Laboratory Manual

For

Digital Image Processing

(MF211)

M.Tech (IT)

SEM II

December 2014

Faculty of Technology
Dharmsinh Desai University
Nadiad
www.ddu.ac.in

Table of Contents

EXPERIMENT-1

Aim: Understand the python with Canopy Editor, Python syntax and writing a program to read image and perform different operations on image also generate histogram of an image.

Tool / Apparatus: Canopy

Common Procedure:

Step 1: Create a folder in Home directory with your Id Number or Name Followed by RollNo.

Step 2: Go to terminal and type a following commands

- ➔ cd /home/user1/Canopy/
- ➔ ./canopy

An Canopy Editor will be start.

Step 3: Click on File Menu → New Editor Window. In Window you will find a button "Create a new file" click on it. New (.py) file will be created. Again go to File Menu →Save As… ,save file to your directory.

Step 4: write your code in Editor Window and run it by pressing a play button in menu bar.

Write a source code:

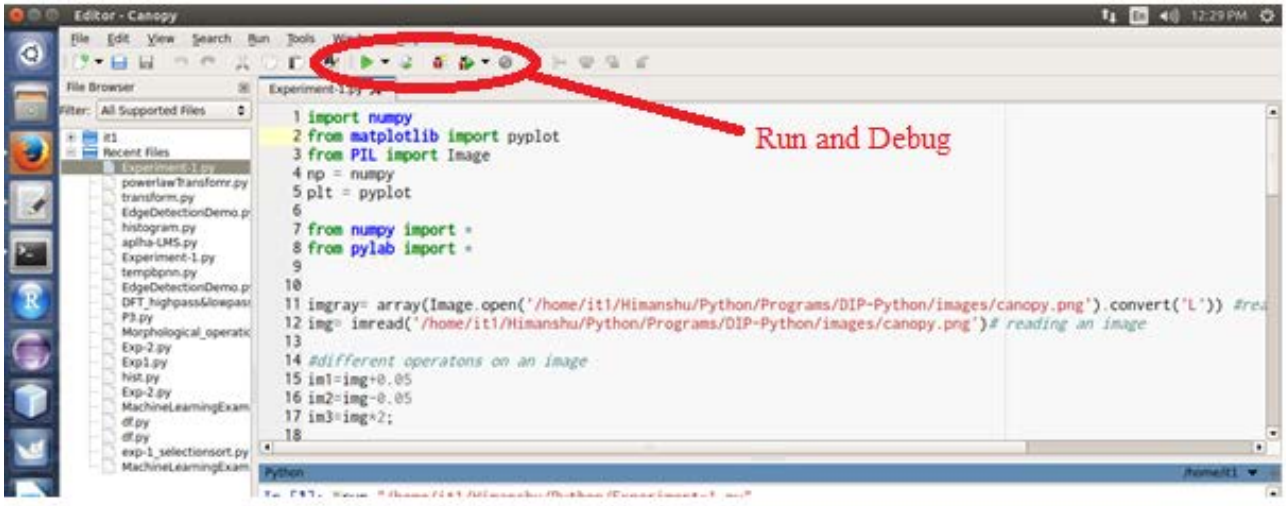
sampleExample.py

```
import numpy
from matplotlib import pyplot
from PIL import Image
np = numpy
plt = pyplot

from numpy import *
from pylab import *

#reading an image file in gray level
imgray=array(Image.open('/home/it1/Himanshu/Python/Programs/DIP-
Python/images/canopy.png').convert('L')) # reading an image
img=imread('/home/it1/Himanshu/Python/Programs/DIP-
Python/images/canopy.png')

#different operatons on an image
im1=img+0.05  #adding 0.05 to every pixels in an image
im2=img-0.05
im3=img*2;

#convertiong image in Black and White image using threshold value
theta = raw_input("Enter value of theta(Threshold Value):") #taking
an input from concole
```

