image patch as a feature. The feature is defined as a vector of n binary tests ($n = 256$) [64].

$$fn(P) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_i, y_i) \tag{2.7}$$

A CMFD using Gabor filters and Scaled-ORB was proposed by Muzaffer et al.(2016) [105]. Their technique starts by computing an histogram equalization on the image, then image is filtered using Gabor filters. Finally, ORB features are extracted and matched to find the tampered areas.

Zhu et al.(2015)[146] proposed a CMFD using Scaled-ORB. Their method

works by establishing a Gaussian scale space. ORB features are extracted in each scale space and those features are matched using HD to find similar features. Finally, RANSAC algorithm is used to reduce false matches. Their CMFD method is effective for geometric transformation but the computational cost is high for high resolution pictures.

Hashmi et al.(2014)[76] proposed a CMFD method that uses the Dyadic Wavelet Transform (DyWT) and SIFT features. DyWT divides the image in sub-bands, then SIFT features are extracted from low frequency sub-band because it contains most of the information. SIFT features are matched to find similar features and to conclude if some copy-move tampering have been done. Their experiment results show a precision of 88% and a recall of 80% on images from dataset MICC-F220 [36].

Amerini et al.(2013)[35] presented a CMFD technique that uses the J-Linkage algorithm with SIFT features. Their method starts by extracting and matching SIFT features. The corresponding features are clustered using J-Linkage algorithm in a transformation domain. The original and the duplicated regions share similar transformations because they have similar conceptual representations.

Warif et al. [135] proposed a CMFD method which uses a symmetry-based matching and SIFT. The overview of their technique is shown in Fig. 2.6.

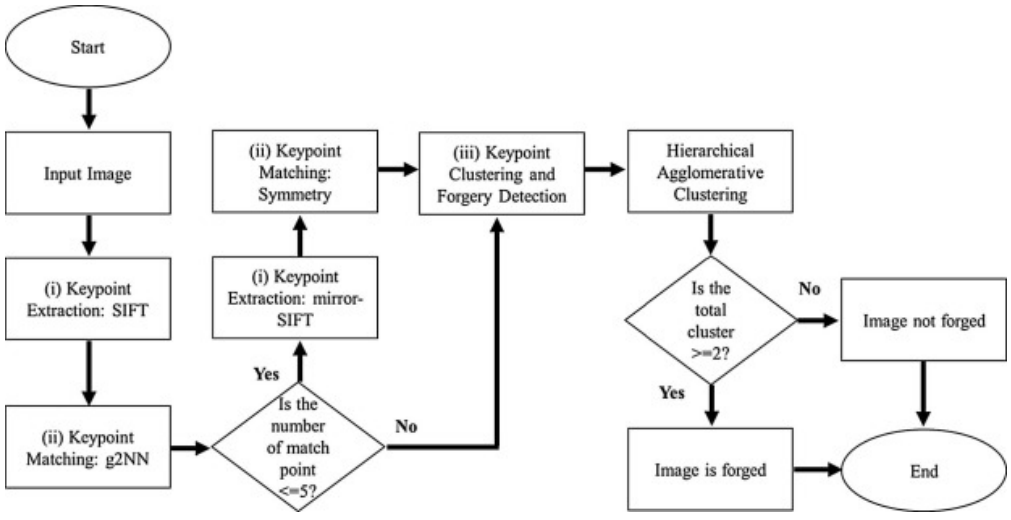


Figure 2.6: SIFT-Symmetry based CMFD

Their experiment results show an average of 80% value of f_1 score for geometrical transformation cases and an average of 65.3% of f_1 score for rotation and reflection cases.

Li et al.(2017)[96] proposed an effective CMFD method that uses the maximally stable color region detector to extract features in image. Then the Zernike moments are used to represent these features. To detect multi-copy regions, the technique uses a matching strategy considering n best-matching features. Finally, the technique computes an hierarchical cluster algorithm and estimate the transformation matrices. The duplicated regions are located at pixel level using these matrices. A CMFD method based on Harris detector and Multi-support Region Order-based Gradient Histogram (MROGH) feature descriptor was proposed by Yu et al.(2016)[140]. Harris method detects keypoints in both textured and smooth regions and MROGH is used as feature descriptor. After feature matching, false positives are filtered and the geometric transformation between CMF regions are estimated. Finally, forgery detection maps to indicate the cloned regions are generated.

Yang et el.(2017)[139] proposed a CMFD method based on hybrid features by combining KAZE and SIFT. Their technique computes KAZE and SIFT to provide more feature points. These features are matched to find CMF. Finally, an image segmentation technique is used to reduce false positives and CMF regions are mapped based on correlation coefficient.

Jin et al.(2017)[85] proposed a CMFD using an improved SIFT and optimized J-Linkage. Their method enhances the discriminative power of SIFT keypoints using the OpponentSIFT as feature descriptor. To reduce the high computational cost of clustering, J-Linkage algorithm is optimized using a matched pair grouping method.

2.2.3 Segment-based CMFD

Pun et al.(2015)[116] proposed an image forgery detection that segments the image and extracts the feature points in segments. The Adaptive Oversegmentation (AO) is used to split the image into segments (non-overlapping and irregular regions).

Feature points are extracted and matched to locate similar segments. Different segments are merged if they have similar features.

2.2.4 Hybrid techniques

The hybrid techniques combine the block-based and keypoint-based CMFD approaches in a single detection method [110]. A typical Hybrid method that computes the DCT on image blocks and extracts keypoints using SURF was proposed by Ojeniyi et al.(2018)[110]. The combination of DCT and SURF techniques in a single method compensates the failures found in each of the two techniques for a successful detection of CMF. Their experiment results show an accuracy of 95.45% on images from dataset MICC-F220 [36].

2.2.5 The main limitations of block-based, keypoint-based, and segment-based CMFD approaches

1. The limitations of block-based techniques include the difficulty of finding the appropriate size of the block. Small blocks don't give robust features. The computational cost is high when using the small blocks. Large blocks cannot be used to detect the presence of small tampered areas. Uniform areas (e.g., background) can be detected as duplicates [108].

2. The limitations of keypoint-based techniques include the large number of keypoints to match, matching between neighboring keypoints produce a lot of false matches, and the need for filtering techniques such as RANSAC to reduce the false positives [109].

3. The main limitations of segments-based methods include the oversegmentation of a single copy-move region and the uniform areas (e.g., background) can be detected as duplicates [109].

2.2.6 Conclusion

A copy-move forgery is composed of one or more regions copied and pasted within the same image. Different methods to detect the copy-move foregry are called

copy-move forgery detection techniques. These techniques are based on image blocks, image keypoints, or image segments. The limitations of existing block-based, keypoint-based, and segment-based copy-move forgery detection techniques have been assessed. The CMFD methods that can tackle these limitations need to be proposed.

2.3 Related Work: splicing detection methods

Image splicing (composition) is the merging of two or more images to produce a tampered image. This composition introduces the following artifacts into the new created image:

- Abnormal transient at the splicing boundaries: some sort of disparities (abnormal sharp boundaries) will occur in those areas that were copied and pasted [123, 143].
- Non-uniform illumination (illumination inconsistencies): since images are captured from different cameras under different lighting effects (distribution of light), this introduces the illumination inconsistencies in the spliced image [123].
- Resampling: it is often inevitable to resize certain regions of image while creating a spliced image. The image is resampled into a new sampling grid. Thus, spliced image hosts some resampling properties that can be measured [112].
- Perturbation of the texture information: the spatial arrangement of intensity or color are modified [107].

And also, some post-processing operations including blurring, noise addition, brightness change, contrast adjustment, and JPEG compression are applied to spliced image to hide the traces of forgery.

Existing methods for detecting image splicing try to differentiate the abnormal edges from the normal ones (boundary-based) [123, 143] or rely on the inconsistencies of the image characteristics (non-uniform illumination, blur estimation, recompression artifacts) [112].

Most recent techniques consider the image splicing problem as a binary classification problem [66, 142]. They consider two phases for the splicing detection task:

- Phase 1: extracting features and training the classifier (classify image as authentic or tampered).
- Phase 2: localizing the spliced areas in the tampered image.

Existing methods for detecting image splicing exhibit some limitations [107]:

- The existing methods with high detection accuracy are computationally expensive. Most of them rely on complex deep learning models which are expensive to train, run on expensive GPU, and require a large amount of data to perform better.
- There is no known standard mechanism to localize the spliced areas.

Table. 2.3 shows most recent image tampering detection techniques for image splicing forgery detection.

Table 2.3: Summary of recent image splicing detection methods

Methods	Author
LBP, DWT, PCA, and SVM	Fahime et al. [66](2015)
Edge and Illuminant Color Estimation	Youseph et al.[123](2015)
Shallow CNN	Zhang et al.[143](2018)
Hybrid CNN-LSTM	Bappy et al.[83](2017)
Markov features in QDCT domain	Li et al.[52](2017)
GLRLM and SVM	Mushtaq et al.[121](2014)
FAST R-CNN	Peng et al.[114](2014)
Convolutional Layer	Bayar et al.[43](2016)
Resampling Features	Bayar et al.[112](2017)
Deep Residual Network	Jaiswal et al.[82](2019)
Hybrid feature set	Jaiswal et al.[81](2020)
Markov features and PCA	Rachna et al.[117](2019)
SURF with ripplet Transform-ii	Jeyalakshmi et al.[30](2019)
Deep Learning Local Descriptor	Yuan et al.[142](2020)
Deep Learning and Haar Wavelet Transform	Eman et al.[63](2019)

2.3.1 Handcrafted features based techniques

An image splicing detection method which uses LBP, DWT, and PCA was proposed by Fahime et al.(2015)[66]. Their method uses LBP, DWT, and PCA to extract features. Then extracted features are fed to a SVM classifier to categorize the image as authentic or tampered. LBP [84, 133] is used as a robust feature for image texture classification.

DWT of a function $f(k,l)$ of size $W \times H$ is given by :

$$w\varphi(j_0,m,n) = \frac{1}{\sqrt{WH}} \sum_{k=0}^{W-1} \sum_{l=0}^{H-1} f(k,l)\varphi j_0,m,n(k,l) \tag{2.8}$$

$$w_\psi^i(j,m,n) = \frac{1}{\sqrt{WH}} \sum_{k=0}^{W-1} \sum_{l=0}^{H-1} f(k,l)\psi^i j,m,n(k,l) \tag{2.9}$$

$i = \{H_0, V_e, D_i\}$, j_0 is arbitrary starting scale, $w\varphi(j_0,mc,nc)$ coefficients that define approximation of $f(k,l)$ at scale j_0, and $w_\psi^i(j,mc,nc)$ coefficients are horizontal, vertical and diagonal details for $j_i \geq j_0$.

SVM is a machine learning algorithm (discriminative classifier) described by a separating hyperplane (see Fig. 2.7) [25, 26]. The coordinates of individual obser-

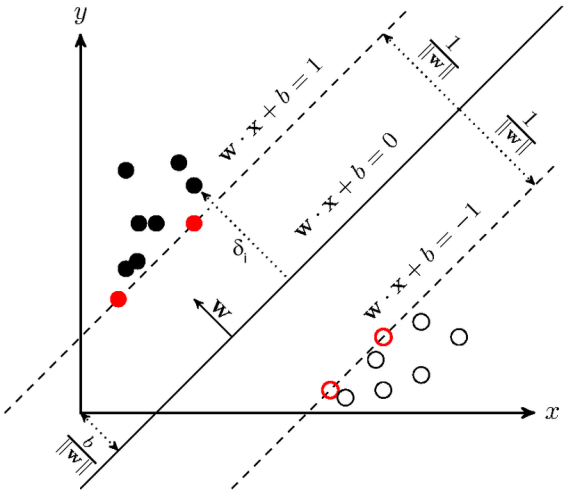


Figure 2.7: Support vector machines

vation are called the support vectors. $\vec{W}$ determines the orientation of hyperplane.

b controls the displacement from origin. $Margin$ can be described by using 2 hyperplanes : $\frac{2}{||\vec{W}||}$. Then solve the optimization problem: max $\frac{2}{||\vec{W}||}$ or min $\frac{||\vec{W}|}{2}$ with Constraints: $W^T x_1 + b \geq 1$, $W^T x_2 + b \leq -1$.

Youseph et al.(2015)[123] presented a method for detecting splicing forged images of humans using illuminant color estimation. Their technique uses the Pixel and Edge based method to estimate a map of illuminant color. Then the edges of the map are detected using Canny edge detector and the gradients of edge points are computed using Histogram of Oriented Gradients (HOG). Color moments features are also extracted. Finally, SVM is used to detect (classify) an image as original or forged using the extracted features.

Celi et al.(2017)[52] proposed an image splicing detection algorithm based on Markov chains in Quaternion Discrete Cosine Transform (QDCT) domain. Their technique works by extracting color information from blocked images. The QDCT coefficients of quaternion blocked images are obtained. Then, the expanded Markov features generated from the transition probability matrices in QDCT domain capture the intra-block and the inter-block correlation between block QDCT coefficients. At the end, SVM is used to classify the Markov feature vector as tampered or not.

Mushtaq et al.(2014)[121] proposed an image splicing detection technique which uses the Grey Level Run Length Matrix (GLRLM) Their technique calculates the GLRLM texture features for an image. Extracted features are fed to a SVM to detect (classify) the forged and non-forged features.

Peng et al.(2017)[112] proposed a technique that detect the forgery resampling and image resampling. Forgery resampling and image resampling leave interpolation artifacts due to interpolation. Their method attempts to capture the traces of resampling using the coefficients of the autoregressive model and histograms as the features. Extracted features are fed to SVM to categorize image as forged or non-forged.

2.3.2 Deep learning based methods

Yuan et al.(2020)[142] proposed a deep learning based image splicing detection method. In the first step, their method pre-trains a CNN model on labelled data

(original or spliced). The pre-trained CNN focuses on the local statistical artifacts and learns the structure for tampered image patches to build a pre-trained CNN-based local descriptor. The test image is split into blocks. For each block, the pre-trained CNN-based is applied to extract feature. The feature maps are obtained from the last convolutional layer and a block pooling technique is used to obtain the discriminative feature vector. In the final step, SVM is used to detect (binary classification as authentic or forged) and a localization scheme is developed. The experimental results show 96.97% accuracy on images from CASIA image tampering detection evaluation dataset [60].

A boundary-based image forgery detection using Fast Shallow Convolution Neural Network (SCNN) was proposed by Zhang et al.(2018)[143]. The SCNN is used to differentiate the boundaries of tampered areas from original edges. SCNN are CNN with limited number of convolution layers. The CNN extracts low-level features (e.g., edge information) in the first layers. They limited the number of convolution layers to two to prevent the SCNN to learn complex spatial features from image. The pooling layers are also discarded because their SCNN is extremely shallow. 20% of the convolution kernel filters are initialized with the Laplacian kernel to expose both edge and resampling features. Table. 2.4 shows the architecture for the SCNN for the model.

Layer (type)	**shape**	**Parameters**
Input	(None,32,32,3)	-
CbCr channels	(None,32,32,2)	-
$Conv_1$	(None,30,30,32)	608
$Activation_1$(ReLU)	(None,30,30,32)	0
$Conv_2$	(None,28,28,32)	9248
$Activation_2$(ReLU)	(None,28,28,32)	0
$Dense_1$	(None,64)	1605696
$Activation_3$(ReLU)	(None,64)	0
Dropout	(None,64)	0
$Dense_2$	(None,1)	65
$Activation_4$(Sigmoid)	(None,1)	0

Table 2.4: SCNN architecture.

After training the SCNN, their method uses a Sliding Window Detection (SWD) of size 32×32 to construct a probability map and localize the tampered regions. Fig. 2.8 shows the process for generating a tampered region bounding box.

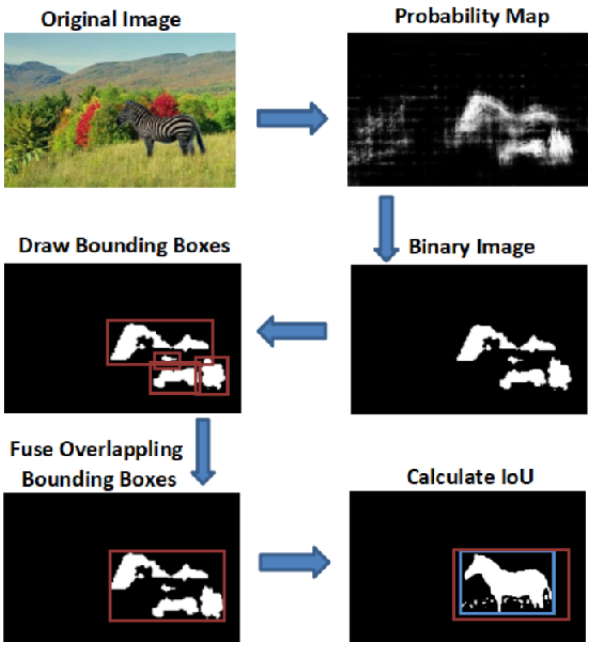


Figure 2.8: Tampered region bounding box.

Bappy et al.(2017)[83] proposed an hybrid method for image forgery detection using CNN and Long-Short Term Memory (LSTM). The tampered regions exhibit discriminative features in boundaries shared with neighboring non-tampered pixels and this model attempts to learn these boundary discrepancies. Using the ground-truth mask information, the network is trained to learn the parameters through back-propagation. Their technique is able to detect different types of image forgery like image splicing and CMF. Fig. 2.9 shows the overview of the framework based on CNN-LSTM.

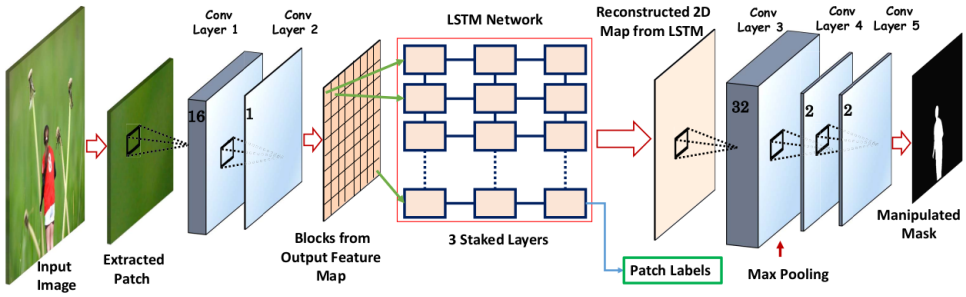


Figure 2.9: Overview of the framework based on CNN-LSTM.

Peng et al.(2018)[114] used Faster R-CNN (Region Based Convolutional Neural Networks) network to detect image splicing. Their method uses a two-stream Faster R-CNN. The first stream is called the RGB stream, it is used to detect tampering artifacts (e.g., unnatural tampered boundaries and strong contrast difference) in RGB image. The second stream is called the noise stream, it is used to find the noise inconsistency between orginal and forged regions. Then they used a bilinear pooling layer to fuse the features from these two streams. Fig. 2.10 illustrates the overview of their method.

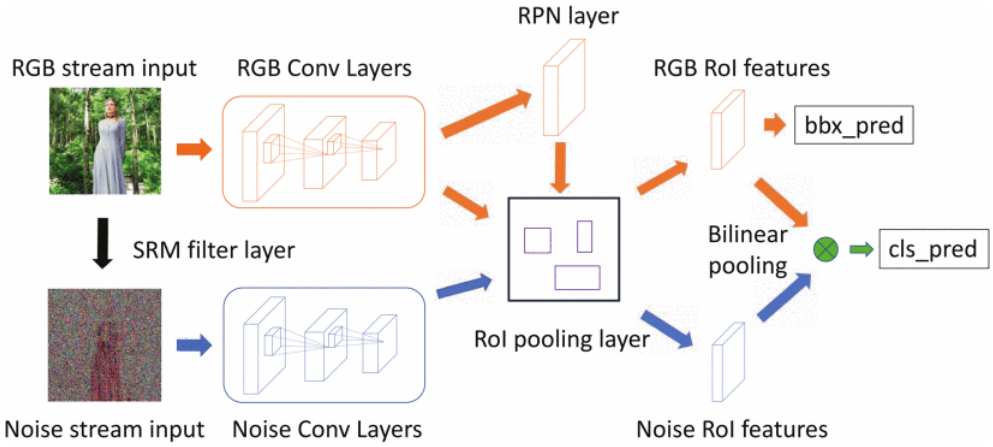


Figure 2.10: Overview of the two-stream faster R-CNN network

Bayar et al.(2016) [43] proposed an image tamper detection technique that uses deep learning with a new form of convolutional layer. Usually, convolutional neural networks learn features related to image's content. To enable the convolutional neural networks to automatically learn features related to image manipulation from

training dataset, they used a constrained CNN that learns the local structural relationships between pixels. These relationships are independent of an image's content.

2.3.3 Main limitations of existing techniques

The existing splicing tamper detection methods with high detection accuracy are computationally expensive. Majority of these methods are based on complex deep learning models. They are expensive to train, require a large amount of data to perform better, and run on expensive GPUs. Also, there is no known standard mechanism to localize the spliced areas [107].

2.3.4 Conclusion

The literature review for image forgery involving image splicing has been discussed. Image splicing is the merging of two or more images to produce a tampered image. Most recent techniques to detect the image splicing are based on machine learning and deep learning, these techniques categorize an image as authentic or tampered based on features extracted from the image. We have also highlighted the overall limitations exhibited by different image splicing detection methods. Thus, new detection techniques are needed in the field to tackle these limitations.

2.4 Summary

This chapter provides an overview of digital image forgery including the CMF, image/photo retouching and image/photo splicing. The related works, which are the state-of-art detection methods for copy-move forgery and image splicing forgery are discussed. The limitations of existing tamper detection techniques are assessed. Finally, the needs to propose new methods that can overcome these limitations are highlighted.

