WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Image Forgery Detection

Ms. A. M Kowshika[1], Dr.B.Ramya[2]

[1]Student, III BCA A, Department of Computer Application, Dr. N.G.P Arts and Science College, Coimbatore-641048
[2]Assistant Professor, Department of Computer Application, Dr. N.G.P Arts and Science College, Coimbatore-641048.
DOI : https://doi.org/10.55248/gengpi.6.0425.1369

ABSTRACT :

This project, titled "Image Forgery Detection" using SIFT Model is developed using Python and MySQL. The goal is to detect forged digital images using machine learning techniques. In today's world, digital images are a major way we share information. However, technology has made it easy to manipulate images, leading to more forgeries. This project tackles this problem using a powerful method called Scale-Invariant Feature Transform (SIFT) that can match images even if they are scaled or rotated. The application organizes images into two folders: training and testing data. The trained folder has images used to teach the system and the test folder has images that need to be checked for forgery. When an image is uploaded, the system first removes any unwanted noise and converts the color image to grayscale. The SIFT detector then scans the grayscale image to find unique features that won't change with scaling or rotation. These features are stored for analysis. The system compares the key points of the test image with those from the train images using Dependent Similarity analysis. If the features match a known forged image, the system predicts the uploaded image is likely fake; otherwise, it's considered genuine.

Keywords: Copy-Move Forgery, Splicing, Retouching, Digital Watermarking, Digital Signatures, Noise Analysis, Pixel-Based Techniques, Deep Learning.

1.INTRODUCTION

Every year, brands lose a significant amount of money to fake products and counterfeits. These fake products are often of poor quality, which damages the brand's reputation and cheats consumers out of their hard-earned money. Consumers unknowingly buy these fake products, thinking they're genuine, and end up paying a high price for something that's not worth it. This not only affects the brand's sales but also erodes customer trust and loyalty. Moreover, counterfeit products can also pose health and safety risks to consumers, making it essential to detect and prevent them.

This problem affects not only brands but also consumers. Forged products can be harmful for us. For example, fake medicines can have harmful ingredients, and counterfeit electronics can catch fire or explode. That's why it's essential to detect and prevent Forged products. The Forged Detection system is designed to help consumers identify edited images and verify the authenticity of the logos they buy.

This project uses advanced technology to analyze images and detect forgeries. It extracts interest points from images using the Scale Invariant Feature Transform Algorithm (SIFT). This helps to identify fake logos and distinguish them from genuine ones. With this system, consumers can verify the authenticity of products, and brands can take safety measures to prevent forgeries, which creates a safer and transparent marketplace.

2. SYSTEM STUDY

2.1 EXISTING SYSTEM

Often, consumers lose their money by paying a lot for fake products. The issue of detecting fake images has become an important research area due to the rise in the number of forgeries. Many efforts have been made to improve fake image detection and still research is going on to look for better ways to identify fake images. The current system uses shape matching and object recognition models to detect fake images. However, these methods only match shapes and objects, which takes more time and require a lot of training data. This leads long time and the algorithm often fails to achieve high accuracy.

2.2 PROBLEM IDENTIFICATION

Currently, it’s hard for consumers and brands to know if a product is genuine or fake. They rely on manual methods, which can be time-consuming and ineffective. This leads to financial losses and damage to brands. Consumers may also face loss from counterfeit products.

To solve this problem, an Image Forgery Detection system is needed. This system can analyze product images and detect signs of forgery. It will make it easier for consumers and brands to identify genuine products and prevent forgeries. With this system, consumers can trust the products they buy, and brands can protect their sales.

2.3 PROPOSED SYSTEM

The drawbacks, which are faced during existing system, can be eradicated by using the proposed system. The main objective of the proposed system is to provide a user-friendly interface for detect fake Image more accurately. The Image detection aims to helps user distinguish forgeries from the original product.

Initially interest point extraction is done using Scale Invariant Feature Transform Algorithm (or SIFT). This SIFT detect and describe local features in images .Finally Each and every Interest points analysis using Dependent Similarity and finally system will predict upload Image is forgery or real effectively. This application can also be helpful for brands struggling to fight against forged products.

3. SYSTEM DESIGN

3.1 MODULE DESCRIPTION

Image Upload Process

The image upload process is the first step in checking if an image is real or fake. This process involves collecting a large set of images from the internet. It contains two folders: trained and test folder. This helps to make sure that the computer is able to correctly identify fake images.The images are stored in a special folder, where they can be easily accessed and used.

Pre-processing and Feature Extraction

Pre-processing and feature extraction are important steps in preparing an image for analysis. Pre-processing makes sure that the image is clear and easy to read. This involves adjusting the brightness or contrast of the image. Feature extraction involves looking and identifying image's that make it unique. This involves the shapes or patterns in the image, or identifying specific features. It uses a technique called Local Binary Pattern (LBP). LBP looks at the image and converts it into a series of binary numbers, which can be used to identify patterns and features.