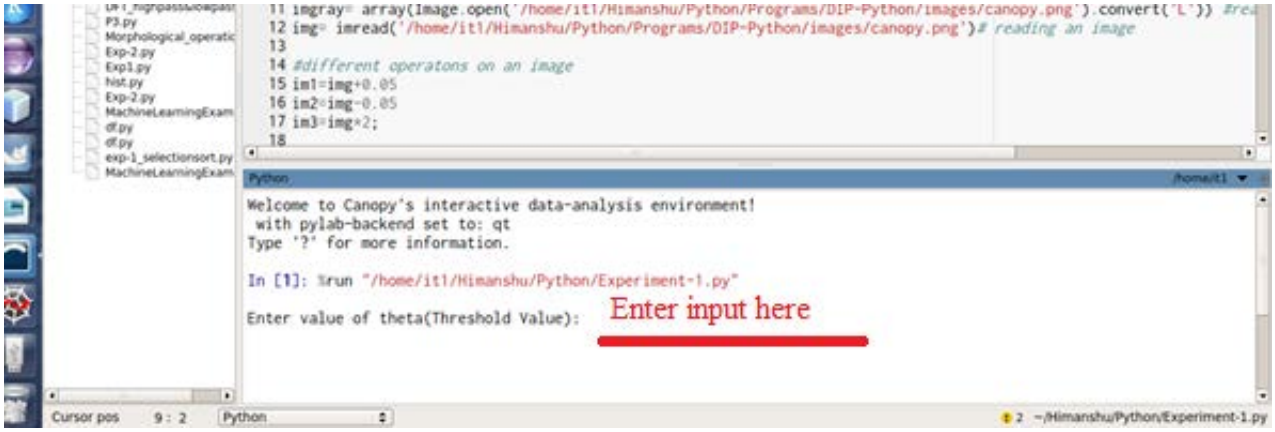
```
theta = int(theta)
imbw= 1*(imgray>theta)
iminv=~imbw #inverse of an B&W image

#saving an output image to a file at given path
imsave('/home/it1/Himanshu/Python/Programs/DIP-
Python/images/bwcanopy.png',imbw)
axis('off')

temp=subplot(3,3,1)     #displaying image at 1st in 3*3 grid of figure
window
temp.set_title('Original') #setting a title of an image
imshow(img)       #showing an image in figure window

temp=subplot(3,3,2)     #displaying image at 2nd in 3*3 gridview
temp.set_title('img+0.05')
imshow(im1)

temp=subplot(3,3,3)
temp.set_title('img-0.05')
imshow(im2)

temp=subplot(3,3,4)
temp.set_title('img*2')
imshow(im3)

temp=subplot(3,3,5)
temp.set_title('Gray Image')
gray()
imshow(imgray)

temp=subplot(3,3,6)
temp.set_title('B&W Image')
imshow(imbw)

temp=subplot(3,3,7)
temp.set_title('Inverse Image')
imshow(iminv)

figure()    #to create new window
print "Mean Value:",imgray.mean() #writing output to the console
hist(imgray.flatten(),128)
print img.shape
print imgray.shape
print img.dtype
print imgray.dtype
print imgray.mean()
print imgray.min()
print imgray.max()
show()  #showing all figure windows
```

Run the script

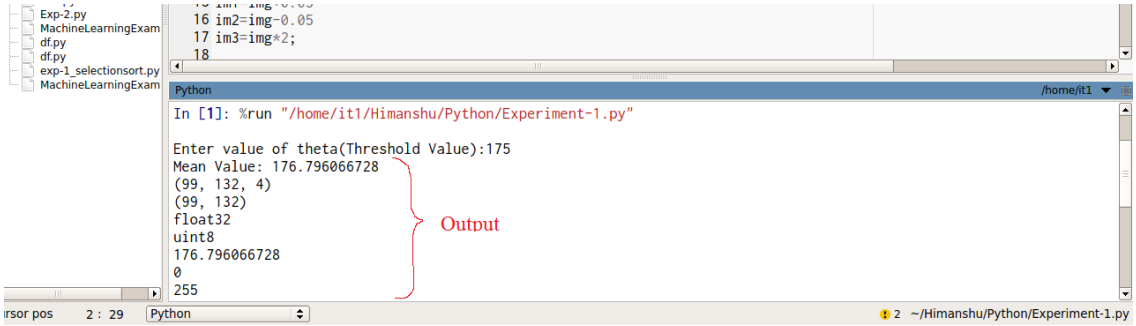
Run the script by pressing play button highlighted in following figure.