Chapter 3

Copy-Move Forgery Detection using DoG Blob Detector and ORB

3.1 Introduction

This chapter covers the use of image blobs in CMFD techniques. Image blobs can be seen as variable size and variable shaped blocks that basely approximate foreground segments. Hence, image blobs can be used to tackle some limitations of existing CMFD methods. Sect. 3.2 describes the advantages of using image blobs as alternative to image blocks in CMFD. Sect. 3.3 discusses the enhancement of blob localization on foreground regions. Interest points extraction stage is detailed in Sect. 3.4. Feature matching process is discussed in Sect. 3.5. The pseudocode for CMFD method using image blobs and ORB is shown in Algorithm 1 and Sect. 3.7 shows the experimental results.

3.2 The use of image blobs in CMFD to tackle the limitations of existing methods

An image blob is a region/area in a digital image that looks different from its neighbors at different scales. Blobs also differ in properties, such as color or brightness, compared to neighboring regions. A technique that detects these regions (blobs) is called a blob detector [3, 4]. The LoG and DoG are most common blob detectors.

To find blobs, LoG convolves an image with a blob filter at multiple scales and looks for extrema of filter response in the resulting scale space. This blob filter is obtained from the second order derivative of a Gaussian filter along x and $y-axis$ and adding them. The Gaussian blur removes the noise and stabilizes the second order derivative which is sensitive to noise. Given a Gaussian filter of a standard deviation σ, with x and $y-axis$, 2D LoG filter is computed by:

$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right] e^{-(x^2+y^2)/2\sigma^2} \tag{3.1}$$

Fig. 3.1 shows the Gaussian filter and LoG filter.

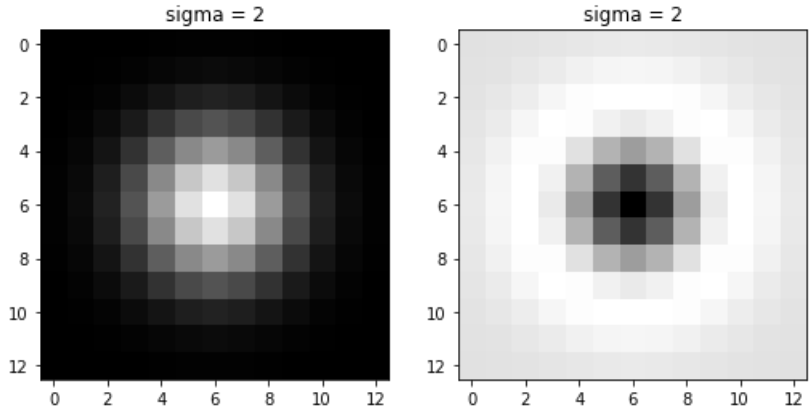


Figure 3.1: (a) Gaussian Filter at σ=2. (b) LoG filter at σ=2.

When there is a corner in the image, the output of the LoG filter is maximum. Fig. 3.2 shows the input signals convolved with a Laplacian of $\sigma = 1$.

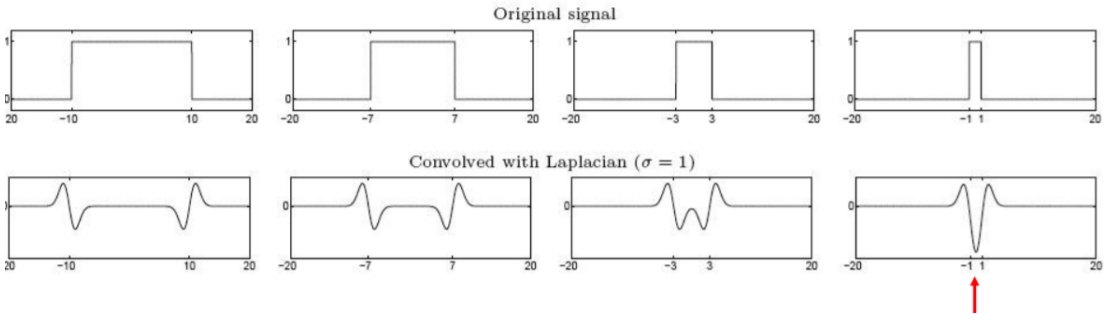


Figure 3.2: The Output will be maximum when there is a corner.

To find the characteristic scale of a blob, an image is convolved by a blob filter with different σs. However, the response of blob filter with image decreases as σ increases. To reduce this effect a *scale normalization* is performed by multiplying the LoG filter by σ^2. The Laplacian achieves maximum response for the binary circle of radius r at $\sigma = 1.414 * r$ [20]. The following two steps summarize the whole process for blob filter:

1. Using different scales (different scales means different σ), convolve image with scale-normalized Laplacian.

2. Extract maxima of squared Laplacian response in scale-space

To detect scale-invariant keypoints , *scale-space* filtering is used [19]. In it, LoG is found for the image with different (σ) values. LoG detects blobs in various sizes due to change of σ. This σ acts as a scaling parameter. Blobs are maxima of the LoG response in *scale-space* and the radius of each blob is approximately $\sqrt{2}\sigma$. Since it is computationally intensive to compute the second order derivatives, LoG is costly. Therefore LoG is approximated by a DoG at different scales [21, 101].

$$g(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\left(\frac{-x^2+y^2}{2\sigma^2}\right)} \tag{3.2}$$

$$DoG = g(x, y, \sigma k) * I(x, y)\breve{}\, g(x, y, \sigma) * I(x, y) \tag{3.3}$$

$g(x, y, \sigma)$ is Gaussian filter, * is convolution operator, k is a scale variable, σ is standard deviation and $I(x, y)$ is image. Blobs are scale-space extrema of DoG [100, 101]. The Fig. 3.3 shows image blob detection using DoG with different σs, whereas the Fig. 3.4 shows the image blobs detection using DoG on a copy-move tampered image.

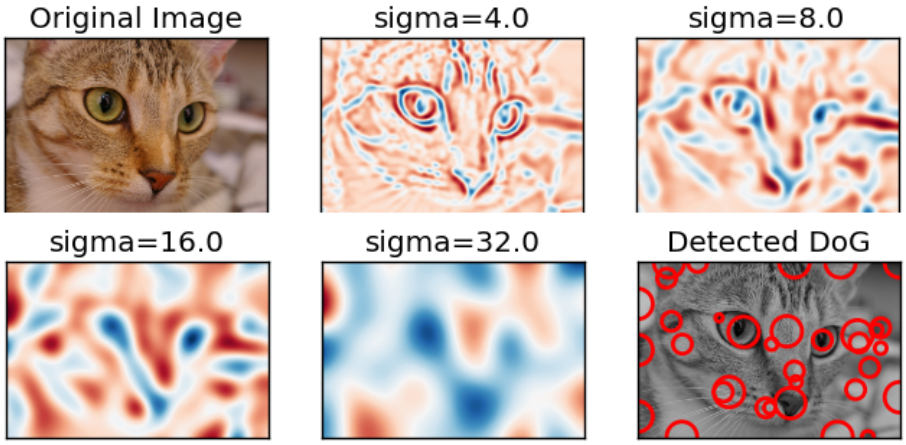


Figure 3.3: DoG blob detection , k = 1.6

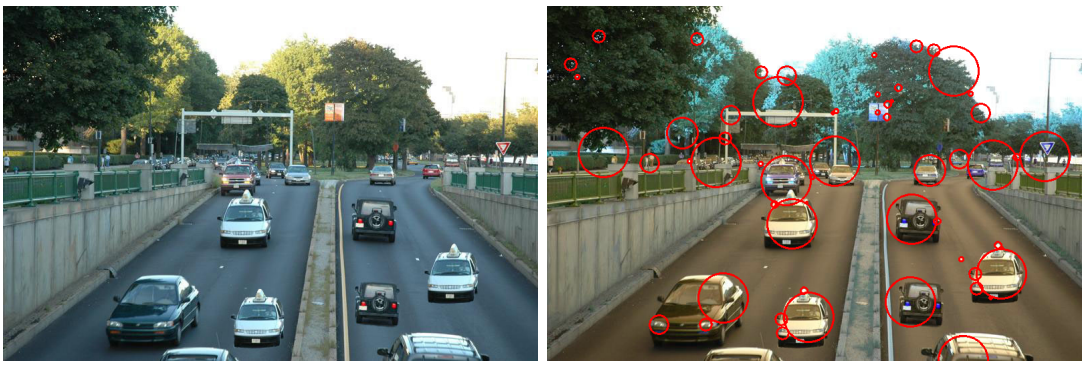


Figure 3.4: (a) Forged image. (b) Red circles are blobs detected using DoG.

3.2.1 Advantages of image blobs over image blocks in CMFD

If we split an image $I(x, y)$ of size $M \times N$ in overlapping fixed-size blocks β with n as slide step size, the number of blocks N_b is given by [69, 103]:

$$N_b = (M - \beta + n)(N - \beta + n) \tag{3.4}$$

N_b increases as image size increases. It is difficult to find the appropriate size of the block. The computational cost is high when using the small blocks. Small blocks don't give robust features. Large blocks cannot be used to detect the presence of small tampered areas. Uniform areas (e.g., background) can be detected as

duplicates. Whereas, blobs are scale-invariant because they are found in scale-space, small forged regions are detected in small blobs and large forged regions are detected in large blobs, blobs separate foreground regions and background, and uniform areas (e.g., background) cannot be detected as duplicates. Given a σ of a Gaussian kernel which detected a blob, the radius of that blob is approximately $\sqrt{2}\sigma$ 100. Fig. 3.5 highlights the difference between image blocks and image blobs.

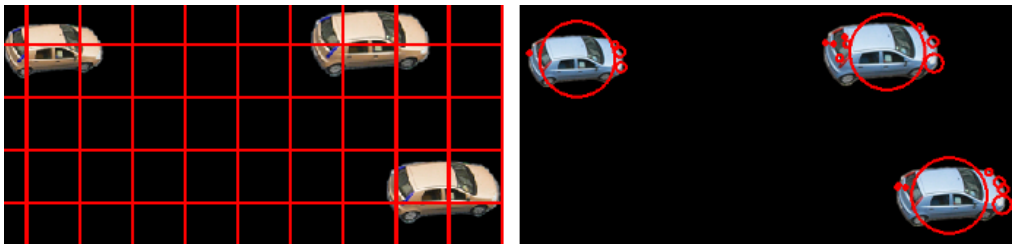


Figure 3.5: (a)Image blocks. (b)Image blobs

3.2.2 Advantages of image blobs over image segments in CMFD

Segmentation splits a digital image into multiple regions called super-pixels (sets of pixels) or segments [12]. To group pixels into meaningful atomic regions, superpixel algorithms are used [31]. Some CMFD techniques use image segmentation [99]. However, blobs exhibit advantages over image segments for CMFD. Blobs separate foreground and background areas (see Fig. 3.6). Thus, blobs enable to discard or reduce the false positives from similar background areas.

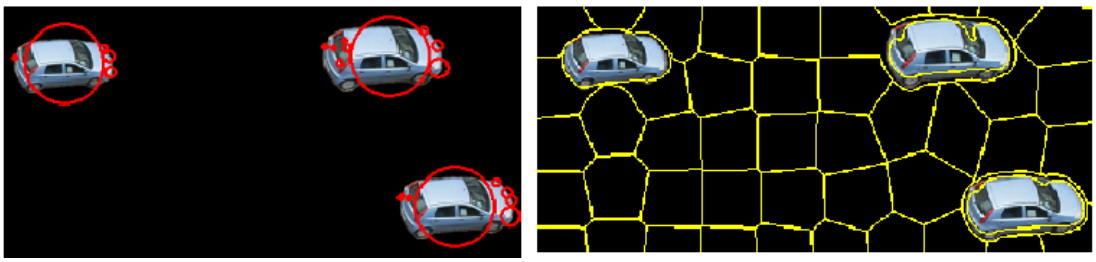


Figure 3.6: (a)Image blobs. (b)Image segments

3.2.3 The ideal blob detector for CMFD

The ideal blob detector should detect the authentic region and its duplicate in different blobs. Also, the blob detector should attempt to enclose each CMF region (authentic and its duplicate) within a single blob. To determine the ideal blob detector, we evaluated the DoG blob detector and LoG blob detector on the standard dataset MICC-F8multi which includes forged images with multi copy-move regions. We extracted forged patches as the ground-truths and compared the detected blobs to the ground-truths using IoU [40]. Fig. 3.7 shows the area of overlap between the large blob detected and the ground-truth. IoU is used to evaluate the performance. Specifically, we want to measure the overlap area between the detected bounding box (green or blue) against the ground-truth (red).

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \tag{3.5}$$

IoU is Intersection over Union.

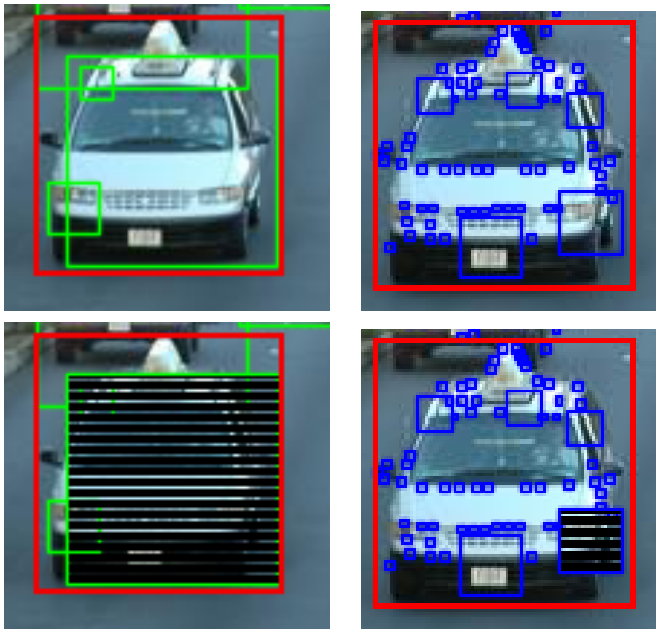


Figure 3.7: Red rectangles are ground-truth, green rectangle are DoG, blue rectangles are LoG, and black lines show intersection area.

The score for each blob detector is shown in Fig. 3.8. The blob detector with the high score is considered as the ideal and is selected to be used with the CMFD techniques.

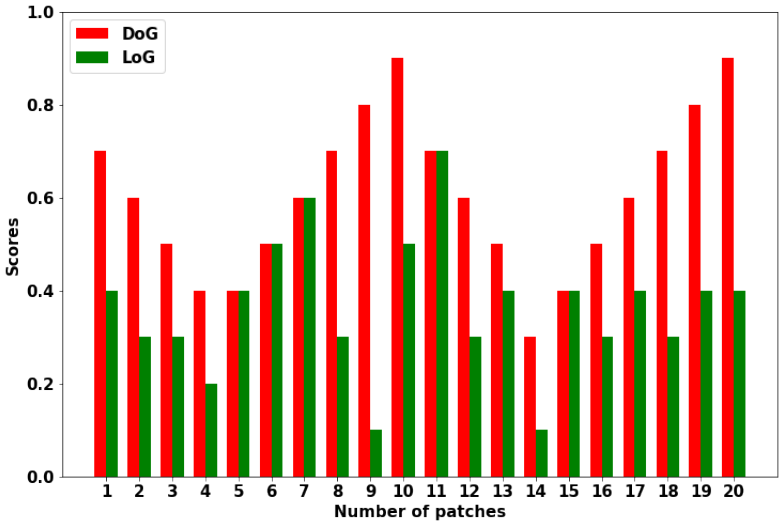


Figure 3.8: IoU Scores from 20 patches extracted from dataset MICC-F8multi. DoG has overall score of 0.6. LoG has overall score of 0.3.

3.3 Enhanced blob localization using Sobel edge detection

An image is pre-processed by edge detector before blob detection. Edge detection (e.g., Sobel) generates a 2D map of the gradient at each point with the regions of high gradient (foreground) visible as white lines [51, 126]. Thus, we used a Sobel image as input of the blob detector to enhance blob localization on foreground regions. Also, the CMF regions (original region and its duplicate) are detected in different blobs because edge detection causes objects in the image to become distinct regions with edges separating them (see Fig. 3.9).

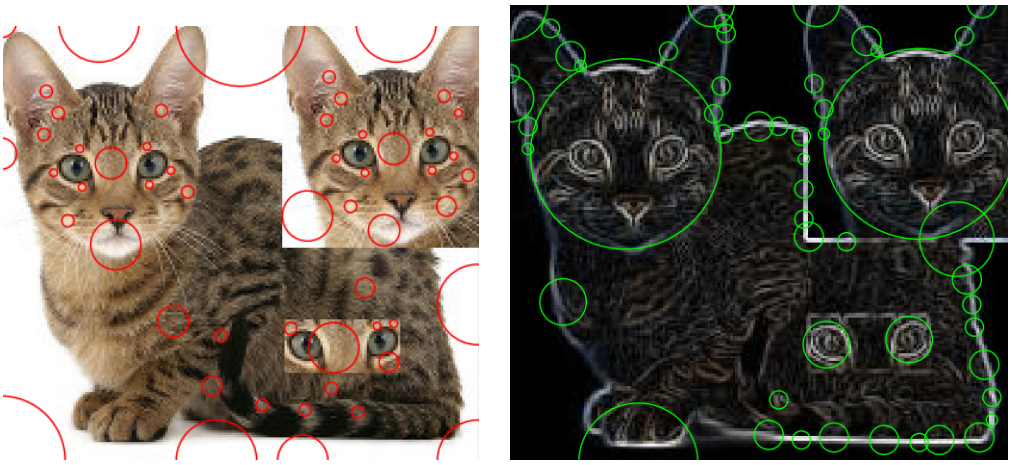


Figure 3.9: (a) Blob detection. (b) Edge + blob detection.

From Fig. 3.9(a): no blob localization (e.g., eyes, heads are not detected inside the blobs). We cannot ensure that the CMF regions (original region and its duplicate) will be detected in different blobs. From Fig. 3.9(b): blob localization (e.g., eyes, heads are detected inside the blobs). We are sure that the CMF regions (original region and its duplicate) will be detected in different blobs because of edges separating them.

3.4 ORB feature extraction

ORB interest points detection and description [48] is used at this stage. ORB have been discussed in Sect. 2.2.2

3.5 Feature matching

To find similar ORB features for CMFD, we match features that reside in different blobs because the copy-move regions are detected in different blobs. To find features that reside within the same blob, we extract 2D spatial coordinates for each blob (x_b, y_b, r) and 2D spatial coordinates for each ORB keypoint (x_k, y_k). The distance (D) from a keypoint location to the center of a blob is given by:

$$D = \sqrt{(x_k - x_b)^2 + (y_k - y_b)^2} \tag{3.6}$$

The keypoint is inside the blob if $D \leq r$. Otherwise the keypoint is outside. $\mathbf{r}$ is the radius of the blob.

HD (D_h) metric is used to determine the similarity between features (ORB descriptors). D_h between i^{th} and j^{th} descriptors of same length is the number of positions in which the corresponding symbols are different. D_h can be computed efficiently using a bitwise XOR operation followed by a bit count [50]. Let v_k be the k^{th} element of the i^{th} descriptor.

$$D_h(i,j) = \sum_{k \leq 256} XOR(v_k(i), v_k(j)), \tag{3.7}$$

$$v_k(i) = v_k(j) \Rightarrow 0 \tag{3.8}$$

$$v_k(i) \neq v_k(j) \Rightarrow 1 \tag{3.9}$$

Then D_h is normalized such that $D_h \in (0,1)$. A match is found between the keypoint pair if D_h is less than a predefined threshold T.

$$D_h \leq T \; where \; T \in (0,1) \tag{3.10}$$

Fig. 3.10(a) shows image blobs with ORB features, whereas Fig. 3.10(b) shows feature matching between different blobs. The straight lines show the corresponding ORB keypoint pairs. The algorithm 1 shows the pseudocode of the present method.

3.6 Algorithm

Algorithm 1: CMFD based on DoG and ORB feature

Result: Image with CMFD results
I : Input Image;
Gi : Compute grayscale of I ;
Si : Apply sobel edge detector to Gi;
$Di_{(x_b,y_b,r)}$: Extract DoG feature and spatial coordinates from Si;
$Oi_{(x_k,y_k)}$: Extract ORB feature and spatial coordinates from Gi;
$List_1 = empty$;
for $Blob\ in\ Di_{(x_b,y_b,r)}$ **do**
    $List_2 = empty$;
    for $Keypoint\ in\ Oi_{(x_k,y_k)}$ **do**
        if $\sqrt{(x_k - x_b)^2 + (y_k - y_b)^2} \leq r$ **then**
            $list_2$.append($Keypoint$) ;
        else
            continue;
    if *list2 != empty* **then**
        $List_1$.append($List2$);
    else
        continue;
for $i\ in\ range(0, len(List_1) - 1)$ **do**
    for $desc_1\ in\ List_1[i]$ **do**
        for $j\ in\ range(i + 1, len(List_1))$ **do**
            for $desc_2\ in\ List_1[j]$ **do**
                D_H = distance.hamming($desc_1, desc_2$);
                if $D_H \leq T$ **then**
                    print 'Match found';
                    drawline(keypoint($desc_1$),keypoint($desc_2$));
                else
                    continue;

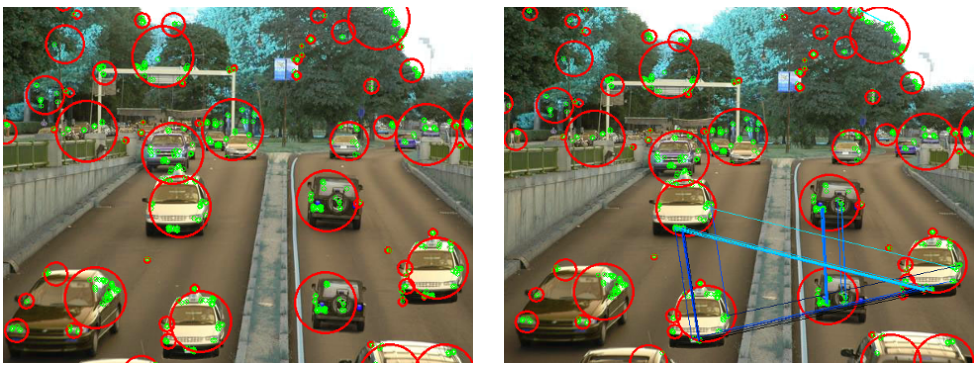


Figure 3.10: (a) Blobs and ORB Feature detection. (b) Feature matching.

