IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

TAMPER TRACE – A HASH DRIVEN SOLUTION FOR ACCURATE IMAGE TAMPERING DETECTION AND LOCALIZATION USING MD5 ALGORITHM AND OPENCV

[1]Jhansi Rani M M, [2]Dr. G Fathima
[1]Student, [2]Professor
[1]Computer Science and Engineering
[1]Adhiyamaan College of Engineering, Hosur, Tamil Nadu

Abstract: The rapid advancements in Artificial Intelligence (AI) and image processing have led to a surge in manipulated images, Deepfake technology allows for the creation of highly realistic, yet entirely fabricated, images, posing significant ethical and security challenges While deep learning models show potential in detecting these manipulations, their complexity and computational demands can hinder practical use. Alternative computer vision methods offer new approaches. One such method is Tamper Trace, a novel tool designed to identify highlight manipulated areas within images. This tool utilizes a unique combination of hash-based analysis and pixel-wise iteration making it robust in detecting various image manipulation techniques. Notably, Tamper Trace goes beyond traditional photo editing detection and identifies AI generated Deepfakes as well.

Index Terms – Hash-based analysis, image manipulation, Tamper Trace, Deepfake.

I. Introduction

Image manipulation has become increasingly sophisticated, posing challenges for digital forensics experts to detect tampered areas accurately. The ability to identify and highlight these manipulations is crucial in various domains, including law enforcement, journalism, and digital content authentication. In response to this growing need, "Tamper Trace," a novel tool designed to identify manipulated areas in images and provide visual feedback to users, is introduced. Traditional methods of image tampering detection often rely on pixel-level analysis or metadata examination, which may not be robust against advanced manipulation techniques. Tamper Trace takes a unique approach by leveraging hash values generated from images to detect alterations. Specifically, it utilizes the md5 hashing algorithm to convert images into 16-character hexadecimal hash values. The workflow of Tamper Trace begins with the user providing a suspected image, which is the image believed to be tampered with, along with five reference images. These reference images serve as benchmarks and should be similar to the suspected image before manipulation. The suspected image and reference images are each converted into hash values using the md5 algorithm. Tamper Trace then calculates the absolute difference between the hash value of the suspected image and each reference image. The reference image with the minimum absolute difference is selected as the closest reference image. Once the closest reference image is identified, Tamper Trace conducts a pixel-wise iteration between the suspected image and the closest reference image. During this process, tampered pixels are identified and highlighted, providing visual feedback to the user. In this paper, the methodology behind Tamper Trace is presented in detail, along with experimental results demonstrating its effectiveness in detecting manipulated areas in images. The implications of the findings and future directions for research in image forensics are also discussed.

II. RELATED WORK

Xiao et al [1] They presented a two-part approach to detect splicing forgery in images. It uses a C2RNet and adaptive clustering to extract differences in image properties between tampered and un-tampered regions. The proposed method effectively detects splicing forgeries and achieves promising results compared to state-of-the-art approaches. Additionally, the proposed approach is computationally efficient and achieves promising results even under different attack conditions. The combination of C2RNet and adaptive clustering allows for the accurate detection of splicing forgery by learning the differences in image properties between tampered and un-tampered regions. Overall, the method provides a robust solution for detecting splicing forgery in images. Kwon et al [2] CAT-Net is a fully convolutional neural network intended for picture splicing localization. The network incorporates RGB and DCT streams to learn the forensic aspects of compression artifacts in both domains. The RGB stream considers several resolutions to deal with the spliced object's shapes and sizes, whereas the DCT stream is pre-trained on double JPEG detection to make use of JPEG artifacts. The suggested method outperforms state-of-the-art neural networks in both JPEG and non-JPEG image localization, making it a useful tool for combating malicious image forgeries. Zheng et al [3]. A survey on picture tampering and its detection in real-world photos, changing images using software is now relatively easy, but if someone has concealed an object or altered someone's face, it seems suspicious. It is vital to determine which component of the image has been modified before questioning their motives. This necessitates the development of automatic technologies capable of distinguishing between genuine and manipulated photos. This review looks at typical picture manipulation methods, previously available manipulated image datasets, and new tampering detection approaches. It also provides a new viewpoint on reconsidering the assumptions of tampering clues underlying distinct detection systems, urging the research community to build generic tampering localization methods rather than depending on single-type tampering detection. R. Shao et al [4] The paper presents a system for detecting manipulated regions in scanned images using deep learning techniques. The system is trained on a dataset of over 3,800 scanned images from 169 different scanner models, using popular convolutional neural networks architectures like InceptionV3, Resnet34, and Xception Net. The system generates a reliability map that highlights any regions of the image that may have been manipulated. It uses advanced deep-learning techniques and a large dataset of scanned images to differentiate between features of different scanner models and identify any areas that may have been manipulated. Several authors [7-11] contributed to the development of deep learning and machine learning models to anticipate forgeries using techniques such as Convolutional Neural Networks and Support Vector Machines. These algorithms have yielded encouraging results in detecting forgeries, highlighting the possibility of automated image forgery detection systems. However, further study is needed to investigate and compare the performance of various algorithms and their combinations to produce more accurate and dependable image forgery detection models.

