Image Enhancement

Skill Project Report

Silicon Institute of Technology, Bhubaneswar

Naman Agrawalla	Reg. No. 1601209430
Kashif Nehal	Reg. No. 1601209427
Arpit Pati	Reg. No. 1601209055
Satyam Sovan Mishra	Reg. No. 1601209440
Subrat Kumar Patra	Reg. No. 1601209220
(B.Tech. 4th sem. CSE-B2)	

April 23, 2018

Contents

List of Figures

Abstract

The field of Digital Image Processing refers to processing digital images by means of digital computer. One of the main application areas in Digital Image Processing methods is to improve the pictorial information for human interpretation. Most of the digital images contain noise. This can be removed by many enhancement techniques. Filtering is one of the enhancement techniques which is used to remove unwanted information (noise) from the image. It is also used for image sharpening and smoothening. The aim of this project is to demonstrate the filtering techniques by performing different operations such as smoothening, sharpening, removing the noise etc. This project has been developed using Python language because of its universal acceptance and easy understandability.

Chapter 1

Digital Image Processing

Digital image processing is the use of computer algorithms to perform image processing on digital images. As a subcategory or field of digital signal processing, digital image processing has many advantages over analogue image processing. It allows a much wider range of algorithms to be applied to the input data and can avoid problems such as the build-up of noise and signal distortion during processing. Since images are defined over two dimensions (perhaps more) digital image processing may be modelled in the form of multidimensional systems. Digital filters are used to blur and sharpen digital images. Filtering can be performed in the spatial domain by convolution with specifically designed kernels (filter array), or in the frequency (Fourier) domain by masking specific frequency regions.

1.1 Advantages

Digital Image Processing in the most layman terms is image editing to improve it's visual appearance but not limited to it. The main advantages are:-

Important features such as edges can be extracted from images which can be used in industry.

Images can be given more sharpness and better visual appearance.

Minor error can be rectified.

Image size can be increased and decreased.

Images can be compressed and decompressed for faster image transfer over the network.

Unrecognisable features can be prominent.

Images can be smoothened.

It allows robots to have vision.

Removing noise from the images.

Sharpening of image.

1.2 Application in Real World

There are many application of image processing in the real world.Some of them are listed below:-

Image Enhancement, Image Restoration, Character recognition, Signature verification, Biometrics, Fingerprint Verification, Face Detection, Medical Application, Digital Cinema, Image Transmission and coding, Robot vision, Hybrid Techniques, Pattern Recognition, Video Processing, and Weather Forecasting.

Chapter 2

Problem Statement

2.1 Goal

The goal of this project is to incorporate an image processing using various image processing technique like deblurring, denoising, exposure correction, high filter boosting ,etc. to increase the quality of the image for the viewer. The image processing unit will primarily perform many tasks improving the contrast of the image and removing any artefacts and noise introduced as a result of contrast enhancement.

Give the students a general understanding of the fundamentals of digital image processing. Introduce the student to analytical tools which are currently used in digital image processing as applied to image information for human viewing.Develop the students ability to apply these tools in the laboratory in image restoration, enhancement and compression.

2.2 Scope

Image processing is being applied in many fields in today's world,

Automotive sector: In developing advanced drivers assist for semi-autonomous cars and also heavily used in autonomous/driver-less cars.

Image enhancing: The camera applications in smart phones and digital cameras using image processing to enhance the image quality, video stabilization and noise removal etc.

Robotics: Mobile robot's navigation in unknown environment (SLAM), control of the robot by processing the video feed from the camera on robot to extract the live scene around it.

Gaming: Advanced gaming consoles like Xbox Kinect uses image processing from motion analysis of the human player.

Problem specific solutions: Image processing is used as a solution to a variety of problems, starting from facial recognition access to defects identification in manufacturing industries.

Manufacturing: To identify defects in the processes and also to control the robots in performing certain tasks. for ex. defects in manufacturing of a Printed Circuit Board (PCB) can be observed using high resolution image processing.

