**HISTOGRAM-BASED IMAGE FORGERY DETECTION**

**USING PYTHON**

**A PROJECT REPORT**

***Submitted by***

**ASWIN K : 715522104008**

**DAVID VEDHA JEROME A : 715522104113**

**DHARINEESH DJ : 715522104116**

**DILIPAN AR : 715522104117**

**DHRUV R : 715522104118**

***in partial fulfilment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH, COIMBATORE 641 062**

**ANNA UNIVERSITY: CHENNAI - 600 025**

**JUNE 2023**

**ABSTRACT**

Image Forgery Detection plays crucial role in image analysis and computer vision. Image processing is used widely in many fields including medicine, photography, design and other industries. Image proxy or image similarity is a one of the problems which can be resolved by comparing their frequency of the intensities of the pixels of both the images.

OpenCV is one of the powerful tools for image processing, computer vision and visualization. This work consists of a Python script that can generate and compare the histogram of two input images and find the absolute difference in order to find the similarity between them.

Python functions have been used to enable the user to prompt the images that needs to be compared. The fed images are then converted into gray scale for better comparison. The histograms of the images are generated respectively. Finally, the result is obtained by calculating the absolute difference and displaying the output. Keywords: OpenCV, computer vision, histogram, image similarity, pixel intensities.

**LITERATURE REVIEW**

A digital image consists of numerous pixels and their histogram gives their frequency of occurrence. OpenCV developed by Intel is a powerful library for computing intensive vision tasks and image processing which can be integrated with Python that contains the inbuilt functions to prompt images, convert them into grayscale and generate the histogram of the images.

Image forgery detection can be performed by comparing the histograms of the two images and estimating their absolute difference. The images are prompted using a python-based UI created using tkinter module

With the absolute difference in the intensities of the pixels, similarity between the images can be inferred.