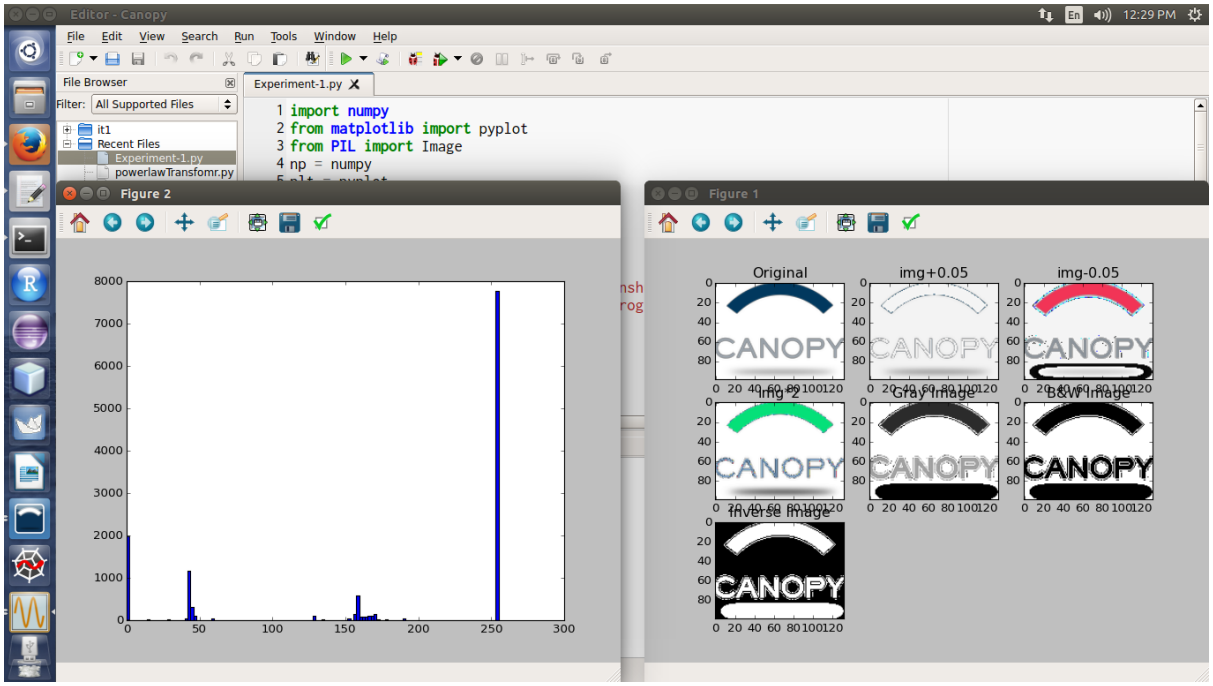


Enter input to the console as shown in following figure.



Output will be displayed on same console as well as all affected figures or generated figures will be displayed in new window.





Install Packages

Go to Tools -> Package manager and search for the package to install and click install.

Conclusion

This sample experiment shows how to do image processing with the help of python. It demonstrates to read images, performing operations on it, generating output image, take inputs from the user and displaying output image.

EXPERIMENT – 2

Aim: Write a python script to Perform Zoom In and Zoom out on an image.

Theory:

Zoom In using Bilinear Interpolation

- Average out the missing pixel value along the row and column

1	2
5	6

1	0	2	0
0	0	0	0
5	0	6	0
0	0	0	0

1	1.5	2	1
3	3.5	4	2
5	5.5	6	3
2.5	2.75	3	1.5

a) Original image b) Add rows and columns with zero c) interpolated image

- Interpolation along rows is given by:-
  - v1(m,2n) = u(m,n);
  - v1(m,2n+1) = ½[u(m,n) + u(m,n+1)];

  Where $0 \le m \le M-1$, $0 \le n \le N-1$
- Interpolation along columns is given by:-
  - v(2m,n) = v1(m,n);
  - v(m,2n+1) = ½[v1(m,n) + v1(m+1,n)];

  Where $0 \le m \le M-1$, $0 \le n \le N-1$

Zoom Out

Remove alternative rows and columns for zoom out.

Procedure:

- For zoom in first read an image and also create new blank image of double size from actual image.
- Copy actual image to double sized image in alternative rows and columns.
- Interpolate the missing pixels using median filter. And display the output.
- For zoom out read an image and create new image of half size. And copy alternate pixels of actual image to new image. And display the output.

