Photoshopped Image Detection

A report submitted for the course named Project II (CS-300)

By

Mohd Tabish Nehal
Bachelor of Technology, VI Semester
Roll No. 17010116

Under the Supervision and Guidance of
Dr. Nongmeikapam Kishorjit Singh

Department of Computer Science and Engineering
Indian Institute of Information Technology Manipur
April, 2020

Abstract

In ancient times, images were used very rarely there was a possibility of less amount of forgery or no forgery in images. It requires much knowledge to create a forged image earlier. Nowadays, Images have gained a very vital importance in our daily life and it is not very difficult to make forged images because of the availability of powerful digital image editing software's that does not require any expert knowledge. So, it becomes very easy to create a tampered image. As a result we have to prove the authenticity of an image. Nowadays,it is very easy for the person to interact and edit the digital image for the reason ranging from financial gain to artistic expression.

However,the advancement of image manipulation technique introduced new methods of using technology for inappropriate purposes.Therefore,there is requirement of method or application that could detect the different types of manipulation done in the digital image.

To handle the different type of manipulation done in image,a system of detecting manipulation in image is introduced.This system uses Image processing to detect anomaly in the image.

Keywords - Image processing.

Declaration

I declare that this submission represents my idea in my own words and where others' idea or words have been included, I have adequately cited and referenced the original source. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/sources in my submission. I understand that any violation of the above will be a cause for disciplinary action by the institute and can also evoke penal action from the sources which have thus not been properly cited or from proper permission has not been taken when needed.

Signature

Mohd Tabish Nehal

Date: 17010116

Department of Computer Science & Engineering
Indian Institute of Information Technology Manipur

Dr. N. Kishorjit Singh
Assistant Professor

Email: kishorjit@iiitmanipur.ac.in

To Whom It May Concern

This is to certify that the report entitled **"Photoshopped Image Detection"** submitted to by "Mohd Tabish Nehal", has been carried out under my supervision and that this work has not been submitted elsewhere for a degree,diploma or a course.

Signature of Supervisor

Dr. Nongmeikapam Kishorjit Singh

Department of Computer Science & Engineering
Indian Institute of Information Technology Manipur

To Whome It May Concern

This is to certify that the project report entitled **"Photoshopped Image Detection"** submitted to the Department of Computer Science and Engineering, Indian Institute of Information Technology, Manipur, in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bona fide work carried out by Mr Mohd Tabish Nehal, Roll No. 17010116. No part of this report has been submitted elsewhere for award of any other degree.

Signature of HoD

Dr. Nongmeikapam Kishorjit Singh
Assistant Professor Head,
Department of CSE
IIIT,Manipur

Department of Computer Science & Engineering
Indian Institute of Information Technology Manipur

To Whom It May Concern

This is to certify that the project report entitled **"Photoshopped Image Detection"** submitted to the Department of Computer Science and Engineering, Indian Institute of Information Technology, Manipur, in partial fulfillment for the award of the degree of Bachelor of Technology in Computer Science and Engineering, is a record of bona fide work carried out by Mr Mohd Tabish Nehal, Roll No. 17010116. No part of this report has been submitted elsewhere for award of any other degree.

Signature of HoD

Dr. Nongmeikapam Kishorjit Singh

Signature of Examiner 1: _____

Signature of Examiner 2: _____

Signature of Examiner 3: _____

Acknowledgement

I would like to express my special thanks of gratitude to my Project Coordinator Dr. Kabita Thaoroijam and my Project Supervisor Dr. Nongmeikapam Kishorjit Singh for their support and valuable suggestions regarding the project work. I extend my sincere thanks to the faculty,family and friends for their constant encouragement and moral support in doing the project.

- Mohd Tabish Nehal

Contents

List of Figures

List of abbreviations

		A
ANMS	Adaptive Non Maximum Suppression	
		G
GUI	Graphical User Interface	
		H
HCDA	Harris Corner Detection Algorithm	
		R
RANSAC	Random Sample Consensus	
RAM	Random Access Memory	
		S
SSD	Sum of Squared Differences	

Chapter 1

Introduction

"To every thing there is a season, and a time to every purpose under the heaven: A time to be born, and a time to die; a time to plant, and a time to pluck up that which is planted: A time to kill, and a time to heal; a time to break down,and a time to build up; A time to weep, and a time to laugh; a time to mourn, and a time to dance; A time to cast away stones, and a time to gather stones together; a time to embrace, and a time to refrain from embracing;A time to get, and a time to lose; a time to keep, and a time to cast away; A time to rend,and a time to sew; a time to keep silence, and a time to speak; A time to love,and a time to hate; a time of war, and a time of peace."

– The Bible, Ecclesiastes 3:1-8

Nowadays,it is very easy to manipulate any type of data.The person can use different type of software available to perform the alteration in the data.In the digital age,the image can be easily be manipulated by adding or removing some element from image which in turn results in a high number of image forgeries.With continuous tremendous increase in digital image,the image editing software like Adobe Photo-shop are also being developed on large scale.Such type of software can alter the image by changing the blocks of image without showing the modification effect in the altered image.Human eyes are not able to identify such alteration in image.Therefore,it has become a challenging task to verify the digital image originality.

Images have become a powerful tool for communication nowadays as they are used everyday in newspapers, magazines, websites and advertisements and provide various information. As the use of images are increasing day by day but trust in images is decreasing day by day. Creating a fake image from original image is known as Image forgery and to check whether the image is original or fake is probably termed as Image forgery Detection or Photoshopped Image Detection. Moreover, digitally manipulated images can trigger off major controversies. Hence, there is a need for more sophisticated and mathematically sound techniques that can perform this task of classification of the image into real ones and digitally manipulated ones.

An image can be manipulated with a wide variety of manipulation techniques such as scaling, rotation, blurring, resampling, filtering, cropping, etc. Image photoshopped detection technique is needed in many fields for protecting copyright and preventing forgery. The verification of originality of images is required in variety of applications such as military, forensic, media, scientific, glamour, etc.Image tampering is a digital art which needs understanding of image properties and good visual creativity. Detection of image tampering deals with investigation on tampered images for possible correlations embedded due to tampering operations.Detecting forgery in digital images is a rising research field with important implications for ensuring the credibility of digital images.

With the rapid development of digital media editing techniques, digital image manipulation becomes rather convenient and easy. While it benefits to legal image processing, malicious users might use such innocent manipulations to tamper digital photograph images. Currently, image forgeries are widespread on the Internet and other security-related applications such as surveillance and recognition that utilize images are therefore impacted. The event and scene information delivered in images might become no longer believable. In the applications such as law enforcement and news recording, it is also necessary to verify the originality and authenticity of digital images, and make

clear the image manipulation history to get more information. To overcome such a problem, digital forensic techniques have been proposed to blindly verify the integrity and authenticity of digital images.

Generally, image manipulations could be classified into

1. Content-changing manipulations.
2. Content-preserving manipulations.

Accordingly, prior works on image manipulation forensics fall into two categories. As the first category, the forensics methods focus on detecting image tampering such as copy–move and splicing, by which the image content is reshaped arbitrarily according to semantic content. In the second category, common manipulations, as compression, blur and contrast enhancement are detected passively. These content-preserving manipulations are often applied as post processing to conceal the residual trail of malicious tampering operations and create realistic forgeries.