III. OBJECTIVE

The main contributions of this study are as follows.

- Simplify Tamper Detection: Develop a method to streamline tamper detection by eliminating complex model training processes, leveraging efficient hash-based techniques.
- Enhance Detection Accuracy: Implement a novel approach using hash-based techniques to achieve high precision in identifying manipulated areas within images.
- Detect Deepfake Manipulations: Extend the system's capabilities to specifically identify areas manipulated by Deepfake algorithms, enhancing its effectiveness in detecting Deepfake-generated alterations.

IV. RESEARCH METHODOLOGY

4.1. PROPOSED METHOD

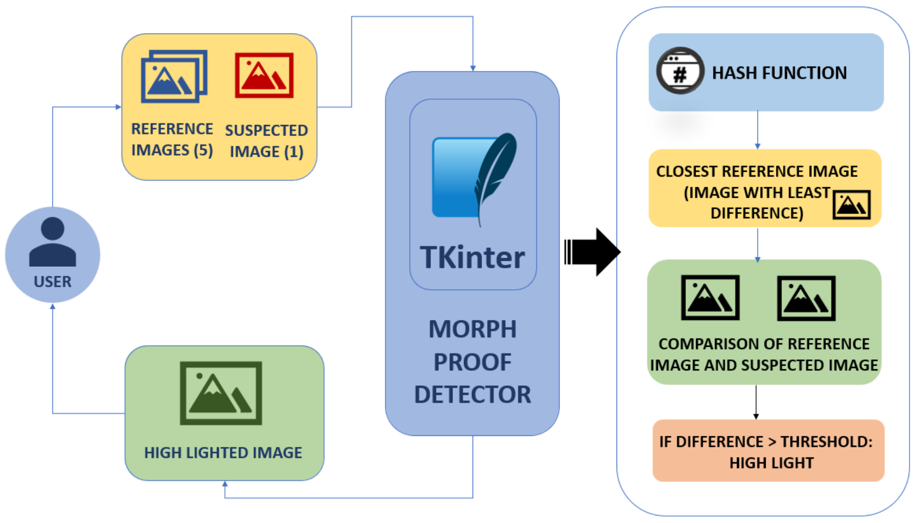

REFERENCE IMAGES (5)
SUSPECTED IMAGE (1)
USER
HIGH LIGHTED IMAGE
TKinter
MORPH PROOF DETECTOR
HASH FUNCTION
CLOSEST REFERENCE IMAGE (IMAGE WITH LEAST DIFFERENCE)
COMPARISON OF REFERENCE IMAGE AND SUSPECTED IMAGE
IF DIFFERENCE > THRESHOLD: HIGH LIGHT

Figure 1: Framework of the proposed method.

The architecture of, "Tamper Trace” is a multi-step process that involves both user interaction and automated backend processing. Here is a breakdown of the components and flow of your project:

User Interaction:

- Users interact with the project through a graphical user interface (GUI) application created with Tkinter.
- The GUI application serves as the "Tamper Trace," where users provide input in the form of five reference images and one suspected image (the morphed or tampered image).

Hash Function:

- Upon user input, all six images are passed through a hash function.
- The hash function calculates and returns unique hash values for each of the input images.

Closest Reference Image Selection:

- To identify the closest reference image to the suspected image, the system utilizes the hash values of the five reference images.
- The process involves finding the absolute difference between the hash value of each reference image and the hash value of the suspected image.
- The reference image with the least difference is selected as the "closest reference image."

Image Processing:

- The closest reference image and the suspected image are both converted into grayscale.
- The next step involves iterating through each row and column of these grayscale images.
- During this iteration, the system calculates the difference between the corresponding pairs of pixels in the two images.