3.7 Experimental Results

3.7.1 Evaluation metrics

To assess the performance of CMFD techniques, the following metrics [55] are used: Forged images correctly detected are t_p (True positives). Images wrongly detected as forged are termed as f_p (False positives). Tampered images that are undetected and marked as not forged are termed as f_n (False negatives), and untampered images correctly detected are t_n (True negatives). tp_r is true positive rate and fp_r is false positive rate. Precision (p_r) indicates the probability that a detected forgery is truly a forgery, while Recall (r_c) indicates the probability that a forged image is detected as being forged. f_1 score is a measure which combines p_r and r_c in a single value and the accuracy termed as *acc*.

$$p_r = \frac{tp}{t_p + f_p}, \quad r_c = \frac{t_p}{t_p + f_n}, \quad f_1 = 2\frac{p_r r_c}{p_r + r_c} \tag{3.11}$$

$$tp_r = \frac{\#\, t_p}{\#\ forged\ images}, fp_r = \frac{\#f_p}{\#\ original\ images} \tag{3.12}$$

$$acc = \frac{tp + tn}{t_p + t_n + f_p + f_n} \tag{3.13}$$

3.7.2 Experimental platform and operating point settings

Experimentation platforms consist of a desktop (Intel Core i5 processor with 8 GB RAM), UBUNTU OS 16.04 LTS, Python 2.7.12 and OpenCV 3.0. Edge detector used is Sobel whereas blob detector is DoG.

The running time analysis results are reported in Tab. 3.4. The experiment to set the operating point value (see Equ. 3.10) is performed on 400 images from CoMoFoD dataset and results are reported in Tab. 3.1. The performance comparison of the present method against other related methods [39] are reported Table. 3.6.

Table 3.1: Experimental results on 400 images (100 originals and 300 forged) from dataset CoMoFoD [132] and T values between 0.1 and 0.9 to set the proper operating point.

Thleshold (T)	t_p	f_p	f_n	p_r (%)	r_c (%)	f_1 (%)
0.1	97	1	203	98.97	32.33	48.73
0.2	126	2	174	98.43	42.00	58.87
0.3	185	7	115	96.35	61.66	75.19
0.4	253	8	47	96.93	84.33	90.19
0.5	274	10	26	96.47	91.33	**93.82**
0.6	278	23	22	92.35	92.66	92.50
0.7	283	52	17	84.47	94.33	89.12
0.8	288	89	12	75.06	96.00	84.24
0.9	289	100	11	74.29	96.33	83.88

The results from Tab. 3.1 show that when $T = 0.5$ the optimal f_1 score is 93.82%. Thus, $T = 0.5$ is the operating point.

3.7.3 Robustness tests

The CMFD results for robustness tests using $T = 0.5$ are reported in Tab. 3.5.
Test 1 includes simple copy-move regions (see Fig. 3.11) with geometric transformation parameters reported in Tab. 3.2. 100 images (50 originals and 50 forged) from the dataset MICC-F220 [36] are used.
Test 2 includes multi copy-move regions (see Fig. 3.11) on 24 images of size 3039×2014 from dataset $C3\_$ nikon [55]. 12 tampered images and 12 originals. Other 8 forged images are from dataset MICC-F8multi [36].
Test 3 includes post-processing operations like noise addition and contrast adjustment. 400 images (100 originals and 300 forged) of size 512×512 from CoMoFoD dataset [39] are used. Post-processing parameters are reported in Tab. 3.3.

Table 3.2: Rotation θ in degrees. Scaling factors S_x, S_y. Images from the dataset MICC-F220

Attack	θ	S_x	S_y	Attack	θ	S_x	S_y
A	0	1	1	F	0	1.2	1.2
B	10	1	1	G	0	1.3	1.3
C	20	1	1	H	0	1.4	1.2
D	30	1	1	I	10	1.2	1.2
E	40	1	1	J	20	1.4	1.2

Table 3.3: Parameters for post-processing operations. Images from CoMoFoD dataset

Methods	Parameters
Jpeg compression	factor= [20, 30, 40, 50, 60, 70, 80, 90, 100]
Noise addition	$\mu = 0$, σ^2 = [0.009, 0.005, 0.0005]
Blurring	averaging filter = [3x3, 5x5, 7x7]
Brightness change	(lower bound, upper bound) = [(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]
Contrast adjustment	(lower bound, upper bound) = [(0.01, 0.95), (0.01, 0.9), (0.01, 0.8)]

Table 3.4: Processing time in seconds (average per image) per number of ORB features extracted from image.

Number of ORB features	100	500	1000
CoMoFoD : Time [s]	0.8	2	5.5
MICC-F220, MICC-F8, $C3$_nikon : Time [s]	1.41	3.45	9.20

Table 3.5: Detection results for robustness tests against geometric transformations, multicopies and post-processing operations.

Robustness tests	t_p	f_p	f_n	p_r (%)	r_c (%)	f_1 (%)
Test 1	48	1	2	97.95	96.00	97.46
Test 2	14	1	6	93.33	70.00	80.00
Test 3	274	10	26	96.47	91.33	93.82

3.7.4 Comparative results

Table 3.6: Comparative results between the present method against other related methods [39] on 400 images from CoMoFoD dataset.

Feature extraction methods	p_r (%)	r_c (%)	f_1 (%)
DCT[72]	78.69	100	88.07
PCA [115]	84.21	100	93.20
SURF [124]	91.94	89.58	90.53
ACC[39]	95.65	91.67	93.62
Proposed DoG & ORB	**96.47**	**91.33**	**93.82**

3.7.5 Sample input and output

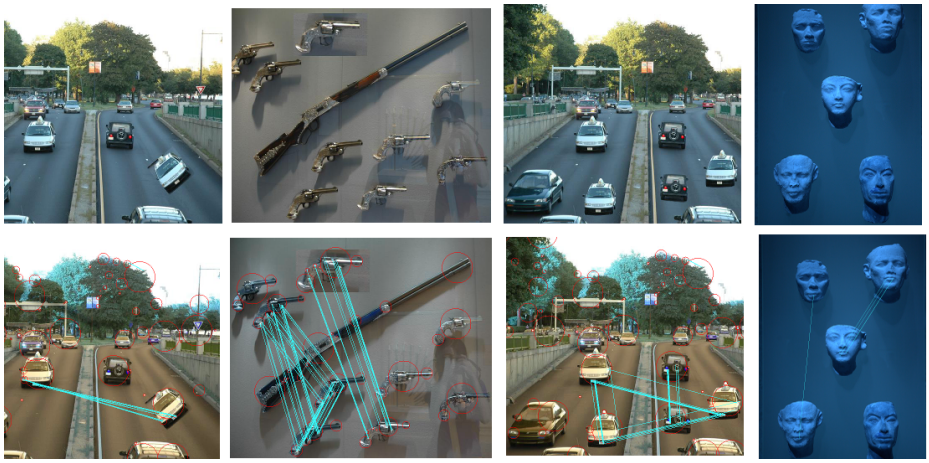


Figure 3.11: First row: tampered images. Second row: CMFD results.

The CMFD results from Tab. 3.5 show that the present method is robust when the tampered regions include post-processing operations. Also, Tab.3.6 indicates that the present technique performs better compared to other related CMFD techniques found in [39].

3.7.6 Conclusion

A method for CMFD using DoG blob detector and ORB features is presented. The present technique uses DoG to detect scale-invariant regions called image blobs. To enhance blob localization on foreground regions, edges are detected before blobs. Then ORB keypoints that reside in different blobs are found. Finally, keypoints from different blobs are matched to find the CMF regions. Various experiments show that the present method is effective for forgeries that include post-processing operations.

3.8 Where image blobs fail in CMFD?

There are two potential weaknesses in the use of image blobs in CMFD [109]. If the authentic and tampered regions overlap in the copy-move forgery, then their corresponding image blobs overlap too. However, such cases are few in the available standard datasets (see Fig. 3.12). In such images, if the tampered area is large enough to contain more than one blob, then they are detected as forgeries because not all blobs overlap. Second is that if the background is detailed, it can have blobs too which result in false positives. Again, we found that in the datasets used there was not a single failure of the present technique due to this effect. The datasets did contain images with rocks, grass, trees, etc, in the background but the method appears empirically robust.

Figure 3.12: (a) Overlapping CMF areas. (b) Blob detection on overlapping CMF areas.

3.9 Summary

This chapter has discussed the potential of using image blobs in CMFD. Image blob detection and the ideal blob detector for CMFD have been discussed. The advantages of blobs over blocks and segments in CMFD are highlighted. A CMFD technique that uses DoG blob detector and ORB features is presented to tackle the limitations of existing CMFD techniques. Finally, potential weaknesses in the use of image blobs in CMFD are discussed.

Chapter 4

Copy-Move Forgery Detection using Image Blobs and BRISK Feature

4.1 Introduction

In this chapter, we show experimentally that CMFD based on image blobs with BRISK feature improves the performance of previously studied ORB features (see chapter 3) in CMFD. Sect. 4.2.6 shows experimentally that the number of keypoints to match is reduced by almost 50% when using image blobs in CMFD. Sect.4.2.7 discusses the reduction of false matches without requiring a filter algorithm.

4.2 CMFD using image blobs and BRISK feature

There are five main steps in the present technique: a pre-processing stage (Sect. 4.2.1), blob detection (Sect. 4.2.2), BRISK feature extraction (Sect. 4.2.3), determine BRISK keypoints located within the same blob (Sect. 4.2.4), and BRISK feature matching between different blobs (Sect. 4.2.5). Fig. 4.1 illustrates the diagram of the proposed technique. The pseudocode for the proposed method is shown in Algorithm 2.

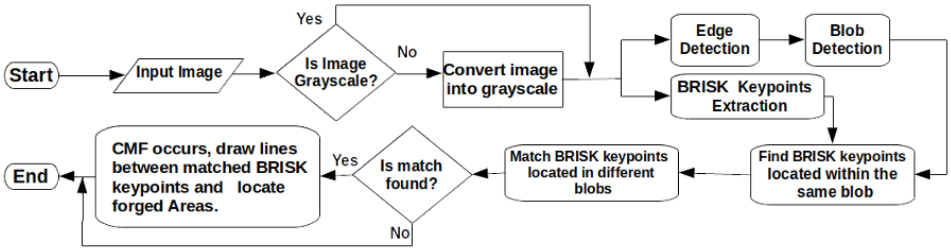


Figure 4.1: CMFD using image blobs and BRISK

4.2.1 Pre-processing

This stage involves the edge detection. The edge detection has been discussed in Sect. 3.3. The output of the edge detector is used as the input to a blob detector for two main reasons:

1. To enhance the blob localization on foreground regions.

2. To locate the CMF regions (authentic region and its duplicate) in different blobs. Edge detection causes CMF regions to become distinct regions with edges separating them.

4.2.2 Blob Detection

The blob detector used in this technique is DoG(Difference of Gaussian) (see Sect. 3.2).

4.2.3 BRISK Feature Extraction

BRISK technique is used for keypoint detection and binary description in image [95]. BRISK uses a scale-space to detect keypoints in octave layers of image pyramid. Via a quadratic function fitting, the location and the scale of each keypoint are obtained.

BRISK descriptor is computed in two steps: first step estimates orientation of keypoints and assists in creating a rotation-invariant descriptor. Second stage consists of intensity comparisons to result in a descriptor that efficiently and effectively captures local region properties.

BRISK descriptor uses concentric circles as a sampling pattern to define N locations. The intensity of each point $\mathbf{p_i}$ in the sampling pattern is smoothed with a Gaussian filter to avoid aliasing effects. The N sample points are gathered into pairs $(\mathbf{p_i}, \mathbf{p_j})$ and grouped into short pairs and long pairs. Short pairs if the distance between $(\mathbf{p_i}, \mathbf{p_j}) < T_{max}$ and long pairs if the distance is greater than T_{min}. To estimate the rotation, long pairs are used. Short-pairs are used to build the descriptor after rotation correction. BRISK descriptor local gradients are computed by:

$$\nabla(\mathbf{p_i}, \mathbf{p_j}) = (\mathbf{p_j} - \mathbf{p_i})\frac{I(\mathbf{p_j}, \sigma j) - I(\mathbf{p_i}, \sigma i)}{||\mathbf{p_j} - \mathbf{p_i}||^2}) \tag{4.1}$$

$\nabla(\mathbf{p_i}, \mathbf{p_j})$ is the local gradient between the sampling pair $(\mathbf{p_i}, \mathbf{p_j})$ and $I(x, \sigma)$ is the smoothed intensity at x using σ. Rotation angle θ is estimated from the average gradient in x and y directions. To obtain rotation-invariant descriptor short pairs are rotated by $-\theta$. Then, to build the descriptor, BRISK uses the set of short pairs $(\mathbf{p_j}, \sigma_j)$, rotates the pairs by the orientation $-\theta$ to get (p_j^θ, σ_j) and makes comparisons of smothed intensity (I) such that each bit $\mathbf{b}$ corresponds to:

$$\mathbf{b} = \begin{cases} 1 & : \mathrm{I}(p_j^\theta, \sigma_j) > \mathrm{I}(p_i^\theta, \sigma_i) \\ 0 & : \text{Otherwise} \end{cases} \tag{4.2}$$

Finally, each keypoint is described by binary descriptor (binary string).

4.2.4 Find BRISK keypoints located within the same blob

To find BRISK keypoints that reside within the same blob (see Fig. 4.3(a)), 2D spatial coordinates for each blob (x_b, y_b, σ) and 2D spatial coordinates for each BRISK keypoint (x_k, y_k) are extracted. (x_b, y_b) is the center of the blob and σ is the standard deviation of the Gaussian kernel which detected the blob. $\mathbf{r}$ is the radius of each blob. $\mathbf{r}$ is approximately $\sqrt{2}\sigma$ [100]. The distance $\mathbf{D}$ from the center of the blob to the keypoint is given by :

$$\mathbf{D} = \sqrt{(x_k - x_b)^2 + (y_k - y_b)^2} \tag{4.3}$$

A keypoint is located within the blob if $\mathbf{D} \leq \mathbf{r}$, and outside the blob if $\mathbf{D} > \mathbf{r}$ [109].

4.2.5 BRISK feature matching

The feature matching process is done to find similar keypoints (see Fig. 4.3(b)). Each BRISK keypoint is described by a binary descriptor $\beta D(i)$ of size Z, the similarity is the Hamming distance (H_d) between two descriptors $\beta D(i)$ and $\beta D(j)$ which is the number of different bits 146.

$$H_d = \sum_{k=1}^{Z} XOR(\beta D_k(i), \beta D_k(j) \tag{4.4}$$

Where $XOR(i, j) = 0$ for $i = j$ and $XOR(i, j) = 1$ for $i \neq j$. Then the 2NN matches and ratio criterion 101 is used to find correct keypoint matches. That is, for each keypoint in blob b_i, find the two nearest neighbors in blob b_j whose distances are d_1 and d_2. The nearest neighbor is defined as the keypoint with minimum Hamming distance H_d. Given a predefined threshold T such that $T \in (0, 1)$, a match is confirmed if

$$d_1/d_2 < T \tag{4.5}$$

Table 4.2 reports the settings for the threshold T for Nearest Neighbor Matching Ratio.

The CMFD methods proposed by Amerini et al.(2011)[36] and Ojeniyi et al.(2018)[110] require a minimum of three matching pairs between different clusters to consider a CMF. Whereas, the present technique requires a minimum of two matching pairs. This is found empirically by experiments on various datasets.

4.2.6 Reducing the number of keypoints to match.

If a copy-move forgery is detected, the CMF regions (authentic region and its duplicate) are located in different blobs [109]. Therefore, BRISK keypoints that reside within the same blob are not matched for CMFD. Tab. 4.1 shows experimentally that the number of BRISK keypoints to match are reduced when image blobs are used. The experiment is performed on 8 images from the dataset MICC-F8multi

[36]. These images are tampered with multi copy-move regions. When blobs are not used Kp_o is the number of keypoints and μt_o is the number of matches. When blobs are used Kp_b is the number of keypoints and and μt_b is the number of matches. $dec(\%)$ shows the number of matches decreased for each image in %. DoG maxσ =40 and BRISK samples radius = 27. The number of matches μt is given by:

$$\mu t = \frac{n!}{(n-k)!k!} \tag{4.6}$$

n is the number of BRISK keypoints for each image and k=2.

Table 4.1: Reduction of the number of BRISK keypoints to match.

$\#Blobs$	$\#Kp_o$	$\#Kp_b$	μt_b	μt_o	$dec(\%)$
281	6438	4256	9054640	20720703	56.3
65	1754	1498	1121253	1537381	27
115	7002	3551	6303025	24510501	74.28
106	851	720	258840	361675	28.43
239	5980	4299	9238551	17877210	48.32
162	6664	4304	9260056	22261116	58.4
408	4435	3377	5700376	9882395	42.31
325	8297	5566	15487398	34415956	54.99
-	-	-	-	-	**49**%

Table 4.1 shows that the present method matches half of all keypoints detected. Recent techniques [80, 87] match all of the keypoints detected in the image.

4.2.7 Tackling the use of filtering techniques

The main source of the false positives in many earlier methods are matches between spatially close areas and similar intensities between neighboring pixels. Different filtering methods like RANSAC have been proposed for reducing the false positives [110]. The use of filtering method increases the computational costs.

However, when using the image blobs in a CMFD technique, the filtering algorithm is unneeded since the neighboring pixels with similar intensities are detected within the same blob, and the spatially close areas that look similar at different scales are also detected within the same blob. Since CMF regions (authentic region and its duplicate) are detected in different blobs, it is unnecessary to match

the keypoints located in the same blob for CMFD [109]. Blobs also separate foreground and background areas which reduces false positives occurring due to similar background areas. Algorithm 2 shows the pseudocode of the present method.

4.3 Algorithm

Algorithm 2: CMFD using image blobs and BRISK feature

```
Result: Image with CMFD results
I : Input Image;
Gi : Compute grayscale of I ;
Si : Apply sobel edge detector to Gi;
Blobs_(x_b,y_b,r) : Extract Blobs feature and spatial coordinates from Si;
BRISK_(x_k,y_k) : Extract BRISK feature and spatial coordinates from Gi;
for Blob in Blobs do
    sameBlob = empty;
    for Keypoint in BRISK do
        if sqrt((x_k − x_b)^2 + (y_k − y_b)^2) ≤ r then
            sameBlob.append(Keypoint) ;
        else
            continue;
    if sameBlob != empty then
        Blobs = Blobs − sameBlob;
        nn_matches = matcher.knnMatch(Blobs, sameBlob);
        nn_match_ratio = 0.43;
        for n1, n2 in nn_matches do
            L = 0;
        if n1.distance < nn_match_ratio ∗ n2 then
            L+ = 1 ;
        else
            if L > 1 then
                print('Match found');
            else
                continue;
    else
        continue;
```

Figure 4.2: (a) Red circles are detected blobs and green circles are BRISK keypoints inside different blobs (b) Green lines indicate BRISK keypoints matching pairs from different blobs.

4.4 Experiments and Results

This section describes the datasets used for experiments. The evaluation metrics, experimental platforms and running time analysis are discussed. Then the present method is compared with existing CMFD techniques. The datasets used for experiments are MICC-F8multi[36], MICC-f220 [36] with parameters reported in Table 3.2, and images from CoMoFoD [132] with parameters reported in Table 3.3.

4.4.1 Evaluation metrics

The metrics used to assess the performance of CMFD methods have been discussed in Sect. 3.7.

4.4.2 Setting the Treshold T for Nearest Neighbor Matching Ratio.

The proposed technique requires setting the threshold (T in Equ. 4.5). It is empirically determined from tests on 220 images (110 original and 110 forged) from MICCF-220 dataset. Tab. 4.2 indicates that when T value starts to increase from 0.2 to 0.43, the f_1 *score* also increases. As T value continues to increase to the

value of 0.44 to 0.6, the f_1 *score* begins to decrease. This indicates that the best performance is obtained when $T = 0.43$.

Table 4.2: Experimental results to determine the threshold value T.

Treshold	MICCF-220					
T	t_p	f_p	f_n	$p_r(\%)$	$r_c(\%)$	$f_1(\%)$
0.2	25	0	85	100.00	22.72	37.02
0.3	86	2	24	97.72	78.18	86.86
0.4	98	6	12	94.23	89.09	91.58
0.43	103	6	7	94.49	93.63	**94.05**
0.44	103	7	7	93.63	93.63	93.63
0.5	109	14	1	88.61	99.09	93.55
0.6	110	20	0	84.61	100.00	91.66

Using the procedure described in Tab. 4.2, the implementations of image blobs and SIFT feature [128] (Blobs+SIFT) gave an f_1 score = 92.43% when $T = 0.4$ and $min\_match >= 3$, whereas image blobs and SURF feature [128] (Blobs+SURF) gave f_1 score = 86.36% when $T = 0.4$ and $min\_match >= 2$.

4.4.3 Experimental platform and running time analysis

The experimentation platforms include a desktop computer with Intel Core i5 processor and 8GB RAM, UBUNTU 18.04.2 LTS OS, Python 3.6.7, OpenCV 3.4.2. Sobel edge detector and DoG blob detector are also used. The running time in seconds is reported in Table 4.3. Images of size 800×600 from the MICC-F220 dataset are used.