Human machine interface: Machines are made smart by adding gestural interface, or human action response interfaces, which decodes the actions of the human user to perform certain tasks.

2.3 Requirement Specification

In our project we require various PYTHON library..

opencv library
matplotlib library
skimage library
PIL library
scipy library
numpy library

Chapter 3

Methodology

3.1 Denoising

3.1.1 Noise

Image noise is random variation of brightness or color information in images, and is usually an aspect of electronic noise. It can be produced by the sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector. Image noise is an undesirable by-product of image capture that obscures the desired information.

Image noise can range from almost imperceptible specks on a digital photograph taken in good light, to optical and astronomical images that are almost entirely noise, from which a small amount of information can be derived by sophisticated processing. Such a noise level would be unacceptable in a photograph since it would be impossible even to determine the subject.

3.1.2 Removal of noise

Noise is generally considered to be a random variable with zero mean. Consider a noisy pixel, $\mathbf{P = P0 + n}$ where P0 is the true value of pixel and n is the noise in that pixel. You can take large number of same pixels (say N) from different images and computes their average. Ideally, you should get $\mathbf{p = P0}$ since mean of noise is zero.

Linear filtering can be used to remove certain types of noise. Certain filters, such as averaging or Gaussian filters, are appropriate for this purpose. For example, an averaging filter is useful for removing grain noise from a photograph. Because each pixel gets set to the average of the pixels in its neighbourhood, local variations caused by grain are reduced.

The methods that we used for removing the noise from the image is **cv2.medianblur**

cv2.medianblur

The function cv2.medianBlur() takes median of all the pixels under kernel area and central element is replaced with this median value. This is highly effective

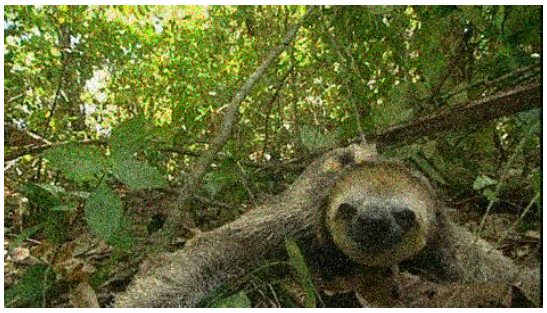

Figure 3.1: Noisy Image

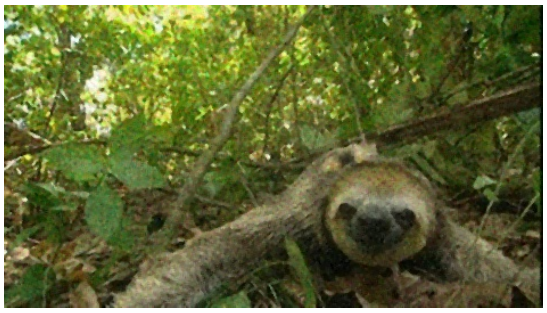

Figure 3.2: After denoising

against salt-pepper noise in the images.This method takes two parameter that is,the image and kernel size.In median blurring, central element is always replaced by some pixel value in the image. It reduces the noise effectively. Its kernel size should be a positive odd integer.

3.1.3 Output

Refer figure 3.1 and 3.2 for before and after denoising.

3.2 Exposure correction

Exposure is the amount of light per unit area (the image plane illuminance times the exposure time) reaching a photographic film or electronic image sensor, as determined by shutter speed, lens aperture and scene luminance. Exposure correction is a technique for adjusting the exposure indicated by a photographic exposure meter, in consideration of factors that may cause the indicated exposure to result in a less-than-optimal image.