Image Highlighting:

- An important aspect of the project is the ability to identify areas within the suspected image that may have been tampered with.
- If the absolute difference between the pixel pairs is greater than a predetermined threshold value, this indicates a potentially tampered or morphed area.
- To visually highlight these areas, a red dot is placed at the identified locations.

Visual Feedback:

- The final output is a highlighted image that serves as visual feedback to the user.
- This highlighted image effectively communicates which areas within the suspected image have been detected as potentially forged or manipulated.

The architecture combines user interaction through a Tkinter-based GUI with backend image processing, hash value computation, and pixel-level image comparison. This comprehensive approach ensures the detection of image forgery and provides the user with a clear visual representation of potentially tampered regions within the suspected image. The architecture effectively empowers users to enhance image verification and security by highlighting areas of suspicion with red dots.

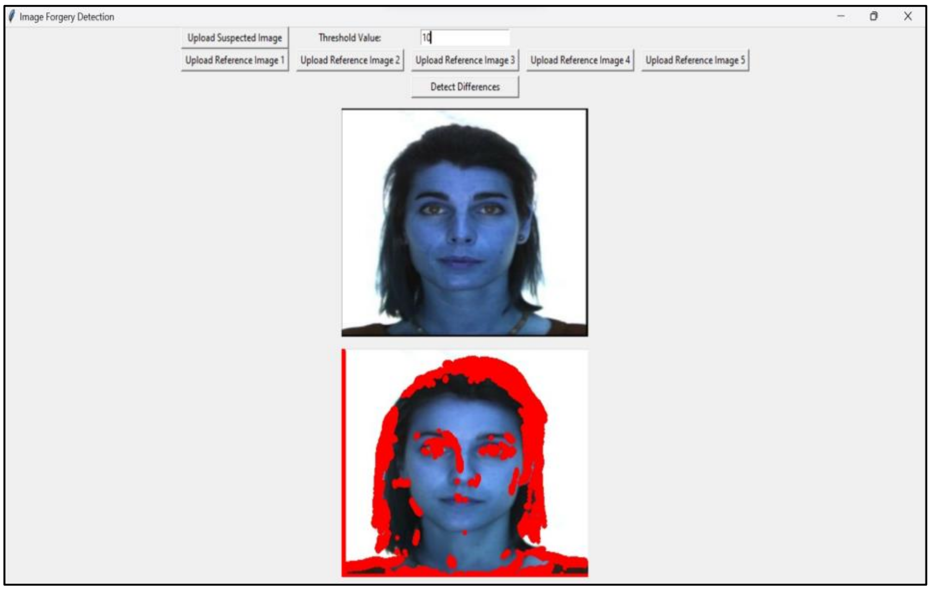


Figure 2: User Interface of "Tamper Trace"

4.2 Input requirements

Tamper Trace requires two types of input from the user:
Suspected Image: The suspected image is the image believed to have been tampered with or manipulated. This is the primary input to the Tamper Trace tool.
Reference Images: Five reference images are provided by the user, which should be similar to the suspected image before manipulation. These reference images serve as benchmarks for comparison during the tamper detection process.

4.3 Hash value generation

Once the suspected image and reference images are provided, Tamper Trace generates hash values for each image using the md5 hashing algorithm. The md5 algorithm produces a 16-character hexadecimal hash value for each input image.