Interest Points Extraction

Interest points extraction is a way of identifying the most important parts of an image. This involves using an algorithm called Scale Invariant Feature Transform (SIFT). SIFT looks at the image and identifies the parts that are most unique and interesting. These can be shapes and patterns. SIFT is able to identify these interest points even if the image is rotated or viewed from a different angle. Its is a powerful tool for analyzing images.

Context Dependent Similarity

Context dependent similarity is a way of comparing two images to see if they are similar. This involves looking at the interest points that were identified in the previous step. This process involves transforming image data into a special format to compare and find key points, such as corners or edges, to identify the image. By doing this, it is possible to determine whether the two images are similar, or one image might be fake.

Image Classification

Image classification is the final step in determining whether an image is real or fake. This module allows image uploads in JPEG or PNG format. Once uploaded, the system checks the image's authenticity by comparing its interest points with those in the database. If the image is forged, the system alerts accordingly. This module provides a user-friendly interface for checking image authenticity.

3.2 DATABASE DESIGN

Admin Login Table:

Field Name	Data type	constraints	Description
username	Varchar(10)	Not Null	Username of the admin
Password	Varchar(10)	Not Null	Password information

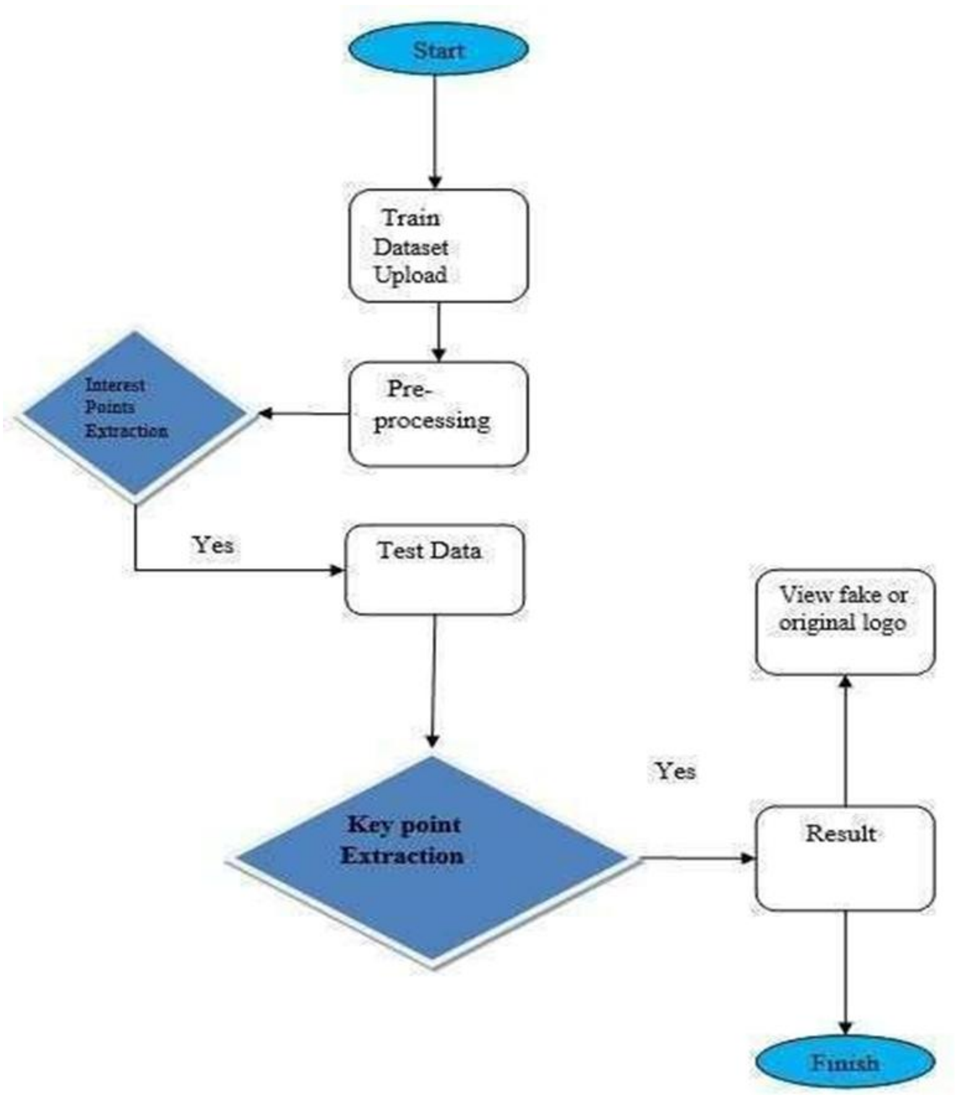
Train forged info Primary Key: Image id

Colum Name	Data type	Description	Constraint
Image id	Int(20)	Contains the image id	Not null
Image name	Varchar(20)	Contains the name	Not null
Image path info	Varchar(10)	Contains the test image path	Not null
Ip_1	Int (20)	Contains train image Interest Points Extraction details	Not null
Ip_2	Int (20)	Contains train image Interest Points Extraction details	Not null