Table 4.3: Comparison of running times

Methods	# Blobs	# Keypoints	Running time(seconds)
DoG+ORB [108]	239	4960	4.65
Blobs+SIFT	239	3027	**4.10**
Blobs+SURF	239	3467	4.18
Proposed(Blobs+Brisk)	239	**5980**	6.24

4.4.4 CMFD results for multi copy-move regions

Tab. 4.4 reports CMFD results under multi copy-move regions in the same image on 45 images (15 original and 30 forged) from CoMoFoD and MICCF8-multi datasets. A t_p is considered only if all forged areas within image are detected.

4.4.5 CMFD results for rotation and scaling operations

We used images from dataset MICC-F220 with parameters reported in Tab. 3.2 to evaluate the proposed method on forgeries that include *rotation* on 44 images(11 original and 33 forged) with $angles = \{10°, 20°, 30°, 40°\}$, *scaling* on 44 images(11 original and 33 forged), and *rotation + scaling* on 44 images (11 original and 33 forged. CMFD results are reported in Table 4.4.

4.4.6 CMFD results for post-processing operations.

Tab. 4.4 reports CMFD results for post-processed copy-move forgeries. 70 images used are from CoMoFod dataset with parameters reported in Tab. 3.3. 70 images (20 originals, 20 forged with jpeg compression, and 30 forged with blur, noise addition & contrast adjustment).

Table 4.4: CMFD results on forged images with rotation, scaling, rotation+scaling, multi copy-move regions, and post-processing operations

	Rotation					scaling				
Methods	t_p	f_p	f_n	tp_r	fp_r	t_p	f_p	f_n	tp_r	fp_r
DoG+ORB [108]	34	2	10	0.77	0.18	24	2	9	0.72	0
Blobs+SIFT	40	0	4	0.9	**0**	32	0	1	**0.96**	**0**
Blobs+SURF	21	1	23	0.47	0.09	26	1	7	0.78	0.09
Proposed(Blobs+BRISK)	42	1	2	**0.95**	0.09	30	1	3	0.90	0.09
	Rotation + scaling					Multiple copy-move regions				
DoG+ORB [108]	17	2	5	0.77	0.18	24	1	6	0.8	0.06
Blobs+SIFT	20	0	2	**0.90**	**0**	24	1	6	0.8	0.06
Blobs+SURF	10	1	12	0.45	0.09	23	2	7	0.76	0.13
Proposed(Blobs+Brisk)	19	1	3	0.86	0.09	25	1	5	**0.83**	0.06
	Jpg compression					Blur, Noise add & contrast adj.				
DoG+ORB [108]	18	2	2	0.9	0.1	29	2	1	**0.96**	0.1
Blobs+SIFT	18	2	2	0.9	0.1	29	2	1	**0.96**	0.1
Blobs+SURF	17	3	3	0.85	0.15	28	2	2	0.93	0.1
Proposed(Blobs+BRISK)	19	2	1	**0.95**	0.1	29	2	1	**0.96**	0.1

4.4.7 Comparative Results

Tab. 4.6 reports the performance comparison between the present method with related methods [110] on 220 images (110 authentic and 110 forged) from the dataset MICC-F220.
Tab. 4.7 reports the comparative results between the present method with methods [39, 108] on 400 images (100 originals and 300 forged) from CoMoFoD dataset.

The CMFD results obtained on MICC-F220 are $t_p : 103$, $t_n : 104$, $f_p : 6$, $f_n : 7$.

The CMFD results obtained on CoMoFoD are $t_p : 276$, $t_n : 91$, $f_p : 9$, $f_n : 24$.

Table 4.5 reports the detection results in terms of TPR and FPR on MICC-F220 dataset.

Table 4.5: Detection results in terms of TPR and FPR on MICC-F220 dataset

Methods	$TPR(\%)$	$FPR(\%)$
DoG+ORB [108]	90	9
Blobs+SIFT	90	**5.4**
Blobs+SURF	86	13
SIFT+Clusters [36]	**100**	8
SIFT+LBP [87]	99.1	**5.4**
Proposed(Blobs+BRISK)	93	**5.4**

Table 4.6: Comparative results between the presented method and existing methods [110] on 220 images from MICC-F220

Technique used	p_r (%)	r_c (%)	acc (%)
DyWT+SIFT [76]	88.89	80.00	85.00
PCA+SIFT [88]	93.04	**97.27**	95.00
DWT+SURF [125]	77.17	64.55	72.60
DyWT+SURF [122]	77.06	76.36	76.71
HDS [110]	93.86	**97.27**	**95.45**
DoG+ORB [108]	90.09	82.72	86.24
Proposed Blobs+BRISK	**94.49**	93.63	94.05

Table 4.7: Comparative results between the presented method and existing methods [39, 108] on 400 images from CoMoFoD dataset

Technique used	p_r (%)	r_c (%)	f_1 (%)
ACC [39]	95.65	91.67	93.62
DoG+ORB [108]	96.47	91.33	93.82
Proposed Blobs+BRISK	**96.84**	**92.00**	**94.35**

From Tab. 4.6, the CMFD results show that the present method offers a comparable matching performance to the best-known algorithms [88, 98, 110] that use filter algorithms to remove false matches. However, the present technique offers the advantage of reducing keypoints to match by around 50% and avoids the use of filter algorithm.

4.4.8 Sample input and output

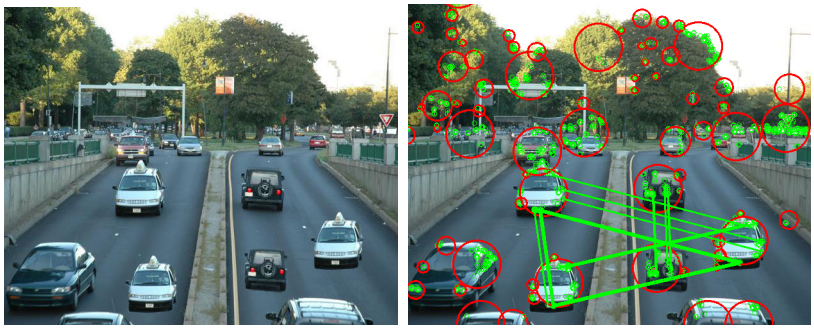


Figure 4.3: (a) Copy-move forged image. (b) Output CMFD

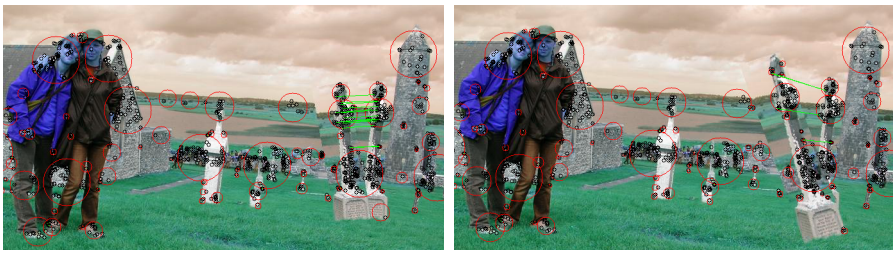


Figure 4.4: CMFD on overlapping cloned and original areas

Figure 4.5: Authenic images detected as tampered

Fig. 4.4 shows CMFD when the original and forged areas overlap, then their corresponding blobs overlap too, and our CMFD method fails since we don't match keypoints located within the same blob. However, if the forged area is large enough to contain more than one blob (see Fig. 4.4), the proposed method detects the CMF because not all blobs overlap. Another observation is shown in Fig. 4.5 where authentic images are detected as tampered, those false matches occur because several small blobs are detected in a single authentic region. In future, to prevent false matches from authentic images, blobs can be combined with object detection or semantic segmentation.

4.4.9 Conclusion

A technique that uses image blobs and BRISK feature to detect the copy-move tampered regions in the image is presented. Image blobs are detected using DoG and keypoints are extracted using BRISK. Keypoints from different blobs are matched to detect similar regions. Since CMF regions (original area and its duplicate) are detected in different blobs, BRISK keypoints that reside in the same blob are not matched for CMFD. This reduces the number of keypoints to match around 50%. The use of filter algorithm is unnecessary because spatially close areas located in the same blob are not matched for CMFD.

4.5 Summary

This chapter has discussed a CMFD technique that uses DoG blob detector and BRISK features. An experiment was performed to show that image blobs enable to

reduce the number of keypoints to match around 50%. The needless and inessential of using filter algorithms when image blobs are used in CMFD is discussed. CMFD results obtained show that image blobs with BRISK features are robust and effective than previously studied CMFD method based on image blobs with ORB features.

Chapter 5

Geometric Transformation Parameters Estimation from Copy-Move Forgery using Image Blobs and Features: AKAZE, BRISK, ORB, SIFT and SURF

5.1 Introduction

In many cases of CMF, post-processing operations including scale, rotation, rotation + scale are applied to the tampered areas to hide the counterfeits in CMFD methods. If the authentic image is not available and we are asked to recover it from a copy-move forged image, then the estimation of geometric transformation parameters between authentic region/area and its duplicate is a key step. However, the potential weakness of using image blobs in CMFD is that many small blobs are detected in a single entire CMF region (see Fig. 5.5(b)). In such scenario, the estimation of geometric transformation parameters between authentic region and its duplicate is impracticable because several blobs are detected in each region. Thus, this chapter discusses a blob post-process step with a 2D affine transformation to enable the blobs-based CMFD to recover the geometric transformation parameters.

In Sect. 5.2.6 a blob post-processing method to enable the blobs-based CMFD to be used with a 2D affine transformation to recover the geometric transformation parameters including rotation, scale, and rotation + scale is described. Tareen et al.(2018)[128] have shown that when analyzing image transformations that include scale and rotation changes, feature detectors behave differently. E.g., SIFT [21], SURF [42] and BRISK [95] are more robust scale-invariant feature detectors over

ORB [64] but they are computationally expensive. Therefore, it is reasonable to enable the algorithm to be flexible and take in different types of features including AKAZE, ORB, BRISK, SURF, and SIFT for geometric transformation estimation.

5.2 Geometric transformation parameters estimation from CMF

The present technique has five main steps: pre-processing, scale-rotation invariant features extraction, image blobs detection, matching scale-rotation invariant features that are located within different blobs, the blobs with matched keypoints are post-processed, and 2D affine transformation computation to estimate the geometric transformations parameters. The present method is wrapper that can use various features in separate implementations for each. These different uses are referred to as: Blobs+SIFT+2D affine, Blobs+BRISK+2D affine, Blobs+AKAZE+2D affine, Blobs+ORB+2D affine, and Blobs+SURF+2D affine. Fig. 5.1 illustrates the pipeline for the proposed method.

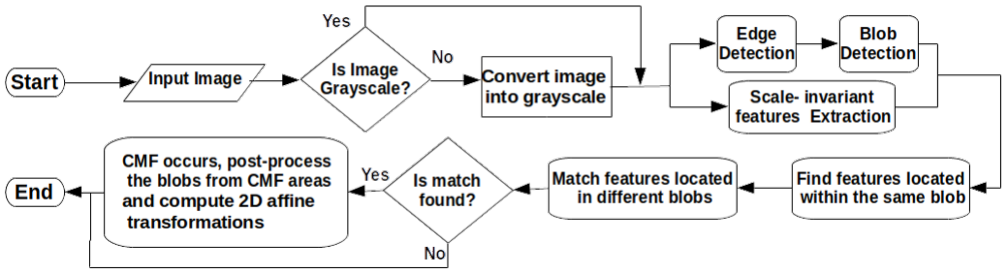


Figure 5.1: Flowchart for the proposed technique for geometric estimation from CMFD

5.2.1 Pre-processing and Edge detection

Color image is converted into grayscale and edge detection is performed. The result is fed as input to the blob detector to enhance the blob localization on foreground regions and ensure that the CMF regions (original region and its duplicate) are detected in different blobs [108].

5.2.2 Scale-Rotation Invariant Feature Extraction

For effective geometric transformation parameters estimation, it is reasonable to enable the algorithm to be flexible and take in different types of features to balance between performance and the computational cost. Most common scale-rotation invariant features including AKAZE [70], ORB [48], BRISK [95], SURF [42] and SIFT [101] are used with the present method.

5.2.2.1 SIFT (Scale-Invariant Feature Transform)

SIFT is a technique which extracts keypoints and computes their descriptors [101]. Here are the steps in SIFT:

- Building a scale space
- Approximation of LoG using DoG
- Extract Keypoints
- Discard edges and low contrast areas
- Orientation assignment
- Generate keypoint descriptor (SIFT feature)

SIFT potential keypoint are local extrema extracted in scale and space using DoG. Regarding different parameters, [13, 101] give some empirical data as optimal values, 4 octaves , 5 scale levels, initial σ is 1.6 and scale variable k is $\sqrt{2}$. Once a keypoint is found, SIFT computes a magnitude and orientation for all pixels around the keypoint. Then, prominent gradient orientation(s) are computed using histogram. For each keypoint, a 16×16 neighbourhood around the keypoint is used as keypoint descriptor.

5.2.2.2 Speeded Up Robust Features (SURF)

SURF is an algorithm that extracts keypoints and computes their descriptors like SIFT [42]. SURF uses box filters to approximate LoG (scale space) (see Fig. 5.2). SURF is fast compared to SIFT because convolution with box filters is computed

using integral images and can be computed simultaneously for various scales. The approximation of both convolution and LoG using box filters is done at a low computational cost using integral images independently of image size.

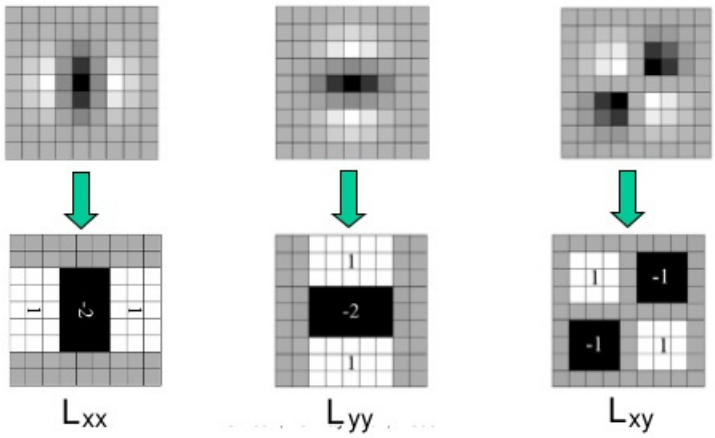


Figure 5.2: Approximate LoG with box filters

Integral image (summed-area table) is an algorithm that generates the sum of values in a rectangular subset of an image region. It accelerates the box type convolution filters computation. SURF uses the determinant of the Hessian matrix for scale and location selection. The Hessian at given pixel is given by:

$$H(f(x,y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$

To adapt to any scale, an image is filtered by a Gaussian kernel. To analyze the scale space, SURF upscales the filter size 9×9, 15×15, 21×21, 27×27, etc, rather than iteratively reducing the image size as in image pyramid. SURF calculates the Haar-wavelet responses in x and y directions to obtain orientation assignment and constructs a feature descriptor (region around the keypoint).

5.2.2.3 KAZE and Accelerated-KAZE (AKAZE)

KAZE is an algorithm that extracts keypoints and computes their descriptors using nonlinear diffusion filtering [33]. The propagation of particles over time within a

substance can be described using diffusion theory. One upstanding example is the heat equation that indicates temperature diffusion in a closed body. KAZE detector uses the Hessian Matrix which is computed at various scale levels to detect keypoints (maxima of the filter response). KAZE feature descriptor consists of a circular neighborhood around the detected keypoint. Nonlinear diffusion filtering can reduce noise while retaining the boundaries of regions but it is computationally expensive to build the nonlinear scale space. Thus, AKAZE was proposed as a speed-up version of KAZE [33, 128]. The standard nonlinear diffusion (SND) is given by:

$$SND = \frac{\partial Lu}{\partial t} = dive(cf(x, y,\ t).\nabla Lu) \tag{5.1}$$

Lu is image luminance, ∇ is gradient operator, cf is conductivity function and $dive$ is divergence.

AKAZE uses Fast Explicit Diffusion (FED) algorithm to compute the nonlinear scale space fast. The AKAZE feature detector is based on the determinant of Hessian Matrix and Scharr filters are used to improve the rotation invariance trait. AKAZE keypoints are maxima of the detector responses in spatial locations. AKAZE uses a Modified Local Difference Binary (MLDB) algorithm to construct the keypoint descriptor [33, 128].

5.2.2.4 Oriented Fast and Rotated Brief (ORB)

ORB consists of a keypoint detector and a binary descriptor [48]. Sect. 3.4 discusses ORB in detail.

5.2.2.5 Binary Robust Invariant Scalable Keypoints (BRISK)

BRISK [95] is composed of a scale-space Keypoint detector and a binary feature descriptor. Sect. 4.2.3 discusses BRISK in detail.

5.2.3 Blob Detection

DoG blob detector is used in the present method. Sect. 3.2 discusses DoG blob detector in detail.

5.2.4 Extract keypoints located within the same blob

To find which keypoints are located within the same blob, a distance (D) between 2D spatial coordinates for each blob and 2D spatial coordinates for each keypoint is calculated (see Fig. 5.3). Sect. 4.2.4 discusses the distance (D) in details.

5.2.5 Feature matching.

Keypoints that are located in different blobs are matched to find similar features (see Fig. 5.5(b)(c)). AKAZE, ORB and BRISK use binary descriptors of size S. Thus, the similarity is given by the Hamming distance H_d (number of different bits) between two feature descriptors $\beta D(i)$ and $\beta D(j)$.

$$H_d = \sum_{k=1}^{S} XOR(\beta D_k(i), \beta D_k(j) \tag{5.2}$$

Where $XOR(i, j) = 0$ for $i = j$ and $XOR(i, j) = 1$ for $i \neq j$.
SIFT and SURF do not use binary descriptor. Every keypoint is described by a vector of size Z. The similarity is the Manhattan distance M_d between two keypoint descriptors $\upsilon(i)$ and $\upsilon(j)$ which is the sum of absolute values.

$$M_d = \sum_{k=1}^{Z} |\upsilon_k(i) - \upsilon_k(j)| \tag{5.3}$$

M_d and H_d are normalized such that $M_d, H_d \in (0, 1)$. The 2NN matches and ratio criterion [101] is used to determine the correct keypoint matches. For each keypoint in blob b_i, find the two nearest neighbors in blob b_j whose distances are d_1 and d_2. The nearest neighbor is defined as the keypoint with minimum Hamming distance H_d for AKAZE, ORB and BRISK. Whereas for SIFT and SURF, the nearest neighbor is defined as the keypoint with minimum Manhattan distance M_d [109]. Given a predefined threshold T such that $T \in (0, 1)$, the point pair (i, j) is a matching pair if the following condition is satisfied:

$$\frac{d_1}{d_2} < T \tag{5.4}$$

The settings for the threshold T for Nearest Neighbor Matching Ratio are discussed in the Sect. 5.3.2. In [109], a minimum of 2 matching pairs between different blobs is required to consider a forgery. For the present method a minimum of 3 matching pairs between different blobs is required to consider a forgery. However, a minimum of 5 matching pairs is required to estimate the geometric transformations parameters. This is found empirically by experiments on the MICC-F220 [36] dataset.

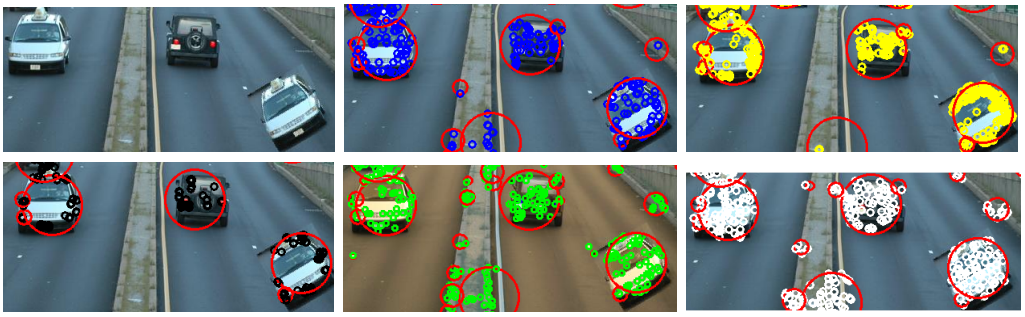


Figure 5.3: Big red rings are detected blobs and small rings are detected features. First row: (a)CMF forged image. (b)Blobs+AKAZE. (c)Blobs+BRISK. Second row : (d)Blobs+ORB. (e)Blobs+SIFT. (f)Blobs+SURF.

5.2.6 Blob post-processing and 2D affine transformations computation

The main limitation of using image blobs in CMFD, is that several small blobs are detected for a single entire CMF region. Then in such cases, we cannot compute the geometric transformation parameters. To tackle this issue, we performed a post-process operation on blobs that enclose the matched keypoints pairs. The blob post-process operation ensures that each entire CMF region (authentic and its duplicate) is contained within a single large blob. If b_i and b_j are the blobs that enclose matched keypoint pairs, we extract the center of each blob (b_ix, b_iy) and (b_jx, b_jy) (see Sect.4.2.4), then the distance d_{ij} between the two blobs is calculated by:

$$d_{ij} = \sqrt{(b_ix - b_jx)^2 + (b_iy - b_jy)^2} \tag{5.5}$$

The distance to the center of the d_{ij} is:

$$M = d_{ij}/2 \tag{5.6}$$

The distance r_i from M to $(b_i x, b_i y)$ is equal to the distance r_j from M to $(b_j x, b_j y)$, i.e, $M = r_i = r_j$. Thus, to obtain new large blobs that don't overlap, we set the radius of each new blob $= M$. We obtain large blobs $(b_i x, b_i y, M)$ and $(b_j x, b_j y, M)$, these two blobs don't overlap and each entire CMF region (authentic region and its duplicate) is contained in one of the two blobs. Image blobs that enclose the matched keypoints pairs before blob post-processing are shown in Fig. 5.4(a) and Fig. 5.5(b). Image blobs which cover each entire CMF regions (authentic and its duplicate) after blob post-processing are shown in Fig. 5.5(c) and Fig. 5.6(b)(c)(d).

