**Interactive Toolkit for Digital Image Processing**

Project submitted to the

SRM University – AP, Andhra Pradesh

for the partial fulfillment of the requirements to award the degree of

**Bachelor of Technology**

In

**Computer Science and Engineering**

**School of Engineering and Sciences**

Submitted by

**Gunuputi Rohith - AP22110010749**

**Bandaru Sandeep - AP22110010763**

**Pilla Mohith Ram – AP22110010789**

**Nalluri Nikhil Siddharth – AP22110010822**

**A picture containing text

Description automatically generated**

Under the Guidance of

**Dr. Suresh Kumar Kanaparthi**

**SRM University–AP**

**Neerukonda, Mangalagiri, Guntur**

**Andhra Pradesh – 522 240**

**[April, 2025]**

**Certificate**

Date: 28-Apr-25

This is to certify that the work present in this Project entitled “**Image Processing Application**” has been carried out by **Gunuputi Rohith** under my/our supervision. The work is genuine, original, and suitable for submission to the SRM University – AP for the award of Bachelor of Technology/Master of Technology in **School of Engineering and Sciences**.

**Supervisor**

(Signature)

Prof. / Dr. Suresh Kumar Kanaparthi

Assistance Professor,

Department of Computer Science.

**Acknowledgement**

I would like to express my special thanks to my teacher **Dr. Suresh Kumar Kanaparthi** for the knowledge, time and efforts he provided throughout the course of this project. His useful advice and suggestions were really helpful to me during the project’s completion.

I would like to thank my parents and close friends, who were always with me from imagination to the implementation of this project. Without their cooperation, this project would not have been completed. The journey of making this project has been beautiful, as well as knowledgeable for me and I have learned a lot from it. I did this project not only to get marks but also to improve my technical knowledge. During this project, I acquired many valuable skills, and I hope that these will help me in my future career. Once again, thanks to everyone who was involved with this project from beginning to end.

**Table Of Contents**

**Certificate ……………………………………………………………………..2**

**Acknowledgement …………………………………………………………...3**

**Abstract ………………………………………………………………………..5**

**Introduction …………………………………………………………………..6**

**Background …………………………………………………………………...7**

**Problem statement …………………………………………………………. 8**

**Architecture …………………………………………………………………. 9**

**Methodology ………………………………………………………………...11**

**Source Code …………………………………………………………………13**

**Results ……………………………………………………………………….17**

**Discussion and Conclusion ……………………………………………….21**

**Future work …………………………………………………………………23**

**References …………………………………………………………………...25**

**Abstract**

Image forgery has become increasingly common with the widespread availability of digital editing tools, leading to challenges in ensuring the authenticity of visual content. This project presents a simple and effective image forgery detection approach using classical image processing techniques that highlight inconsistencies introduced during tampering.

The increasing accessibility of sophisticated image editing tools has led to a surge in digital image manipulations, posing a significant challenge to the authenticity and credibility of visual media. From fake news to altered legal evidence, forged images can have serious consequences, making image forgery detection a vital aspect of modern digital forensics. This project focuses on identifying such manipulations using a classical image processing pipeline that analyzes visual inconsistencies often introduced during tampering.

The proposed system begins with **Histogram Equalization**, which enhances the overall contrast of the image, allowing hidden or subtle irregularities to become more visible. It then applies **Edge Detection** using three different techniques—**Sobel**, **Prewitt**, and **Laplacian filters**—to capture sudden changes in pixel intensity that often indicate unnatural boundaries or copied regions. These edge maps help identify possible areas where splicing or cloning may have occurred.

Following edge detection, **Otsu’s Thresholding** is employed to segment the image based on pixel intensity distribution. This automated method determines the optimal threshold to distinguish between foreground and background, helping isolate regions of interest that might have been tampered with. To further refine the results, the system incorporates **Median Filtering** and **Adaptive Median Filtering**, which suppress noise while preserving important edge details, making the detection more reliable and visually interpretable.