Result info Foreign Key: Image id

Column Name	Data type	Description	Constraint
Image id	Int(20)	Contains the image id	Not null
Image name	Varchar(20)	Contains the name	Not null
Image path info	Varchar(10)	Contains the test image path	Not null
Ip_1	Int(20)	Contains train image Interest Points Extraction details	Not null
Ip_2	Int(20)	Contains train image Interest Points Extraction details	Not null
Classification result	Varchar(10)	Store result fake or original forged	Not null

3.3 STRUCTURE DIAGRAM



4. SYSTEM TESTING AND IMPLEMENTATION

4.1 SYSTEM TESTING

System testing is a critical phase in the software development cycle that ensures the entire system functions as intended and meets the specified requirements. For the Image Forgery Detection Project, system testing involves evaluating the complete system—covering all modules.

The primary goal of system testing in this project is to validate that the image upload, forgery detection, database storage, and user interface work as expected under various conditions. By performing different types of testing, we can ensure that the system performs efficiently, provides correct results (real or forged images), and gives a user- friendly experience.

4.2 TESTING METHODS

1. **Image Upload Testing:** It ensures the system can accept and record essential details of uploaded images from dataset which proceeds to next part of the project.
2. **Forgery Detection Testing:** This testing validates the core functionality of the system, which is accurately classifying images as real or forged using the SIFT algorithm.
3. **Database Verification:** It confirms that important metadata is stored properly, maintaining system integrity and enabling future data retrieval.
4. **User Interface Testing:** Guarantees a smooth and intuitive user experience, which is critical for usability.

5. **Performance Testing:** This ensures whether the system handles large images or extracting efficiently without compromising speed or functionality.
6. **Edge Case Testing:** It helps to identify and address the potential issues with unexpected inputs, making the system more robust and stable.
7. **Error Handling Testing:** This testing is to ensure whether the system responds if user uploads non-image files or invalid image formats instead of uploading from data.

4.3 SYSTEM IMPLEMENTATION

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively. The system can be implemented only after thorough testing is done and if it is found to work according to the specification. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover and an evaluation of change over methods a part from planning

System Testing:

In this the system is tested to ensure all components work together. This includes testing image uploads, forgery detection, and database storage.

Error Detection and Debugging:

It identifies any bugs or issues happens during testing are fixed. For example, if the system incorrectly classifies a real image as forged or vice versa, those issues will be addressed before deployment.

Database Setup Testing:

The MySQL database is set up to store essential information such as image details like Image ID, detection results (key points). It ensures that all uploaded images and their results are stored securely.

Integration Testing:

The frontend is connected to the backend where the image processing happens. This integration ensures that when a user uploads an image, the system can process the image using the SIFT algorithm, and display the results.

Security Testing:

Security features are implemented to protect the system and user data. This includes ensuring that user authentication is required before they can upload images.

User Training & Documentation:

This testing is to ensure that the user can interact with the system easily. It helps the users to upload images through the system efficiently.

Performance Monitoring & Maintenance:

Once the Image Forgery Detection System is running, it will be regularly checked to make sure everything is working properly. If there are any issues, such as the system running slowly, errors showing up, or security problems, they will be fixed quickly.

5. CONCLUSION

The Image Forgery Detection Project shows a highly effective approach to solve real world challenges in digital image authentication. By integrating multiple image processing techniques, the project efficiently handles processes like image uploading, preprocessing, feature extraction, and classification. It ensures robust detection of forgeries by employing advanced methods to analyze and compares the features in images. The system is also userfriendly, with step by step selection options that simplifies navigation. It provides a reliable, user-friendly, and secure method for identifying counterfeit images, helping both consumers and brands to protect their money. By automating the process of image forgery detection, this web application reduces the risk of fraud, saving time and money while improving trust and credits of the brands.