Help:

- Array(), np.zeros(), imsave()

EXPERIMENT – 3

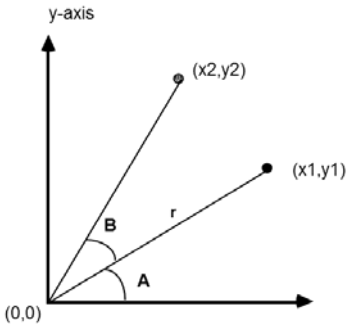
Aim: Write a python script to Perform Geometric Transforms on an image.

Theory:

Rotation

- To rotate a line or polygon, we must rotate each of its vertices.
- To rotate point (x_1,y_1) to point (x_2,y_2)

  we observe:



From the illustration we know that:

- $\sin (A + B) = y_2/r$
- $\cos (A + B) = x_2/r$
- $\sin A = y_1/r$
- $\cos A = x_1/r$

- From the double angle formulas: $\sin (A + B) = \sin A\cos B + \cos A\sin B$

  $\cos (A + B) = \cos A\cos B - \sin A\sin B$
- Substituting: $y_2/r = (y_1/r)\cos B + (x_1/r)\sin B$
- Therefore: $y_2 = y_1\cos B + x_1\sin B$
- We have **$x_2 = x_1\cos B - y_1\sin B$**

  $y_2 = x_1\sin B + y_1\cos B$

Scaling/Resize

- Changing the size of an object is called a scale. We scale an object by scaling the x and y coordinates of each vertex in the object.

Procedure:

- Read an image
- Apply inbuilt functions to perform rotation and resize om image

Help:

- Image.open(), resize(), rotate(), imsave()

EXPERIMENT – 4

Aim: Write a python script to Perform histogram equalization of an image.

Theory:

- The histogram of an image shows us the distribution of grey levels in the image.
- Massively useful in image processing, especially in segmentation.
- Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

The formula for histogram equalisation is given where

- r_k: input intensity
- s_k: processed intensity
- k: the intensity range (e.g 0.0 – 1.0)
- n_j: the frequency of intensity j
- n: the sum of all frequencies

$$s_k = T(r_k) = \sum_{j=1}^{k} p_r(r_j) = \sum_{j=1}^{k} \frac{n_j}{n}$$

Procedure:

- Read an image in gray format.
- Calculate the gray levels from the image.
- Calculate PDF and CDF value.
- Round off the CDF value and map it with actual gray levels.
- Generate image from final equalized values.
- Display the resultant image.

Help:

- array(), zeros() ,imsave()

EXPERIMENT – 5

Aim: Write a python script to Perform power law Transformation on an image.

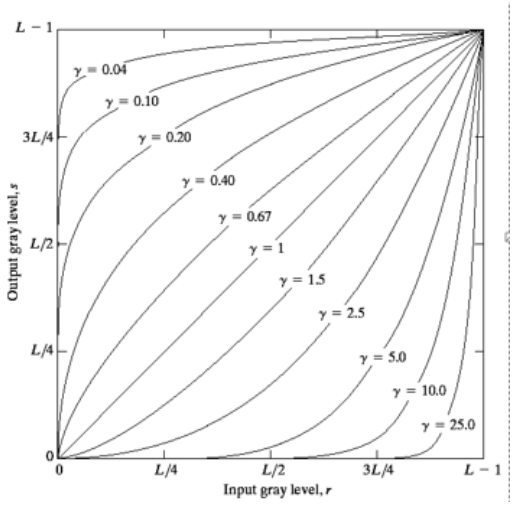
Theory:

- Power law transformations have the following form

$$s = c * r^{\gamma}$$

- Map a narrow range of dark input values into a wider range of output values or vice versa

Varying γ gives a whole family of curves



Procedure:

- Read an image in gray format.
- Take gamma value from the user.
- Apply formula of power law function to every pixel if an image and generate new image.
- Display the resultant image.

Help:

- array(), zeros(), imsave()

EXPERIMENT – 6

Aim: Write a python script to Fourier Transformation on an image and apply high and law pass filter to it.

Theory:

DFT

The *Discrete FourierTransform* of *f(x, y),* for $x = 0, 1, 2 \ldots M-1$ and $y = 0,1,2 \ldots N-1$, denoted by *F(u, v),* is given by the equation:

$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

for $u = 0, 1, 2 \ldots M-1$ and $v = 0, 1, 2 \ldots N-1$.