Furthermore, the modular structure of the system allows for future expansion, including integration with machine learning or AI-based models to improve accuracy and adapt to evolving forgery techniques. By leveraging foundational image processing principles, this project contributes to the broader goal of developing transparent and trustworthy tools for digital content verification and media integrity

**Introduction**

In the modern digital era, images have become a critical medium for communication, evidence, and information sharing. However, the ease of editing and manipulating digital images has led to a growing concern about image authenticity. Image forgery, the act of altering a digital image to mislead or deceive, is commonly seen across areas such as media, social networking, legal evidence, and even scientific publications. Detecting such tampered images has become crucial to maintain trust and integrity in various fields.

Traditional image forgery detection techniques often rely on complex machine learning models or deep learning frameworks, which require large amounts of training data, computational power, and specialized expertise. In contrast, our project focuses on building a **simple, efficient, and interpretable system** for image forgery detection using **only Digital Image Processing (DIP) techniques** without the need for any artificial intelligence or machine learning methods.

This project aims to identify suspected forged regions within an image by leveraging classical image processing operations such as **grayscale conversion, histogram equalization, edge detection (Sobel, Prewitt, Laplacian filters), Otsu’s thresholding, and median-based noise filtering**. The processed image is then analyzed to extract features like **edge density**, **threshold density**, and **contour count**, which help in classifying the image as **forged** or **authentic**. Additionally, the system visually highlights suspicious regions for better understanding and verification by users.

Through this project, we demonstrate that even without sophisticated AI models, it is possible to create an effective and accessible solution for forgery detection that can be easily used in real-world scenarios, especially where quick offline analysis is needed.

**Background**

Digital images play a crucial role in modern communication, documentation, media, and legal processes. However, with the widespread availability of powerful editing tools, altering and manipulating images has become easier than ever before. Techniques such as cloning, splicing, retouching, and copy-move attacks can modify images in subtle or significant ways, often leaving minimal visible traces to the human eye.

Historically, detecting image forgery relied heavily on manual inspection or the expertise of forensic analysts. Over time, automated techniques using statistical methods, pattern recognition, and machine learning have been developed to assist in forgery detection. While machine learning and deep learning approaches offer high accuracy, they require large annotated datasets, significant computational resources, and can sometimes act as black boxes with limited interpretability.

In many real-world cases, a lightweight, fast, and understandable solution is preferred, especially when dealing with simple forgeries or when computational resources are limited. Digital Image Processing (DIP) techniques offer a practical alternative by focusing directly on the visual and structural properties of an image, such as texture inconsistencies, unnatural edges, and abrupt intensity transitions.

Our project builds upon this foundation by utilizing classical DIP methods like grayscale conversion, histogram equalization, edge detection, thresholding, and filtering to expose suspicious regions. By extracting and analyzing basic features such as edge density and threshold density, the system can make a reliable and explainable decision about the authenticity of an image, without relying on AI models or complex training.

This approach ensures that the solution remains simple, transparent, and accessible, while still providing valuable assistance in identifying potential forgeries in digital images.

**Problem statement**

With the increasing use of digital images in media, legal documentation, research, and social platforms, ensuring the authenticity of these images has become a major concern. Image forgery — the process of manipulating images to mislead or falsify information — is a growing threat that can have serious consequences, including the spread of misinformation, fraudulent activities, and compromised evidence in legal scenarios.

Although advanced machine learning and deep learning methods exist for detecting forgery, they require large amounts of labeled data, high computational resources, and expert-level tuning, making them less accessible for simple or offline use cases. There remains a significant gap for a **lightweight, fast, and interpretable forgery detection system** that does not depend on artificial intelligence or complex models.

This project addresses the problem of detecting forged regions in digital images using **only Digital Image Processing (DIP) techniques**. By applying traditional operations like edge detection, thresholding, and filtering, and by extracting features such as edge density and threshold density, the system aims to reliably detect and highlight suspected tampered regions without the need for any learning-based models.