REFERENCE

- Python Tricks: A Buffet of Awesome Python Features by Dan Bader

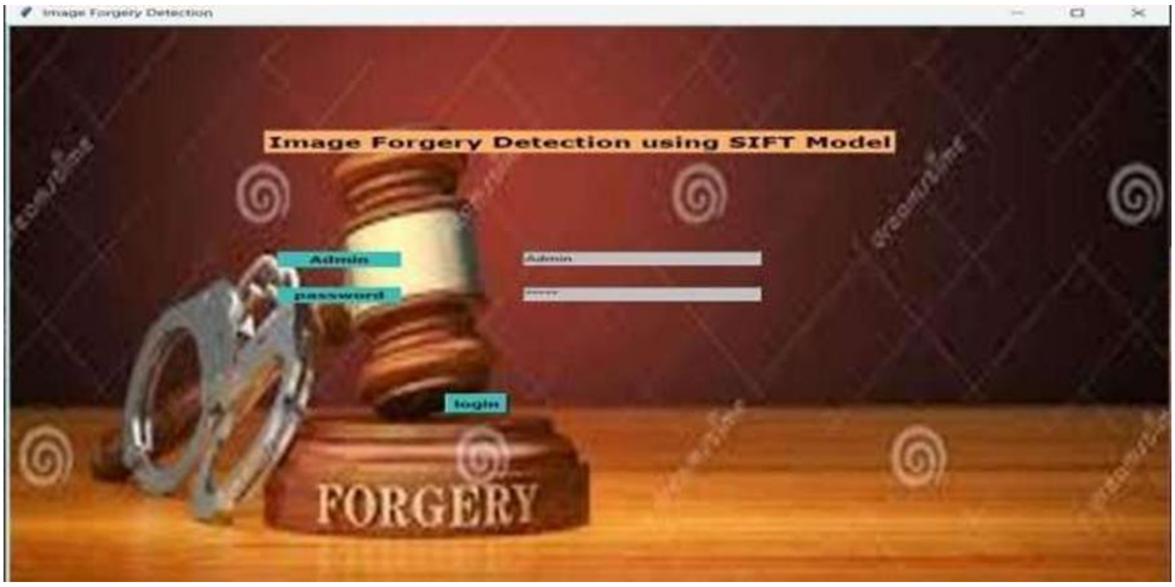
REFERENCE WEBSITES

- https://creatorpdf.com/B0785Q7GSY
- https://creatorpdf.com/B0785Q7GSY
- https://www.w3schools.com/python
- www.tutorialspoint.com