Each pixel in a image has brightness level, called luminance. This value is between 0 to 1, where 0 means complete darkness (black), and 1 is brightest (white).Different camera or video recorder devices do not correctly capture luminance. (they are not linear) Different display devices (monitor, phone screen, TV) do not display luminance correctly neither. So, one needs to correct them, therefore the gamma correction function.Gamma correction function is used to

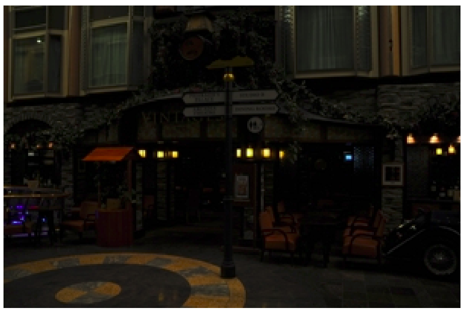
Figure 3.3: Before exposure correction

correct image's exposure.Thus increasing gamma will give darker image and decreasing gamma will give a lighter image.The method which we defined using for exposure correction is are scaling the pixel intensities to the range [0, 1.0],then applying the transform, and then scaling back to the range [0, 255].

3.2.1 Method Used

OpenCV provides the method is:

cv2.cvtColor(src, code, dst, dstCn)

Parameters:

src: input image: 8-bit unsigned, 16-bit unsigned , or single-precision floating-point.

dst: output image of the same size and depth as src.

code: color space conversion code (see the description below).

dstCn: number of channels in the destination image; if the parameter is 0, the number of the channels is derived automatically from src and code.

Various color space we used for exposure correction are:

1. RGB(red, green, blue)

RGB (red, green, and blue) refers to a system for representing the colors to be used on a computer display. Red, green, and blue can be combined in various proportions to obtain any color in the visible spectrum. Levels of R, G, and B can each range from 0 to 100 percent of full intensity. Each level is represented by the range of decimal numbers from 0 to 255 (256 levels for each color), equivalent to the range of binary numbers from 00000000 to 11111111, or hexadecimal 00 to FF. The total number of available colors is 256 x 256 x 256, or 16,777,216 possible colors.

2. HSV(hue, saturation, value)

Hue: the dominant wavelength, the redness of red, greenness of green, etc. Saturation: how pure the color is, or how much white is contained in the color. For example, red and royal blue are more saturated than pink and sky blue, respectively. Luminance: the amount or intensity of light or control the brightness of the image.

3.2.2 Output

Refer figure 3.3 and 3.4 for before exposure correction and after exposure correction.

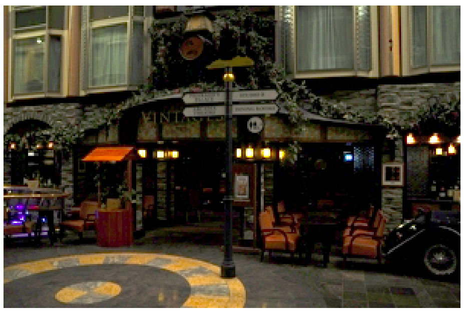

Figure 3.4: After exposure correction

3.3 Boosting

In image processing, it is often desirable to emphasize high frequency components representing the image details without eliminating low frequency components (such as sharpening). The high-boost filter can be used to enhance high frequency component.Here we sharpen edges of a image through the amplification and obtain a more clear image.The high-boost filter is a simple sharpening operator in signal and image processing.We are also enhancing the image further by a sharpening filter.Thus a boosted image will always look sharp and crisp with popped up colors from original one. Another way of making a highpass filter is to simply subtract a lowpass filtered image from the original.Thus we subtract the gaussian lowpass of image from the original image, to get an equivalent highpass filter. That's what's referred to as a ”gaussian high pass”.

3.3.1 Method Used

The method used for boosting is:

cv2.filter2D(src, ddepth, kernel, dst, anchor, delta, borderType)

Parameters:

src :- input image.

dst :- output image of the same size and the same number of channels as src.

ddepth :- desired depth of the destination image.

kernel :- convolution kernel.

anchor :- anchor of the kernel that indicates the relative position of a filtered point within the kernel; the anchor should lie within the kernel; default value (-1,-1) means that the anchor is at the kernel center.

delta :- optional value added to the filtered pixels before storing them in dst.