**Thus, the problem can be framed as:**

"How can we design a simple, efficient, and interpretable system to detect image forgeries using only classical digital image processing techniques, without relying on machine learning or deep learning frameworks?"

**Architecture:**

The architecture of the Image Forgery Detection system is designed to follow a simple yet effective sequential pipeline, utilizing only classical Digital Image Processing (DIP) techniques to analyze and classify an image as real or forged. Each stage in the pipeline focuses on preparing the image, enhancing important features, and extracting meaningful information to assist in forgery detection.

The architecture consists of the following major components:

**1. Input Image**

* The system takes a digital image (in formats like PNG, JPG, BMP) as input.
* The uploaded image is assumed to be either a real or potentially forged image.

**2. Preprocessing**

* **Grayscale Conversion:** The input image is converted into a grayscale image to remove color information, reduce complexity, and focus on intensity patterns.
* **Histogram Equalization:** Enhances the contrast of the grayscale image, making hidden tampering signs more visible.

**3. Edge Detection**

* **Sobel, Prewitt, and Laplacian Filters** are applied to detect edges and boundaries within the image.
* Tampered regions often show unnatural or broken edges, which can be captured during this phase.

**4. Thresholding**

* **Otsu’s Thresholding** is used to automatically separate suspected tampered regions (foreground) from the background.
* Converts the image into a binary form where white pixels represent suspicious regions.

**5. Noise Filtering**

* **Median Filtering:** Removes random noise while preserving important edges.
* **Adaptive Median Thresholding:** Further refines regions, especially under varying lighting conditions, making suspected zones cleaner.

**6. Feature Extraction**

* **Edge Density:** Measures the overall edge content per pixel.
* **Threshold Density:** Calculates the proportion of white pixels (suspected regions) in the thresholded image.
* **Suspicious Region Count:** Counts the number of significant contours detected.

These numerical features are critical for making the final forgery decision.

**7. Forgery Decision Logic**

* Based on thresholds for edge density, threshold density, and suspicious contour count:
  + If edge density > 0.2, threshold density > 0.5, or more than 5 suspicious regions are detected, the image is classified as **forged**.
  + Otherwise, the image is classified as **authentic**.

**8. Output**

* The system displays:
  + Visual results at each stage (grayscale image, edge-detected image, thresholded image, filtered image, and highlighted regions).
  + Final decision: **Forgery Detected** or **No Forgery Detected**.
  + Number of suspicious regions detected.

**Methodology**

The project "Image Forgery Detection Using Digital Image Processing" was developed following a structured step-by-step pipeline. The methodology focuses on detecting suspicious regions in an image using traditional image processing techniques without relying on Machine Learning or AI models

**1. Preprocessing the Image**

* **Objective**: Prepare the input image for analysis by enhancing important features.
* **Techniques Used**:
  + **Grayscale Conversion**: The color image is converted into grayscale to simplify processing and remove unnecessary color information.
  + **Histogram Equalization**: Enhances contrast in the grayscale image, making any subtle tampering signs more visible.

**2. Edge Detection**

* **Objective**: Identify abrupt changes in intensity that may indicate tampered boundaries.
* **Techniques Used**:
  + **Sobel Filter**: Detects horizontal and vertical edges separately.
  + **Prewitt Filter**: Captures vertical and horizontal gradients.
  + **Laplacian Filter**: Detects fine edges and sharp transitions.
* **Combination**: The results of all edge detectors are combined to strengthen suspicious boundary detection.

**3. Thresholding**

* **Objective**: Segment the image to highlight important (possibly tampered) regions.
* **Techniques Used**:
  + **Otsu’s Thresholding**: An automatic thresholding method that separates foreground (suspected regions) from the background, producing a binary image.

**4. Noise Filtering**

* **Objective**: Remove random noise and enhance meaningful regions for better analysis.
* **Techniques Used**:
  + **Median Filter**: Smoothens the image while preserving edges.
  + **Adaptive Median Thresholding**: Adjusts locally to varying lighting conditions, ensuring better noise removal.