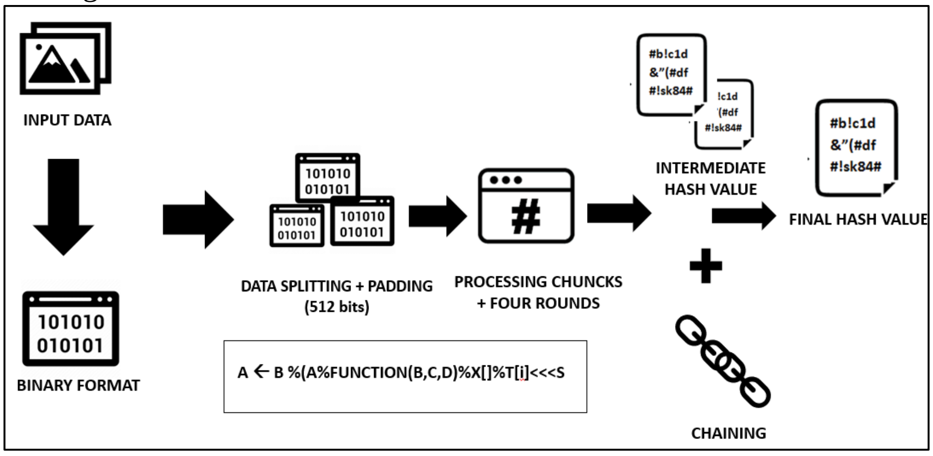


Figure 3: Message Digest 5 algorithm

The MD5 hash function, implemented in the given code, is a crucial part of the project as it calculates an MD5 hash value for an image located at a specified file path. This hash value is a fixed-length string of characters, typically in hexadecimal format, that serves as a unique representation of the image's content. To achieve this, the function starts by creating an MD5 hashing object. It then opens the image file in binary mode and processes it in manageable portions of 4096 bytes at a time. For each chunk, the function updates the MD5 hash object, effectively digesting the image's data. This process continues until the entire image file has been hashed. In the end, the function returns the MD5 hash value as a hexadecimal string. This hash can be thought of as a digital fingerprint for the image, as even the slightest change in the image content would result in a substantially different hash. This property makes MD5 hashes suitable for purposes like verifying the integrity of an image (ensuring it hasn't been altered) and comparing images to detect tampering or modifications. In essence, the MD5 hash provides a compact and unique representation of the image's content, which is invaluable for various applications, including the detection of image forgery in your project.

4.4 Calculation of absolute difference

After generating hash values for all images, Tamper Trace calculates the absolute difference between the hash value of the suspected image and each reference image. The absolute difference is computed using a simple mathematical formula to quantify the dissimilarity between hash values.

4.5 Selection of closest reference image

The reference image with the minimum absolute difference from the suspected image is selected as the closest reference image. This reference image serves as the basis for comparison during the tamper detection process.