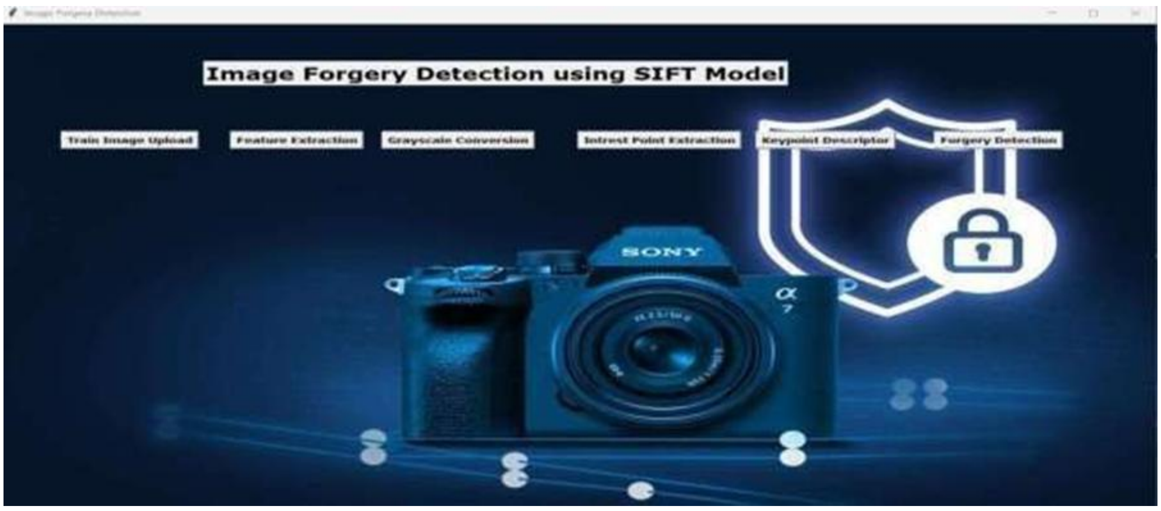
- https://www.w3schools.com/python

APPENDIX

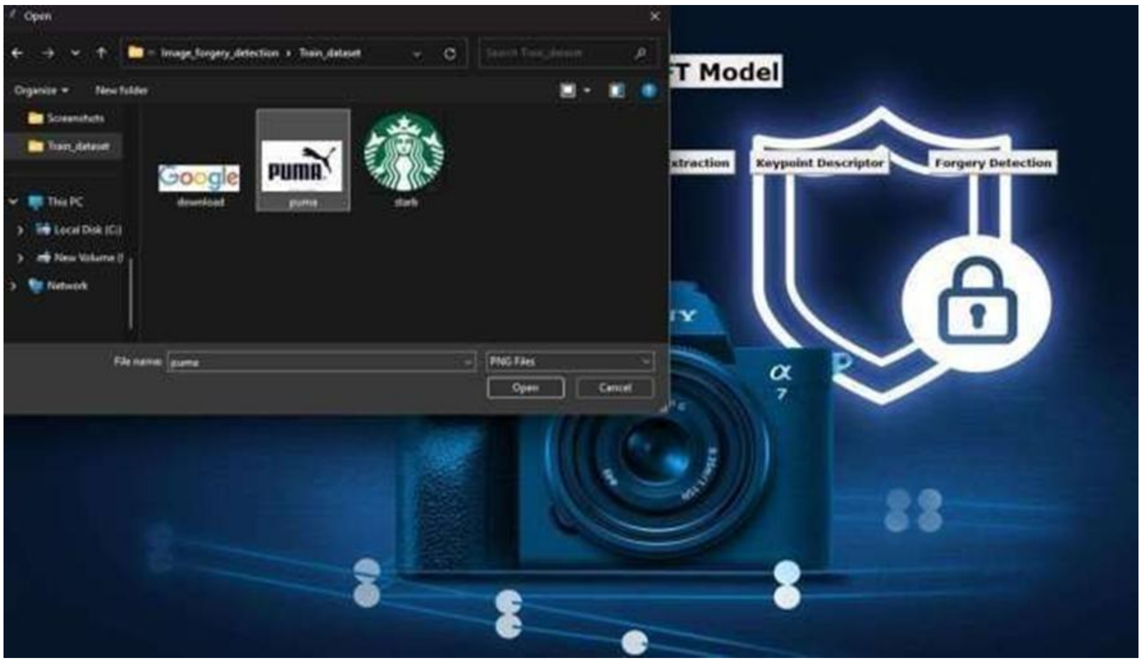
A. SCREENSHOTS



