ISSN: 3067-2384
Research Article

World Journal of Sensors Network Research

# Photo Forensics and Similarity Calculation Using Python and C++ Programming Languages

**Hadžib Salkić***

*CEPS – Center for Business Studies Kiseljak*

***Corresponding author:** Hadžib Salkić, CEPS – Center for Business Studies Kiseljak.

**Submitted:** 13 September 2024 **Accepted:** 23 September 2024 **Published:** 07 October 2024

**https://doi.org/10.63620/MKWJSNR.2024.1006**

***Citation:*** *Hadžib, S. (2024). Photo Forensics and Similarity Calculation Using Python and C++ Programming Languages. Wor Jour of Sens Net Res, 1(1), 01-03.*

***Abstract***

*This paper investigates the application of algorithms for comparing the similarity between two photographs using Python and C++ programming languages. Through implementation using libraries such as OpenCV and NumPy, the percentage similarity between two images is calculated. The paper shows the key steps in image processing, as well as the evaluation of the results. It also analyzes the code in both programming languages and their advantages and disadvantages.*

**Keywords:** Forensics, Python, C++, Comparison

## Introduction

Image comparison is an important task in image processing, with applications in many fields such as biometrics, face recognition, and image search. This paper presents a simple method for comparing two images using histogram analysis and similarity comparison.

## Methodology

### Introduction to Histogram analysis

Histogram analysis is often used to compare the distribution of colors in two images. A histogram represents the distribution of pixels in an image by color. By comparing the histogram distributions of two images, a measure of similarity between them can be obtained.

### Implementation in Python

The code below shows the implementation steps:

python

Copy the code

```
import cv2 # Import OpenCV library
import numpy as np # Import the NumPy library

# Function to load and calculate image histogram
def calculate_histogram(image_path):
    image = cv2.imread(image_path) # Load image from given path
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert image from BGR to RGB format
    histogram = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256]) # Calculating the histogram
    cv2.normalize(histogram, histogram) # Normalize the histogram
    return histogram

# Function to compare two histograms and calculate similarity
def compare_histograms (hist1, hist2):
  similarity = cv2.compareHist(hist1, hist2, cv2.HISTCMP_CORREL) # Comparing histograms using correlation return similarity

# The main part of the program
image1_path = 'image1.jpg' # Path to the first image
image2_path = 'image2.jpg' # Path to the second image

# Loading images and calculating histograms
histogram1 = calculate_ histogram (image1_path)
histogram2 = calculate_ histogram (image2_path)

# Histogram comparison
similarity = compare_histograms (histogram1, histogram2)

# Print the results
print (f"Probability of similarity between images: {similarity * 100:.2f} %")
```

## Results and Discussion

Through this simple algorithm, we get a numerical value that represents the similarity between two images. Histogram analysis is an effective method, but it has limitations, especially when dealing with images with different lighting or perspective. Image comparison using histogram analysis in Python is a simple but powerful tool for calculating the similarity between two images. In future work, the use of more advanced techniques such as feature-based methods or the application of neural networks could be explored.

This example of a scientific paper covers the basic aspects of creating an algorithm for image comparison using the Python programming language. It can be extended by further research and application of more advanced methods in image similarity analysis. In order to compare the implementation of comparing two images using the Python programming language with another programming language, we will choose C++, one of the most powerful and widely used image processing languages due to its efficiency and speed. We will compare two approaches, one in Python and the other in C++, in order to analyze their specificities.

### Python and Implementation

Python is popular for image processing due to its simplicity and large number of libraries, such as OpenCV and NumPy. Let's look at the Python code again:

python
Copy the code

```
import cv2 # Import OpenCV library
import numpy as np # Import the NumPy library

# Function to load and calculate image histogram
def calculate_histogram(image_path):
    image = cv2.imread(image_path) # Load image from given path
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert image from BGR to RGB format histogram = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256]) # Calculating the histogram cv2.normalize(histogram, histogram) # Normalize the histogram
  return histogram

# Function to compare two histograms and calculate similarity
def compare_histograms(hist1, hist2):
    similarity = cv2.compareHist(hist1, hist2, cv2.HISTCMP_CORREL) # Comparing histograms using correlation
  return similarity

# The main part of the program
image1_path = 'image1.jpg' # Path to the first image
image2_path = 'image2.jpg' # Path to the second image

# Loading images and calculating histograms
histogram1 = calculate_ histogram (image1_path)
histogram2 = calculate_ histogram (image2path)

# Histogram comparison
similarity = compare_histograms(histogram1, histogram2)
# Print the results
Print (f"Probability of similarity between images: {similarity * 100:.2f} %")
```

### Python Implementation Specifics

1. **Simplicity:** Python code is easy to read and understand. Using high-level libraries such as OpenCV and NumPy allows fast implementation of complex operations with few lines of code.
2. **Productivity:** Python allows for rapid iteration and testing, making it suitable for development and research.
3. **Performance:** Python is an interpreted language, so it is slower to perform compared to languages like C++. However, libraries like NumPy use optimized C/C++ implementations under the hood, which significantly speeds up operations.