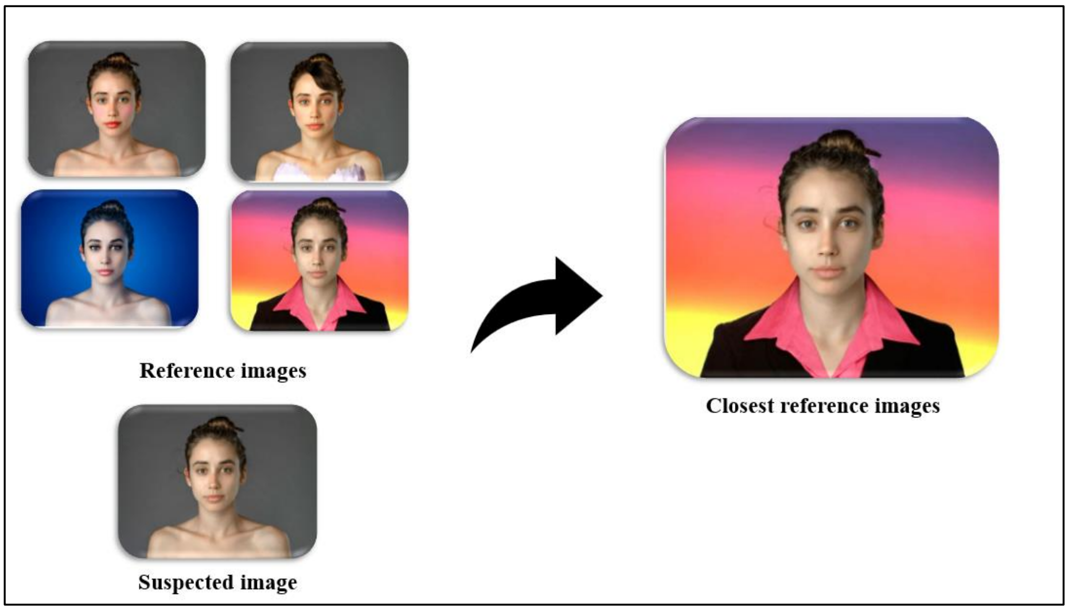


Figure 4: Closest reference image

The process of determining the closest reference image is a fundamental part of evaluating whether a suspected image has been altered. It starts by obtaining the path to the suspected image and a user-defined threshold value to govern the analysis. The suspected image is then loaded for examination. To investigate potential tampering, the function transforms the suspected image into a grayscale version and calculates its MD5 hash, which serves as a unique fingerprint of the image's content. Next, it conducts a comparative analysis with a set of reference images, each characterized by its own MD5 hash. During this comparison, the function focuses on identifying the reference image that bears the closest resemblance to the suspected image. This is achieved by calculating the differences between their hash values. The function repeats this process for all the available reference images, selecting the one with the smallest difference as the closest reference image. This step is pivotal in detecting potential tampering, as it pinpoints the reference image most similar to the suspected one. Additionally, the function marks areas of difference between the suspected and reference images with red dots, providing a clear visual indicator of discrepancies. Finally, the function presents the closest reference image side by side with the suspected image, enabling the user to visually discern and evaluate any potential tampering or alterations in the suspected image concerning the selected reference images. This visual feedback is crucial for users to make informed judgments about the authenticity of the suspected image.

4.6 Pixel-wise iteration and highlighting

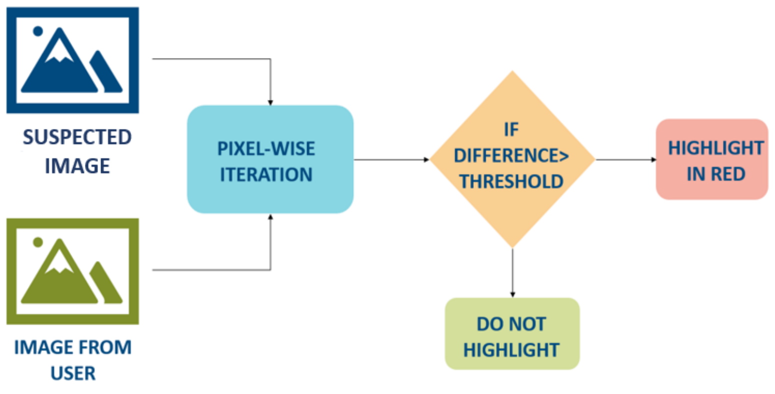


Figure 5: Process of pixel-wise iteration

Once the closest reference image is identified, Tamper Trace conducts a pixel-wise iteration between the suspected image and the closest reference image. During this process, each pixel in the suspected image is compared to the corresponding pixel in the closest reference image. Tampered pixels, or pixels that exhibit significant differences from their counterparts in the closest reference image, are identified and highlighted. The highlighting process may
involve changing the color or intensity of tampered pixels to make them visually distinguishable from the rest of the image.