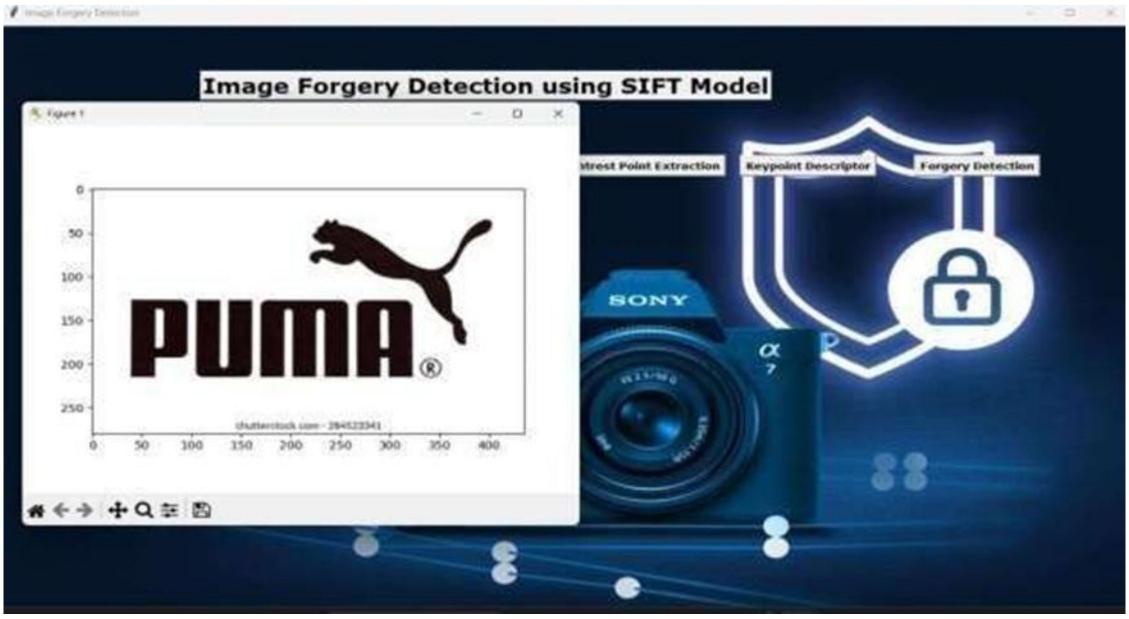
HOME PAGE



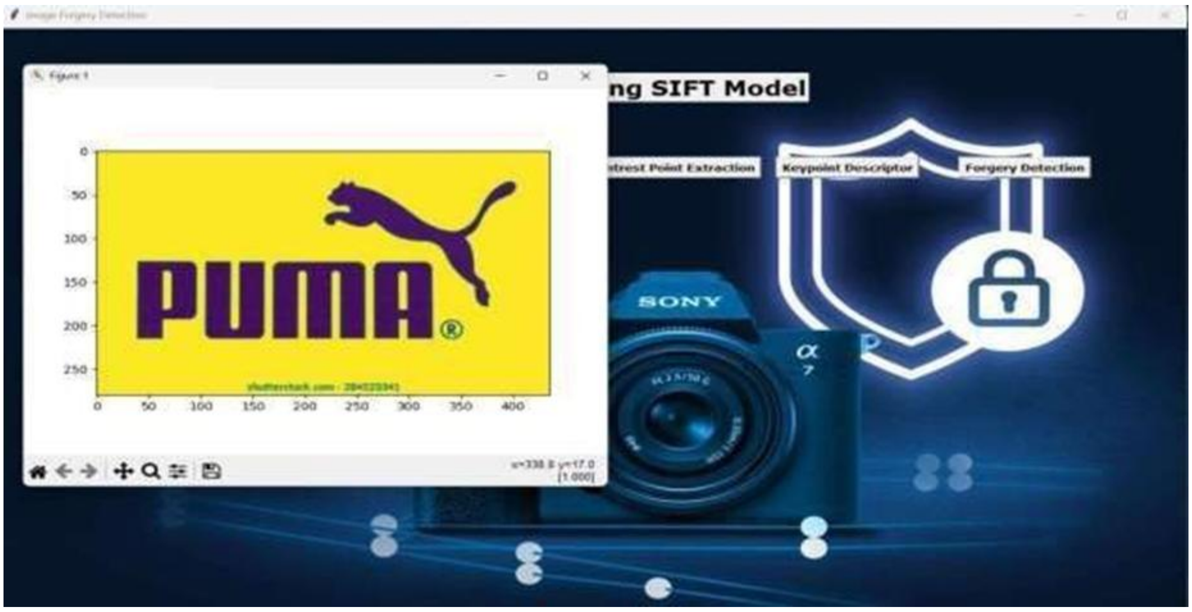
UPLOADING IMAGE FROM DATASET



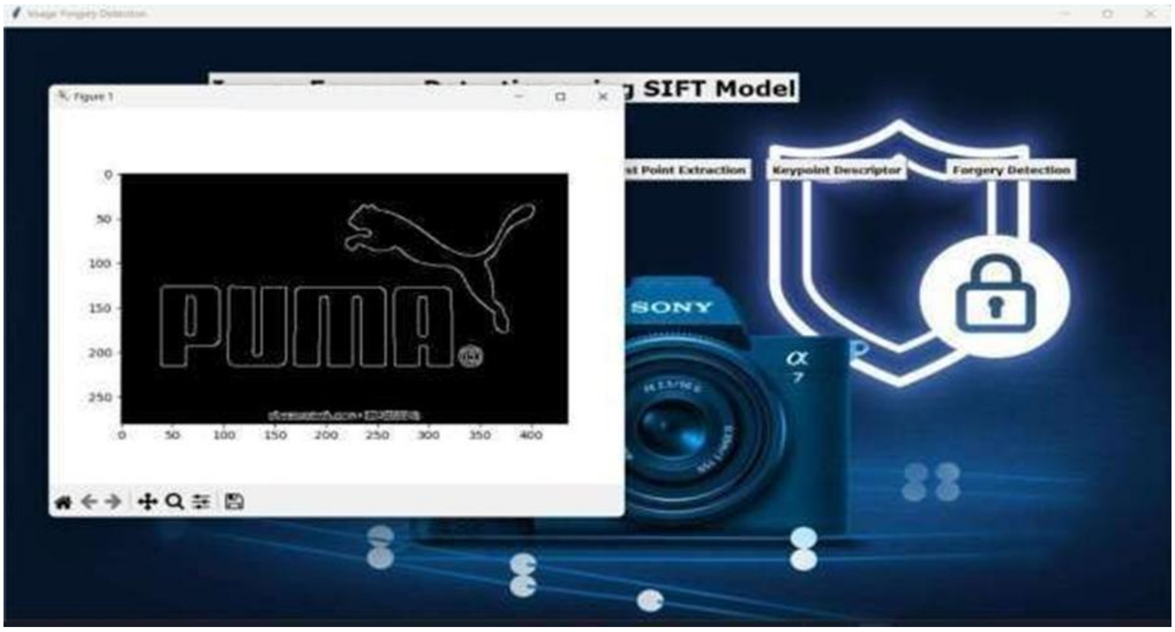