**5. Feature Extraction**

* **Objective**: Quantify the suspiciousness of the image for final classification.
* **Features Calculated**:
  + **Edge Density**: Total edge strength normalized by the image size.
  + **Threshold Density**: Proportion of white pixels in the thresholded image.
  + **Suspicious Contour Count**: Number of large suspicious regions identified after filtering.

**6. Forgery Decision Logic**

* **Objective**: Classify the image as forged or real based on extracted features.
* **Decision Rules**:
  + If **Edge Density > 0.2** and **Threshold Density > 0.5**, or if **Suspicious Contour Count > 5**, the image is classified as **forged**.
  + Otherwise, it is classified as **real**.

**7. Visualization of Results**

* **Objective**: Display each stage’s output for better understanding and verification.
* **Stages Shown**:
  + Original Image
  + Grayscale Image
  + Histogram Equalized Image
  + Edge Detection Output
  + Thresholded Image
  + Filtered Image, Final Suspected Regions Highlighted

**Source Code**

import cv2

import numpy as np

import matplotlib.pyplot as plt

def preprocess\_image(img\_path):

"""Loads the image, converts to grayscale, and applies histogram equalization."""

image = cv2.imread(img\_path)

if image is None:

raise FileNotFoundError(f"Error: The image '{img\_path}' was not found or is not a valid image.")

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

equalized = cv2.equalizeHist(gray)

return image, gray, equalized

def detect\_edges(image):

"""Detects edges using Sobel, Prewitt, and Laplacian filters."""

sobelx = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=5)

sobely = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=5)

laplacian = cv2.Laplacian(image, cv2.CV\_64F)

prewittx = cv2.filter2D(image, -1, np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]]))

prewitty = cv2.filter2D(image, -1, np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]]))

combined\_edges = cv2.convertScaleAbs(sobelx) + cv2.convertScaleAbs(sobely) + \

cv2.convertScaleAbs(laplacian) + cv2.convertScaleAbs(prewittx) + \

cv2.convertScaleAbs(prewitty)

return combined\_edges

def threshold\_image(image):

"""Applies Otsu's thresholding for segmentation."""

\_, thresholded = cv2.threshold(image, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

return thresholded

def apply\_filters(image):

"""Applies Median and Adaptive Median Filters to reduce noise."""

median\_filtered = cv2.medianBlur(image, 5)

adaptive\_median = cv2.adaptiveThreshold(median\_filtered, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,

cv2.THRESH\_BINARY, 11, 2)

return adaptive\_median

def highlight\_suspected\_regions(original, mask):

"""Draws bounding boxes around suspected forgery regions."""

result = original.copy()

contours, \_ = cv2.findContours(mask, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

suspicious\_count = 0

for cnt in contours:

area = cv2.contourArea(cnt)

if area > 200: # ignore tiny noise

x, y, w, h = cv2.boundingRect(cnt)

cv2.rectangle(result, (x, y), (x+w, y+h), (0, 0, 255), 2)

suspicious\_count += 1

return result, suspicious\_count

def display\_results(images, titles):

"""Displays images in grid format."""

plt.figure(figsize=(15, 6))

for i in range(len(images)):

plt.subplot(2, 4, i+1)

if len(images[i].shape) == 2:

plt.imshow(images[i], cmap='gray')

else:

plt.imshow(cv2.cvtColor(images[i], cv2.COLOR\_BGR2RGB))

plt.title(titles[i])

plt.axis('off')

plt.tight\_layout()

plt.show()

def detect\_forgery(img\_path):

"""Runs the full pipeline to detect image forgery and classify it."""

original, gray, equalized = preprocess\_image(img\_path)

edges = detect\_edges(equalized)

thresholded = threshold\_image(equalized)

filtered = apply\_filters(thresholded)

# Feature Extraction

edge\_density = np.sum(edges) / edges.size

threshold\_density = np.sum(thresholded) / 255 / thresholded.size

# Highlight suspected zones

highlighted\_image, suspected\_count = highlight\_suspected\_regions(original, filtered)