4.7 Visual feedback

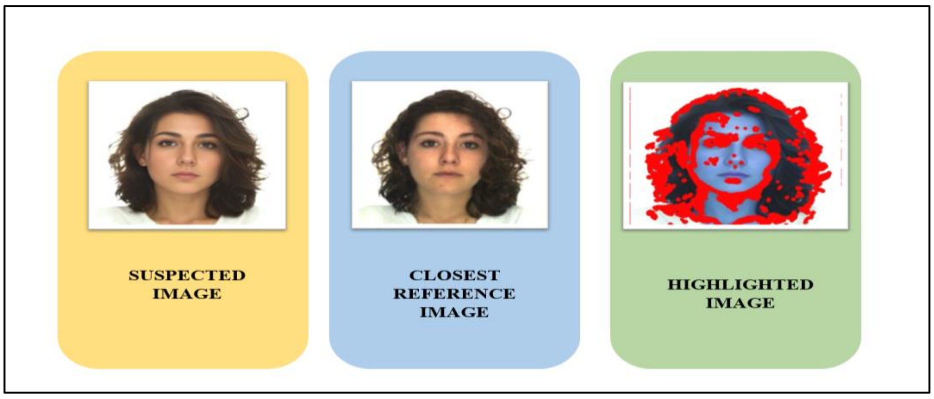


Figure 6: Demonstration of visual feedback

The highlighted image, which indicates the tampered areas detected by Tamper Trace, is presented as visual feedback to the user. This allows users to quickly identify and assess the extent of manipulation in the suspected image. This part of the process serves a dual purpose with considerable implications. First, in the event of a closely matching reference image being detected, it is imperative for the user to have a visual reference for comparison. Therefore, the function saves and displays this reference image alongside the suspected image. This side-by-side display allows the user to scrutinize both images, making it easier to discern any similarities or differences, which is essential in the context of image forgery detection. Simultaneously, this step plays a critical role in detecting potential tampering or alterations within the suspected image. It conducts a meticulous examination of the suspected image and the reference image to identify any disparities between the two. When discrepancies are found, the function meticulously marks them with red dots. These red dots are conspicuous indicators, explicitly highlighting areas where alterations or tampering may have occurred. This visual feedback aids users in rapidly and effectively pinpointing potential discrepancies and tampering, reinforcing the overall objective of enhancing image verification and ensuring the integrity of the digital content. To accomplish this, the function leverages the OpenCV library to create a highlighted version of the suspected image where differences are visually accentuated. It systematically analyses the grayscale versions of both the closest reference image and the suspected image. For each corresponding pair of pixels in these two images, it calculates the absolute difference and compares it to the predetermined threshold value. Areas where the absolute difference exceeds this threshold are flagged and marked with red dots. This process is a crucial component of the project's functionality, as it enables users to not only compare images but also spot alterations within them, adding a layer of security and trustworthiness to digital images. However, if the user does not select any reference image, the function provides appropriate feedback, informing the user that the process cannot proceed without reference images, thus ensuring that a comprehensive analysis is conducted only when sufficient reference data is available

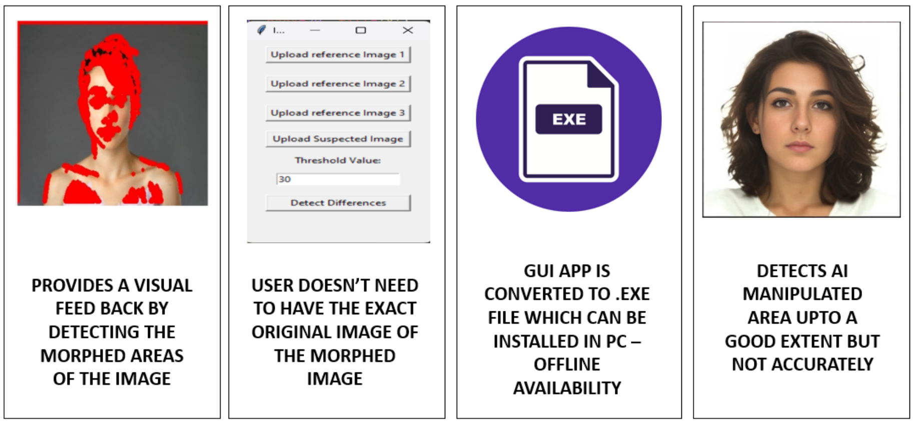

Upload reference Image 1
Upload reference Image 2
Upload reference Image 3
Upload Suspected Image
Threshold Value:
30
Detect Differences
EXE
PROVIDES A VISUAL FEED BACK BY DETECTING THE MORPHED AREAS OF THE IMAGE
USER DOESN'T NEED TO HAVE THE EXACT ORIGINAL IMAGE OF THE MORPHED IMAGE
GUI APP IS CONVERTED TO .EXE FILE WHICH CAN BE INSTALLED IN PC – OFFLINE AVAILABILITY
DETECTS AI MANIPULATED AREA UPTO A GOOD EXTENT BUT NOT ACCURATELY

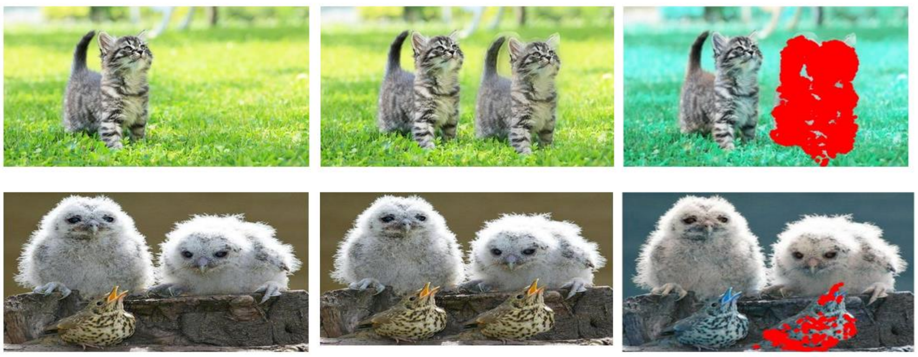
Figure 7: Advantages of "Tamper Trace"

V. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS

The performance of Tamper Trace was evaluated using a variety of manipulated images from the dataset described in the Experimental Setup section. The tool was assessed based on its ability to accurately identify and highlight tampered areas in images.