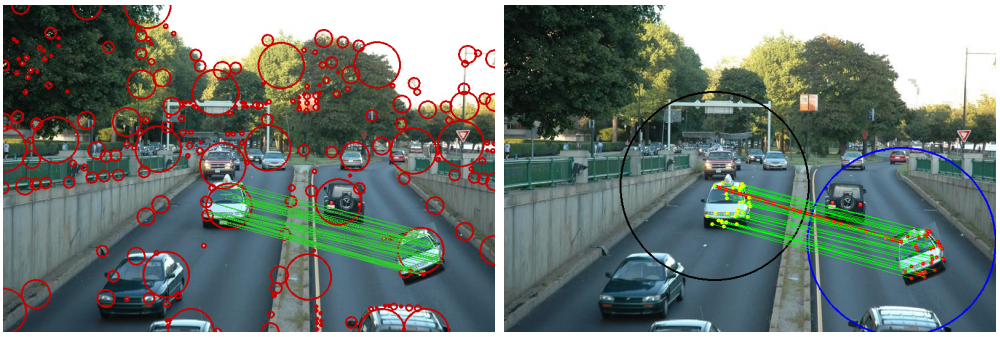


Figure 5.4: (a) Feature matching before blobs post-processing. (b) blobs after post-processing.

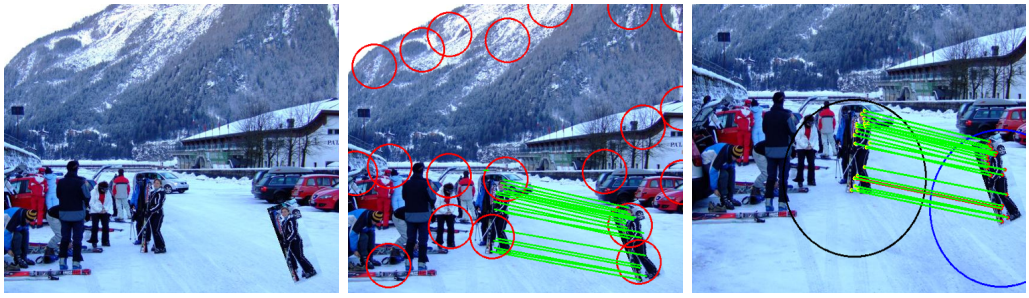


Figure 5.5: Matching keypoints located in different blobs. (a) CMF image. (b)Feature matching before blobs post-processing. (c)Feature matching after blobs post-processing.

Image blobs that don't enclose matched keypoint pairs are not involved in blob post-processing. They are discarded.

Affine transformation is a transformation expressed by a fusion of a linear transformation and a vector addition [2]. Therefore, it can be used to express:

- Rotation (linear transformation)
- Scale (linear transformation)
- Translation (vector addition)

Since an affine transformation can represent a relation between two images [28], we consider to use it to extract the geometric transformation parameters between authentic region and its duplicate. Our technique uses a 2D affine transformation to recover the geometric transformation parameters between CMF regions (authentic and its duplicate). The following 2×3 matrix is the transformation estimated using the corresponding keypoints pairs:

$$\begin{bmatrix} S \times cos(\theta) & -S \times sin(\theta) & t_x \\ S \times sin(\theta) & S \times cos(\theta) & t_y \end{bmatrix}$$

Where S is the scaling factor (S_x, S_y are scale in x, y axes respectively). θ is the rotation angle. t_x and t_y are translations in x and y axes respectively. To find translation, rotation, and scale values we solve

$$\begin{bmatrix} a & b & x \\ c & d & y \end{bmatrix} = \begin{bmatrix} S \times cos(\theta) & -S \times sin(\theta) & t_x \\ S \times sin(\theta) & S \times cos(\theta) & t_y \end{bmatrix}$$

$$Translation : x = t_x, y = t_y \tag{5.7}$$

$$Scale : S_x = \sqrt{a^2 + b^2}, S_y = \sqrt{c^2 + d^2} \tag{5.8}$$

$$Rotation : tan\theta = -\frac{b}{a} = \frac{d}{c} \tag{5.9}$$

Algorithm 3 shows the pseudocode of the present method.

5.2.7 Algorithm

Algorithm 3: Geometric transformation parameters estimation from CMFD

Result: CMFD results with estimated geometric transformations parameters

I : Input Image;
Gi : Compute grayscale of I ;
Si : Apply sobel edge detector to Gi;
$Blobs_{(x_b,y_b,r)}$: Extract Blobs spatial coordinates from Si;
$FEAT_{(x_k,y_k)}$: Extract feature (AKAZE, BRISK, ORB, SIFT and SURF) from Gi;
for *Blob in Blobs* **do**
    $sameBlob = empty$;
    for $Keypoint$ *in* $FEAT$ **do**
        if $\sqrt{(x_k - x_b)^2 + (y_k - y_b)^2} \leq r$ **then**
            $sameBlob$.append($Keypoint$) ;
        else
            continue;
    if $sameBlob$ *!=* $empty$ **then**
        $Blobs = Blobs - sameBlob$;
        $nn\_matches = matcher.knnMatch(Blobs, sameBlob)$;
        $nn\_match\_ratio = T$;
        $L = empty$;
        for $n1, n2$ *in* $nn\_matches$ **do**
            if $n1.distance < nn\_match\_ratio * n2$ **then**
                $L.append(n1)$;
            else
                continue;
        if L *!=* $empty$ **then**
            print('Match found');
            Blobs post-processing;
            Compute 2D affine transformation
        else
            continue;
    else
        continue;

5.3 Experimental Results

Firstly, this section introduces the dataset used and evaluation metrics. Then, the settings for the thresholds T in Equation 5.4 are reported. The detection performance is presented, and the geometric transformation parameters recovery results are reported.

5.3.1 Dataset and evaluation metrics

The images used in the experiments are from the dataset MICC-F220 [36] of size 800×599 pixels and images from the dataset MICC-F2000 [36] of size 2048×1536 pixels. Tab. 5.3 reports the geometric transformations parameters θ for rotation in degrees, and (S_x, S_y) scaling factor in pixels. Sect. 3.7 discusses the metrics which are used to assess the performance of a CMFD method. For geometric transformation parameters, the *absolute error* $|e|$ is the difference between original value of a parameter and its estimated value.

5.3.2 Threshold T settings, detection performance, and comparison

This method requires setting the threshold T in Equ. 5.4. Tests on 220 images (110 authentic and 110 tampered) from MICCF-220 dataset [36] are used to empirically determine the value of T. Table 5.1 shows the threshold set up for Blobs+SIFT+2D affine. As the T value begins to increase from the value of 0.1 to 0.4, the *Acc* (accuracy) also increases. As T value continues to increase to the value of 0.5 to 0.9, the *Acc* begins to decrease. This indicates that the best performance is obtained when $T = 0.5$.

Table 5.1: Experimental results to determine the threshold value T.

Treshold	MICCF-220				
T	t_p	f_p	f_n	t_n	$Acc\%$
0.1	5	0	105	110	52.14
0.2	37	0	73	110	66.61
0.3	80	1	30	109	85.90
0.4	100	7	10	103	92.27
0.5	106	12	4	98	**92.27**
0.6	106	25	4	85	86.81
0.7	110	37	0	73	83.18
0.8	110	85	0	25	61.36
0.9	110	95	0	15	56.81

Using the procedure described in Tab. 5.1, $T = 0.5$ for Blobs+SIFT+2D affine, $T = 0.6$ for Blobs+BRISK+2D affine, $T = 0.7$ for Blobs+AKAZE+2D affine, $T = 0.6$ for Blobs+ORB+2D affine and $T = 0.6$ for Blobs+SURF+2D affine. The accuracy results are reported in Tab. 5.8.

5.3.3 Experimental platform and analysis of running time.

The experiments are performed using a desktop with Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz, 64-bit processor with 8GB RAM. The software environments are Python 3, OpenCV-contrib and Ubuntu 18.04.3 LTS OS. Sobel edge detector and the DoG blob detector are used. Tab. 5.2 reports the running time in seconds on images of size 800×532 pixels belonging to the MICC-F220 dataset.

Table 5.2: Running time analysis on image car from MICC-F220 dataset

Methods	# of blobs	# of keypoints detected	Running time (seconds)
DoG+ORB [108]	239	4960	**4.65**
Blobs+ORB+2D affine	239	4960	5.9
Blobs+BRISK [109]	239	5980	**6.24**
Blobs+BRISK+2D affine	239	5980	6.83
Blobs+AKAZE+2D affine	239	1498	2.48
Blobs+SIFT+2D affine	239	2855	4.6
Blobs+SURF+2D affine	239	3299	4.24

The results indicate that this method does not add a significant computational time to the existing blobs-based CMFD methods.

5.3.4 Geometric transformations parameters estimation

For each attack in Tab. 5.3 the experiment is carried out on 11 images.

Table 5.3: Geometric transformations parameters. Rotation θ in degrees and (S_x, S_y) scale factor in pixels.

Attack	θ	S_x	S_y		Attack	θ	S_x	S_y
A	0	1	1		F	70	1	1
B	10	1	1		G	90	1	1
C	20	1	1		H	0	1.2	1.2
D	30	1	1		I	0	1.3	1.3
E	40	1	1		J	10	1.2	1.2

I. Simple copy-move: the translation parameters are described by the attack A from Tab. 5.3. 11 images are forged by duplicating a random region and pasting it into the same image (see Fig. 5.6(b)). The estimated translation parameters are reported in Tab. 5.5.

II. Rotation transformation: the parameters for rotation are described by the attacks {B,C,D,E,F,G} from Table 5.3. 11 images forged with $\theta = 10°$, 11 images forged with $\theta = 20°$, 11 images forged with $\theta = 30°$, 11 images forged with $\theta = 40°$, 11 images forged with $\theta = 70°$, and 11 images forged with $\theta = 90°$. For $70°, 90°$ images are from MICC-F2000 [36] with the size 2048×1536 pixels (see Fig. 5.6(c-d)). The estimated parameters are reported in Table. 5.4.

III. Scale transformation: the scale transformation parameters are described by the attacks {H, I} from Tab. 5.3. The scale parameters are estimated on 11 images forged with duplicated regions scaled to $(S_x = 1.2, S_y = 1.2)$, and 11 images forged with $(S_x = 1.3, S_y = 1.3)$. The parameter estimation results are reported in Tab. 5.6.

IV. Rotation + scale transformation: the rotation + scale transformation parameters are described by the attack J from Tab. 5.3. 11 images forged with rotation $(\theta = 10°)$ + scale$(Sx = 1.2, Sy = 1.2)$. The estimated parameters are reported in Tab. 5.7.

Table 5.4: Rotation parameters estimation. Images (i) from MICC-F220 dataset. For 70° and 90° images from MICC-F2000 dataset. (-) indicates that not enough matches are found to estimate the parameters. B is blob, and 2Da is 2D affine transforms

Methods	Rotation $\theta = 10°$										
	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11
B+AKAZE+2Da	9.7°	9.4°	10.0°	-	-	10.0°	-	10.0°	-	10.0°	-
B+ORB+2Da	-	7.6°	10.1°	11.3°	-	9.8°	−°	9.4°	9.5°	10.4°	-
B+BRISK+2Da	9.8°	10.3°	10.1°	10.9°	-	9.6°	-	9.8°	10.1°	10.0°	9.7°
B+SURF+2Da	-	9.3°	9.7°	-	9.7°	9.7°	-	4.1°	9.6°	9.7°	-
B+SIFT+2Da	9.6°	10.1°	9.9°	10.0°	-	10.0°	9.9°	10.0°	9.4°	9.9°	-
Rotation $\theta = 20°$											
B+AKAZE+2Da	-	19.7°	19.5°	-	-	20.1°	-	20.0°	-	19.9°	-
B+ORB+2Da	-	20.7°	20.4°	20.8°	-	20.1°	19.0°	20.0°	19.3°	-	-
B+BRISK+2Da	20.2°	18.9°	19.7°	-	-	19.9°	-	19.9°	19.2°	19.7°	-
B+SURF+2Da	-	17.6°	19.4°	-	-	20.0°	-	4.1°	19.9°	19.7°	-
B+SIFT+2Da	19.8°	20.1°	20.0°	20.5°	-	19.9°	20.0°	20.0°	19.7°	19.9°	-
Rotation $\theta = 30°$											
B+AKAZE+2Da	29.7°	30.1°	29.9°	-	-	30.0°	-	29.9°	-	30.0°	-
B+ORB+2Da	-	30.1°	29.9°	30.3°	-	30.2°	-	30.6°	31.1°	30.6°	-
B+BRISK+2Da	30.0°	30.4°	30.5°	29.8°	-	30.1°	27.1°	29.9°	30.2°	29.9°	-
B+SURF+2Da	30.2°	30.0°	-	-	-	-	-	4.1°	-	29.4°	-
B+SIFT+2Da	30.0°	30.0°	30.0°	-	30.0°	29.9°	29.8°	30.1°	29.9°	29.9°	-
Rotation $\theta = 40°$											
B+AKAZE+2Da	39.9°	39.9°	39.9°	-	-	40.1°	-	39.8°	-	40.0°	-
B+ORB+2Da	-	39.8°	39.9°	41.0°	-	39.9°	-	39.7°	39.8°	40.6°	-
B+BRISK+2Da	39.8°	40.2°	40.1°	-	-	40.1°	40.6°	39.9°	39.4°	40.1°	-
B+SURF+2Da	-	38.3°	39.8°	-	-	39.0°	-	4.1°	-	40.4°	38.6°
B+SIFT+2Da	39.8°	39.6°	40.0°	-	40.0°	39.9°	39.8°	39.9°	40.1°	40.0°	39.8°
Rotation $\theta = 70°$											
B+AKAZE+2Da	-	-	69.9°	69.9°	69.9°	70.0°	70.0°	70.0°	70.0°	69.9°	69.9°
B+ORB+2Da	70.2°	-	69.9°	70.5°	68.0°	70.0°	70.1°	70.1°	-	69.1°	69.9°
B+BRISK+2Da	70.9°	70.1°	69.9°	-	69.9°	69.9°	69.9°	69.8°	70.5°	70.1°	70.0°
B+SURF+2Da	70.0°	-	70.0°	-	70.2°	70.2°	69.2°	69.9°	-	69.6°	-
B+SIFT+2Da	-	70.2°	69.9°	-	69.9°	70.0°	69.9°	70.0°	69.8°	70.0°	70.0°
Rotation $\theta = 90°$											
B+AKAZE+2Da	-	-	90.0°	90.0°	90.0°	90.0°	90.0°	89.9°	90.0°	89.9°	90.0°
B+ORB+2Da	89.8°	90.9°	90.0°	90.2°	-	90.1°	89.6°	90.3°	-	89.8°	90.0°
B+BRISK+2Da	-	89.8°	90.0°	88.9°	90.0°	90.0°	89.9°	90.0°	90.2°	90.1°	89.9°
B+SURF+2Da	89.6°	90.0°	89.9°	89.9°	90.0°	90.0°	90.0°	90.0°	90.5°	89.9°	90.1°
B+SIFT+2Da	90.0°	90.0°	90.0°	-	89.9°	89.9°	90.0°	89.9°	90.0°	89.9°	89.9°

Table 5.5: Translation parameters estimation. Images (i) from MICC-F220 dataset. (-) indicates that not enough matches are found to estimate the parameters. B is blob and 2Da is 2D affine transformation

Methods	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11
Translation (tx, t_y)											
B+AKAZE+2Da	366,22	-	229,43	-	44,2	225,1	127,386	561,10	311,80	-	220,143
B+ORB+2Da	-	197,42	231,44	-	-	-	126,388	553,16	311,76	398,39	222,148
B+BRISK+2Da	366,22	181,47	230,42	116,9	43,3	-	127,388	557,12	309,78	400,37	219,149
B+SURF+2Da	367,22	198,39	229,44	-	39,6	225,2	127,386	560,8	39,60	-	231,140
B+SIFT+2Da	366,23	197,40	230,44	115,8	46,1	225,2	127,385	562,7	311,79	396,39	218,142

Table 5.6: Scale parameters estimation. Images (i) from MICC-F220 dataset. (-) indicates that not enough matches are found to estimate the parameters. B is blob and 2Da is 2D affine transformation

Methods	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11
Scale (Sx,Sy)=1.2											
B+AKAZE+2Da	-	-	1.20	-	-	-	1.19	-	1.20	-	1.20
B+ORB+2Da	-	1.20	1.20	-	-	-	1.19	1.17	1.20	1.21	1.17
B+BRISK+2Da	1.20	1.18	1.20	1.22	1.18	1.20	1.20	1.18	1.20	1.21	1.19
B+SURF+2Da	1.20	1.22	1.19	-	-	1.19	1.19	-	1.15	-	1.19
B+SIFT+2Da	1.20	-	1.20	1.21	-	1.20	1.20	1.19	1.20	1.20	1.20
Scale (Sx,Sy)=1.3											
B+AKAZE+2Da	-	-	1.21	-	-	-	1.37	-	1.33	-	1.39
B+ORB+2Da	-	-	-	-	-	-	1.34	1.21	1.29	1.38	1.34
B+BRISK+2Da	1.21	-	1.20	-	-	1.20	1.38	1.22	1.33	1.39	1.37
B+SURF+2Da	1.21	1.24	1.19	-	-	1.19	1.35	-	1.15	-	1.31
B+SIFT+2Da	1.21	-	1.21	1.33	-	1.20	1.37	1.23	1.32	1.38	1.39

Table 5.7: Rotation and scale parameters estimation. Images (i) from MICC-F220 dataset. (-) indicates that not enough matches are found to estimate the parameters. B is blob and 2Da is 2D affine transformation

Methods	Rotation($\theta = 10°$), Scale (Sx,Sy)=1.2										
	i1	i2	i3	i4	i5	i6	i7	i8	i9	i10	i11
B+AKAZE+2Da	-	-	10.0°	-	-	-	9.9°	-	10.2°	-	10.2°
	-	-	1.20	-	-	-	1.20	-	1.21	-	1.21
B+ORB+2Da	-	13.8°	11.3°	-	8.2°	-	9.5°	-	9.0°	9.7°	10.2°
	-	1.21	1.19	-	1.27	-	1.20	-	1.21	1.20	1.19
B+BRISK+2Da	10.4°	7.2°	9.9°	-	10.8°	10.0°	9.6°	-	9.8°	9.7°	9.5°
	1.20	1.17	1.20	-	1.19	1.20	1.20	-	1.21	1.19	1.22
B+SURF+2Da	-	-	9.7°	-	9.9°	-	9.4°	-	4.1°	-	10.0°
	-	-	1.20	-	1.19	-	1.20	-	1.15	-	1.22
B+SIFT+2Da	9.9°	-	9.8°	9.5°	10.7°	9.8°	10.0°	9.6°	9.9°	11.2°	9.8°
	1.20	-	1.20	1.20	1.20	1.20	1.20	1.21	1.21	1.22	1.20

The results from Tab. 5.5, Tab. 5.6, Tab. 5.7 and Tab. 5.4 show how reliable the estimates are (the actual values of parameters and their estimated values). The results indicate that the estimated parameters of the affine transformation are reliable. Tab. 5.9 compare the present method with the method [36], which is the best geometric transformations estimator CMFD found in literature. The details for tp_r and the absolute error $|e|$ (the difference between original value of a parameter and its estimated value) are given below.

Tab. 5.5 shows translation (t_y, t_x) in x, y axes respectively. The actual values of translation parameters are not provided with the dataset MICC-F220 [36] to calculate the $|e|$. However, it is noticeable that the parameters recovered by different methods are apparently identical. **Blobs+SIFT+2Da** gives the better results: $tp_r = 100\%$.

From Tab. 5.6 (scale) results $(Sx, Sy) = 1.2$, **Blobs+BRISK+2Da** gives better results: $tp_r = 100\%$ and the overall $|e| = 0.1$.

For scale $(Sx, Sy) = 1.3$, **Blobs+SIFT+2Da** gives the better results: $tp_r = 81.8\%$ and the overall $|e| = 3.33$.

From Tab. 5.7 (rotation + scale) results, **Blobs+SIFT+2Da** gives better results: $tp_r = 90.9\%$, the overall $|e| = 1.6$ for scale, and overall $|e| = 13.6°$ for rotation.

From Tab. 5.4 (rotation) results, **Blobs+SIFT+2Da** gives better results: $tp_r = 84.4\%$ and the overall $|e| = 450°$.

5.3.5 Comparative results

Table 5.8: CMFD results in terms of accuracy on MICC-F220 dataset

Methods	Accuracy (%)
DoG+ORB [108]	86.24
Blobs+BRISK [109]	**94.09**
Blobs+ORB+2D affine tranforms	84.54
Blobs+AKAZE+2D affine tranforms	80.0
Blobs+SIFT+2D affine tranforms	92.27
Blobs+SURF+2D affine tranforms	81.81
Blobs+BRISK+2D affine tranforms	92.27

Results from Tab. 5.8 indicate that the present method exhibits a comparable matching performance to the existing blobs-based CMFD that cannot recover the geometric transformation parameters. To show how reliable the estimates are, the results from Tab. 5.9 compare the present method with the method [36], which is the best geometric

transformation parameters estimator CMFD found in literature.