PREPROCESSING AND EXTRACTION OF FEATURE



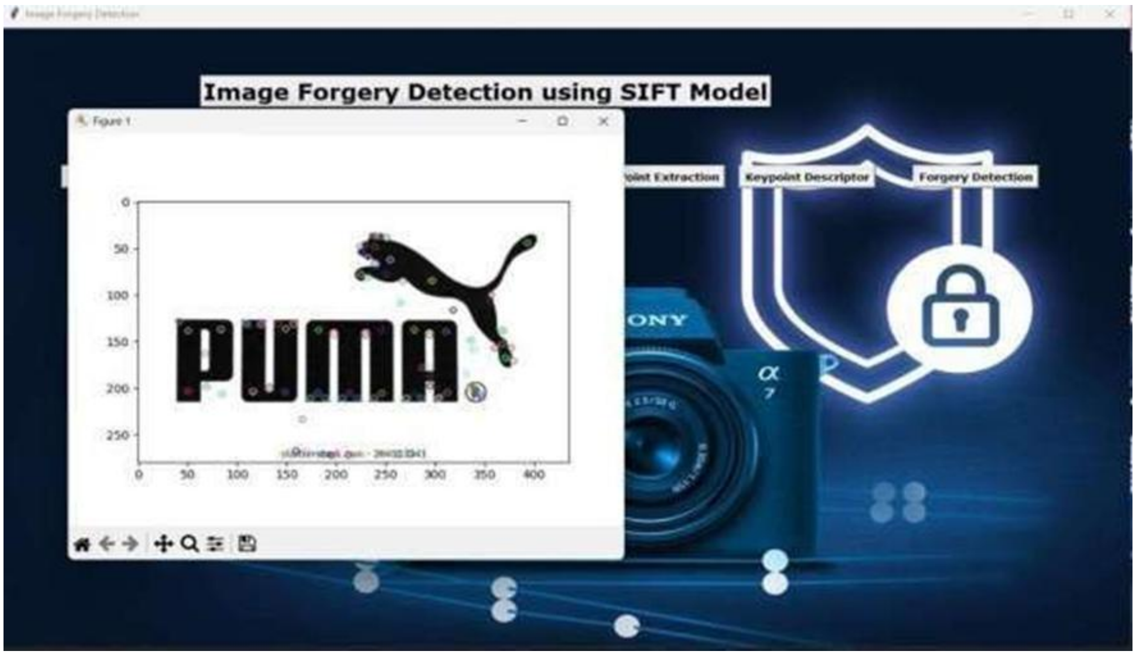
GRAYSCALE CONVERSION TECHNIQUE: Which includes enhancing brightness of the image.