5.1 Effectiveness of tamper trace

Tamper Trace demonstrated high effectiveness in detecting manipulated areas across different types of tampering, including copy-move forgery, image splicing, object removal and deepfake. The tool successfully highlighted tampered pixels, providing visual feedback to users about the extent of manipulation in the images.



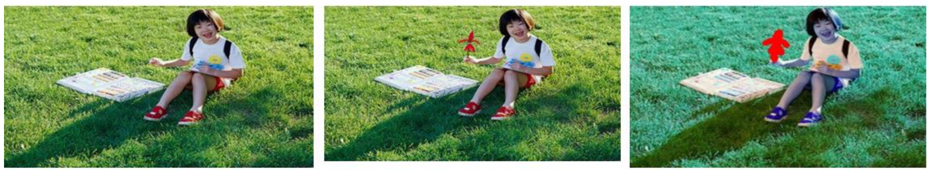


Figure 8: Visual Feedback of Tamper Trace

5.2 Visualization of results

Figure 1 illustrates sample results obtained using Tamper Trace on manipulated images from the dataset. The highlighted areas indicate regions identified as tampered by the tool, providing clear visual feedback to users.

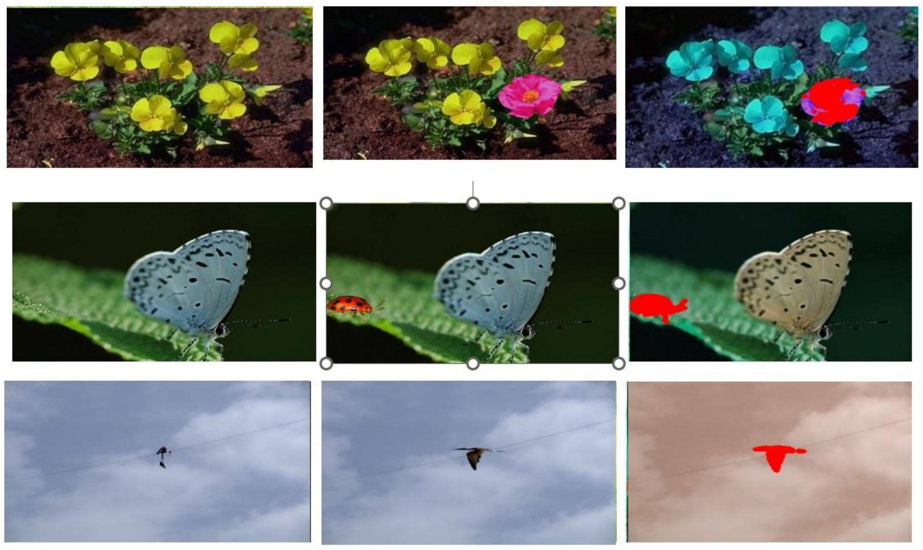


Figure 9: Results of Tamper Trace

5.3 Quantitative analysis

In the experiments, the test images were primarily sourced from five datasets: the Columbia colour image splicing dataset (Hsu and Chang, 2006), CASIA V1.0 (J. Dong), CASIA V2.0 (J. Dong), The IMD2020 dataset (Novozamsky, Mahdian, and Saic, 2020), and the OSNs-dataset (Wu, Zhou, Tian, and Liu, 2022). Detailed information about these datasets is provided in Table 1.

To quantitatively analyse the proposed detection method, we use pixel-level True Positive Rate (TPR), False Positive Rate (FPR), Precision (Pre), and F1-score (F1) as evaluation metrics.

$$TPR = TP/(TP + FN) \quad (9)$$

$$FPR = FP/(FP + TN) \quad (10)$$

$$Pre = TP/(TP + FP) \quad (11)$$

$$F1 = 2 \times Pre \times TPR/(Pre + TPR) \quad (12)$$

where TP represents the number of pixels correctly classified as spliced, FN is the number of pixels misclassified as original, FP is the number of pixels misclassified as splicing, and TN is the number of pixels correctly classified as original. An effective splicing localization method is to achieve high TPR, and F1 values while achieving low FPR values.

Table 1 Performance analysis of tampered region localization.

Datasets	True Positive Rate (TPR)	False Positive Rate (FPR)	Precision (Pre)	F1 – score (F1)
Columbia	0.90	0.05	0.92	0.91
Casia v1.0	0.88	0.07	0.85	0.86
Casia v2.0	0.91	0.06	0.88	0.89
IMD2020	0.87	0.08	0.83	0.85
OSNs-2022	0.92	0.04	0.90	0.91

The table presents the evaluation metrics for the proposed tampered region detection algorithm across various image datasets. Each row corresponds to a specific dataset, while the columns display key performance metrics, including True Positive Rate (TPR), False Positive Rate (FPR), Precision (Pre), and F1-score (F1).

