9/10

EXPERIMENT NUMBER - 6

EXPERIMENT NAME - IMAGE PROCESSING USING RASBERRY PI

DATE - 24/11/2022, THURSDAY

---

* AIM:

Introduce basic image processing techniques using Raspberry Pi.

* INTEGRATED DEVELOPMENT ENVIRONMENT (IDE):

Name - Thonny 4.0.1

Publisher - Aivar Annamaa

support link - https://thonny.org

* IMPORT NECESSARY LIBRARIES:

① Install Pillow library using the following command -

```
sudo pip3 install pillow
```

② Install an image viewing utility called xv for the built-in function show() to work.

```
sudo apt-get install xli -y
cd /usr/local/bin
sudo ln -s /usr/bin/xli xv
```

③ Import Tkinter library for GUI support in Python.

```
sudo apt-get install python-3-pil.imagetk
```

(a) Display Image and its Properties:

→ ALGORITHM -

① From PIL import Image, provides a class with the same name which is used to represent a PIL image.

② From PIL import ImageTk, contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

③ Import tkinter as tk, standard Python interface to the Tcl/Tk GUI toolkit.

④ Open and identify the given image file.

Figure 1. Sample Image from Web

⑤ Print the image characteristics for the following functionalities :-
(i) mode - string specifying the pixel format used by the image
(ii) format - File format of the source file
(iii) size - size in pixels is given as a 2-tuple (width, height)
(iv) Info - A dictionary holding data associated with the image
(v) getbands() - Returns a tuple containing the name of each band in the image

⑥ Construct a toplevel Tk widget and a Tkinter-compatible photo image. Widget that is used to implement display boxes where you can place text or images.

⑦ Organize widgets in blocks before placing them in the parent widget and loop forever, waiting for events from the user, until the user exits the program.

→ PYTHON CODE -

```
from PIL import Image
from PIL import ImageTk
import tkinter as tk
im = Image.open('Sample_Image_Web.tiff')

print(im.mode)
print(im.format)
print(im.size)
print(im.info)
print(im.getbands())

root = tk.Tk()
root.title("Display Image")
photo = ImageTk.PhotoImage(im)

l = tk.Label(root, image = photo)
l.pack()
l.photo = photo
root.mainloop()
```