1.1 Literature Review

Digital image forgery is now a nightmare to individuals (e.g. fake images of celebrities and public figures), societies (fake images targeting religion or race), journalism, scientific publication etc. [1]. According to a survey, Flickr has some 350 million photographs with more than 1 million added daily (record 2007) and Facebook has more than 50 million cumulative upload of images (record 2010) [1]. The maintenance of such a large amount of digital data has become critical, complex and challenging.

S. Bayram, et al. [2] proposed a technique for the detection of doctoring in digital image. Doctoring typically involves multiple steps, which typically involve a sequence of elementary image-processing operations, such as scaling, rotation, contrast shift, smoothing, etc. The methodology used is based on the three categories of statistical features including binary similarity, image quality and wavelet statistics. The three categories of forensic features are as follows:

1. **Image Quality Measures**: These focuses on the difference between a doctored image and its original version. The original not being available, it is emulated via the blurred version of the test image.

2. **Higher Order Wavelet Statistics**: These are extracted from the multi-scale decomposition of the image.

3. **Binary Similarity Measure**: These measures capture the correlation and texture properties between and within the low significance bit planes, which are more likely to be affected by manipulations.

To deal with the detection of doctoring effects, firstly, single tools to detect the basic image-processing operations are developed. Then, these individual "weak" detectors assembled together to determine the presence of doctoring in an expert fusion scheme.

M. Stamm and K. Liu [3] focuses on recovering the possible information about the unmodified version of image and the operations used to modify it, once image alterations have been detected. An iterative method based on probabilistic model is proposed to jointly estimate the contrast enhancement mapping used to alter the image as well as the histogram of the unaltered version of the image. The probabilistic model identifies the histogram entries that are the most likely to occur with the corresponding enhancement artifacts.

Weiqi Luo [1] proposed an algorithm to extract image features by using seven characteristics features computed from the statistical analysis of pixels in an image block. The first three features are the average of red, green, and blue components respectively and the other four features are computed based on the division of that block into two parts in 4 directions: horizontal, vertical, and two diagonal directions. To obtain the correct matching, the main shift vector which has the highest frequency of occurrence is also defined. It gives low computational complexity and more robust against various post region duplication image processing graphics operations.

Reshma P.D and Arunvinodh C [4] describes a technique for detecting digital image forgery using illuminant color.In this paper, they presented an improved forgery detection method that makes use of machine learning classifiers. Training images texture and gradient features extracted and train the classifier with these features.Then classifier tests the image and made accurate.The main advantage of this work is that it completely avoids user interaction and provides a crisp Statement on the authenticity of the image.

Zhipenget al. [5] provides another method to detect image tampering. It detects global or local blur manipulation using no-reference image quality metric.It extracts image features from MSCN (mean subtracted contrast normalized) coefficients of different regions to quantify tampered regions.But it does not work good for images with poor resolution.

Fan et al. [6] proposed to correlate statistical noise features with exchangeable image file format header features for manipulation detection.

H. Cao et al. [7] designed a new ensemble manipulation detector to simultaneously detect a wide range of manipulation types on local image patches.

1.2 Summary

This chapter presents the various ways through which manipulations can be performed in digital image and how image forgery is used for malicious purposes.It also presents a brief review on different image manipulation detection techniques.

Chapter 2

Methodology and Design

Outline: This chapter presents the following:

1. A brief introduction about digital image forgery.
2. Algorithm used for detection of Copy Move forgery.
3. System Design.
4. Summary.

2.1 Digital Image Forgery

A digital image can be altered by various ways.It involves adding, changing, or deleting some important features from an image without leaving any obvious trace.Taking into account the methods used to make forged images, digital image forgery can be isolated into three primary classifications:

1. Copy-Move forgery.
2. Image resampling.
3. Image splicing.

2.1.1 Copy-Move forgery

In Copy-Move forgery,some part or object of image of any shape and size is copied and are pasted in the same image at another locations as shown in Figure 2.1. As the copied

(a) The original images (b) The copy-move forged images

Figure 2.1: Copy Move

part originated from the same image, its essential properties such as noise, color and texture don't change and make the recognition process troublesome.

2.1.2 Image Resampling

Image resampling is like physically changing the number of pixels in an image.It allows you to pick and choose the details you would like to include in an image.

2.1.3 Image Splicing

Image splicing is another type of image forgery where a new fake image is created using cut and paste system from one or more images.Figure 2.2 illustrates a sample of image splicing where spliced image (right-most) showing John Kerry and Jane Fonda together at an anti-Vietnam war rally.The left part is authentic image of Kerry and middle one is showing authentic image of Fonda.

Figure 2.2: Image Splicing

2.2 Proposed Method

As we discussed the different ways through which images can be altered, many authors have already worked on it and presented their different techniques of detecting forgery.I have chosen to work on detection of Copy Move Forgery.Since in Copy Move Forgery,a part of image is copied and pasted to different area in the same image so getting the accurate similar region is the key point and it can be achieved if the proposed method is able to perceive a method to recognize copy-move manipulated images and spot the duplicated region accurately.

The proposed method will use key-point detection technique for forgery detection.There are many key-point detection technique available like Harris Corner detection, Scale-Invariant Feature Transform etc.I will be using Harris Corner since it is easy to understand and simple to implement.The steps or methods which i am using in detection of Copy Move Forgery are given:

1. Harris corner detector algorithm [1].
   (a) Color to gray scale.
   (b) Spatial derivative calculation.
   (c) Structure tensor setup.
   (d) Harris response calculation.
   (e) Non-maximum suppression.
2. Feature Descriptor.
3. Matching.
4. Transformation Estimation.

2.2.1 Harris Corner Detector Algorithm

An interest point in an image is a point with well defined position and can be detected robustly. It means it does not change its characteristics even if the image gone through geometrical transformation or illumination adjustment. It can be a corner, an isolated point of local intensity maximum or minimum, a line ending or a point on a curve where the curvature is locally maximal.

The Harris corner detection algorithm uses the fact that, the intensity of image changes a lot in multiple directions at a corner, while it changes greatly in a certain direction at an edge. To formulate this anomaly of change in intensity, a local window is used to get the result from shifting the local window. When the window is shifted in an arbitrary direction around a corner point, the image intensity changes greatly. Around the edge point, it will change greatly when the window is shifted perpendicularly. By using this perception, Harris corner detection algorithm uses a second-order moment matrix as the basis of its corner decisions.