3.3.2 Output

Refer figure 3.5 and 3.6 for before and after boosting.

3.4 Deblurring

Blur is a common and unwanted artefact of image acquisition. There are many reasons why images become blurred such as movement, slow shutter speed or

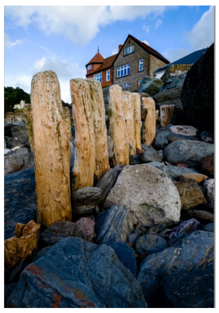

Figure 3.5: Before Boosting

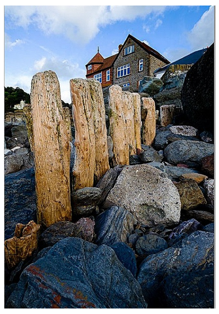

Figure 3.6: After Boosting

incorrect focal distance. Because blurry images are confusing, they are often less appealing and so it is desirable to deblur images for purely aesthetic reasons. Outside of photography, variable blurring creates problems for feature tracking algorithms which rely on the existence of consistent image structures. Blur removal may be used to precondition these algorithms and greatly improve performance.

The most common approach of blur removal is to treat blur as a noisy convolution operation. This model has both its advantages and disadvantages which we discuss in greater detail in the subsequent sections. From the perspective of convolution, there are two basic types of blur removal i.e non-blind and blind, with the main difference being that in the case of non-blind deconvolution the blur kernel is known.We try to measure the amount of bluriness in the image by taking the variance of laplacian of image.We use the method cv2.Laplacian().var() to estimate the bluriness in the image. Let blurred image matrix is **b** blur kernel matrix is **k** original image matrix is [o]

$$b = conv(o, k)$$

So, original image can be recovered by ,

$$o = deconv(b, k)$$

3.4.1 Richardson-Lucy Method

Richardson and Lucy both independently discovered what is now referred to as the Richardson- Lucy (RL) deconvolution method (Richardson discovered it first and then Lucy discovered on his own soon after). RL deconvolution, unlike the Wiener method, is more robust to noise because it explicitly models it as a Poisson process. Although this does not very accurately reflect real noise, it works better than the power spectrum used by the Wiener method.We are also using the Wiener filter for getting a better result.Then finally we are sharpening and boosting image a bit which will make the deblurred image look even more crisp and sharper.

3.4.2 Method Used

The methods used in deblurring are:

1.**cv2.Laplacian**(src, ddepth, dst, ksize, scale, delta, borderType)

The parameters are:

src : Source image.

dst: Destination image of the same size and the same number of channels as src .

ddepth: Desired depth of the destination image.

ksize: Ap erture size used to compute the second-derivative filters. See getDerivKernels for details. The size must be positive and odd.

scale: Optional scale factor for the computed Laplacian values. By default, no scaling is applied. See getDerivKernels for details.

delta: Optional delta value that is added to the results prior to storing them in dst .

borderType: Pixel extrapolation method.

2.**restoration.wiener**(img, psf, clip)

the parameters used here are:

img: The source image

psf: the point spread function

clip:True by default. If True, pixel values of the result above 1 or under -1 are thresholded for skimage pipeline compatibility.

3.**restoration.richardson-lucy**(image, psf, iterations=50, clip=True)

the parameters used are:

image: The source image

psf: the point spread function

iterations: Number of iterations. This parameter plays the role of regularisation.

clip: True by default. If true, pixel value of the result above 1 or under -1 are threshold for skimage pipeline compatibility.

4.**gaussian-filter**(input, sigma)

the parameters used are:

input: the input image to filter

sigma: scalar or sequence of scalars Standard deviation for Gaussian kernel. The standard deviations of the Gaussian filter are given for each axis as a sequence, or as a single number, in which case it is equal for all axes.

3.4.3 Output

Refer figure 3.7 and 3.8 for before and after deblurring.