# Tuned decision rule

if (edge\_density > 0.2 and threshold\_density > 0.5) or suspected\_count > 5:

forgery\_status = "⚠ Forgery Detected"

else:

forgery\_status = "✅ No Forgery Detected"

print(f"Forgery Status: {forgery\_status}")

print(f"Edge Density: {edge\_density:.3f}, Threshold Density: {threshold\_density:.3f}, Suspected Regions: {suspected\_count}")

# Display all stages

display\_results(

[original, gray, equalized, edges, thresholded, filtered, highlighted\_image],

['Original', 'Grayscale', 'Histogram Equalized', 'Edges', 'Otsu Threshold', 'Filtered', 'Suspected Regions']

)

# ✅ Run it with your uploaded image

detect\_forgery('img.png')

**Results**

1. **Original Image**

****

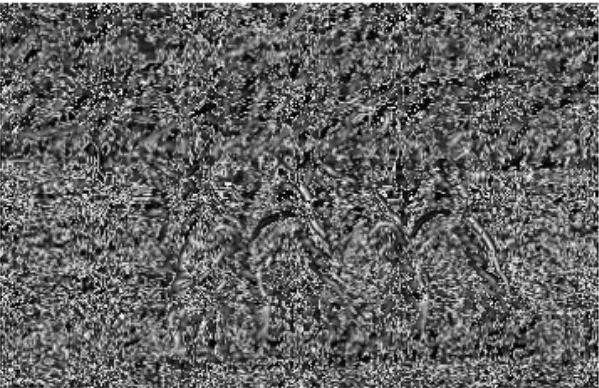
1. **Gray scale Image**

****

1. **Histogram Equalized**

****

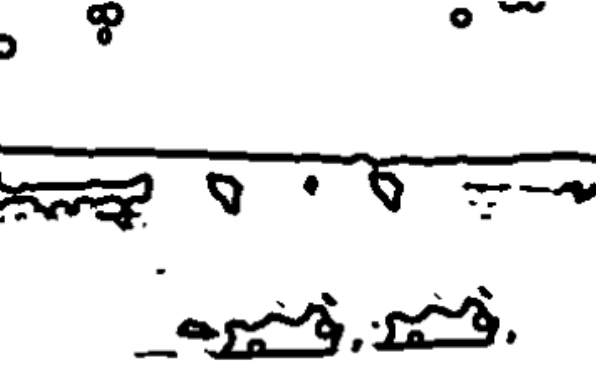
4.**Edges**



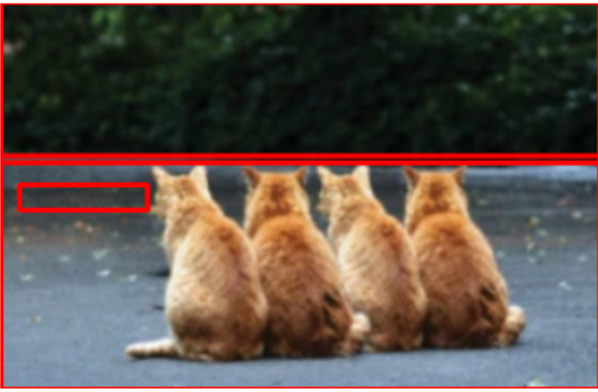
**5.Otsu threshold**

****

**6.Filtered**

****

**7.Suspected Regions**



**Discussion and Conclusion**

**Discussion:**

The project "Image Forgery Detection Using Digital Image Processing" successfully demonstrates how traditional image processing techniques can be utilized to detect tampering in digital images without the need for complex machine learning models.  
By using a structured pipeline — involving preprocessing, edge detection, thresholding, noise filtering, feature extraction, and decision logic — the system is able to highlight suspicious regions that could indicate image forgery.