Before computing the auto-correlation matrix M, the derivatives of pixel intensity Ix and Iy will be computed.To find the interest point, we first compute the corner strength function which is harmonic mean of the matrix [8]

$$f_M(x,y) = \frac{detM_l(x,y)}{trM_l(x,y)} = \frac{\lambda 1 \lambda 2}{\lambda 1 + \lambda 2}$$

[1] https://en.wikipedia.org/wiki/Harris_Corner_Detector

where lambda1 and lambda2 are the eigenvalues of M.These points are selected as interest point where

$$f_M(x, y)$$

is a local maxima in window of 3×3 neighbourhood and above a threshold value. When the local-maxima is detected, its position will be refined to sub-pixel accuracy by fitting a 2D quadratic to the corner strength function in the local window of 3×3 neighbourhood and finding its maxima. The orientation can be detected by two methods, the first method is to find the eigen vectors which denote the largest eigenvalue of the Harris matrix M, and the second method is simply using the gradient at that point as its orientation. Here, i have used second method.The computational cost of matching is directly proportional to the number of interest points extracted from an image, and it should be restricted to minimize the computational cost of matching. At the same time, it is essential that the interest points should be spatially well distributed over the image. To get the above-mentioned requirements, adaptive non-maximal suppression (ANMS) will be used to select a fixed number of interest points from an image.

2.2.2 Feature Descriptor

Once the interest points have been found then extraction of descriptor for each interest point is performed.Basically, these descriptors are used to match against the descriptors of all others interest points to find a match.

There are many different descriptors are available. I will use the rotationally invariant box descriptor as it performs more accurately especially with respect to rotated features.The box descriptor will be created by sampling a 40x40 pixel window that is Gaussian blurred and then will be downsampled to an 8 x 8 window, which will be subsequently normalized such that the features are zero mean unit variance [9]. The down sampling gives robustness for the descriptor in matching to other descriptors. The normalization also helps to robustly match patches where there are changes in hue, saturation, and lighting.

2.2.3 Matching

After the extraction of descriptor, nearest neighbour with outlier rejection will be used to find match between corresponding interest points. And to find nearest neighbours of descriptor, Euclidean distance metric will be used.

Since a part of image can be copied and can be pasted at more than one locations, and we are not going to design algorithm to capture multiple copy-move regions of the same source. Instead, the algorithm will be designed in such a way that it can capture single copy-move regions of the same source. To capture single copy-move regions of the same source, algorithm will implement ratio matching called 2NN where the ratio between the first and second nearest neighbors for a given interest point is compared with a threshold value, which determines whether or not the match should be rejected or not due to its non-discernibility from other points [9].

2.2.4 Transformation Estimation

Once the candidate matches have been found as corresponding points, transformation estimation is used to identify the scaling, rotation, and translation to guess the copy-moved region of the image. To perform transformation estimations, a technique called random sample consensus (abbreviated RANSAC) will be used which acts as a filter for correspondence points that are not correlated with a transformation.

RANSAC randomly selects 6 interest points to compute a homography transformation in the hopes that it will produce the best transformation matrix [10]. Once the homography is computed, the sum of squared differences (SSD) is used to filter an inlier set based on an error threshold from all the corresponding points. In every subsequent homography computation, the size of the newly generated inlier set is to the size of the largest encountered inlier set thus far. If the new set is larger, the algorithm sets the new set to be the largest. This process repeats for a specified number of iterations.

2.3 System Design

The design of the proposed method is heavily based around the different type of algorithm used. The block diagram of Proposed Method is shown below. Once the image is imported, it passes through a series of algorithm to get to check it's originality. HCDA stands for Harris Corner Detection Algorithm.

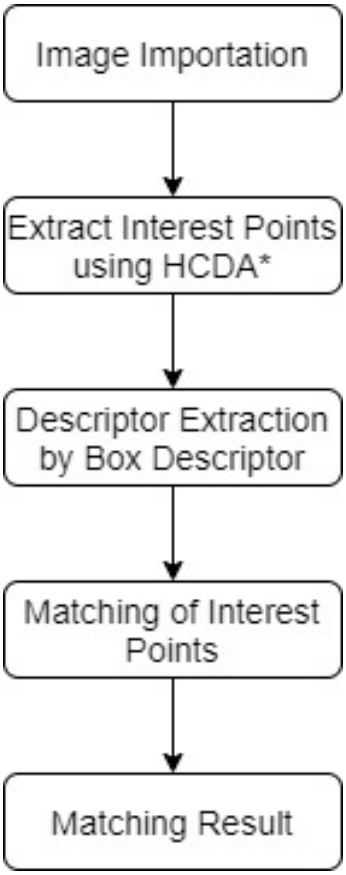


Figure 2.3: Block Diagram of Proposed Method

2.3.1 Operating Environment

Operating Environment for the proposed system is given as follows:

- Operating System: Windows, Linux.
- Platform: Python3.

2.3.2 Design and Implementation Constraints

The System will be developed in Python3. The Basic Libraries of Python3 will be used for computation purpose. There will be different module implemented which will communicate to each other through well defined communications.

2.3.3 Assumptions and Dependencies

Since the system will be developed using Python3 and it's basic libraries therefore requires Python3 and it's basic libraries to be installed on the user's system. The system requires latest version of both technology.

2.3.4 Hardware Interface

The Computer System having RAM 4 GB or more than that are compatible.

2.4 Summary

In this chapter, we described different digital image forgery such as copy move forgery, image resampling, image splicing.We also discussed how to extract interest points using harris corner detector algorithm, descriptors extraction corresponding to interest points and how matching, transformation estimations will be performed.

Chapter 3

System Analysis & Implementation

Outline: This chapter presents the following:

1. System Specification.
2. System Validation.
3. System Evolution.
4. Implementation.
5. Summary.

System Analysis or Software Analysis is the software engineering process which has following activities:

- Software Specification.
- Software Validation.
- Software Evolution.

3.1 Software Specification

3.1.1 Introduction

3.1.1.1 Intended Audience and Reading Suggestions

This project is a prototype for the Copy Move Forgery Detection system and it can be used by anyone who wants to detect copy move forgery in image. This has been implemented under the guidance of college professors. This project is useful for the image forensics team and as well as to the security team.

3.1.2 Overall Description

3.1.2.1 Product Perspective

The Copy Move Forgery Detection system imports the image and finds key points using HCDA.

3.1.3 External Interface Requirements

3.1.3.1 Hardware Interfaces

- Windows or Linux.
- Any browser which supports Computer Generated Image.

3.1.3.2 Software Interfaces

The System requires latest version of Python3 and it's basic libraries to be installed on the user's system.

3.1.3.3 Communications Interfaces

This project supports all types of web browsers. The system requires an internet connection to install new plugins, update already installed ones and update some of its components.

3.1.4 Other Nonfunctional Requirements

3.1.4.1 Software Quality Attributes

- **AVAILABILITY:** The system should be able to show the tampering of the image being searched.
- **CORRECTNESS:** The system should show correct tampering of the image.
- **USABILITY:** The system should be easy to use.

3.2 Software Validation

Software Verification and Validation (V & V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control.

Software validation checks that the software product satisfies or fits the intended use, that is the software meets the user requirements, not as specification artifacts or as needs of those who will operate the software only; but, as the needs of all the stakeholders.The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

3.3 Software Evolution

The need for software evolution comes from the fact that no one is able to predict how user requirements will evolve a prior. In other words, the existing systems are never complete and continue to evolve. As they evolve, the complexity of the systems will grow unless there is a better solution available to solve these issues. The main objectives of software evolution are ensuring functional relevance, reliability and flexibility of the system. Software evolution can be fully manual, partially automated or fully automated.