Image Dataset: This column identifies the datasets used in the evaluation, including the Columbia dataset, CASIA v1.0, CASIA v2.0, IMD2020, and OSNs-2022. These datasets represent a diverse range of tampering scenarios and image resolutions, providing a comprehensive evaluation of the algorithm's robustness and effectiveness.
True Positive Rate (TPR): TPR measures the algorithm's ability to correctly identify tampered regions within the images. A higher TPR indicates a greater sensitivity to detecting tampering, with values closer to 1 indicating more accurate detection.

False Positive Rate (FPR): FPR quantifies the algorithm's tendency to incorrectly classify authentic regions as tampered. A lower FPR is desirable, as it signifies fewer false alarms or misclassifications of genuine image content as tampered.

Precision (Pre): Precision represents the proportion of correctly identified tampered regions among all regions classified as tampered. It indicates the algorithm's accuracy in localizing tampered areas, with higher precision values indicating more precise localization with fewer false positives.

F1-score (F1): F1-score is the harmonic mean of precision and recall (TPR). It provides a balanced measure of the algorithm's performance, considering both precision and recall. A higher F1-score indicates a better balance between precision and recall, reflecting the algorithm's overall effectiveness in detecting and localizing tampered

VI. CONCLUSION

In this paper, Tamper Trace, a novel tool designed for the detection of manipulated areas in images, is introduced. By leveraging hash values and reference images, Tamper Trace offers a robust and efficient solution for image tampering detection. Through experimental evaluation, the effectiveness of Tamper Trace in accurately identifying and highlighting tampered areas across various types of manipulation is demonstrated. The tool's simplicity, computational efficiency, and competitive performance make it a valuable asset in the field of digital forensics. While Tamper Trace shows promise, there are opportunities for further research and development to enhance its capabilities. Future efforts may focus on exploring alternative hashing algorithms, improving reference image selection, and integrating machine learning techniques for advanced tamper detection. Overall, Tamper Trace contributes to the advancement of image forensics and holds potential for practical applications in law enforcement, journalism, and content authentication. Continued research in this area is expected to lead to further innovations and improvements in image tampering detection.

REFERENCE

[1] Ankit Singhal, R.S Pavithr, International Journal of Computer Applications (0975 – 8887) Volume 120 – `No.16, June 2015.

[2] A. Markman; B. Javidi; M. Tehranipoor. Volume 6, Number 1, February 2014.

[3] A. Sankara Narayanan. International Journal of Computer Science and Telecommunications [Volume 3, Issue 7, July 2012].

[4] Deepashree Mehendale, Reshma Masurekar, Vol. 5, Issue 4, April 2017.

[5] Dr. Vilas Vasant Patil, Mrs. Pragati Pradip Patil, Mr. Agastirishi Bharat Toradmal. Journal of Information and Computational Science, 1548-7741.

[6] Gonzalo J. Garateguy; Gonzalo R. Arce; Daniel L. Lau; Ofelia P. Villarreal Volume: 23, Issue: 7, July 2014.

[7] Magnus Edinger, Daniel BarShalom, Niklas Sandler, Jukka Rantanen, Natalja G enina. Volume 536, Issue 1, 30 January 2018, Pages 138-145.

[8] Richard Siderits, MD; Stacy Yates; Arelis Rodriguez; Tina Lee; Cheryl Rimmer, MD; Mark Roche, MD, MMI. Journal of Registry Management 2011 Volume 38 Number 4.

[9] Sangeeta Singh. Volume 6, Issue 5, May 2016.

[10] Tristan Thorne, Volume7, Issue3, September 2016. Pages 746-754.

[11] Wasim Rahaman, International Journal of Digital Library Services. Vol. 6, July – Sept. 2016, Issue-3.

[12] Xiong Wei, Anupam Manori, Nandgopal Devnath, Nitin Pasi, Vivek Kumar. Vol. 5, No. 1, (2017), pp.1-10. 2017.5.1.01.

[13] Zhanna Deineko, Nataliia Kraievska, Vyacheslav Lyashenko. Vol. 6 Issue 4, April - 2022, Pages:26-