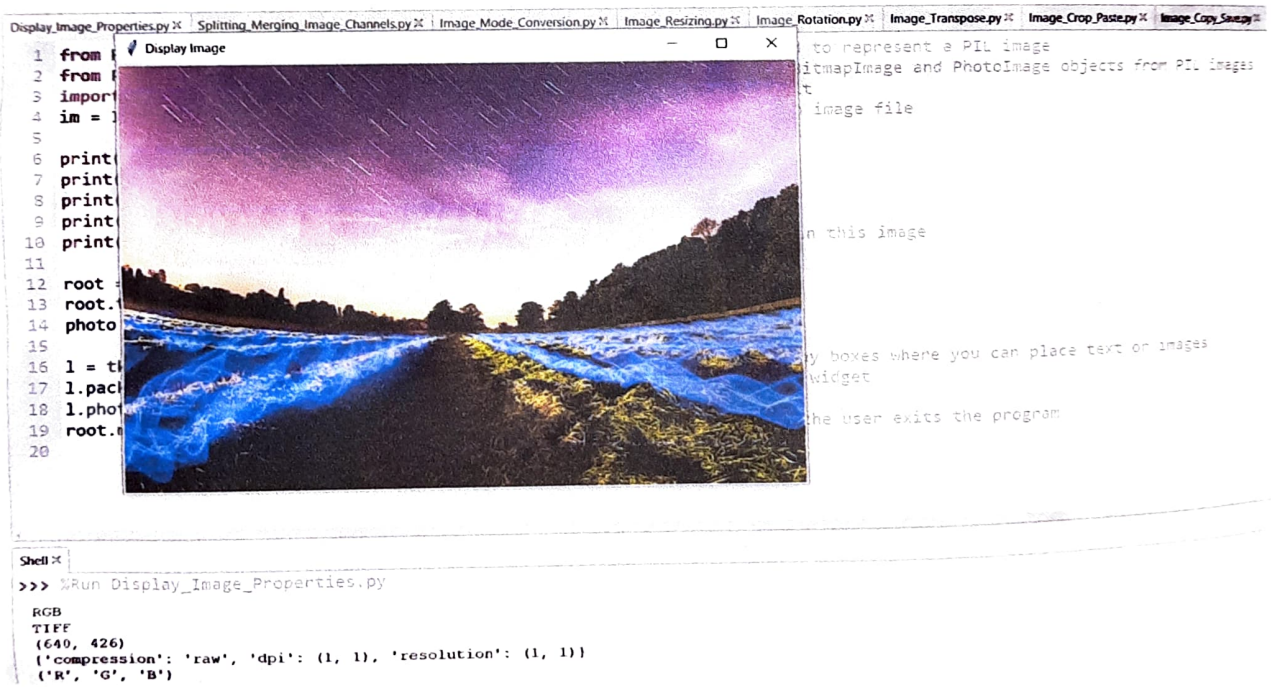


Figure 2 - Display Image and its Properties

(8) Splitting and merging Image channels:

→ ALGORITHM -

① From PIL import Image, provides a class with the same name which is used to represent a PIL image.

② From PIL import ImageTk, contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

③ Import tkinter as tk, standard Python interface to the Tcl/Tk GUI toolkit.

④ Opens and identify the given image file. Construct a toplevel Tk widget and split the image into individual bands.

⑤ Create a Tkinter-compatible photo image and widget that is used to implement display boxes where you can place text or images.

⑥ Organize widgets in blocks before placing them in the parent widget and merge a set of single band images into a new multiband image.

⑦ Loop forever, waiting for events from the user, until the user exits the program.

→ PYTHON CODE -

```
from PIL import Image
from PIL import ImageTk
import tkinter as tk
im = Image.open('sample_Image-web.tiff')

root = tk.Tk()
root.title("RED channel Demo")
r,g,b = im.split()

photo_1 = ImageTk.PhotoImage(im)
l-1 = tk.Label(root, image = photo_1)
l-1.pack()
l-1.photo = photo_1
```