3.4 Implementation

This project was supposed to be designed in three parts where each part is responsible for different aspects.This project was implemented using Waterfall model as shown in figure. Detection of corner points using HCDA with non-maximum suppression was the first part to be designed and it has been completed without employing non-maximum suppression. Implementing ANMS was a difficult one and still it is.

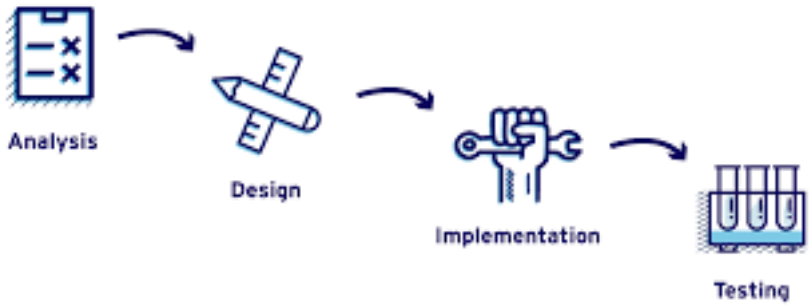


Figure 3.1: Waterfall Model

Extraction of descriptor for each interest point to be performed was second part to be designed and it has not been completed because of inability to implement ANMS. After the extraction of descriptor, matching of corresponding interest points using nearest neighbour with outlier rejection was the third part to be designed and it has not been completed because of incompleteness of descriptor extraction.

Moreover, the first part implemented is able to detect copy move forgery if present. The last two part were suppose to detect copy move forgery with accuracy and precision.

While implementing this project, the following gantt chart timeline as shown in figure 3.2 was followed.

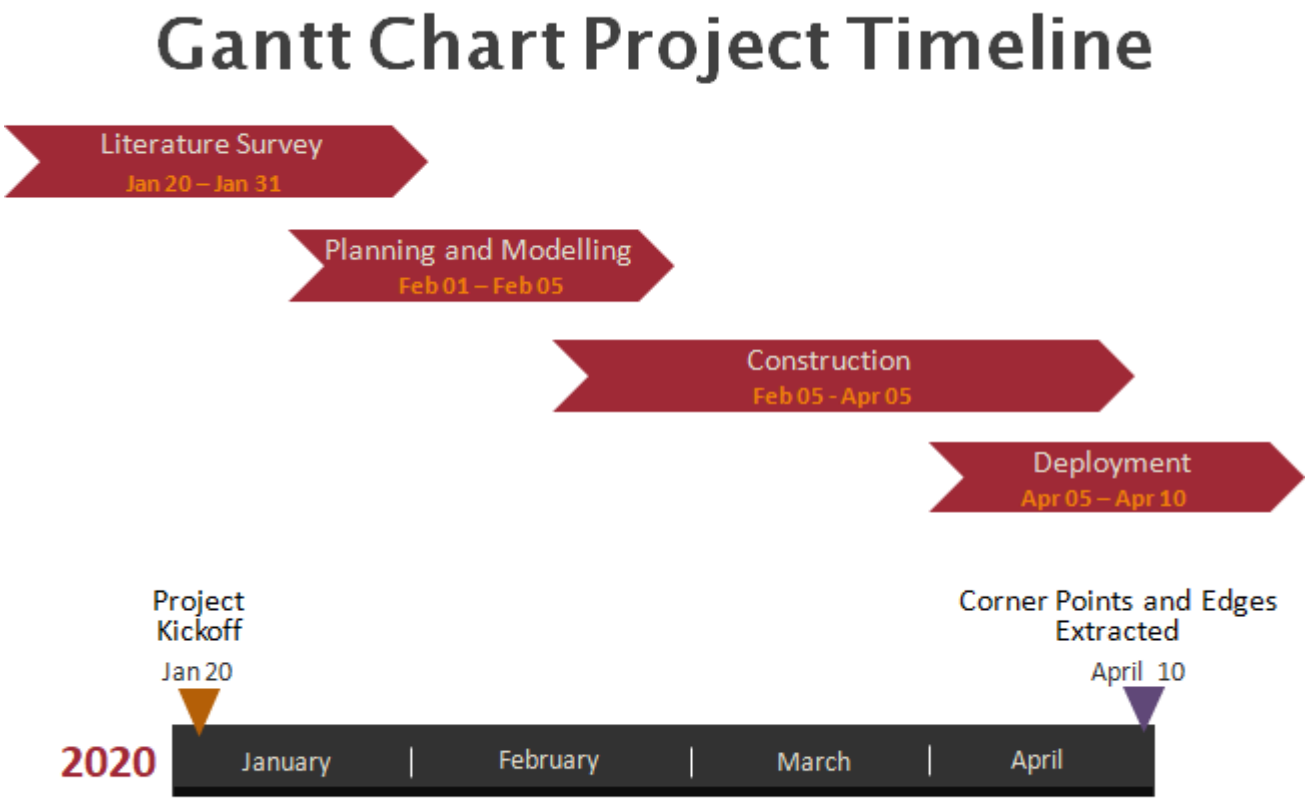


Figure 3.2: Project Timeline.

3.4.1 Color to gray scale

There are generally two method present to convert RGB image into gray scale:

- Average method
- Weighted method or Luminosity method

Here, Weighted method or luminosity method has been used because this method converts into gray scale properly as compared to Average method where we want to get result as

gray scale, but it gives a black image. Since red color has more wavelength of all the three colors, and green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes.

It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two. So, New gray scale image = ((0.3 * R) + (0.59 * G) + (0.11 * B)). According to this equation, Red has contribute 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%.

3.4.2 Spatial derivative calculation

Spatial derivative calculation can be performed using small convolution filters of size 2 x 2 or 3 x 3, such as the Laplacian, Sobel, Roberts and Prewitt operators. Here, i have used derivative mask of 3 x 3.

To find gradient of image in x direction, the following mask has been used:

[[-1 0 1]
[-1 0 1]
[-1 0 1]]

To find gradient of image in y direction, the following mask has been used:

[[1 1 1]
[0 0 0]
[-1 -1 -1]]

3.4.3 Structure tensor setup

Structure tensor setup is nothing but calculating auto-correlation matrix M of size 2 x 2 given by

[[Ixx Ixy]
[Ixy Iyy]]

where Ixx is convolution of product of gardient of image of x direction with that of x direction using gaussian kernel of size 3 x 3, Ixy is convolution of product of gardient

of image of x direction with that of y direction using gaussian kernel of size 3 x 3 and Iyy is convolution of product of gardient of image of y direction with that of y direction using gaussian kernel of size 3 x 3.

For calculating M, the following weighing function w(x,y) and gaussian kernel of size 3 x 3 has been used:

$$w(x,y) = g(x,y) = \frac{e^{\frac{-(x^2+y^2)}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

3.4.4 Harris response calculation

To calculate Harris response, we needed following values:

- detM, Determinant value of auto-correlation matrix M
- traceM = Ixx + Iyy
- k, k is sensitive factor to separate corners from edges and has value close to zero

For our purpose, we have chosen k = 0.05.

$$Harrisresponse = detM - k(traceM^2)$$

For value r in response, if r > 0, then the point is corner else if r < 0, then it is an edge.