Table 5.9: Geometric transformation parameters estimation, B is blob, and 2Da is 2D affine transforms

Methods	Rotation θ in degree				scale in pixels		Rotation + scale	
	10°	20°	30°	40°	1.2	1.3	10°	1.2
Amerini et al.[36]	9.963°	**20.009°**	30.092°	39.932°	1.202	1.304	9.910°	**1.203**
B+AKAZE+2Da	10.006°	20.088°	**29.982°**	39.883°	1.204	1.336	10.299°	1.215
B+ORB+2Da	9.411°	20.041°	30.612°	39.739°	1.209	**1.298**	9.00°	1.215
B+BRISK+2Da	9.811°	19.932°	29.942°	**39.959°**	1.201	1.335	9.832°	1.212
B+SURF+2Da	4.143°	4.143°	4.143°	4.143°	1.157	1.157	4.143°	1.157
B+SIFT+2Da	**10.002°**	20.016°	30.192°	39.910°	**1.200**	1.327	**9.990°**	1.211

Considering the results from Tab. 5.9, the present method gives better estimates on rotation: $\{10°, 30°, 40°\}$ and scale:$\{1.2, 1.3\}$ with their respective $|e|$ are $0.002°, 0.018°, 0.41°, 0,$ and 0.002. The method [36] gives better estimate on $\{20°\}$ with $|e|= 0.009°$. For performance comparison, the present method does not match the keypoints located within the same blob, this reduces the number of keypoints to match about 50% [109] and does not use filters to remove false matches; whereas method [36] matches all keypoints in image before clustering and uses filter technique to eliminate false matches.

5.3.6 Sample ouputs

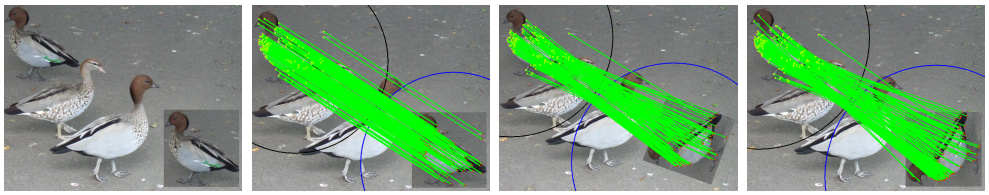


Figure 5.6: (a) CMF image. (b) Simple CMFD. (c)CMFD $\theta = 70°$. (d) CMFD $\theta = 90°$.

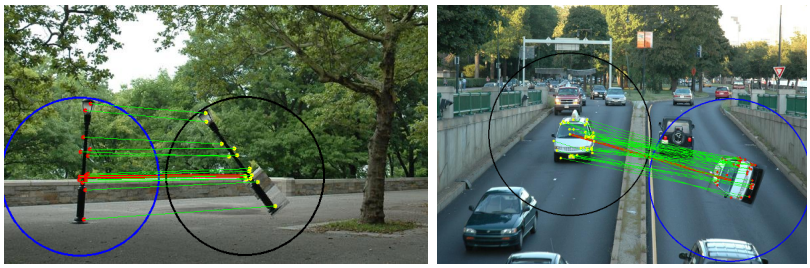


Figure 5.7: (a)40°, estimated = 40.03°. (b)70°, estimated = 69.94°.

5.3.7 Conclusion

A CMFD method that detects a copy-move forgery and estimates the geometric transformation parameters between CMF regions (original region and its duplicate) is presented. Image blobs are detected using DoG and keypoints are detected using various feature detectors (AKAZE, ORB, BRISK, SURF, and SIFT). Keypoints from different blobs are matched to find similar regions. However, the main limitation of using image blobs in blob-based CMFD is the inability to perform the geometric transformation parameters estimation because several small blobs are detected in each CMF region. To tackle the above-mentioned limitation, the present method performs a post-process operation on blobs that enclose the matched keypoints pairs to ensure that each entire CMF region (original and its duplicate) is contained within a single large blob. Blob post-process operation is followed by the computation of a 2D affine transformation bewteen CMF regions to estimate the geometric transformation parameters. The present method shows a very high degree of flexibility because it can easily take in various features including AKAZE, ORB, BRISK, SURF and SIFT to enhance the effectiveness.

5.4 Summary

In this chapter, we have demonstrated that image blobs with various features including AKAZE, ORB, BRISK, SURF and SIFT can be used to estimate the geometric transformation parameters (scale, rotation,translation and scale + rotation) from the copy-move forgery. A blob post-process operation and a 2D affine transformation are used to enable blob-based CMFD to recover the geometric transformation parameters between original region and its duplicate. The experimental results show the potential of using image blobs to recover the geometric transformation parameters in CMFD. The results also show that the estimated parameters of the affine transformation are reliable.

Chapter 6

Image Splicing Detection using Illumination Component and LBP

6.1 Introduction

Image splicing consists of two or more different images spliced together. Image splicing introduces sharp edges, changes image structure and introduces illumination inconsistencies [136]. How to discriminate these introduced edges from authentic edges, to capture the information of texture and structure of colors, and to extract illumination information are the important clues in image splicing detection. This chapter describes an efficient technique for image splicing detection using illumination component and LBP. Sect. 6.2.1 describes the *YCbCr* color space. The extraction of the illumination component is discussed in Sect. 6.2.2. Sect. 6.2.4 describes the classification stage.
Most recent techniques consider the image splicing problem as a binary classification problem with two main phases [143]:

1. Phase 1: classify image as tampered or authentic based on features extracted.

2. Phase 2: localize the tampered regions in forged image.

To solve the first phase, various techniques based on deep learning have been proposed [43, 114]. However, the existing methods with high detection accuracy are computationally expensive. Majority of them are based on complex deep learning models. They are expensive to train and require a large amount of data to perform better. They also run on expensive GPUs. Since this increases the cost to the users, the use of a simplistic machine learning model that considers a trade-off between the cost and accuracy is

needed. The main goal is to detect spliced images using an efficient model, which has a very simple structure and uses a small feature vector, yet obtain a plausible accuracy compared to the performance of deep learning models. The need of large training sets is thus eliminated.
In Sect. 6.2 an efficient method that uses illumination component, chroma channel features and LBP to detect images as spliced (tampered) or authentic is discussed .

6.2 Image splicing detection technique using illumination component and LBP

The present technique starts by converting RGB image to $YCbCr$. This conversion is important because the human visual system is more sensitive to overall intensity Y (luma) changes than to colour changes ($CbCr$) [29, 54]. Thus, most tampering clues which are imperceptible by naked eye are concealed in chromatic channel ($CbCr$). A spliced image is composed by two or more images taken from different cameras within different lighting conditions. Therefore, there is an illumination inconsistency into the spliced image. The present method uses the *Illumination-Reflectance model* [54] of image formation to estimate the illumination component from the image. Illumination is the lighting condition during image capture. Reflectance is the reflectance of the object(s) on the scene. Illumination varies slowly (Low-frequency) across the image as compared to reflectance which change considerably at object edges (High-frequency) [54]. Therefore, the log domain and Fourier transform are used to separate the illumination and reflectance, e.g., Homomorphic filter [10]. To achieve efficiency, the present method uses Y (luma) instead of using the log domain and Fourier transform to extract the illumination component. It is reasonable because the intensity information Y (luma) is related to low frequency [66]. An edge-preserving filter, e.g., *Bilateral filter* [131], should be used to ensure that clear edges are kept between surfaces under different lighting conditions in the estimated illumination. Since Bilateral filter is slower compared to other smoothing filters [22], we can approximate an edge-preserving filter using the max and min filters because these filters act like dilation and erosion [54]. Therefore, they can locally smooth the image and keep the important edges. We consider to filter Y from YCbCr with max and min filters (approximate edge-preserving filter), then retain the output image as the estimated illumination.

The present method can be summarized as follows in algorithm 5: a max-min filter is used to approximate edge-preserving filter, then we apply this filter to the lumi-

nance channel to estimate the illumination component based on Illumination-Reflectance model. The LBP normalized histogram computed from the illumination component and chrominance channel is used as a feature vector. For effective modeling and discrimination, the present method takes in various machine learning classifiers including Decision Tree (DT), SVM, Logistic Regression (LR), LDA, K-Nearest Neighbors (KNN) and Naive Bayes (NB) [18].

The techniques [104] and [118] exhibit a high accuracy in splicing forgery detection. However, they are computationally expensive, the former uses a Steerable Pyramid Transform (SPT) that involves multi-scale and multi-orientation image decomposition [61]. The latter uses CNN with eight hidden layers and requires a large amount of data samples to train the CNN model. Therefore, the present work introduces a new efficient method. Fig. 6.1 illustrates the diagram of the proposed method.

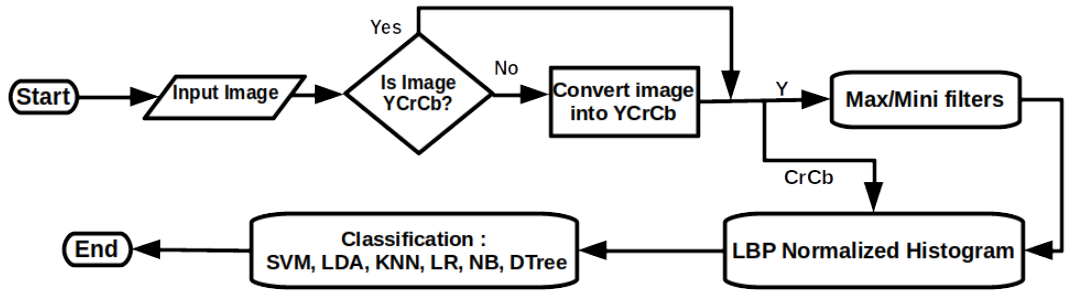


Figure 6.1: Flowchart of the present method

6.2.1 Convert input image to YCbCr

$YCbCr$ isolates luma/luminance, the intensity information, from chroma/chrominance, the colour information. Y is the luma component, Cb and Cr are the blue-difference and red-difference chroma components. The authors in [29, 32, 66, 136] showed that Y is related to low frequency (Y is more sensitive to the human eye); whereas $CbCr$ is related to high frequency (Cb and Cr are less sensitive to the human eyes). Chrominace channel is a good feature for splicing detection because tampering clues which are undetectable by naked eyes are hidden in chromatic channel. The input image is converted from the RGB to $YCbCr$ color space [11] by:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B \tag{6.1}$$

$$Cb = 0.492(B - Y) \tag{6.2}$$

$$Cr = 0.877(R - Y) \tag{6.3}$$

6.2.2 Extracting illumination component from luma

The interaction between light and object surfaces can be described using the Illumination-Reflectance model (e.g., Homomorphic filter) [10, 54]. This model assumes that the intensity at any pixel, which is the amount of light reflected by a point on the object, is the product of the illumination of the scene and the reflectance of the object(s). This model considers an image as a function F expressed by the product of illumination L and reflectance R components as follows:

$$F(x, y) = L(x, y)R(x, y) \tag{6.4}$$

F is the image, L is illumination of the scene and R is reflectance of object(s) on the scene. Illumination is low-frequency (varies slowly across the image), whereas reflectance is high-frequency (changes at object edges). This difference enables to split these two components. The log (lg) domain is used to transform the multiplicative components to additive components as follows:

$$lg(F(x, y)) = lg(L(x, y)R(x, y)) \tag{6.5}$$

$$lg(F(x, y)) = lg(L(x, y)) + lg(R(x, y)) \tag{6.6}$$

The structure of an Homomorphic filter is shown in Fig. 6.2 .

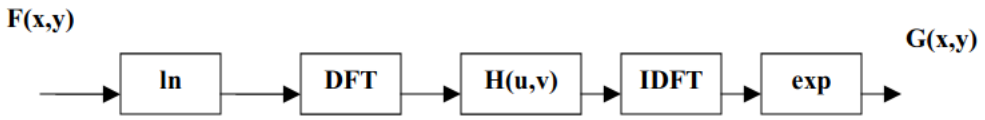


Figure 6.2: Homomorphic filtering

l_n is log, DFT is Discrete Fourier Transform, $H(u, v)$ is high-pass kernel, and $IDFT$ is Inverse Discrete Fourier Transform.

To retain the illumination component, a Low Pass Filter (LPF) can be used to remove high frequencies (reflectance) and keep the low frequencies (illumination). However, Homomorphic filtering requires twice Fourier transform; and the Retinex [137] and Homomorphic methods assume that the illumination component is globally smooth.
However, in practice, if the orientations of the object surfaces differ from each other, their light-receiving conditions will be also different. Thus, the real illumination com-

ponent often has sharp edges, especially at the object boundaries. Chen et al.(2016)[54] concluded that a proper illumination component should satisfy two conditions:

1. Clear edges are kept between surfaces under different lighting conditions.

2. Details within a single surface are blurred.

Since LPF tends to blur edges, it is logical to utilize an *edge-preserving filter* to extract the illumination. Thus, edge-preserving filter such as *Bilateral filter* [131] can be used instead of LPF. Bilateral filter is typically effective in image smoothing while keeping important edges but the operation is slower compared to other smoothing filters.

To make a trade-off between the effeciency and the detail removal ability, it is ideally suited to approximate a fast edge-preserving filter using the *max and min filters* [54]. Max and min filters are given by:

$$f(x,y) = max_{(i,j)\in S_{xy}}\{K(i,j)\}, \quad f(x,y) = min_{(i,j)\in S_{xy}}\{K(i,j)\} \tag{6.7}$$

To achieve a high running speed, there are very fast implementations for max and min filters [54, 141]. Since luma (Y) is related to low frequency in the image, we used Y with the max and min filters to estimate the illumination component as shown in algorithm 4.

Algorithm 4: pseudocode

Result: Estimated Illumination

Input I:(R,G,B) ← Y CbCr ;

foreach *pixel* $x \in Y$ **do**
  $M_a(x) \leftarrow GetMaxValueInTheMask$

foreach *pixel* $x \in M_a$ **do**
  $M_i(x) \leftarrow GetMinValueInTheMask$

$Ill \leftarrow a * (M_i + t)$;

$R \leftarrow I/Ill$

Ill is estimated illumination, R is reflectance, $t = 0.05$ is a small positive number used to prevent zeros in division and $a = 1.1$ is a small constant slightly larger than 1 used to avoid the resulting image being too bright [54]. Dominant edges are retained because the max and min filters act like dilation and erosion. Fig. 6.3 illustrates the illumination component from luma.

Figure 6.3: Illumination estimation
(a) Input image. (b) Luma. (c) Illumination.

Fig. 6.3(b) is luma Y (the intensity information). It is obtained by converting RGB to YCbCr, then sparate Y from CbCr. Fig. 6.3(c) is the estimated illumination from Y. It is obtained by filtering Y component using the max and min filters.

6.2.3 Local Binary Patterns (LBP)

LBP provides features (visual descriptors) that are used for image texture classification [66, 97]. Image texture gives us information about the spatial arrangement of color in the image. Given p_c as a central pixel value, P number of neighborhood pixels and r the radius of the neighborhood. LBP is calculated by:

$$LBP_{p,r} = \sum_{i=1}^{p-1} S(p_i - p_c).2^i \tag{6.8}$$

$$S(p_i - p_c) = \begin{cases} 1 & : p_i \geq p_c \\ 0 & : p_i < p_c \end{cases} \tag{6.9}$$

Fig. 6.4 illustrates the LBP.Fig. 6.5 illustrates the feature vector for the present method.

6.2.4 Tamper detection

Input image is detected as original or forged based on the features extracted. To categorize an input image as authentic or spliced, the present method uses the LBP normalized histogram as the feature vector for classification. A standard approach is to run multiple classifiers and compare their performance against one another, then pick the classifier

which has the best performance [18]. Pedregosa et al.(2011)[5] provide access to numerous different classification algorithms such as KNN [94], SVM [65], LDA [129], LR [113], DT [73], Random Forests [49] and NB [15].

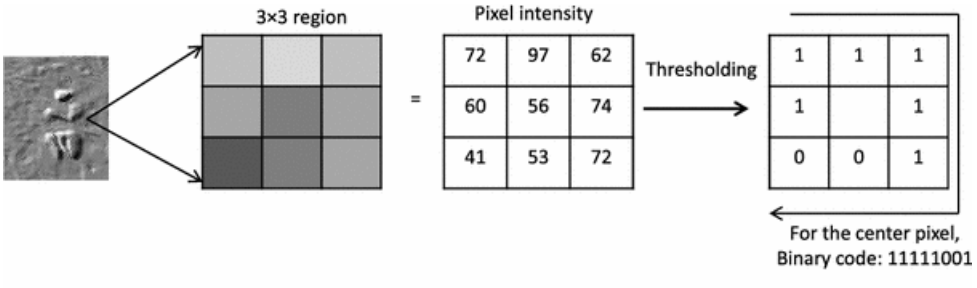


Figure 6.4: Local Binary Patterns

Fig. 6.5 illustrates the feature vector.

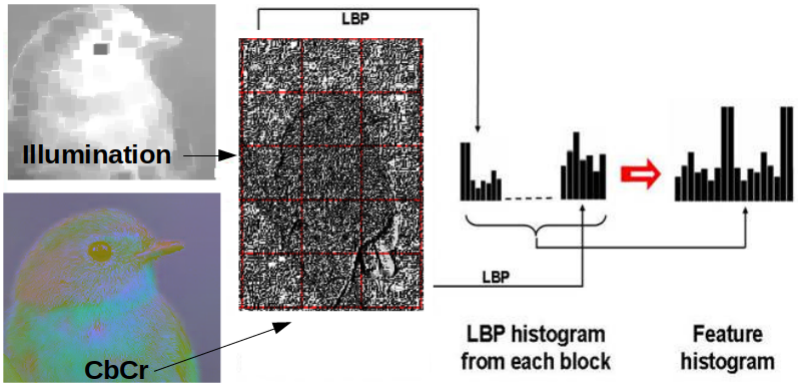


Figure 6.5: Feature vector (LBP histogram)

Algorithm 5 shows the the pseudocode of the present method.

6.3 Algorithm

Algorithm 5: Splicing detection based on illumination and LBP

Result: Classify as authentic or forged

Input I:(R,G,B) $\leftarrow$ Y CbCr **foreach** *pixel* $x \in Y$ **do**
  $M_a(x) \leftarrow GetMaxValueInTheMask$

foreach *pixel* $x \in M_a$ **do**
  $M_i(x) \leftarrow GetMinValueInTheMask$

$Ill \leftarrow a * (M_i + t)$

$LBP\_L' \leftarrow LBP(Ill)$

$LBP\_Cb \leftarrow LBP(Cb)$

$LBP\_Cr \leftarrow LBP(Cr)$

$LBP\_All \leftarrow merge(\ LBP\_L',\ LBP\_Cb,\ LBP\_Cr)$

$Feature\_vector \leftarrow normalizedHistogram(LBP\_All)$

$All\_classify \leftarrow SVM,\ LDA,\ NB,\ LR,\ KNN,\ DTREE;$

$All\_classify(X, Y)$

Ill is the estimated illumination, $a = 1.1$, $t = 0.05$[54], X : training data, Y : class labels of X.

6.4 Experimental Results

In this section, the dataset used is decribed in Sect. 6.4.1, the platform and running time analysis are discussed in Sect. 6.4.2, and experiment results are reported in Sect. 6.4.3.

6.4.1 Dataset and evaluation metrics

Experiments are performed using 12614 images from the dataset CASIA v2.0 [60]. 5123 are forged colored images, whereas 7491 are authentic colored images. The sizes (width, height) of these images varie from 240×160 to 900×600 and they are in different formats including JPEG, BMP and TIFF. The forgery detection on CASIA v2.0 is more challenging because this dataset contains mostly low resolution images and tampered regions are post-processed. The metrics to assess the performance of an image forgery detection technique have been described in Sect. 3.7. Fig. 6.6 plots the images distribution in authentic and tampered categories.

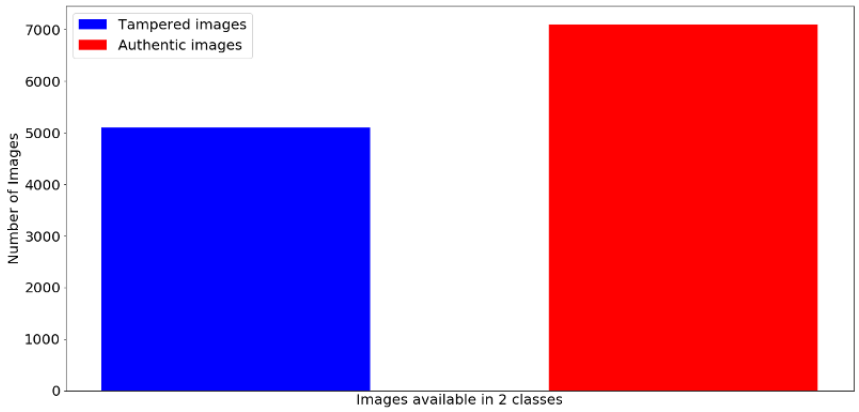


Figure 6.6: Data class distribution in dataset CASIA 2.0

6.4.2 Experimental platform and running time analysis.

The experiments are performed using a desktop with Intel(R) Core(TM) i5-5200U CPU @ 2.20GHz, 64-bit processor with 8GB RAM. Python 3.6, Scikit-learn 0.21.2 and Ubuntu 18.04.3 LTS OS software environments are used. Tab. 6.1 reports the running time in minutes on 12614 images from dataset CASIA v2.0 with classifiers including SVM, KNN, LDA, DT and NB. Training size = 60% and test size = 40%

Table 6.1: Analysis of running time. M is minute, and S is second. 12614 images from dataset CASIA v2.0 are used

Feature vector size	Feature extraction (M)	Training time (M)	Prediction time (S)
256	42.18	1.50	13.42
512	55.0	2.0	20.44
768	76.10	3.30	52.61

The reported training time, and prediction time, are for all classifiers combined (SVM, KNN, LDA, Decision Tree and Naive Bayes).

6.4.3 Experiment results

The experiments are performed on CASIA v2.0 public benchmark dataset for forgery detection. Different machine learning algorithms including SVM, KNN, LDA, DT and NB are used with the present model. The detection results (accuracy and the Area under the ROC Curve(AUC)) [47] are reported. Tab. 6.2 reports the accuracy of the model

when the feature vector size is 256, and the Fig. 6.7 is the graphical plot of the ROC curve.

Table 6.2: Accuracy when the feature vector size is 256.