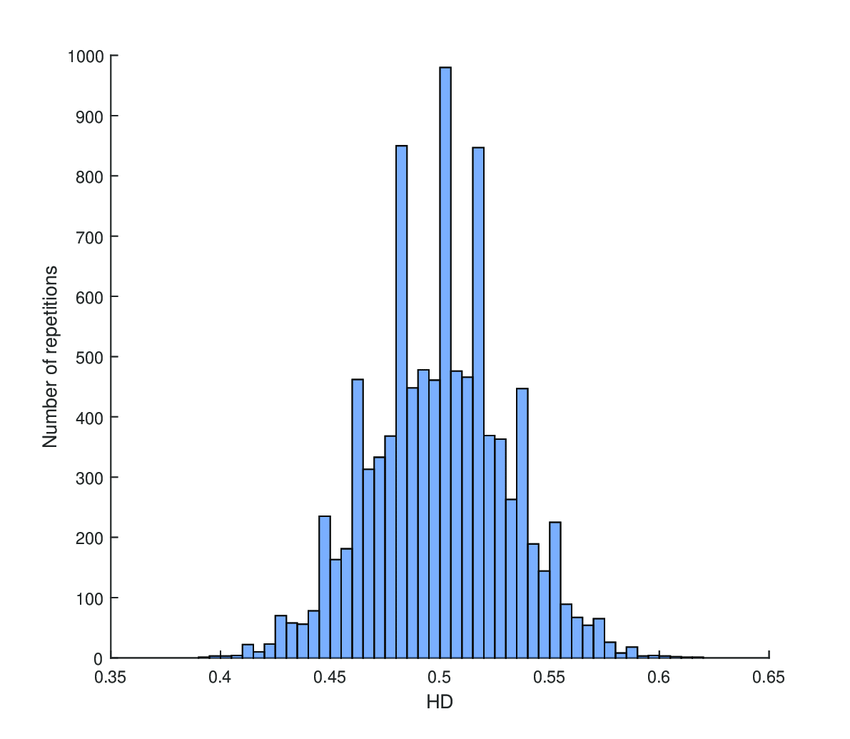


Figure 1 – **Histogram**

**SOURCE CODE**

**from tkinter import \***

**from tkinter import filedialog**

**from PIL import Image,ImageTk**

**from tkinter.filedialog import askopenfile**

**import tkinter.messagebox as messagebox**

**import cv2**

**import tkinter as tk**

**window=Tk()**

**window.geometry("1920x1080")**

**window.title("IMAGE FORGERY DETECTION")**

**window.configure(bg='#8fbc8f')**

**image=Image.open("PSG.jpeg")**

**resized=image.resize((250,250),Image.LANCZOS)**

**new\_pic=ImageTk.PhotoImage(resized)**

**def new\_window1():**

**window.destroy()**

**window1=Tk()**

**window1.geometry("1920x1000")**

**window1.configure(bg='#8fbc8f')**

**window1.title("IMAGE FORGERY DETECTION")**

**l1=Label(window1,text="Image Forgery Detection",font=("Angsana New",40,"bold"),bg="#8fbc8f").place(relx=0.5,rely=0.15,anchor="n")**

**image1=Image.open("project.png")**

**resized1=image1.resize((300,250),Image.LANCZOS)**

**new\_pic1=ImageTk.PhotoImage(resized1)**

**l2=Label(window1,image=new\_pic1).place(relx=0.5,rely=0.45,anchor="center")**

**button1=Button(window1,text="Open File",font=("Georgia",20),bg="black",fg="white",activebackground="#8fbc8f",command=img\_to\_text\_converter).place(relx=0.5,rely=0.75,anchor="s")**

**window1.mainloop()**

**data=""**

**def img\_to\_text\_converter():**

**global data,Img,Img1,img1,img,G,window1,button1,button2,tk**

**path=filedialog.askopenfilename()**

**fath=filedialog.askopenfilename()**

**if not path:**

**data = "YOU DIDN'T SELECT FIRST IMAGE"**

**messagebox.showinfo("ERROR!",data)**

**elif not fath:**

**data = "YOU DIDN'T SELECT SECOND IMAGE"**

**messagebox.showinfo("ERROR!",data)**

**elif not path and fath:**

**data = "YOU HAVE SELECTED NOTHING"**

**messagebox.showerror("ERROR!",data)**

**else:**

**Img=cv2.imread(path,-1)**

**Img1=cv2.imread(fath,-1)**

**img=cv2.calcHist([Img], [0, 1, 2], None, [256, 256, 256], [0, 256, 0, 256, 0, 256])**

**img1=cv2.calcHist([Img1],[0, 1, 2], None, [256, 256, 256], [0, 256, 0, 256, 0, 256])**

**G=cv2.compareHist(img,img1,cv2.HISTCMP\_CORREL)**

**if G ==1.0:**

**data = "They are the same images"**

**else:**

**data = "Image Forgery Is Done"**

**messagebox.showinfo("Information",data)**

**#place function here is used to place the label at the required place.**

**l1=Label(window,text="PSG Institute of Technology and Applied Research",font=("Angsana New",40,"bold"),bg="#8fbc8f").place(relx=0.5,rely=0.15,anchor="n")**

**l2=Label(window,image=new\_pic).place(relx=0.5,rely=0.5,anchor="center")**

**button=Button(window,text="START",font=("Georgia",20),bg="black",fg="white",activebackground="#8fbc8f",command=new\_window1).place(relx=0.5,rely=0.80,anchor="s")**

**window.mainloop()**

**CHAPTER 1**

**INTRODUCTION**

* 1. **PROJECT MOTIVATION**

This report discusses the process of finding the similarity between two images based on the histogram using OpenCV an Open-source Computer Vision library and a powerful image processing tool developed by Intel. Image similarity detection can be done using many other ways but one of the simplest and yet efficient way is using their histogram. This would be crucial in image comparison, image retrieval, image analysis and finding image replication.

Pixel intensities of an image can be represented graphically which provide us important information about the frequency of occurrence of the intensities of the pixels of the image. This is known as histogram of an image. Histogram of an image can be generated in Python using calcHist() function.