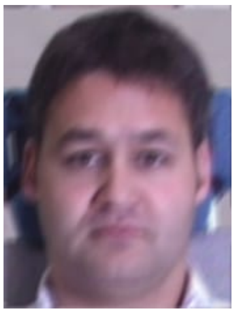

Figure 3.7: Before deblurring

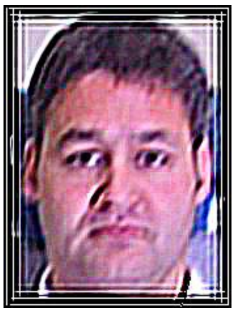

Figure 3.8: After deblurring

Chapter 4

Conclusion

This project has helped us to learn about different aspects of image processing. We have used various processing techniques like Denoising, Boosting, Exposure correction, Deblurring. This processes helped us to process a noisy/blurred image to a enhanced image. This processes were implemented using various *Python* libraries. Finally an enhanced image was produced. We could have achieved even higher accuracy percentage, although this project may not make any significant contribution in the any research or real world problems but the experience will be definitely be valuable for us.

Appendix A

Denoising

```
#Final Denoising

import cv2
import matplotlib.pyplot as plt
img = cv2.imread('/Users/satyamsovan123/Desktop/i.png')
img1=cv2.medianBlur(img,ksize=3)
cv2.imwrite('/Users/satyamsovan123/Desktop/img1.jpg',img1)
```

Appendix B

Exposure Correction

```
#Final Exposure Correction

import cv2
import numpy as np
from PIL import ImageEnhance,Image

def adj_gamma(image, gamma):

    invGamma = 1.0/ gamma
    table = np.array([((i / 255.0) ** invGamma) * 255
    for i in np.arange(0, 256)]).astype("uint8")
    return cv2.LUT(image, table)

x = '/Users/satyamsovan123/Desktop/rough/o3.png'  #location of the image
original = cv2.imread(x)
gamma = 1.6
adjusted = adjust_gamma(original, gamma=1.6)
cv2.imwrite('/Users/satyamsovan123/Desktop/rough/o4.png',adjusted)

img1=Image.open('/Users/satyamsovan123/Desktop/rough/o4.png')
contrast = ImageEnhance.Contrast(img1)
cntr=contrast.enhance(1.5)
cntr.save('/Users/satyamsovan123/Desktop/rough/o5.png')

img2=Image.open('/Users/satyamsovan123/Desktop/rough/o5.png')
bright=ImageEnhance.Brightness(img2)
brght=bright.enhance(1.2)
brght.save('/Users/satyamsovan123/Desktop/rough/o6.png')
```

Appendix C

Boosting

```
#Final Boosting

import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage
from PIL import Image
import cv2
from PIL import ImageEnhance,Image


# Load the data...
im = cv2.imread('/Users/satyam/Desktop/rough/o6.png')
data = np.array(im, dtype=int)
lowpass = ndimage.gaussian_filter(data, 10)
gauss_highpass = 1.5*data - (0.55)*lowpass
cv2.imwrite('/Users/satyam/Desktop/rough/output1.png',gauss_highpass)

imr=Image.open('/Users/satyam/Desktop/rough/output1.png')
sharpness = ImageEnhance.Sharpness(imr)
shrp=sharpness.enhance(1.0)
shrp.save('/Users/satyam/Desktop/rough/o7.png')
```