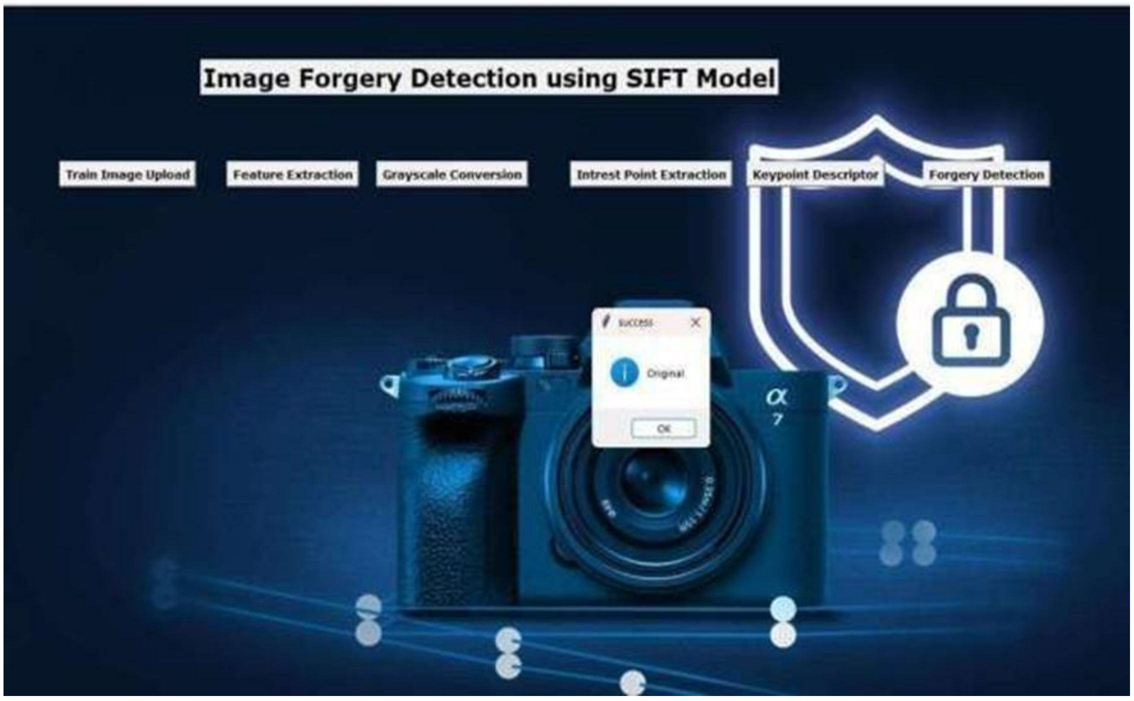
INTEREST POINT EXTRACTION WHICH USES SIFT ALGORITHM



KEYPOINT DESCRIPTION TECHNIQUE



RESULT OF THE DETECTED IMAGE



B. SOURCE CODE

from skimage import filters, feature

import imagehash from tkinter import * from tkinter

import messagebox, filedialog, simpledialog import cv2

import matplotlib.pyplot as plt import pandas as pd

import tkinter as tk from tkinter.filedialog import askopenfilename

import pymysql import csv from csv import writer

import os from skimage.filters.edges

import prewitt_h, prewitt_v from skimage.io import imread, imshow from PIL

```
import ImageTk, Image class ViewData:

def _init_(self):

def dataupload():

global filename1 f_types = [('Jpg Files', '*.jpg'), ('PNG Files', '*.png')] filename1 = filedialog.askopenfilename(filetypes=f_types)

msg = "Image uploded Sucessfully" messagebox.showinfo("Success", msg) def viewdata():

im = cv2.imread(filename1) plt.imshow(im) plt.show() msg="Feature        Extracted   Sucessfully" messagebox.showinfo("Success", msg)

def viewdata1():

image2 = imread(filename1, as_gray=True) plt.imshow(image2) plt.show()

msg="Grayscale      Conversion         Sucess"

messagebox.showinfo("Success", msg)


def viewdata2():

image2 = imread(filename1, as_gray=True)

pre_hor = prewitt_h(image2) pre_ver = prewitt_v(image2) # Sobel Kernel

ed_sobel = filters.sobel(image2) # Datamining algorithm

can = feature.canny(image2) plt.imshow(can, cmap='gray'); plt.show()

msg        =          "Intrest     Point      Extraction Sucess" messagebox.showinfo("Success", msg)

def viewdata3():

image2 = cv2.imread(filename1)

gray1 = cv2.cvtColor(image2, cv2.COLOR_BGR2GRAY) # keypoints

sift = cv2.SIFT_create()

keypoints_1, descriptors_1 = sift.detectAndCompute(image2, None) print(descriptors_1)

img_1 = cv2.drawKeypoints(gray1, keypoints_1, image2) plt.imshow(img_1) plt.show()

msg = "Key Point Extracted"

messagebox.showinfo("Success", msg) def act2():

global vehnorth f_types = [('Jpg Files', '*.jpg'), ('PNG Files', '*.png')]        filename1  =

filedialog.askopenfilename(filetypes=f_types)

pathSpecified        =        "D:\Image_forgery_detection\Train_dataset" fnames = os.listdir(pathSpecified)

# print(listOfFileNames) count = 0

similarity = 80

threshold = 1 - similarity / 100 with Image.open(filename1) as img:

hash1 = imagehash.average_hash(img)         print(hash1) for      image      in        fnames:           with
Image.open(os.path.join("D:\Image_forgery_detection\Train_dat aset", image)) as img1:

hash2 = imagehash.average_hash(img1)

print("Similarity Checking", hash2) if hash1 == hash2:

count = 1

print("{} image found {}%      similar to {}".format(image, similarity, filename1))

if count == 1:

# print("Logo is Original ") s1="Original" messagebox.showinfo("success",s1) else:
```

```
# print("Logo is Not Origianl ") s1 = "Fake"

messagebox.showinfo(" success",s1) win = Tk()

# app title win.title("Image Forgery Detection") # window size

win.maxsize(width=1300, height=800)

win.minsize(width=1300, height=800) win.configure(bg='#59B5B4')

image1 = Image.open("2.jpg") img = image1.resize((1500, 800)) test = ImageTk.PhotoImage(img) label1 = tk.Label(win, image=test) label1.image =
test

# Position image        label1.place(x=0, y=0) # image1 = Image.open("3.png")

test = ImageTk.PhotoImage(img) label1 = tk.Label(win, image=test) label1.image = test

heading = Label(win, text="Image Forgery Detection using SIFT Model", font='Verdana 20 bold') heading.place(x=230, y=60)

btnbrowse = Button(win, text="Train Image Upload", font=' Verdana 10 bold', command=lambda: dataupload())

btnbrowse.place(x=70, y=170)

btncamera = Button(win, text="Feature Extraction", font='Verdana 10

bold', command=lambda:

viewdata())             btncamera.place(x=260, y=170)

btnsend = Button(win, text="Grayscale Conversion", font='Verdana 10 bold', command=lambda: viewdata1())

btnsend.place(x=430, y=170)

btnsend = Button(win, text="Intrest Point Extraction", font='Verdana 10 bold', command=lambda: viewdata2())             btnsend.place(x=650,
y=170) btnsend = Button(win, text="Keypoint Descriptor", font='Verdana 10 bold', command=lambda: viewdata3())        btnsend.place(x=850,
y=170) btnsend = Button(win, text="Forgery Detection", font='Verdana 10 bold', command=lambda: act2())      btnsend.place(x=1050, y=170)

# btnsend = Button(win, text="Upload Data Set", font='Verdana 10 bold', command=lambda:act3())

# btnsend.place(x=1070, y=170)

#          btnsend      =          Button(win,          text="Phishing          Detection", font='Verdana 10 bold', command=lambda:act4())

,# btnsend.place(x=1250, y=170) win.mainloop()
```