Inverse DFT

It is really important to note that the Fourier transform is completely **reversible**

The inverse DFT is given by:

$$f(x,y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$$

for $x = 0, 1, 2 \ldots M-1$ and $y = 0, 1, 2 \ldots N-1$

Ideal Low pass filter

The transfer function for the ideal low pass filter can be given as:

$$H(u,v) = \begin{cases} 1 \text{ if } D(u,v) \le D_0 \\ 0 \text{ if } D(u,v) > D_0 \end{cases}$$

Where is a positive constant and *D(u,v)* is the distance between a point (u,v) in the frequency domain and centre of frequency rectangle is given as:

$$D(u,v) = [(u - M/2)^2 + (v - N/2)^2]^{1/2}$$

Gaussian Lowpass Filters

The transfer function of a Gaussian lowpass filter is defined as:

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}$$

Butterworth Lowpass Filters

The transfer function of a Butterworth lowpass filter of order *n* with cutoff frequency at distance D_0 from the origin is defined as:

$$H(u,v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$

Ideal High Pass Filters

The ideal high pass filter is given as:

$$H(u,v) = \begin{cases} 0 \text{ if } D(u,v) \le D_0 \\ 1 \text{ if } D(u,v) > D_0 \end{cases}$$

where D_0 is the cut off distance as before.

Gaussian High Pass Filters

The Gaussian high pass filter is given as:

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

where D_0 is the cut off distance as before

Butterworth High Pass Filters

The Butterworth high pass filter is given as:

$$H(u,v) = \frac{1}{1+[D_0 / D(u,v)]^{2n}}$$

where n is the order and D_0 is the cut off distance as before.

Procedure:

- Read an image in gray format.
- apply DFT to an image which will return complex numbers. Sift all complex numbers to center and calculate magnitude spectrum of an image applying log function to shifted values.
- Take user input of cutoff radios d0 and order n.
- Calculate the different filters like butterworth low pass filter, idle high filter, etc..
- Apply calculated filter to shifted complex numbers.
- Now again apply inverse DFT to complex number and generate image.
- Display the resultant image.

Help:

- array(), zeros(), fft2(), fftshift(), log(), abs(),ifft2(),exp(), imsave()

EXPERIMENT – 7

Aim: Write a python script to perform morphological operations on an image**:**

Theory:

Erosion

Erosion of image *f* by structuring element *s* is given by $f \ominus s$

The structuring element s is positioned with its origin at *(x, y)* and the new pixel value is determined using the rule:

$$g(x,y) = \begin{cases} 1 \text{ if } s \text{ fits } f \\ 0 \text{ otherwise} \end{cases}$$

Dilation

Dilation of image *f* by structuring element *s* is given by $f\ s$

The structuring element s is positioned with its origin at *(x, y)* and the new pixel value is determined using the rule:

$$g(x,y) = \begin{cases} 1 \text{ if } s \text{ hits } f \\ 0 \text{ otherwise} \end{cases}$$

Opening

The opening of image *f* by structuring element *s,* denoted $f \bigcirc s$ is simply an erosion followed by a dilation

$$f \bigcirc s = (f \ominus s) \oplus s$$

Closing

The closing of image *f* by structuring element *s,* denoted $f \bullet s$ is simply a dilation followed by an erosion

$$f \bullet s = (f \quad s) \ominus s$$

Procedure:

- Read an image
- Apply erosion, dialation, opening and closingon an image with structuring element using existing function available in **skimage** library.

Help:

- array(), erosion(), dialition(), opening(), closing()

EXPERIMENT – 8

Aim: Write a python script to perform thinning and thickening operations on an image.