Appendix D

Deblurring

```
#Final Deblurring

import numpy as np
import matplotlib.pyplot as plt
import cv2
from scipy import misc
from scipy.signal import convolve as conv2
from skimage import color, data, restoration
from PIL import ImageEnhance, Image
from scipy import ndimage
img32=misc.imread('/Users/satyam/Desktop/a.png')
img = cv2.imread('/Users/satyam/Desktop/a.png')
b,g,r=cv2.split(img)

#Detecting amount of bluriness in image
bluriness_img=cv2.Laplacian(img, cv2.CV_64F).var()
print(bluriness_img)

#cv2 uses BGR
imgb=b #Blue Channel
imgg=g #Green Channel
imgr=r #Red Channel

#point spread function
psf = (np.ones((5,5)) / 25.0)

# Restore Image using Wiener
deconv_b=restoration.wiener(imgb,psf,1,clip=False)
deconv_g=restoration.wiener(imgg,psf,1,clip=False)
deconv_r=restoration.wiener(imgr,psf,1,clip=False)

# Restore Image using Richardson-Lucy
deconv_b= restoration.richardson_lucy(deconv_b,psf,iterations=30,clip=False)
deconv_g= restoration.richardson_lucy(deconv_g,psf,iterations=30,clip=False)
deconv_r= restoration.richardson_lucy(deconv_r,psf,iterations=30,clip=False)
```

```
a=4
deconv_b=deconv_b+a*(deconv_b-ndimage.gaussian_filter(deconv_b,1))
deconv_g=deconv_g+a*(deconv_g-ndimage.gaussian_filter(deconv_g,1))
deconv_r=deconv_r+a*(deconv_r-ndimage.gaussian_filter(deconv_r,1))

#Edge Detection
edges = cv2.Canny(img,250,400)
#Formula (getting required content from edge)
alpha = 1.2
ed = (255-edges)/255.0*alpha+1-alpha

db_b=cv2.medianBlur(deconv_b*ed,1)
db_g=cv2.medianBlur(deconv_g*ed,1)
db_r=cv2.medianBlur(deconv_r*ed,1)

#Adding Deconvoled and edges
Img_b = cv2.addWeighted(ed,0.1,deconv_b*ed,0.8,1)
Img_g = cv2.addWeighted(ed,0.1,deconv_g*ed,0.8,1)
Img_r = cv2.addWeighted(ed,0.1,deconv_r*ed,0.8,1)

#writing red blue and green matrix
#cv2.imwrite('/Users/satyam/Desktop/b.png',Img_b)
#cv2.imwrite('/Users/satyam/Desktop/g.png',Img_g)
#cv2.imwrite('/Users/satyam/Desktop/r.png',Img_r)


#Merging all 3 channels
x=cv2.merge((Img_b,Img_g,Img_r))
#y=cv2.imread('/Users/satyam/Desktop/dfd.png')
cv2.imwrite('/Users/satyam/Desktop/o1.png',x)

increasing the brightness and contrast after few iterations
def adjust_gamma(image, gamma):

    invGamma = 1.0/ gamma
    table = np.array([((i / 255.0) ** invGamma) * 255
    for i in np.arange(0, 256)]).astype("uint8")
    return cv2.LUT(image, table)

y = '/Users/satyam/Desktop/o1.png'
original = cv2.imread(y)
#cv2.imshow('original',original)

gamma = 2.5 #
adjusted = adjust_gamma(original, gamma=1.6)
#cv2.imshow("gammam image 1", adjusted)
cv2.imwrite('/Users/satyam/Desktop/o2.png',adjusted)
```

```
import matplotlib.pyplot as plt
import numpy as np
from scipy import ndimage
from PIL import Image
import cv2
from PIL import ImageEnhance,Image


# Load the data...
im = cv2.imread('/Users/satyam/Desktop/o2.png')
data = np.array(im, dtype=int)
lowpass = ndimage.gaussian_filter(data, 10)
gauss_highpass = 1.5*data - (0.55)*lowpass
cv2.imwrite('/Users/satyam/Desktop/o3.png',gauss_highpass)

imr=Image.open('/Users/satyam/Desktop/o3.png')
sharpness = ImageEnhance.Sharpness(imr)
shrp=sharpness.enhance(7)
shrp.save('/Users/satyam/Desktop/osad4.png')

plt.subplot(1,2,1)
plt.imshow(img32)
plt.subplot(1,2,2)
plt.imshow(shrp)
plt.show()
```