3.5 Summary

In this chapter, we have discussed about different stages of Software or System Analysis like System Specification, System Validation and System Evolution. We also discussed the implementation part and difficulties while implementing.

Chapter 4

Result Analysis and Evaluation

Outline: This chapter presents the following:

1. Introduction
2. Evaluation
3. Issues
4. Summary

4.1 Introduction

Result Analysis is the process of identifying whether the software developed is giving the final product correctly as expected or not. In figure 4.1, I tried to read an image and output it to the console. The System worked as expected.

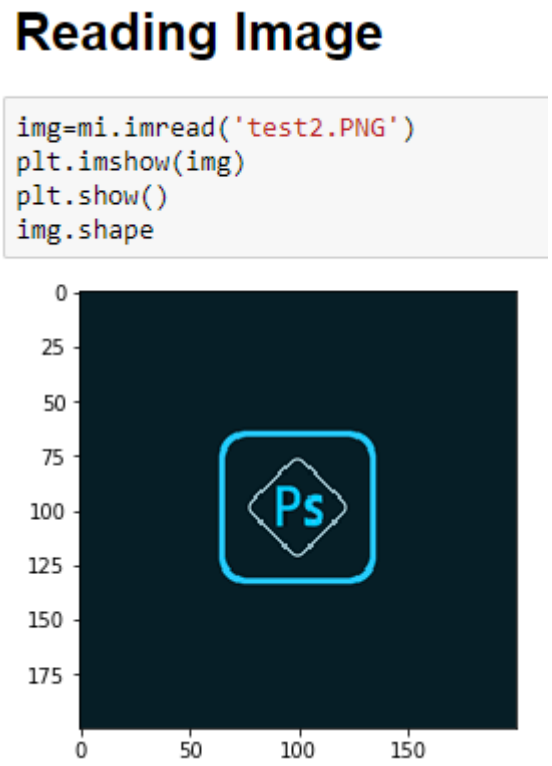


Figure 4.1: Original Image

The image being read is having 3 channel namely Red, Green and Blue as we have developed the system for this 3 channel image only.

Processing gray scale image has many benefits. It reduces computation time Therefore, i have firstly converted the 3 channel image into gray scale using Weighted method as shown in figure 4.2. We got the result as expected.

After color conversion to gray scale, I went for finding the first order derivative of image in X direction (horizontal direction). First order derivative of image in X direction is nothing but the change in intensity of image at every pixel in horizontal direction as shown in figure 4.3. We got the result as expected.

```
for i in range(height):
    for j in range(width):
        img2D[i][j]=(0.3*img[i][j][0] + 0.59*img[i][j][1] + 0.11*img[i][j][2])
```

```
plt.imshow(img2D,cmap='gray')
plt.show()
img2D.shape
```

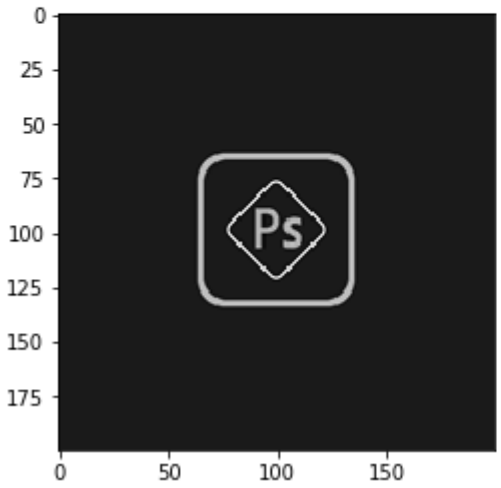


Figure 4.2: Gray scale Image

Similarly, after finding Gradient of Image in X direction, I went for finding the first order derivative of image in Y direction (vertical direction). First order derivative of image in Y direction is nothing but the change in intensity of image at every pixel in vertical direction as shown in figure 4.4. We got the result as expected.

After finding gradient of image in both direction (horizontal and vertical), auto-correlation matrix M has been calculated whose result has been shown in the code file itself because of large size of output.

Using this auto correlation matrix M, Harris corner response has been calculated for each pixel and from Harris corner response, corner points and edge have been detected as shown in figure 4.5.

Ix:Gradient of image in x direction

```
Ix=derivative(img2D,height,width,fx)
plt.imshow(Ix)
plt.show()
```

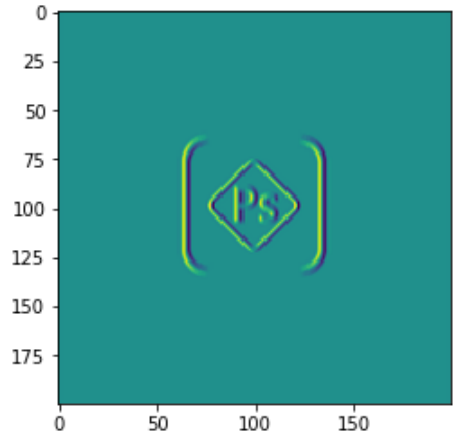


Figure 4.3: Gradient of Image in X (horizontal) direction

Iy:Gradient of image in y direction

```
Iy=derivative(img2D,height,width,fy)
plt.imshow(Iy)
plt.show()
```