Theory:

Thinning

The thinning of a set *A* by a structuring element *B* can be defined in terms of the hit-or-miss transform:

$$A \otimes B = A - (A \circledcirc B)$$
$$= A \cap (A \circledcirc B)^c$$

A more useful expression for thinning A symmetrically is based on a sequence of structuring elements:

$$\{B\} = \{B^1, B^2, B^3, ..., B^n\}$$

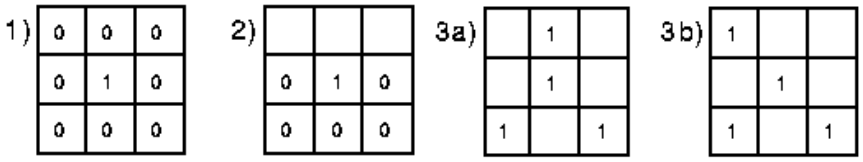
$$A \otimes \{B\} = ((...((A \otimes B^1) \otimes B^2)...) \otimes B^n)$$

Thickening

The thickening of a set *A* by a structuring element *B* can be defined in terms of the hit-or-miss transform:

$$A \otimes B = A + (A \circledcirc B)$$
$$= A \cup (A \circledcirc B)^c$$

Some structuring elements that can be used for locating various binary features



1) is used to locate isolated points in a binary image.
2) is used to locate the end points on a binary skeleton.
    - Note that this structuring element must be used in all its orientations, and thus the four hit-and-miss passes are required.
3) a) and 3b) are used to locate the triple points on a skeleton.
    - Both structuring elements must be run in all orientations so eight hit-and-miss passes are required.

Procedure:

- Read an image.
- Use existing function for hit and miss transform.
- Union the result of H&M transform with image for thickening and for thinning do intersection.
- compare the images using.

Help:

- array(), binary_hit_or_miss() and library is '**skimage.ndimage**'.

EXPERIMENT – 9

Aim: Write a python script to perform point, line and Edge detection operations on an image**.**

Theory:

Most of the shape information of an image is enclosed in edges. So first we detect these edges in an image and by using these filters and then by enhancing those areas of image which contains edges, sharpness of the image will increase and image will become clearer.

Here are some of the masks for edge detection that we will discuss in the upcoming tutorials.

- Prewitt Operator
- Sobel Operator
- Robinson Compass Masks
- Krisch Compass Masks
- Laplacian Operator.

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Prewitt

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Sobel

−1	0
0	1

0	−1
1	0

Roberts

Procedure:

- Read an image.
- Use existing function to filter an image by passing kernel as argument. Use **PIL** library for reference.
- For point and line detection apply different kernels.
- For edge detection use **skimage** library. And use different filter functions like sobel, Roberts.

Help:

- Open(), filter(), ImageFilter.kernel(), Roberts(), sobel()

EXPERIMENT – 10

Aim: Write a python script to perform boundary extraction and region filling on an image**.**

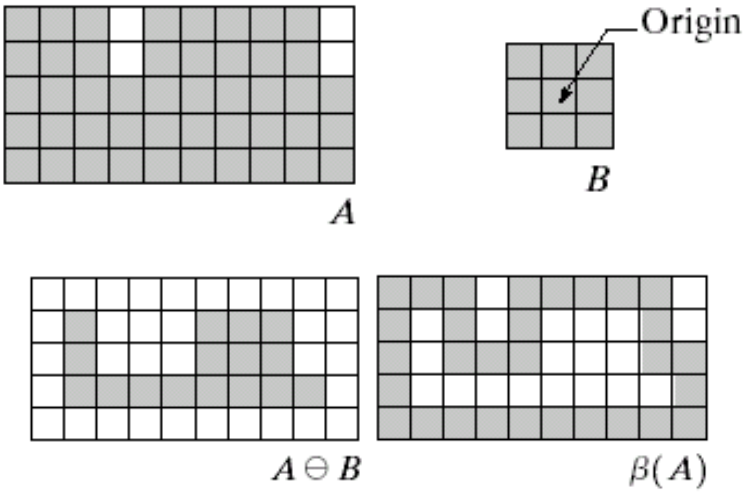
Theory:

Boundary Extraction

Extracting the boundary (or outline) of an object is often extremely useful

The boundary can be given simply as

- $\beta(A) = A - (A \ominus B)$



Region Filling

The key equation for region filling is

$$X_k = (X_{k-1} \oplus B) \cap A^c \qquad k = 1,2,3.....$$

Where X_0 is simply the starting point inside the boundary, B is a simple structuring element and A^c is the complement of A

This equation is applied repeatedly until X_k is equal to X_{k-1}

Finally the result is unioned with the original boundary

Procedure:

- Read an image.
- Create structuring element for boundary extraction.
- Erode the image first and subtract it from original image.
- Use existing function to fill an image from **skimage.morphology** library.