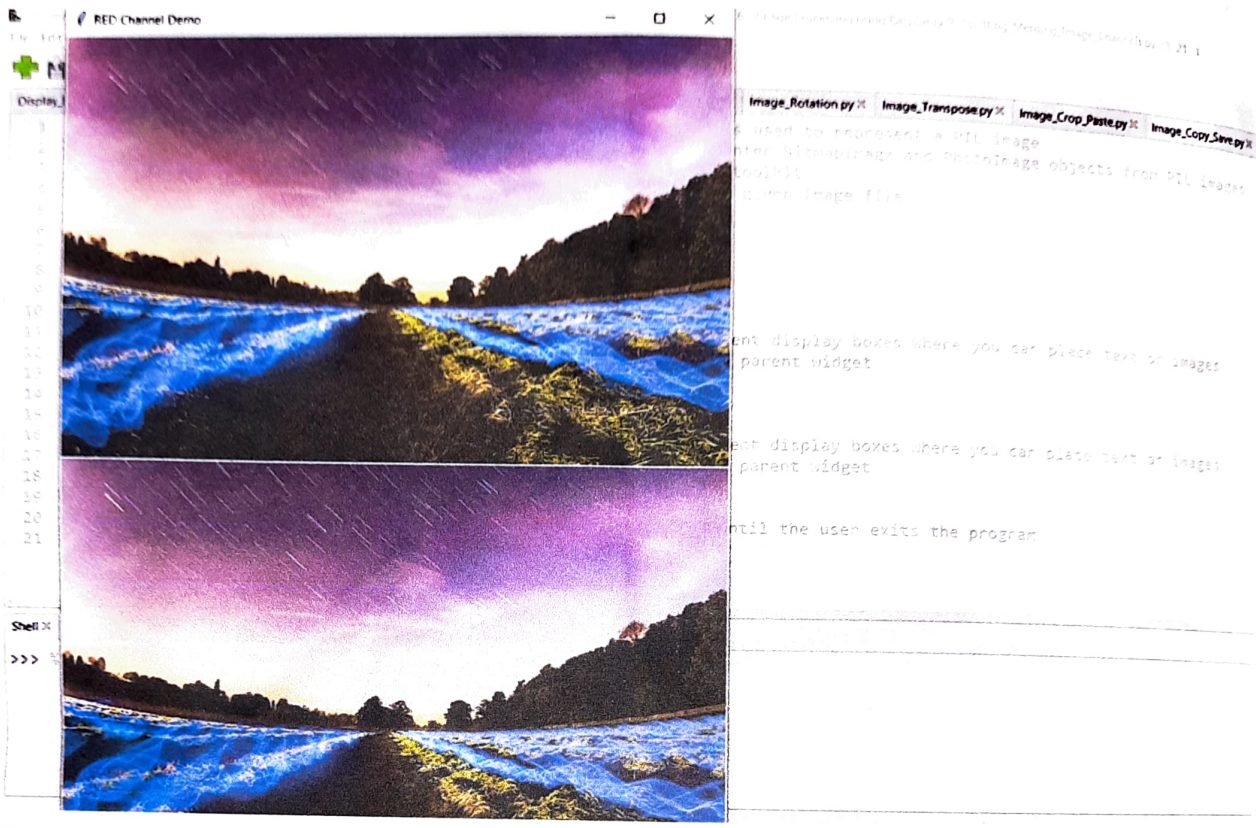


Figure 3 - Splitting and Merging Image Channels.

```
photo_2 = ImageTk.PhotoImage(Image.merge("RGB", (r,g,b)))
l_2 = tk.Label(root, image = photo_2)
l_2.pack()
l_2.photo = photo_2

root.mainloop()
```

(c) Image Mode Conversion:

→ ALGORITHM -

① From PIL import Image, provides a class with the same name which is used to represent a PIL image.

② From PIL import ImageTk, contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

③ Import tkinter as tk, standard Python interface to the Tcl/Tk GUI toolkit.

④ Open and identify the given image file. Return a converted copy of the image, translating a color image to greyscale.

⑤ Construct a toplevel Tk widget and a Tkinter-compatible photo image widget that is used to implement display boxes where you can place text or images.

⑥ Organizes widgets in blocks before placing them in the parent widget. Loop forever, waiting for events from the user, until the user exits the program.

→ PYTHON CODE -

```
from PIL import Image
from PIL import ImageTk
import tkinter as tk
im = Image.open('Sample_Image_Web.tiff')

res = im.convert("L")
root = tk.Tk()
root.title("Colorspace Conversion Demo")
photo = ImageTk.PhotoImage(res)
```

Figure 4 - Image Mode Conversion

```
l = tk.Label(root, image = photo)
l.pack()
l.photo = photo
root.mainloop()
```

(d) Image Resizing:

→ ALGORITHM -

① From PIL import Image, provides a class with the same name which is used to represent a PIL image.

② From PIL import ImageTk, contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

③ Import tkinter as tk, standard Python interface to the Tcl/Tk GUI toolkit.

④ In a function "show_value", return a resized copy of the image and a Tkinter-compatible photo image.

⑤ Construct a toplevel Tk widget. Open and identify the given image file.

⑥ Widget that is used to implement display boxes where you can place text or images. Organize widgets in blocks before placing them in the parent widget.

⑦ Provide a graphical slider object that allows you to select values from a specific scale.

(i) label - You can display a label within the scale widget by setting this option to the label's text.

(ii) from_ - A float or integer value that defines one end of the scale's range.

(iii) to - A float or integer value that defines one end of the scale's range. Can be either greater than or less than the from_ value.

(iv) resolution

(v) command - A procedure to be called every time the slider is moved.

(vi) orient - Scale runs along the x dimension.

⑧ Pack widgets relative to the earlier widget and loop forever, waiting for events from the user, until the user exits the program.

Figure 5 - Image resizing

→ PYTHON CODE -

```
from PIL import Image
from PIL import ImageTk
import tkinter as tk

def show_value (size):
    print ('Resize: ', size)
    img = im.resize (int (size), int (size)))
    photo = ImageTk.PhotoImage (img)
    l['image'] = photo
    l.photo = photo

root = tk.Tk ()
root.attributes ('-fullscreen', True)
im = Image.open ('Sample_Image_Web.tiff')
photo = ImageTk.PhotoImage (im)

l = tk.Label (root, image = photo)
l.pack ()
l.photo = photo

w = Tk.Scale (root,
              label = "Resize",
              from_ = 128,
              to = 512,
              resolution = 1,
              command = show_value,
              orient = tk.HORIZONTAL)
    )

w.pack ()
root.mainloop ()
```

(c) Image Rotation :

→ ALGORITHM -

① Import tkinter as tk, standard Python interface to the Tcl/Tk GUI toolkit.

② From PIL import Image, provides a class with the same name which is used to represent a PIL image.

③ From PIL import ImageTk, contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

④ In a function "show_value", return a rotated copy of the image and create a Tkinter-compatible photo image.