One of the key strengths of the project is the simplicity and interpretability of the detection process. Each stage of the pipeline contributes meaningfully:

* Preprocessing improves contrast and visibility of anomalies.
* Edge detection captures unnatural transitions which are often signs of tampering.
* Thresholding and filtering isolate and refine the potential forged regions.
* Feature extraction and simple rules allow automatic classification without any training dataset.

Additionally, the project provides visual outputs at each processing stage, making it easy for users to understand and trust the detection process.  
The system is lightweight, works offline, and executes quickly, making it practical for small to medium-scale image verification tasks.

However, the system has certain limitations:

* It may miss very subtle forgeries where changes are carefully blended.
* Fixed threshold values might not adapt well across all types of images (e.g., different lighting, textures).
* It is less effective against deepfakes, AI-generated edits, and highly professional forgeries.

Despite these challenges, the results demonstrate that Digital Image Processing alone can be a powerful tool for preliminary forgery detection, especially in environments where computational simplicity and transparency are prioritized.

**Conclusion:**

This project achieves its primary objective of detecting forged regions in images using basic Digital Image Processing techniques . Through a carefully designed step-by-step pipeline, suspicious areas are effectively identified without relying on Artificial Intelligence or Machine Learning .The method is fast, easy to understand, and effective for many common types of image tampering.

While more sophisticated attacks (like deepfakes) require advanced techniques, this project proves that **DIP-based forgery detection remains a strong and explainable first line of defense** .Future improvements could involve introducing **adaptive thresholds**, **machine learning integration**, or **metadata analysis** for even better accuracy.

Overall, the project highlights the power of traditional image processing and provides a strong foundation for more advanced forgery detection systems.

**Future work**

**1. Adaptive Thresholding and Feature Tuning**

* Implement dynamic or adaptive thresholds based on image properties rather than using fixed values.
* Automatically adjust parameters like edge density and contour area based on the input image’s size, type, or complexity to improve detection accuracy across diverse datasets.

**2. Machine Learning and Deep Learning Integration**

* Introduce Machine Learning models to learn complex patterns of forgery from large datasets.
* Deploy Convolutional Neural Networks (CNNs) for forgery classification and localization, improving detection of subtle or skillfully edited images.
* Train classifiers to distinguish between natural and forged images based on learned feature sets.

**3. Copy-Move Forgery Detection**

* Extend the system to detect copy-move forgeries, where parts of an image are duplicated elsewhere within the same image — a very common type of tampering.
* Implement block-based or keypoint-based matching techniques for this purpose.

**4. Metadata Analysis (EXIF Data Checking)**

* Analyze image metadata (EXIF information) to detect inconsistencies, such as changes in camera parameters, timestamps, or editing software signatures.
* Metadata discrepancies can provide additional clues to support forgery detection.

**5. Deepfake and AI-Generated Image Detection**

* Expand the system to identify AI-generated (GAN-based) fake images and deepfake faces, which are becoming increasingly prevalent and harder to detect manually.

**6. Web and Mobile Application Development**

* Develop a user-friendly web application or mobile app to make forgery detection tools easily accessible to non-technical users.
* Offer features like drag-and-drop image analysis, batch processing, and report generation.

**7. Improved Visualization and Reporting**

* Provide more advanced visualization tools to not only highlight suspicious regions but also explain why certain areas are considered forged.
* Generate downloadable analysis reports summarizing detection results for use in legal, journalistic, or academic settings.

**8. Dataset Expansion and Benchmarking**

* Test the system on publicly available forgery detection datasets to benchmark and validate its performance.
* Regularly update and retrain the system based on new types of forgeries and editing techniques.

**References**

1. **Rafael C. Gonzalez, Richard E. Woods – *Digital Image Processing***
2. **OpenCV Documentation –** [**https://docs.opencv.org**](https://docs.opencv.org)
3. **NumPy Documentation –** [**https://numpy.org**](https://numpy.org)
4. **Python Official Docs –** [**https://docs.python.org**](https://docs.python.org)
5. **Research papers on Image Forgery Detection using Classical Techniques**