Help:

- Open(), erosion(), binary_fill_holes().

EXPERIMENT – 11

Aim: Write a python script to perform segmentation on a colour image**.**

Theory:

Region Based Technique

Region based methods are based continuity. These techniques divide the entire image into sub regions depending on some rules like all the pixels in one region must have the same gray level. Region-based techniques rely on common patterns in intensity values within a cluster of neighbouring pixels. The cluster is referred to as the region, and the goal of the segmentation algorithm is to group the regions according to their anatomical or functional roles.

Procedure:

- Read a color image as float.
- Use SLIC function to create segments.
- Mark the boundaries using function. Use library **skimage.segmentation**

Help:

- Open(), img_as_float(), slic(), mark_boundaries().

EXPERIMENT – 11

Aim: Installing Python and Open-CV on Ubuntu System**.**

Ref: https://help.ubuntu.com/community/OpenCV

Installation Python

```
sudo add-apt-repository ppa:fkrull/deadsnakes
sudo apt-get update
sudo apt-get install python2.7
```

Installation Open-CV

Many people are having problem with installing OpenCV even from Ubuntu Software Centre. Here a simple .sh script file get all dependancy files from internet and compile the source finally install opencv on your system. So that users can easily write their CV files from C,C++, and Python

Step 1

Download the latest opencv.sh from https://github.com/jayrambhia/Install-OpenCV/blob/master/Ubuntu/ or Copy the following script to gedit and save as opencv.sh

Toggle line numbers

```
1 version="$(wget -q -O - http://sourceforge.net/projects/opencvlibrary/files/opencv-unix | egrep -m1 -o '\"[0-9](\.[0-9]+)+' | cut -c2-)"
2 echo "Installing OpenCV" $version
3 mkdir OpenCV
4 cd OpenCV
5 echo "Removing any pre-installed ffmpeg and x264"
6 sudo apt-get -qq remove ffmpeg x264 libx264-dev
7 echo "Installing Dependenices"
8 sudo apt-get -qq install libopencv-dev build-essential checkinstall cmake pkg-config yasm libjpeg-dev libjasper-dev libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev python-dev python-numpy libtbb-dev libqt4-dev libgtk2.0-dev libfaac-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev x264 v4l-utils ffmpeg cmake qt5-default checkinstall
9 echo "Downloading OpenCV" $version
10 wget -O OpenCV-$version.zip http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/$version/opencv-"$version".zip/download
11 echo "Installing OpenCV" $version
12 unzip OpenCV-$version.zip
13 cd opencv-$version
14 mkdir build
15 cd build
16 cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/usr/local -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=ON -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON -D WITH_QT=ON -D WITH_OPENGL=ON ..
17 make -j2
18 sudo checkinstall
19 sudo sh -c 'echo "/usr/local/lib" > /etc/ld.so.conf.d/opencv.conf'
20 sudo ldconfig
21 echo "OpenCV" $version "ready to be used"
```

Note: If you are running 13.10 and you don't have a nvidia card then ensure you install ocl-icd-libopencl1 (sudo apt-get install ocl-icd-libopencl1) before running this script. Ubuntu 13.10 will install nvidia-319-updates as a dependency for libopencv-dev by default if ocl-icd-libopencl1 is not installed (see bug report).
Note: As of Utopic (14.10) libxine-dev is replaced with libxine2-dev

Step 2

Open terminal.

Toggle line numbers

```
1 $ chmod +x opencv.sh
2 $ ./opencv.sh
```

This will complete opencv installation

4 Running OpenCV

Python

Loading an image in Python

Toggle line numbers

```
1 from cv2.cv import *
2
3 img = LoadImage("/home/USER/Pictures/python.jpg")
4 NamedWindow("opencv")
5 ShowImage("opencv",img)
6 WaitKey(0)
```

Toggle line numbers

```
1 $ python filename.py
```

Note that the test program waits for a key press to end.

References

Reference Books

- "Digital Image Processing" by Rafael C. Gonzalez and Richard E. Woods
- "Image Processing, Analysis and Machine Vision" by Milan Sonka, Vaclav Hlavac and Roger Boyle
- "Fundamentals of Digital Image Processing" by Anil K. Jain