**1.2 PROBLEM STATEMENT AND OBJECTIVES**

PROBLEM STATEMENT:

Finding any forgery occurred between two images by comparing their histogram values is the problem addressed in this report. The objective of this project is to develop a OpenCV based Python solution to compare the histogram of two images.

OBJECTIVES:

1. Develop a Python code that loads two images and convert them into gray scale

2. Estimate the histogram using calcHist() function and normalize them to compare the two input images

3. Finding the difference between the values of histogram. If the values changes between the two images then there is a forgery occurred or if no changes between the values of two images, they’re said to same images.

4. To discuss the possible improvement and extension to this method for future work

5. To provide a vivid and brief explanation about the process and effectiveness of the code implementation.

6. Report conclusion by indicating the significance of histogram-based image similarity in various real time applications.

* 1. **SCOPE AND APPLICATIONS**

SCOPE OF THE PROJECT:

This project analyses the development and applications of a code written using Python which utilizes the OpenCV library which is able to provide a quantitative result with the comparison of the histogram of the two images that are obtained as the input.

Practical Applications: This project discusses the potential applications of image similarity method based on the histogram comparison in wide range of domains such as image retrieval, image analysis, image duplication and it also provides solution to the real-world scenarios where this approach can be useful. By expanding the scope of the project, the performance and efficiency of this histogram-based solution can be extensively increased and can be made applicable to many fields constructively.

* 1. **LIMITATIONS**

This histogram-based method is sensitive only towards the change in the pixel intensities of the two images. It may not responsive to the structural changes such as rotational or scaling transformations of the images.

The complexity of computing the histogram of the images increases with increase in the size of the images. Hence, optimization in the program is required in order to perform the functions for larger images.

The project mainly focuses on the histogram comparison between the gray scale of the images. Alterations are needed in order to perform the operation on colored images by estimating separate histograms for each color channel.

The above limitations should be considered before performing the operations and interpreting the results in real-world applications.

**CHAPTER 2**

**PROJECT ARCHITECTURE, DESIGN AND IMPLEMENTATION**

The methodology of the project involves the following steps, algorithm and techniques for comparing the differences in the histogram of the two images generated by the Python code.

2.1 IMAGE LOADING AND PREPROCESSING:

The askopenfilename() function is being utilized in reading the images either as grayscale or RGB. In order to maintain the consistency in the image quality and dimensions, preprocessing is done such as cropping, resizing and normalization if needed.

2.2 HISTOGRAM COMPUTATION AND THEIR DIFFERENCE:

The histograms of the two images are generated using the calcHist() function which gives us the graphical information about the intensities of the pixel values of the two images respectively. Then the difference between these two images is noted and the program checks for the similarity based on the absolute value of the difference.

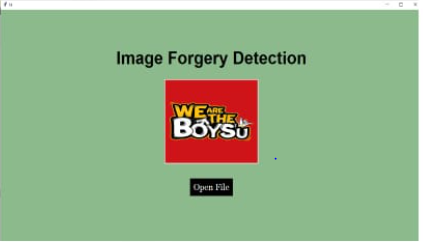
The two images given as inputs are estimated to be similar if the histogram difference is zero, since their intensities matches.

2.3 SIMILARITY VISUALIZATION:

Developing visual representations to demonstrate the similarity between the two images that are given as inputs in the prompt and overlaying the histograms of the images to visually compare their distributions.

By applying this methodology and techniques, the similarity between two images can be computed approximately based on their histograms and also provides insights to its performance, limitations and practical applications.

**INPUT**

**OUTPUT**



**REFERENCE**

Ways to Compare histogram using OpenCV and Python- https://pyimagesearch.com/2014/07/14/3-ways-compare-histograms-using-opencv-python/

Image source: Figure 1 – Histogram, https://www.researchgate.net/publication/334817710/figure/fig3/AS:786903892774918@1564623916877/Histogram-of-HD-between-all-the-obtained-codes.png