⑤ construct a toplevel Tk widget and a Tkinter-compatible photo image. Refer to the name provided to the window.

⑥ Open and identify the given image file. Widget that is used to implement display boxes where you can place text or images.

⑦ Organize widgets in blocks before placing them in the parent widget.

⑧ Provide a graphical slider object that allows you to select values from a specific value.

(i) label - You can display a label within the scale widget by setting this option to the label's text.

(ii) from_ - A float or integer value that defines one end of the scale's range.

(iii) to - A float or integer value that defines one end of the scale's range. Can be either greater than or less than the from_ value.

(iv) resolution

(v) command - A procedure to be called every time the slider is moved.

(vi) orient - Scale runs along the x dimension.

⑨ Packs widgets relative to the earlier widget and loop forever, waiting for events from the user, until the user exits the program.

PTO

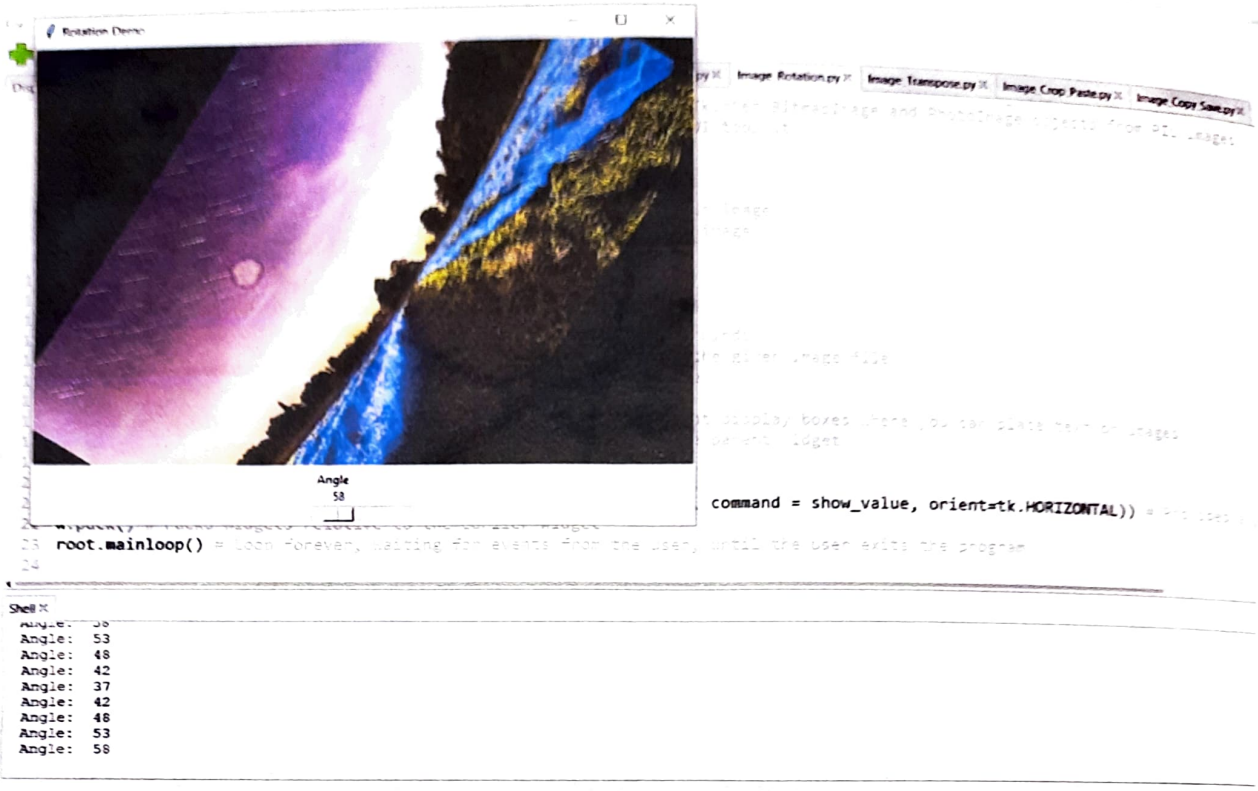


Figure 6 - Image Rotation

→ PYTHON CODE –

```
from PIL import Image
from PIL import ImageTk
import tkinter as tk

def show_value (angle):
    print ('Angle : ', angle)
    img = im.rotate (float (angle))
    photo = ImageTk.PhotoImage (img)
    l['image'] = photo
    l.photo = photo

root = tk.Tk ()
root.title ('Rotation Demo')
im = Image.open ('Sample_Image_web.tiff')
photo = ImageTk.PhotoImage (im)

l = tk.Label (root, image = photo)
l.pack ()
l.photo = photo

w = (tk.Scale (root,
               label = "Angle",
               from_ = 0,
               to = 360,
               resolution = 1,
               command = show_value,
               orient = tk.HORIZONTAL)
    )

w.pack ()
root.mainloop ()
```

(f) Image Transpose :

→ ALGORITHM -

① Import tkinter as tk, standard Python interface to the Tcl/Tk GUI toolkit.