Classifiers	Accuracy (%)
Logistic Regression	62.20
Decision Tree	58.30
K Nearest Nearbor	60.89
Linear Discriminant Analysis	**65.83**
Naive Bayes	55.52)
Support Vector Machines	59.27

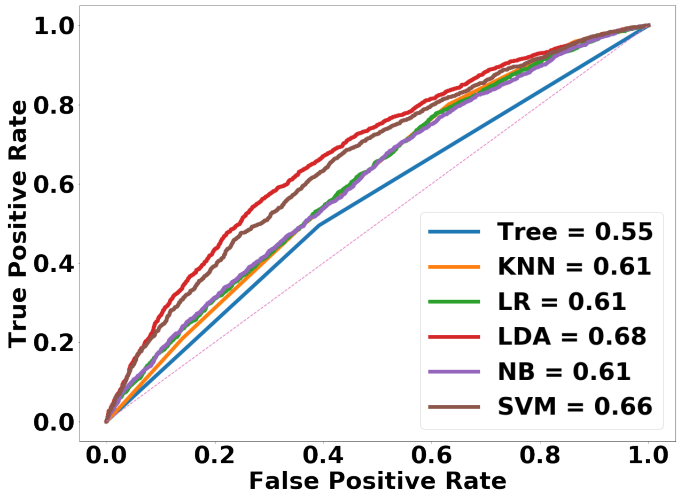


Figure 6.7: ROC feature vector size is 256

The results from Tab. 6.2 show the performance for different classifiers when the feature vector length is 256. LDA gives better performance with accuracy = 65.83% on 12614 images from CASIA v2.0.

Tab. 6.3 reports the accuracy of the model when the feature vector size is 512, and the ROC curve is shown in Fig. 6.8.

Table 6.3: Accuracy when the feature vector size is 512.

Classifiers	Accuracy (%)
Logistic Regression	89.13
Decision Tree	88.58
K Nearest Nearbor	86.08
Linear Discriminant Analysis	**92.98**
Naive Bayes	76.55
Support Vector Machines	90.22

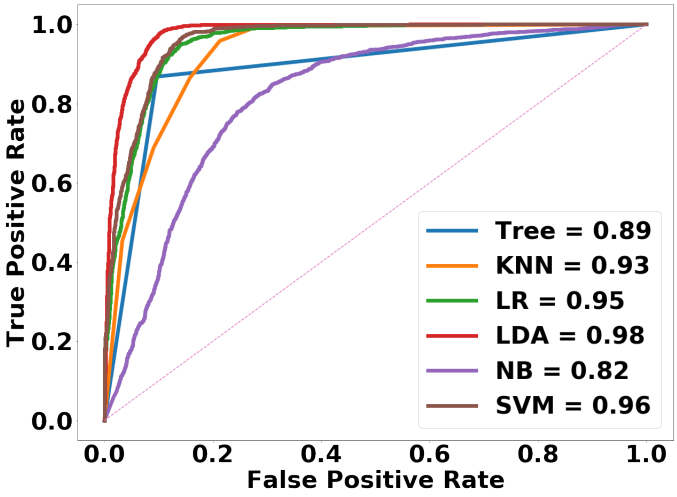


Figure 6.8: ROC feature vector size is 512

The results from Tab. 6.3 show the performance for different classifiers when the feature vector length is 512. LDA gives better performance with accuracy = 92.92% on 12614 images from CASIA v2.0.

Tab. 6.4 reports the accuracy of the model when the feature vector size is 768, and the Fig. 6.9 is the graphical plot of the ROC curve.

Table 6.4: Accuracy when the feature vector size is 768.

Classifiers	Accuracy (%)
Logistic Regression	90.82
Decision Tree	91.07
K Nearest Nearbor	86.08
Linear Discriminant Analysis	**93.79**
Naive Bayes	76.93
Support Vector Machines	91.45

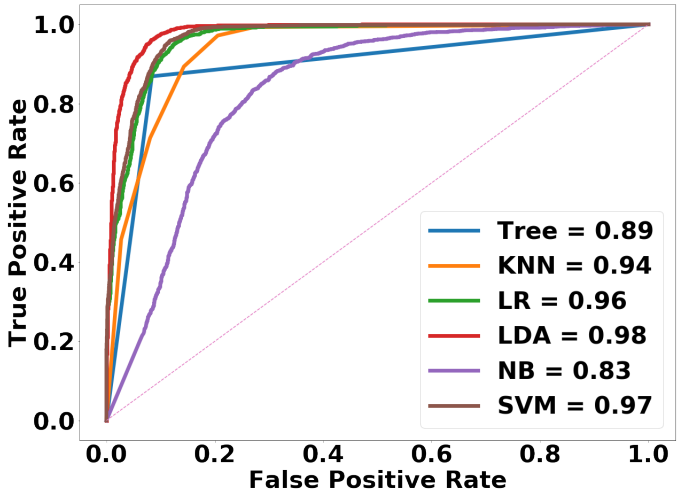


Figure 6.9: ROC feature vector size is 768

The results from Tab. 6.4 show the performance for different classifiers when the feature vector length is 768. LDA gives better performance with accuracy = 93.79% on 12614 images from CASIA v2.0.

6.4.4 Comparative Results

From the results obtained in Sect. 6.4.3, it is noticeable that the present method performs better when the classifier used is LDA. The accuracy obtained are 94.59% on 4117 images and 93.79% on 12614 images. Tab. 6.6 compares the performance of the present technique against other related techniques. Tab.6.5 reports the tn, fp, fn, tp, p_r, r_c, and f_1 score detection results obtained on 12614 images from dataset CASIA v2.0.

Table 6.5: tn, fp, fn, tp, p_r, r_c, and f_1 score, on 12614 images from CASIA v2.0. Testsize=0.4.

Classifiers	tn	fp	fn	tp	precision%	Recall%	F1 score
Logistic Regression	2734	275	192	1845	93	91	92
Decision Tree	2757	252	251	1786	92	92	92
K Nearest Nearbor	2582	427	199	1838	93	86	89
Linear Discriminant Analysis	**2789**	**220**	**96**	**1941**	**97**	93	95
Naive Bayes	2268	741	438	1599	84	75	79
Support Vector Machines	2734	275	191	1846	**97**	90	93
Deep Learning + HWT [63]	-	-	-	-	**97**	**99**	**98**
Markov +DCT+DWT [78]	-	-	-	-	-	92.5	-

The results from Tab. 6.5 show that the method [63] presents better f_1 score. However, this technique uses deep learning (6 convolution layers with 600 kernels). Thus, it is computationally expensive compared to the present method (see Tab. 6.6).

Table 6.6: Accuracy comparison with other models known in literature on dataset CASIA v2.0

Models	Feature vector: size	*acc* (%)
CNN+RGB [118]	Pretrained CNN: 16384 Feature fusion : 400	**97.83**
Markov features in DCT and DWT domain[78]	-	89.76
Fast SCNN[143]	SCNN : 25088	85.83
Improved double quantization detection[130]	-	79.72
Deep Learning Local Descriptor[142]	-	96.97
STP + LBP[104]	LBP Histogram: 3,584 to 480	97.33
Deep Learning + HWT [63]	HWT: 4,096 to 1,024.	96.36
2D Noncausal Markov Mode[138]	14,240	93.36
Presented Method	LBP Histogram : **256** LBP Histogram: **512** LBP Histogram: **768**	65.83 92.98 94.59

The methods [118] and [104] present high accuracy. However, the present method is computational efficient than these two methods, because the former requires a large data samples to train the CNN model and the latter uses SPT that involves multiple image decompositions (scale and orientation) [61]. Fig. 6.10 shows spliced images successfully classified as forged.

Figure 6.10: Spliced images successfully classified as forged

6.4.5 Failure cases

We observed that many cases of misclassification are authentic images categorized as tampered image. Fig. 6.11 shows authentic images misclassified as tampered images.

Figure 6.11: Authentic images misclassified as tampered images

Failure causes are small image size, photographs taken with background blur effect, and uniform background. There is no considerable case of tampered images misclassified as authentic images.

6.4.6 Conclusion

An efficient method for image splicing detection method was presented. The technique uses the chroma channel, illumination component and LBP to discriminate spliced image from authentic image. The input image is first converted into luminance and chroma channels; considering the Illumination-Reflectance model, the illumination component is extracted from luminance using an approximate edge-preserving filter with the max and

min filters; then LBP normalized histogram computed from illumination and chroma is used as a feature vector for classification using different machine learning algorithms. Extensive experiments demonstrate that the present algorithm is computationally efficient and effective for splicing forgery detection.

6.5 Summary

In this chapter, we have demonstrated that the illumination component and LBP can be used for image splicing detection. The combination of illumination component and LBP enables to construct an image splicing detection technique that uses a small feature vector and has a simple structure. Images involved in splicing are taken from different cameras within different lighting conditions, this creates illumination inconsistency into the spliced image. Thus, illumination component is a good feature for image splicing tamper detection. The experiment results show that the present image splicing tamper detection technique is computationally efficient and effective for splicing tamper detection.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis, we presented several techniques for detecting and localizing forged regions in copy-move and image-splicing forgeries. We also presented a technique to obtain geometric transformation parameters with high accuracy for copy-move forgeries. Our results in Chapters 3 and 4 demonstrate that image blobs overcome many limitations of block, keypoint and segment based approaches. Moreover, blobs allowed us to handle one of the most persistent problems with other approaches – namely that of uniform and background regions. Blobs ensure that in a majority of cases, the original and forged regions belong to different blobs. Such a property allowed us to greatly reduce computational costs. Our results also show that blobs may be used with many standard off-the-shelf features and do not require many specialized fine-tuned features or parameters.

The use of illumination information makes our work on image-splicing forgery detection different from most of the existing literature which is based on image content such as edge inconsistencies and localized blurring near spliced boundaries. We believe that ours is the first method that used illuminant component via $YCbCr$ color space, edge-preserving smoothing filter and local binary patterns. The results reinforce our intuition that illumination inconsistencies provide generic and vital information about tampering.

7.2 Recommendations and Future Work

While forgery detection results obtained from proposed techniques are on par with the state-of-the-art, the failures indicate that there is more work to be done in the field.

The performance of the deep networks, while higher, is achieved with larger datasets and greater compute power. The convolution filters in such networks provide *learned* features which are highly tuned to the application at hand [92].It will be interesting to explore such features in combination with blobs and study the performance. Finally, we would also like to add a localization mechanism in the image-splicing detection algorithms.

On the other hand, it would also be interesting to study if image blob computing mechanisms may be implemented as a part of a deep network— blob detection is built from a convolution operation — and combine it with the illumination components to see the effect on the performance of deep networks.

List of Publications

1. Niyishaka Patrick and Chakravarthy Bhagvati. **Digital Image Forensics Technique for Copy-Move Forgery Detection Using DoG and ORB.** ICCVG 2018, Warsaw, Poland, September 17-19, 2018, Proceedings. 10.1007/978 − 3 − 030 − 00692 − 1_41.

2. Niyishaka Patrick and Chakravarthy Bhagvati. **Copy-Move forgery detection using image blobs and BRISK feature**. *Multimedia Tools and Applications (MTAP)*, https://doi.org/10.1007/s11042-020-09225-6.

3. Niyishaka Patrick and Chakravarthy Bhagvati. **Geometric transformation estimation from copy-move forgery using image blobs features and keypoints**. *Multimedia Tools and Applications (MTAP)*, Communicated.

4. Niyishaka Patrick and Chakravarthy Bhagvati. **Image splicing detection technique based on illumination component and LBP**. *Multimedia Tools and Applications (MTAP)*, https://doi.org/10.1007/s11042-020-09707-7.

References

[1] **16 Famous First Photographs in History: From the Oldest Photo Ever to the World's First Instagram**. https://mymodernmet.com/first-photograph-photography-history/. Accessed: 2019-07-12. (1)

[2] **Affine transformation**. https://en.wikipedia.org/wiki/Affine_transformation. Accessed: 2019-08-15. (68)

[3] **Blob detection**. https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_blob Accessed: 2019-07-29. (31)

[4] **Blob Detection Using OpenCV (Python, C++)**. https://www.learnopencv.com/blob-detection-using-opencv-python-c/. Accessed: 2019-07-29. (31)

[5] **Classifier comparison**. https://scikit-learn.org/stable/auto_examples/classification/plot_classifie Accessed: 2020-01-02. (84)

[6] **Facebook-Statistics**. https://www.brandwatch.com/blog/facebook-statistics/. Accessed: 2019-07-11. (2)

[7] **Facebook Users Are Uploading 350 Million New Photos Each Day**. https://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9?IR=T. Accessed: 2019-07-11. (2)

[8] **First Photograph**. https://www.hrc.utexas.edu/exhibitions/permanent/. Accessed: 2019-07-12. (1)

[9] **Heliography**. https://en.wikipedia.org/wiki/Heliography. Accessed: 2019-07-12. (1)

[10] **Homomorphic filtering**. https://en.wikipedia.org/wiki/Homomorphic_filtering. Accessed: 2020-01-02. (79, 81)

[11] **Human Vision and Color**. https://biomachina.org/courses/imageproc/121.pdf. Accessed: 2020-01-20. (80)

[12] **Image segmentation**. https://en.wikipedia.org/wiki/Image_segmentation. Accessed: 2019-07-29. (35)

[13] **Introduction to SIFT (Scale-Invariant Feature Transform)**. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html. Accessed: 2019-08-15. (62)

[14] **The inventor of the first self-contained (portable) digital camera**. https://en.wikipedia.org/wiki/Steven_Sasson. Accessed: 2020-01-10. (1)

[15] **Naive Bayes Classifier**. https://en.wikipedia.org/wiki/Naive_Bayes_classifier. Accessed: 2020-01-02. (84)

[16] **OpenCV: Open source computer vision library**. https://github.com/opencv/opencv. Accessed: 2019-07-10. (2)

[17] **OpenCV: seamless cloning**. https://www.learnopencv.com/seamless-cloning-using-opencv-python-cpp/. Accessed: 2019-07-09. (2)

[18] **Overview of Classification Methods in Python with Scikit-Learn**. https://stackabuse.com/overview-of-classification-methods-in-python-with-scikit-learn/. Accessed: 2020-01-02. (80, 84)

[19] **Scale Space**. https://en.wikipedia.org/wiki/Scale_space. Accessed: 2019-07-29. (33)

[20] **Scale-Space Blob Detection**. https://medium.com/@nikhilkumar0042/scale-space-blob-detection-b93a4a0829ba. Accessed: 2019-07-30. (33)

[21] **SIFT: Theory and Practice. LoG approximations**. http://aishack.in/tutorials/sift-scale-invariant-feature-transform-log-approximation/. Accessed: 2019-07-29. (33, 60)

[22] **Smoothing images**. https://docs.opencv.org/master/d4/d13/tutorial_py_filtering.html. Accessed: 2019-08-15. (79)

[23] **Social-Media-Statistics**. https://dustinstout.com/social-media-statistics/. Accessed: 2019-07-11. (2)

[24] **Steve on Image Processing and MATLAB: Homomorphic filtering-part 1**. https://blogs.mathworks.com/steve/2013/06/25/homomorphic-filtering-part-1/. Accessed: 2019-07-13. (7)

[25] **Support Vector Machines**. https://en.wikipedia.org/wiki/Support-vector_machine. Accessed: 2019-07-16. (25)

[26] **Support Vector Machines**. https://scikit-learn.org/stable/modules/svm.html. Accessed: 2019-07-16. (25)

[27] **A tutorial on binary descriptors – part 3 – The ORB descriptor**. https://gilscvblog.com/2013/10/04/a-tutorial-on-binary-descriptors-part-3-the-orb-descriptor/. Accessed: 2020-01-02. (19)

[28] **What is an Affine Transformation?** https://docs.opencv.org/2.4/doc/tutorials/imgproc-/imgtrans/warp_affine/warp_affine.html. Accessed: 2019-10-15. (68)

[29] **What is YCbCr color space**. https://medium.com/breaktheloop/what-is-ycbcr-964fde85eeb3. Accessed: 2021-02-15. (79, 80)

[30] Jeyalakshmi A. and Chitra D.Ramya. **Image splicing detection based on surf with ripplet Transform-ii**. *nternational Journal of Recent Technology and Engineering (IJRTE)*, **7**, 2019. (24)

[31] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. **SLIC superpixels compared to state-of-the-art superpixel methods**. *IEEE transactions on pattern analysis and machine intelligence*, **34**(11):2274–2282, 2012. (35)

[32] Amani Alahmadi, Muhammad Hussain, Hatim Aboalsamh, Ghulam Muhammad, and George Bebis. **Splicing image forgery detection based on DCT and Local Binary Pattern**. pages 253–256, 12 2013. (80)

[33] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. **KAZE Features**. In *Computer Vision – ECCV 2012*, pages 214–227, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. (63, 64)

[34] Mohammed Hazim Alkawaz, Ghazali Sulong, Tanzila Saba, and Amjad Rehman. **Detection of copy-move image forgery based on discrete cosine transform**. *Neural Computing and Applications*, **30**(1):183–192, Jul 2018. (13, 14)

[35] Irene Amerini, Lamberto Ballan, Roberto Caldelli, Alberto Del Bimbo, Luca Del Tongo, and Giuseppe Serra. **Copy-move forgery detection and localization by means of robust clustering with J-Linkage**. *Signal Processing: Image Communication*, **28**(6):659–669, 2013. (13, 20)

[36] Irene Amerini, Lamberto Ballan, Roberto Caldelli, Alberto Del Bimbo, and Giuseppe Serra. **A SIFT-Based Forensic Method for Copy–Move Attack Detection and Transformation Recovery**. *Information Forensics and Security, IEEE Transactions on*, **6**:1099–1110, 10 2011. (12, 13, 18, 20, 22, 42, 50, 51, 53, 56, 66, 70, 72, 75, 76)

[37] Roy Aniket, Dixit Rahul, Naskar Ruchira, and Subhra Chakraborty Rajat. *Digital Image Forensics: Theory and Implementation*, **755**. June 2018. (4)

[38] Anuradha, Singh Baljinder, and Sood Ritika. **A Hybrid Algorithm For Image Forgery Detection**. *International Journal of Computer Science and Mobile Computing*, **7**:122–128, 03 2018. (12)

[39] V. Malviya Ashwini and A. Ladhake Siddharth. **Pixel Based Image Forensic Technique for Copy-move Forgery Detection Using Auto Color Correlogram**. *Procedia Computer Science*, **79**:383–390, 2016. Proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016. (xvi, xvii, 41, 42, 44, 56, 57)

[40] Md Atiqur Rahman and Yang Wang. **Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation**. **10072**, pages 234–244, 12 2016. (36)

[41] Zhihua Ban, Jianguo Liu, and Li Cao. **A novel Gaussian mixture model for superpixel segmentation**. *CoRR*, **abs/1612.08792**, 2016. (12)

[42] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. **SURF: Speeded Up Robust Features**. In *Computer Vision -ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. (18, 60, 62)

[43] Belhassen Bayar and Matthew C. Stamm. **A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer**. pages 5–10, 06 2016. (24, 29, 78)

[44] Aparna Bharati, Richa Singh, Mayank Vatsa, and Kevin W. Bowyer. **Detecting Facial Retouching Using Supervised Deep Learning**. *IEEE Transactions on Information Forensics and Security*, **11**:1903–1913, 09 2016. (4, 10)

[45] Xiuli Bi, Chi-Man Pun, and Xiao-Chen Yuan. **Multi-level dense descriptor and hierarchical feature matching for copy-move forgery detection**. *Information Sciences*, **345**:226–242, 2016. (13, 16)

[46] David Blatner. **Photoshop: It's Not Just a Program Anymore.**, August 2000. (1)

[47] Andrew P. Bradley. **The use of the area under the ROC curve in the evaluation of machine learning algorithms**. *Pattern Recognition*, pages 1145–1159, 1997. (86)

[48] G. Bradski, K. Konolige, V. Rabaud, and E. Rublee. **ORB: An efficient alternative to SIFT or SURF**. In *2011 IEEE International Conference on Computer Vision (ICCV 2011)(ICCV)*, **00**, pages 2564–2571, 11 2011. (7, 17, 38, 62, 64)

[49] Leo Breiman. **Random Forests**. *Machine Learning*, pages 5–32, Oct 2001. (84)

[50] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. **BRIEF: Binary Robust Independent Elementary Features**. **6314**, pages 778–792, 09 2010. (39)

[51] John Canny. **A computational approach to edge detection**. In *Readings in computer vision*, pages 184–203. Elsevier, 1987. (37)

[52] Li Ce, Ma Qiang, Xiao Limei, Li Ming, and Zhang Aihua. **Image splicing detection based on Markov features in QDCT domain**. *Neurocomputing*, **228**:29–36, 2017. Advanced Intelligent Computing: Theory and Applications. (24, 26)

[53] Sekhar Chandra and T N Sankar. **Review of Image Splicing Forgery Detection Techniques**. *Journal of Emerging Technologies and Innovative Research*, **3**, 2016. (4)

[54] Dongliang Chen, Shanzhen Lan, Pin Xu, and Yongqin Zhang. **Illumination-Reflectance Based Image Enhancement Method for Character Recognition**. pages 207–211, 10 2016. (7, 79, 81, 82, 85)

[55] Vincent Christlein, Christian Riess, Johannes Jordan, Corinna Riess, and Elli Angelopoulou. **An evaluation of popular copy-move forgery detection approaches**. *IEEE Transactions on information forensics and security*, **7**(6):1841–1854, 2012. (16, 41, 42)

[56] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. **Copy-move forgery detection based on patchmatch**. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 5312–5316. IEEE, 2014. (13)

[57] Davide Cozzolino, Giovanni Poggi, and Luisa Verdoliva. **Efficient dense-field copy–move forgery detection**. *IEEE Transactions on Information Forensics and Security*, **10**(11):2284–2297, 2015. (13)

[58] S. Debbarma, A. B. Singh, and K. M. Singh. **Keypoints based copy-move forgery detection of digital images**. pages 1–5, 11 2014. (17)

[59] Rahul Dixit, Ruchira Naskar, and Swati Mishra. **Blur-invariant copy-move forgery detection technique with improved detection accuracy utilising SWT-SVD**. *IET Image Processing*, **11**(5):301–309, 2017. (13, 15)