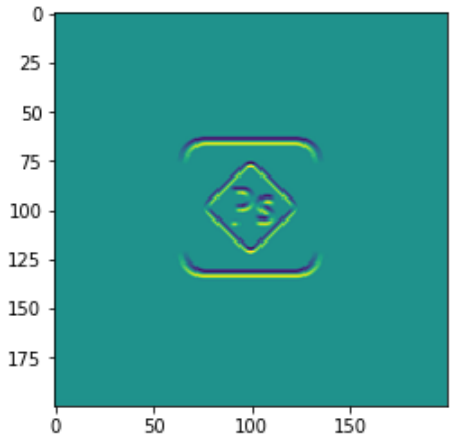


Figure 4.4: Gradient of Image in Y (vertical) direction

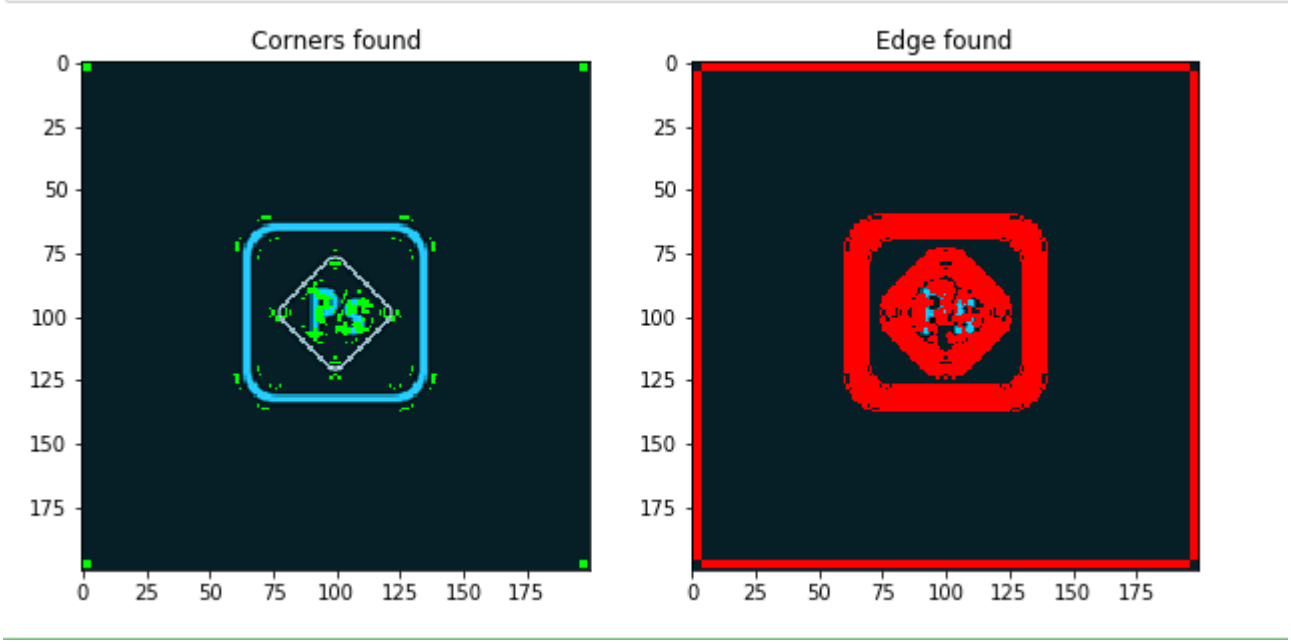


Figure 4.5: Corner points and Edges detected.

4.2 Evaluation

Now, suppose if there would have been an identical square shaped element at another place in same image then we would have got the same number of corner points and edges as shown for the already existing square shaped element as shown in figure 4.5. The system is not able to integrate the first part of implementation with the last two parts.

4.3 Issues

When the image has too many elements scattered all over the image and copy move forgery is present then it would not be possible for this system to perfectly identify the corner points and edges of the element which has been copy move forged.

4.4 Summary

In this chapter, we have discussed about the outputs and the extent up to which outputs are correct. We also discussed the evaluation part and the issues related to the system.

Chapter 5

Conclusion

Knowledge of specific domain improves the overall skill and result.This system has been implemented on the Jupyter Notebook, an open-source web application provided by Anaconda.The requirements and specification have been mentioned clearly.This project is implemented using Python3 and it's basic libraries.

The goal of this system to detect copy move forgery in digital image has been achieved but not with great accuracy as great things take time and underlying difficulties were solved. While implementing copy move forgery detection system, I learnt a great deal of knowledge and basics about Image Processing. The system works great for lightly populated image while it is not promising for densely populated image.

5.1 Future direction

- Graphical User Interface (GUI) can be integrated to the Copy Move forgery detection system.
- ANMS, Feature Descriptor and Transformatin Estimation can be integrated if allowed.
- This project can be put up on the cloud platform, so that it will be accessible by everyone.
- This project can be modified to process every type and number of channeled image.

Appendix A

Screenshot of the Implemented System

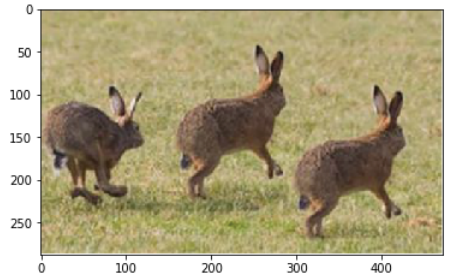
File Edit View Insert Cell Kernel Widgets Help Python [default]

Basic Libraries

In [1]:
```
import matplotlib.image as mi
import matplotlib.pyplot as plt
import numpy as np
```

Reading Image

In [2]:
```
img=mi.imread('rabbit.png')
plt.imshow(img)
plt.show()
img.shape
```



Out[2]: (288, 472, 3)

Figure A.1: Reading Image

File Edit View Insert Cell Kernel Widgets Help Python [default]

Size of Image

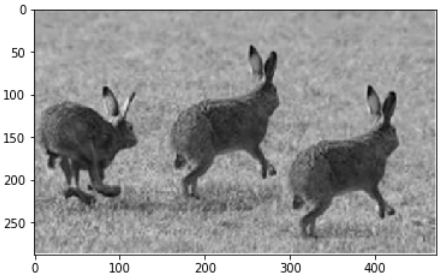
In [3]:
```
height=img.shape[0]
width=img.shape[1]
```

Color to grayscale

In [4]:
```
img2D=np.zeros((height,width))
```

In [5]:
```
for i in range(height):
    for j in range(width):
        img2D[i][j]=(0.3*img[i][j][0] + 0.59*img[i][j][1] + 0.11*img[i][j][2])
```

In [6]:
```
plt.imshow(img2D,cmap='gray')
plt.show()
img2D.shape
```



Out[6]: (288, 472)

Figure A.2: Color to gray scale

File Edit View Insert Cell Kernel Widgets Help Python [default]

Derivatives of pixel intensity:- Ix and Iy

In [7]:
```
#derivative masks or kernels
fx=np.array([(-1,0,1),(-1,0,1),(-1,0,1)])
fy=np.array([(1,1,1),(0,0,0),(-1,-1,-1)])
print(fx)
```

```
[[-1  0  1]
 [-1  0  1]
 [-1  0  1]]
```

In [8]:
```
def derivative(img2D,height,width,kernel):
    Id=np.zeros((height,width))
    m=0
    n=0
    sum_result=0
    for i in range(1,height-1):
        for j in range(1,width-1):
            for k in range(i-1,i+2):
                for l in range(j-1,j+2):
                    sum_result+=img2D[k][l]*kernel[m][n]
                    n=n+1
                n=0
                m=m+1
            m=0
            Id[i][j]=sum_result/3
            sum_result=0
    return Id
```