### C++ Implementation

C++ is a language known for its efficiency and performance, especially when it comes to tasks such as image processing and working with memory. The following code implements image comparison using C++ and OpenCV.

cpp
Copy the code

```
#include <opencv2/opencv.hpp>
#include <iostream>

// Function to load and calculate image histogram
cv:Mat calculateHistogram(const std::string& imagePath) {
    cv: Mat image = cv:imread(imagePath); // Load the image from the given path
cv: cvtColor (image, image, cv: COLOR_BGR2RGB); // Convert image from BGR to RGB format

// Calculating the histogram
   cv:Mat hist;
   int histSize[] = {8, 8, 8};
   float range [] = {0, 256};
   const float* ranges [] = {range, range, range};
   int channels [] = {0, 1, 2};

cv: calcHist (&image, 1, channels, cv: Mat (), hist, 3, histSize, ranges, true, false);
cv: normalize (hist, hist); // Histogram normalization

   return hist;
}

// Function to compare two histograms and calculate similarity
double compareHistograms(const cv::Mat& hist1, const cv::Mat& hist2) {
     return cv:compareHist(hist1, hist2, cv::HISTCMP_CORREL); // Compare histograms using correlation
}

int main () {
std: string image1Path = "image1.jpg”; / / The path to the first image
std: string image2Path = "image2.jpg”; / / Path to another image

// Load images and calculate histogram
cv: Mat histogram1 = calculateHistogram(image1Path);
cv: Mat histogram2 = calculateHistogram (image2Path);
```

```
// Histogram comparison
    double similarity = compareHistograms(histogram1, histogram2);

// Print the results
std:cout << "Probability of similarity between images: " << similarity * 100 << "%" << std::endl;

   return 0;
}
```

**C++ Implementation Specifics**

1. **Efficiency:** C++ is a compiled language that allows faster code execution compared to Python. This is especially important for applications that require high performance.
2. **Control over Memory:** C++ provides fine-grained control over memory, allowing optimization for specific applications.
3. **Complexity:** Writing code in C++ requires more lines of code and more careful handling of resources, which can increase implementation complexity.
4. **Industry Use:** C++ is widely used in applications where performance is critical, such as video games, real-time image processing, and resource- constrained systems.
5. **Analysis and Comparison**

- **Simplicity vs. Performance:** Python offers simpler and faster implementation, but is slower in execution. C++ is more complex to code, but significantly faster and more efficient.
- **Usability:** Python is ideal for rapid development and prototyping, while C++ is better for performance-critical applications.
- **Libraries:** Both Python and C++ use OpenCV for image processing, but the way of integration is different. Python uses a high-level API that is easier to use, while C++ requires more detail in coding.
- **Productivity:** Python allows for rapid code iteration and modification, which is useful in research and academic environments. C++ is better for final versions of software that will be used in production.

## Conclusion

Both languages have their advantages and disadvantages. Python is excellent for development and testing due to its simplicity and speed of implementation, while C++ is the optimal choice for situations where performance is critical. The choice of language depends on the specific requirements of the project, but in many cases a combination of both languages can provide the best results, with Python used for development and testing and C++ for final implementation.

## Reference


1. Shanmugamani, R. (2018). Deep Learning for Computer Vision [E-book]. Amazon.in. https://www.amazon.in/Deep-Learning-Computer-Vision-techniques-ebook/dp/B072L1CG5X
2. Bradski, G., & Kaehler, A. (2019). Learning OpenCV 4: Computer Vision with OpenCV Library. IEEE Xplore. https://ieeexplore.ieee.org/document/5233425/
3. Lakshmanan, V., Gillard, R., & Seltzer, M. (2021). Practical Machine Learning for Computer Vision. O'Reilly Media. https://www.oreilly.com/library/view/practical-machine-learning/9781098102357/
4. Szeliski, R. (2020). Computer Vision: Algorithms and Applications. http://szeliski.org/Book/
5. Géron, A. (2022). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Amazon.in. https://www.amazon.in/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/9355421982
6. Prince, S. J. D. (2019). Computer Vision: Models, Learning, and Inference. https://udlbook.github.io/cvbook/
7. Sonka, M., Hlavac, V., & Boyle, R. (2014). Image Processing, Analysis, and Machine Vision.
8. Solem, J. E. (2020). Programming Computer Vision with Python. http://programmingcomputervision.com/downloads/ProgrammingComputerVision_CCdraft.pdf
9. Villán, A. F. (2019). Mastering OpenCV 4 with Python. Amazon.in. https://www.amazon.in/Mastering-OpenCV-Python-practical-processing/dp/1789344913
10. Bishop, C. (2006). Pattern Recognition and Machine Learning. Microsoft Research. https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/