[60] Jing Dong, Wei Wang, and Tieniu Tan. **CASIA Image Tampering Detection Evaluation Database**. pages 422–426, 07 2013. (27, 85)

[61] Fadoua Drira, Florence Denis, and Atilla Baskurt. **Image watermarking technique based on the steerable pyramid transform**. pages 165–176, 11 2004. (80, 90)

[62] Mahmoud Emam, Qi Han, and Xiamu Niu. **PCET based copy-move forgery detection in images under geometric transforms**. *Multimedia Tools and Applications*, **75**(18):11513–11527, 2016. (13, 16)

[63] Hala H. Zayed Eman I. Abd El-Latif, Ahmed Taha. *International Journal of Computer Network and InformationSecurity(IJCNIS)*, **5**:28–35, 2019. (24, 90)

[64] Rublee Ethan, Rabaud Vincent, Konolige Kurt, and Bradski Gary. **ORB: An efficient alternative to SIFT or SURF**. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011. (18, 19, 61)

[65] Theodoros Evgeniou and Massimiliano Pontil. **Support Vector Machines: Theory and Applications**. **2049**, pages 249–257, 01 2001. (7, 84)

[66] Hakimi Fahime, Hariri Mahdi, and GharehBaghi Farhad. **Image splicing forgery detection using local binary pattern and discrete wavelet transform**. *2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran*, pages 1074–107, 2015. (6, 7, 13, 24, 25, 79, 80, 83)

[67] Hany Farid. **Digital doctoring: how to tell the real from the fake**. 2006. (2)

[68] Hany Farid. **Digital Doctoring : can we trust photographs ?** 2007. (2)

[69] Maher Al Azrak Faten, Sedik Ahmed, I. Dessowky Moawad, Ashraf A. M. Khalaf Ghada, M. El Banby, S. Elkorany Ahmed, and E. Abd. El-Samie Fathi. **An efficient method for image forgery detection based on trigonometric transforms and deep learning**. *Multimedia Tools and Applications*, 2020. (13, 17, 34)

[70] Pablo Fernández Alcantarilla. **Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces**. 09 2013. (62)

[71] Martin A. Fischler and Robert C. Bolles. **Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography**. *Commun. ACM*, **24**(6):381–395, 1981. (5)

[72] J Fridrich, D. Soukal, and J Lukas. **Detection of copy-move forgery in digital images**. *Digital Forensic Research Workshop, Cleveland, OH*, pages 19–23, 2003. (44)

[73] Bhumika Gupta, Aditya Rawat, Akshay Jain, Arpit Arora, and Naresh Dhami. **Analysis of Various Decision Tree Algorithms for Classification in Data Mining**. *International Journal of Computer Applications*, **163**:15–19, 04 2017. (84)

[74] Farid Hany. **Image Forgery Detection: A survey.** *IEEE SIGNAL PROCESSING MAGAZINE*, March 2009. (1, 3, 4)

[75] Chris Harris and Mike Stephens. **A Combined Corner and Edge Detector**. In *Proceedings of the Alvey Vision Conference*, pages 23.1–23.6. Alvety Vision Club, 1988. doi:10.5244/C.2.23. (17)

[76] Mohammad Farukh Hashmi, Vijay Anand, and Avinas G. Keskar. **Copy-move Image Forgery Detection Using an Efficient and Robust Method**

Combining Un-decimated Wavelet Transform and Scale Invariant Feature Transform. *AASRI Procedia*, **9**:84–91, 12 2014. (20, 56)

[77] Khizar Hayat and Tanzeela Qazi. **Forgery detection in digital images via discrete wavelet and discrete cosine transforms**. *Computers and Electrical Engineering*, **62**:448–458, 2017. (3, 11, 13)

[78] Zhongwei He, Wei Lu, Wei Sun, and Jiwu Huang. **Digital image splicing detection based on Markov features in DCT and DWT domain**. *Pattern Recognition*, **45**:4292–4299, 2012. (90)

[79] Ordway Hilton. *Scientific Examination of Questioned Documents, Revised Edition*. Forensic and Police Science Series. Taylor & Francis, 1992. (4)

[80] Huang Hui-Yu and Ciou Ai-Jhen. **Copy-move forgery detection for image forensics using the superpixel segmentation and the Helmert transformation**. *EURASIP Journal on Image and Video Processing*, 2019. (13, 51)

[81] A.K. Jaiswal and R. Srivastava. **A technique for image splicing detection using hybrid feature set.** *Multimed Tools Appl 79, 11837–11860 (2020).* (24)

[82] Ankit Kumar Jaiswal and Rajeev Srivastava. **Image Splicing Detection using Deep Residual Network**. *Proceedings of 2nd International Conference on Advanced Computing and Software Engineering (ICACSE) 2019*, 2019. (24)

[83] Bappy Jawadul H., Roy-Chowdhury Amit K., Bunk Jason, Nataraj Lakshmanan, and Manjunath B.S. **Exploiting Spatial Structure for Localizing Manipulated Image Regions**. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4980–4989, 2017. (24, 28)

[84] Mingxing Jia, Zhixian Zhang, Pengfei Song, and Junqiang Du. **Research of Improved Algorithm Based on LBP for Face Recognition**. In *Biometric Recognition*, pages 111–119. Springer International Publishing, 2014. (25)

[85] Guonian Jin and Xiaoxia Wan. **An improved method for SIFT-based copy-move forgery detection using non-maximum value suppression and optimized J-Linkage**. *Signal Processing: Image Communication*, **57**:113–125, 2017. (13, 21)

[86] Dong Jing, Wang Wei, and Tan Tieniu. **CASIA Image Tampering Detection Evaluation Database**. In *2013 IEEE China Summit and International Conference on Signal and Information Processing*, pages 422–426, July 2013. (4)

[87] Young Park Jun, An Kang Tae, Ho Moon Yong, and Il. Kyu Eom. **Copy-Move Forgery Detection Using Scale Invariant Feature and Reduced Local Binary Pattern Histogram**. *Symmetry*, 2020. (13, 17, 51, 56)

[88] Harpreet Kaur, Jyoti Saxena, and Sukhjinder Singh. **Simulative comparison of copy-move forgery detection methods for digital images**. *International Journal of Electronics, Electrical and Computational System*, **4**:62–66, 2015. (56, 57)

[89] Rajdeep kaur and Amandeep Kaur. **COPY-MOVE FORGERY DETECTION USING ORB AND SIFT DETECTOR**. pages 804–8213, 2016. (18)

[90] Hui Kong, Hatice Cinar Akakin, and Sanjay Sarma. **A Generalized Laplacian of Gaussian Filter for Blob Detection and Its Applications**. *Cybernetics, IEEE Transactions on*, **43**:1719–1733, 01 2013. (7)

[91] Iryna Korshunova, Wenzhe Shi, J Dambre, and Lucas Theis. **Fast Face-Swap Using Convolutional Neural Networks**. pages 3697–3705, 10 2017. (2)

[92] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. **ImageNet Classification with Deep Convolutional Neural Networks**. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, **25**. Curran Associates, Inc., 2012. (94)

[93] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. **ImageNet Classification with Deep Convolutional Neural Networks**. *Commun. ACM*, **60**(6):84–90, may 2017. (2)

[94] J. Laaksonen and E. Oja. **Classification with learning k-nearest neighbors**. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, **3**, pages 1480–1483 vol.3, June 1996. (84)

[95] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. **BRISK: Binary Robust Invariant Scalable Keypoints**. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2548–2555, Washington, DC, USA, 2011. IEEE Computer Society. (7, 11, 17, 48, 60, 62, 64)

[96] Jingwei Li, Fan Yang, Wei Lu, and Wei Sun. **Keypoint-based copy-move detection scheme by adopting MSCRs and improved feature matching**. *Multimedia Tools and Applications*, **76**(20):20483–20497, Oct 2017. (13, 21)

[97] Wang Li and He Dong-Chen. **Texture classification using texture spectrum**. *Pattern Recognition*, **23**(8):905–910, 1990. (83)

[98] Cong Lin, Wei Lu, Xinchao Huang, Ke Zhen Liu, Wei Sun, Hanhui Lin, and Zhiyuan Tan. **Copy-move forgery detection using combined features and transitive matching**. *Multimedia Tools and Applications*, pages 1–16, 2018. (57)

[99] Cong Lin, Wei Lu, Wei Sun, Jinhua Zeng, Tianhua Xu, and Jian-Huang Lai. **Region duplication detection based on image segmentation and keypoint contexts**. *Multimedia Tools and Applications*, **77**(11):14241–14258, Jun 2018. (35)

[100] Tony Lindeberg. **Scale-space: A framework for handling image structures at multiple scales**. *Encyclopedia of Computer Science and Engineering*, **4**:2495–2504, 2008. (7, 33, 35, 49)

[101] David G. Lowe. **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision*, **60**(2):91–110, Nov 2004. (7, 18, 33, 50, 62, 65)

[102] Muthana Mahdi and Saad Alsaad. *Detection of Copy-Move Forgery in Digital Image Based on SIFT Features and Automatic Matching Thresholds*, pages 17–31. 01 2020. (13)

[103] Toqeer Mahmood, Zahid Mehmood, Mohsin Shah, and Zakir Khan. **An efficient forensic technique for exposing region duplication forgery in digital images**. *Applied Intelligence*, **48**(7):1791–1801, Jul 2018. (13, 16, 34)

[104] Ghulam Muhammad, Muneer Al-Hammadi, Muhammad Hussain, and George Bebis. **Image Forgery Detection Using Steerable Pyramid Transform and Local Binary Pattern**. *Machine Vision and Applications*, **25**:985–995, 05 2014. (80, 90)

[105] Gul Muzaffer, Ozge Makul, Beste Gençtürk, and Guzin Ulutas. **Copy Move Forgery Detection Using Gabor Filter and Scaled ORB**. pages 23–29, 03 2016. (19)

[106] Tian-Tsong Ng and Shih-Fu Chang. **A Data Set of Authentic and Spliced Image Blocks**. Technical report, Columbia University, June 2004. (10, 16)

[107] Bhagvati C. Niyishaka, P. **Image splicing detection technique based on Illumination-Reflectance model and LBP.** *Multimed Tools Appl*, pages 44–51, 2020. (6, 23, 24, 30)

[108] Patrick Niyishaka and Chakravarthy Bhagvati. *Digital Image Forensics Technique for Copy-Move Forgery Detection Using DoG and ORB: International Conference, ICCVG 2018, Warsaw, Poland, September 17 - 19, 2018, Proceedings*, pages 472–483. 09 2018. (xvii, 2, 11, 12, 17, 22, 54, 55, 56, 57, 61, 71, 75)

[109] Patrick Niyishaka and Chakravarthy Bhagvati. *Copy-Move Forgery Detection Using image blobs and BRSIK feature*. Springer, 2020. (5, 12, 14, 22, 45, 50, 52, 65, 66, 71, 75, 76)

[110] Joseph Ojeniyi, Bolaji O Adedayo, Ismaila Idris, and Shafi'i Abdulhamid. **Hybridized Technique for Copy-Move Forgery Detection Using Discrete Cosine Transform and Speeded-Up Robust Feature Techniques**. *International Journal of Image, Graphics and Signal Processing*, **10**:22–30, 04 2018. (xvii, 3, 4, 5, 11, 17, 22, 50, 51, 56, 57)

[111] Viola Paul and Jones Michael. **Robust Real-time Object Detection**. In *International Journal of Computer Vision*, 2001. (2)

[112] Anjie Peng, Yadong Wu, and Xiangui Kang. **Revealing Traces of Image Resampling and Resampling Antiforensics**. *Advances in Multimedia*, **2017**:1–13, 01 2017. (6, 23, 24, 26)

[113] Joanne Peng, Kuk Lee, and Gary Ingersoll. **An Introduction to Logistic Regression Analysis and Reporting**. *Journal of Educational Research - J EDUC RES*, **96**:3–14, 09 2002. (84)

[114] Zhou Peng, Han Xintong, Vlad I. Morariu, and Davis Larry S. **Learning Rich Features for Image Manipulation Detection**. *CoRR*, **abs/1805.04953**, 2018. (24, 29, 78)

[115] A. c. Popescu and Hany Farid. **Exposing Digital Forgeries by Detecting Duplicated Image Regions**. 2004. (5, 44)

[116] Chi-Man Pun, Xiao-Chen Yuan, and Xiu-Li Bi. **Image forgery detection using adaptive oversegmentation and feature point matching**. *IEEE Transactions on Information Forensics and Security*, **10**(8):1705–1716, 2015. (5, 12, 13, 21)

[117] Mehta Rachna and Agarwa Navneet. **Image Splicing Detection with Markov features and PCA**. *International Journal of Innovative Research inElectrical, Electronics, Instrumentation and Control Engineering IJIREEICE*, **7**, 2019. (24)

[118] Yuan Rao and Jiangqun Ni. **A deep learning approach to detection of splicing and copy-move forgeries in images**. *2016 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2016. (80, 90)

[119] Judith A Redi, Wiem Taktak, and Jean-Luc Dugelay. **Digital image forensics: a booklet for beginners**. *Multimedia Tools and Applications*, **51**(1):133–162, Jan 2011. (2, 3, 11)

[120] Fries Robert and Modestino James. **Image enhancement by stochastic homomorphic filtering**. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **27**(6):625–637, December 1979. (7)

[121] Mushtaq S. and Mir A. H. **Novel method for image splicing detection**. In *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2398–2403, Sep. 2014. (24, 26)

[122] Gurmeet Kaur Saini and Manish Mahajan. **Study of Copy Move Image Forgery Detection Based On Surf Algorithm**. *International Journal of Modern Electronics and Communication Engineering(IJMECE)*, pages 46–49, 2016. (56)

[123] Youseph Shahana N. and Roy Cherian Rajesh. **Pixel and Edge Based Illuminant Color Estimation for Image Forgery Detection**. *Procedia Computer Science*, **46**:1635–1642, 2015. Proceedings of the International Conference on Information and Communication Technologies, ICICT 2014, 3-5 December 2014 at Bolgatty Palace & Island Resort, Kochi, India. (6, 23, 24, 26)

[124] B.L. Shivakumar and S. SanthoshBaboo. **Detection of region duplication forgery in digital images using SURF**. *International Journal of Computer Science Issues (IJCSI)*, **8**:199–205, 2011. (17, 44)

[125] M Singh and EH Singh. **Detection of Cloning Forgery Images using SURF+ DWT and PCA**. *International Journal of Latest Engineering Research and Applications (IJLERA)*, **1**:1–10, 2016. (56)

[126] Irwin Sobel. **An Isotropic 3x3 Image Gradient Operator**. *Presentation at Stanford A.I. Project 1968*, 02 2014. (37)

[127] Fadl Sondos and Semary Noura. **Robust Copy-Move forgery revealing in digital images using polar coordinate system**. *Neurocomputing*, **265**:57–65, 2017. New Trends for Pattern Recognition: Theory and Applications. (11, 13, 16)

[128] Shaharyar Ahmed Khan Tareen and Zahra Saleem. **A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK**. pages 1–10, 03 2018. (17, 54, 60, 64)

[129] Alaa Tharwat, Tarek Gaber, Abdelhameed Ibrahim, and Aboul Ella Hassanien. **Linear discriminant analysis: A detailed tutorial**. *Ai Communications*, **30**:169–190,, 05 2017. (7, 84)

[130] V. L. L. Thing, Y. Chen, and C. Cheh. **An Improved Double Compression Detection Method for JPEG Image Forensics**. In *2012 IEEE International Symposium on Multimedia*, pages 290–297, Dec 2012. (90)

[131] Carlo Tomasi and Roberto Manduchi. **Bilateral Filtering for Gray and Color Images**. In *ICCV*, pages 839–846, 1998. (79, 82)

[132] Dijana Tralic, Ivan Zupancic, Sonja Grgic, and Mislav Grgic. **CoMoFoD -New Database for Copy-Move Forgery Detection**. pages 49–54, 09 2013. (xvi, 14, 42, 53)

[133] Li Wang and Dong-Chen He. **Texture Classification Using Texture Spectrum**. *Pattern Recogn.*, **23**(8):905–910, aug 1990. (25)

[134] Xiang Yang Wang, Li Xian Jiao, Xue Bing Wang, Hong Ying Yang, and Pan Pan Niu. **A new keypoint-based copy-move forgery detection for color image**. *Applied Intelligence*, **48**(10):3630–3652, Oct 2018. (12, 13, 14, 17)

[135] Nor Bakiah Abd Warif, Ainuddin Wahid Abdul Wahab, Mohd Yamani Idna Idris, Rosli Salleh, and Fazidah Othman. **SIFT-symmetry: a robust detection method for copy-move forgery with reflection attack**. *Journal of Visual Communication and Image Representation*, **46**:219–232, 2017. (11, 13, 17, 20)

[136] Wei Wang, J. Dong, and T. Tan. **Effective image splicing detection based on image chroma**. In *2009 16th IEEE International Conference on Image Processing (ICIP)*, pages 1257–1260, Nov 2009. (78, 80)

[137] Alexander Wong, David Clausi, and Paul Fieguth. **Adaptive Monte Carlo Retinex Method for Illumination and Reflectance Separation and Color Image Enhancement**. pages 108–115, 05 2009. (81)

[138] Shenghong Li Xudong Zhao, Shilin Wang and Jianhua Li. **Passive Image-Splicing Detection by a 2-D Noncausal Markov Mode**. *IEEE Transactions On Circuits And Systems For Video Technology*, **25**, 2015. (90)

[139] Fan Yang, Jingwei Li, Wei Lu, and Jian Weng. **Copy-move forgery detection based on hybrid features**. *Engineering Applications of Artificial Intelligence*, **59**:73–83, 2017. (13, 21)

[140] Liyang Yu, Qi Han, and Xiamu Niu. **Feature point-based copy-move forgery detection: covering the non-textured areas**. *Multimedia Tools and Applications*, **75**(2):1159–1176, 2016. (13, 21)

[141] Hao Yuan and Mikhail Atallah. **Running Max/Min Filters Using 1+o(1) Comparisons per Sample**. *IEEE transactions on pattern analysis and machine intelligence*, **33**:2544–2548, 08 2011. (82)

[142] RAO YUAN, NI JIANGQUN, and ZHAO HUIMIN. **Deep Learning Local Descriptor for Image Splicing Detection and Localization**. *IEEE Access*, **8**, 2020. (6, 24, 26, 90)

[143] Zhongping Zhang, Yixuan Zhang, Zheng Zhou, and Jiebo Luo. **Boundary-based Image Forgery Detection by Fast Shallow CNN**. pages 2658–2663, 08 2018. (6, 23, 24, 27, 78, 90)

[144] Junliu Zhong, Yanfen Gan, Janson Young, Lian Huang, and Peiyu Lin. **A new block-based method for copy move forgery detection under image geometric transforms**. *Multimedia Tools and Applications*, **76**(13):14887–14903, 2017. (13, 16)

[145] Haoyu Zhou, Yue Shen, Xinghui Zhu, Bo Liu, Zigang Fu, and Na Fan. **Digital image modification detection using color information and its histograms**. *Forensic science international*, **266**:379–388, 2016. (13, 15)

[146] Ye Zhu, Shen Xuanjing, and Haipeng Chen. **Copy-move forgery detection based on scaled ORB**. *Multimedia Tools and Applications*, **75**:3221–3233, 01 2016. (19, 50)

Application of Image Blobs and Illumination Component in Image Tamper Detection Techniques

by Patrick Niyishaka

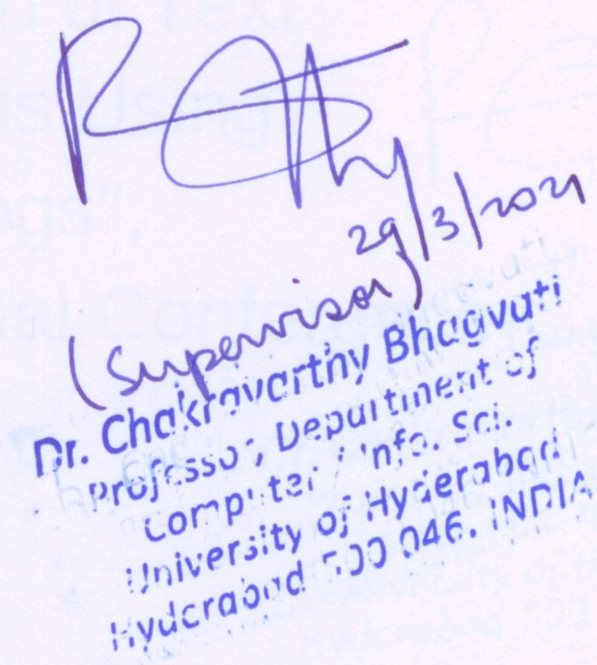
Submission date: 27-Mar-2021 02:01PM (UTC+0530)

Submission ID: 1543720051

File name: Patrick_final.pdf (55.33M)

Word count: 2240

Character count: 5515



Application of Image Blobs and Illumination Component in Image Tamper Detection Techniques

ORIGINALITY REPORT

6%	4%	2%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.iis.sinica.edu.tw Internet Source	2%
2	Submitted to Heriot-Watt University Student Paper	1%
3	uec.repo.nii.ac.jp Internet Source	1%
4	Submitted to Columbus State University Student Paper	1%
5	Kuthirummal, S.. "Fourier domain representation of planar curves for recognition in multiple views", Pattern Recognition, 200404 Publication	1%
6	Abu Saleh Md Tayeen, Saleem Masadeh, Abderrahmen Mtibaa, Satyajayant Misra, Moumita Choudhury. "Comparison of Text Mining Feature Extraction Methods Using Moderated vs Non-Moderated Blogs", Proceedings of the 9th International Conference	1%

29/3/2024
(Supervisor)

Dr. Chakravarthy Bhagvati
Professor, Department of
Computer & Info. Sci.
University of Hyderabad
Hyderabad 500 046. INDIA.