Figure A.3: Spatial gradient of image

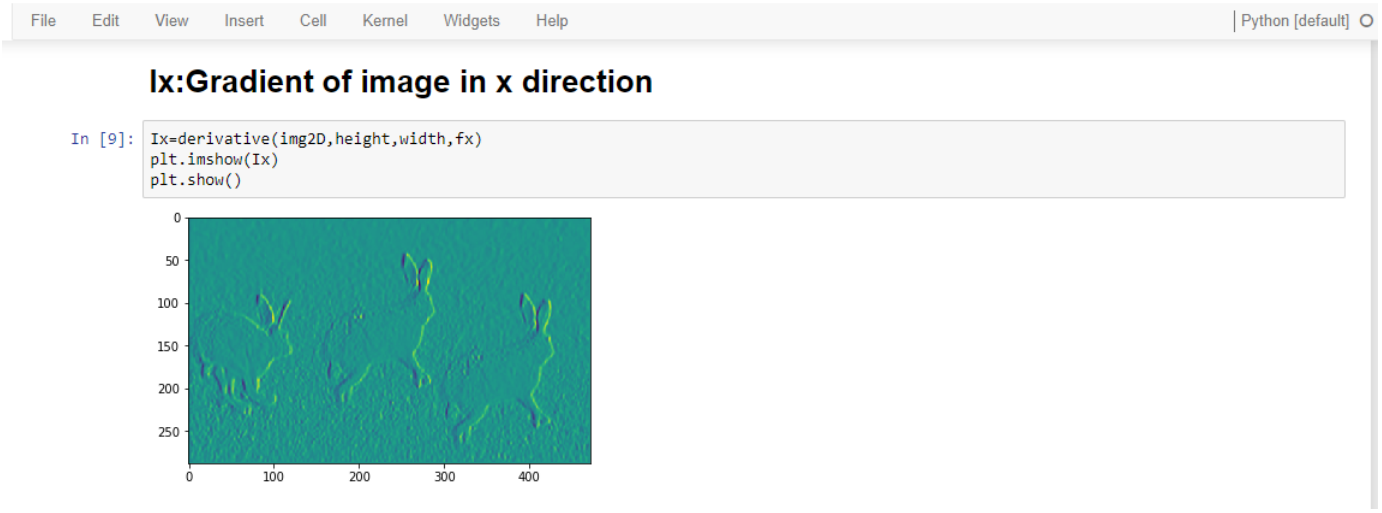


Figure A.4: Gradient of image in X or horizontal direction

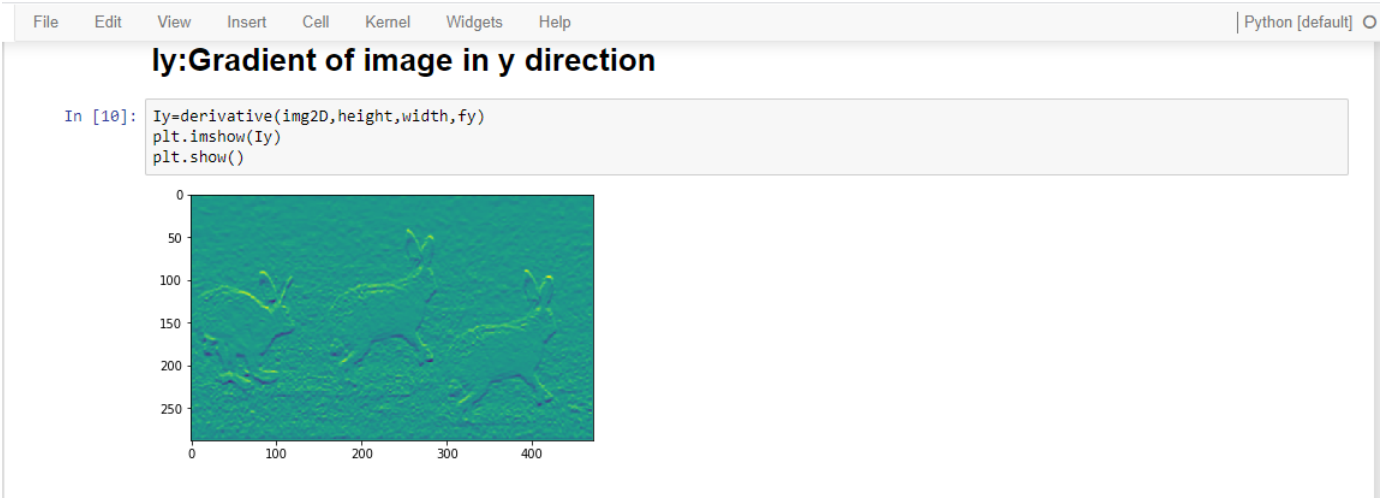


Figure A.5: Gradient of image in Y or vertical direction

File Edit View Insert Cell Kernel Widgets Help | Python [default]

Calculating Auto-correlation matrix

Weighing function w(x,y)=g(x,y,σ)=exp(−(x**2+y**2)/2(σ**2))/2Π(σ**2)

In [11]:
```
def weighingfn(x,y,σ):
    result=np.exp(-(x**2+y**2)/(2*(σ**2)))/(2*np.pi*(σ**2))**0.5
    return result
```

In [12]:
```
def gaussian_filter(derivative_product,σ):
    sums=0
    kernel=np.zeros((3,3))
    for y in range(-1,2,1):
        for x in range(-1,2,1):
            kernel[y+1][x+1]=weighingfn(x,y,σ)
            sums+=kernel[y+1][x+1]

    for i in range(3):
        for j in range(3):
            kernel[i][j]/=sums
    print(kernel)
    Idd=np.zeros((height,width))
    m=0
    n=0
    sum_result=0
    for i in range(1,height-1):
        for j in range(1,width-1):
            for k in range(i-1,i+2):
                for l in range(j-1,j+2):
                    sum_result+=derivative_product[k][l]*kernel[m][n]
                    n=n+1
                n=0
                m=m+1
            m=0
            Idd[i][j]=sum_result/3
                n=0
                m=m+1
            m=0
            Idd[i][j]=sum_result/3
            sum_result=0
    return Idd
```

Computing the sum of the products of derivatives at each pixel

In [13]:
```
Ixx=gaussian_filter(Ix**2,1)
```

```
[[0.07511361 0.1238414  0.07511361]
 [0.1238414  0.20417996 0.1238414 ]
 [0.07511361 0.1238414  0.07511361]]
```

In [14]:
```
Iyy=gaussian_filter(Iy**2,1)
```

```
[[0.07511361 0.1238414  0.07511361]
 [0.1238414  0.20417996 0.1238414 ]
 [0.07511361 0.1238414  0.07511361]]
```

Figure A.6: Structure tensor setup

Figure A.7: Elements of Structure tensor setup

File Edit View Insert Cell Kernel Widgets Help | Python [default]

In [15]:
```
Ixy=gaussian_filter(Ix*Iy,1)
```

```
[[0.07511361 0.1238414  0.07511361]
 [0.1238414  0.20417996 0.1238414 ]
 [0.07511361 0.1238414  0.07511361]]
```

Auto-correlation Matrix M=[[Ixx,Ixy],[Ixy,Iyy]]

In [16]:
```
M=[[Ixx,Ixy],[Ixy,Iyy]]
print(M)
```

```
[[array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
          0.        ],
         [0.        , 0.00124614, 0.00101823, ..., 0.00265782, 0.00406724,
          0.        ],
         [0.        , 0.00193955, 0.00156721, ..., 0.00378554, 0.00583255,
          0.        ],
         ...,
         [0.        , 0.00126605, 0.00150144, ..., 0.00200789, 0.0029613 ,
          0.        ],
         [0.        , 0.00069781, 0.00088706, ..., 0.00094375, 0.00134414,
          0.        ],
         [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
          0.        ]]), array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
           0.        ,  0.        ],
         [ 0.        , -0.00063249, -0.00068598, ...,  0.00085616,
           0.00097191,  0.        ],
         [ 0.        , -0.00049656, -0.00053855, ...,  0.0004045 ,
           0.00048488,  0.        ],
         ...,
         [ 0.        ,  0.00138499,  0.00190283, ..., -0.00099736,
          -0.00156289,  0.        ],
         [ 0.        ,  0.00169426,  0.00245547, ..., -0.00086606,
          -0.00144497,  0.        ],
```

Figure A.8: Output of Auto-Correlation Matrix

File Edit View Insert Cell Kernel Widgets Help | Python [default]

```
         ...,
         [ 0.        ,  0.00138499,  0.00190283, ..., -0.00099736,
          -0.00156289,  0.        ],
         [ 0.        ,  0.00169426,  0.00245547, ..., -0.00086606,
          -0.00144497,  0.        ],
         [ 0.        ,  0.        ,  0.        , ...,  0.        ,
           0.        ,  0.        ]])], [array([[ 0.        ,  0.        ,  0.        , ...,  0.        ,
           0.        ,  0.        ],
         [ 0.        , -0.00063249, -0.00068598, ...,  0.00085616,
           0.00097191,  0.        ],
         [ 0.        , -0.00049656, -0.00053855, ...,  0.0004045 ,
           0.00048488,  0.        ],
         ...,
         [ 0.        ,  0.00138499,  0.00190283, ..., -0.00099736,
          -0.00156289,  0.        ],
         [ 0.        ,  0.00169426,  0.00245547, ..., -0.00086606,
          -0.00144497,  0.        ],
         [ 0.        ,  0.        ,  0.        , ...,  0.        ,
           0.        ,  0.        ]]), array([[0.        , 0.        , 0.        , ..., 0.        , 0.        ,
          0.        ],
         [0.        , 0.00058379, 0.00094425, ..., 0.00113062, 0.00066183,
          0.        ],
         [0.        , 0.00036856, 0.00059667, ..., 0.00071628, 0.00041104,
          0.        ],
         ...,
         [0.        , 0.00498446, 0.0081301 , ..., 0.00919044, 0.00575991,
          0.        ],
         [0.        , 0.00796292, 0.01309118, ..., 0.01255226, 0.00800332,
          0.        ],
         [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
          0.        ]])]]
```

Figure A.9: Output of Auto-Correlation Matrix

File Edit View Insert Cell Kernel Widgets Help | Python [default]

Harris Response Calculation

In [17]:
```
response=[]
#Find determinant and trace of auto-correlation matrix i.e M, use to get corner response
# k is sensitive factor to separate corners from edges and has value close to zero
k=0.05
#determinant
detM = (Ixx*Iyy)-(Ixy**2)
# trace
traceM = Ixx + Iyy
response=detM-k*(traceM**2)
```

Figure A.10: Harris Response

File Edit View Insert Cell Kernel Widgets Help | Python [default]

Corner and Edge

In [18]:
```
image_for_corner=np.copy(img)
image_for_edge=np.copy(img)
for rowindex,response_value in enumerate(response):
    for colindex,r in enumerate(response_value):
        if r>0: # It is corner
            image_for_corner[rowindex,colindex]=[0,1,0]
        elif r<0: # It is an edge
            image_for_edge[rowindex,colindex]=[1,0,0]
fig,axis=plt.subplots(nrows=1,ncols=2,figsize=(10,10))
axis[0].set_title("Corners found")
axis[0].imshow(image_for_corner)
axis[1].set_title("Edge found")
axis[1].imshow(image_for_edge)
plt.show()
```

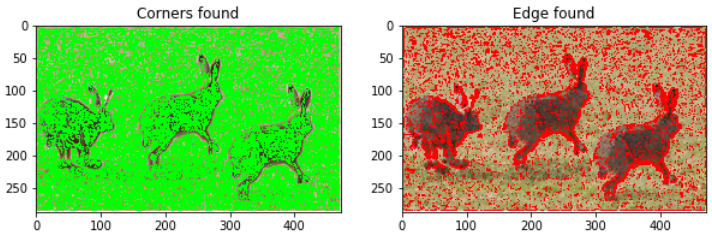


Figure A.11: Interest Points Extracted

Appendix B

User manual

B.1 Introduction

The system has been developed on Jupyter Notebook using Python3 and it's basic libraries. Therefore, it is necessary for users to have Python3 and it's basic libraries present on their system. It's not necessary for users to install Jupyter Notebook. It's up to their wish. Now, lets have a look on installation instructions.

B.2 Step to install implemented system

1. Download the zip file named Mohd Tabish(17010116-6th).zip and place it to your desired path on your system.
2. Unzip file.
3. Now, if you have Jupyter Notebook installed on your system:
   (a) Open command terminal.
   (b) Go to the directory where you have stored extracted zip file and go to the folder code.
   (c) Enter jupyter notebook on terminal and after some time a tab will open on your default web browser.
   (d) Open main.ipynb from the code folder. This is the main file of project where you will get to see output.
   (e) Now run main.ipynb file. If you would have every required and basic libraries of python installed, you will encounter no error.
4. Now, if you do not have Jupyter Notebook but have Python shell:
   (a) Open command terminal.

(b) Go to the directory where you have stored extracted zip file and go to the folder code.

(c) Enter the following command in the terminal without quotes "python main.py".

(d) If you would have every required and basic libraries of python installed, you will encounter no error.

(e) You will see every output one by one after you close the output console one by one.

5. If you want to see the output on other images also then first put your image in the Image folder which is in Code folder then after that on this line mi.imread('Image/rabbit.png') in main.ipynb, replace rabbit.png with your image name with careful consideration of uppercase and lowercase.

Bibliography

[1] J. H. Luo, Weiqi and G. Qiu, "Robust detection of region-duplication forgery in digital image," *ICPR 2006. 18th International Conference,Pattern Recognition, 2006 on*, vol. 4 IEEE, no. 3, 2006.

[2] B. S. S. Bayram, I. Avcubas and N. Memon, "Image manipulation detection," *J. Electron. Imag.*, vol. 15, no. 4, pp. 04 110 201—04 110 217, 2006.

[3] M. C. Stamm and K. J. R. Liu, "Forensic estimation andreconstruction of a contrast enhancement mapping," *IEEE Int. Conf. Acoust., Speech Signal, Dallas, TX, USA,*, vol. 8, no. 4, pp. 1698—1701, Mar. 2010.

[4] R. P.D and A. C, *IMAGE FORGERY DETECTION USING SVM CLASSIFIER.* IEEE International Conference on Innovations in Information, Embedde and Communication Systems (ICIIECS), 2015.

[5] Y. Zhipeng, Chen and R. Ni, *Forensics blurred images based on no-reference image quality assessment.* in IEEE, 2013.

[6] H. C. J. Fan and A. C. Kot, "Estimating exif parameters based on noise features for image manipulation detection," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 4, pp. 608—618, Apr. 2013.

[7] H. Cao and A. C. Kot, "Manipulation detection on image patches using fusionboost," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 992—1002, Jun. 2012.

[8] C. S. Prakash, H. Om, S. Maheshkar, and V. Maheshkar, "Keypoint-based passive method for image manipulation detection," *Cogent Engineering*, vol. 5, no. 1, p. 1523346, 2018.

[9] R. S. M. Brown and S. Winder., *Multi-Image Matching using Multi-Scale Oriented Patches.* International Conference on Computer Vision and Pattern Recognition (CVPR2005) pp. 510-517.

[10] M. Fischler and R. Bolles., "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM.*, vol. 24, Issue 6, June 1981. 1 June 1981.