② From PIL import Image, provides a class with the same name which is used to represent a PIL image.

③ From PIL import ImageTk, contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

④ Construct a toplevel Tk widget. Refer to the name provided to the window. Open and identify the given image file.

⑤ Transpose image (flip or rotate in 90-degree steps).
PARAMETERS : method - One of
Transpose.FLIP_LEFT_RIGHT
Transpose.FLIP_TOP_BOTTOM
Transpose.ROTATE_90
Transpose.TRANSVERSE
Transpose.ROTATE_180
Transpose.ROTATE_270
Transpose.TRANSPOSE

⑥ Create a Tkinter-compatible photo image. Widget that is used to implement display boxes where you can place text or images.

⑦ Organizes widgets in blocks before placing them in the parent widget.

→ PYTHON CODE -

```
from PIL import Image
from PIL import ImageTk
import tkinter as tk

root = tk.Tk()
root.title('Transpose Demo')
im = Image.open('Sample_Image_Web.tiff')
out = im.transpose(Image.Transpose.ROTATE_180)

photo = ImageTk.PhotoImage(out)
l = tk.Label(root, image=photo)
l.pack()
l.photo = photo
```

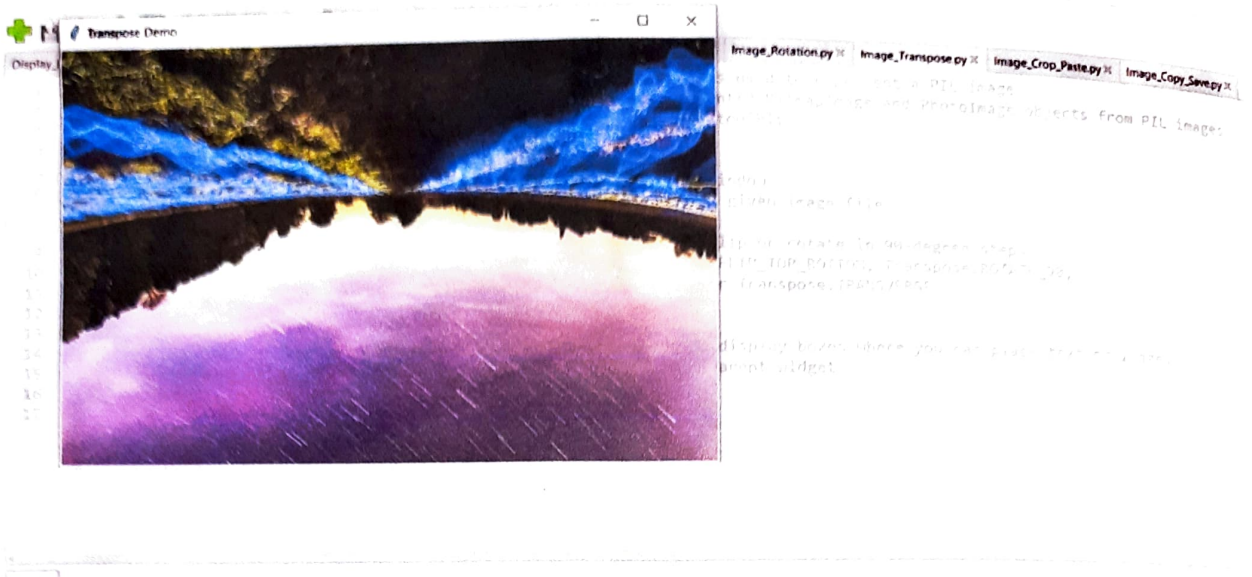


Figure 7 - Image Transpose

(g) Image Crop and Paste :

→ ALGORITHM -

① Import Image from PIL, provides a class with the same name which is used to represent a PIL image.

② Open and identify the given image file. Return a rectangular region from the image.

③ Transpose image (flip or rotate in 90-degree steps).

PARAMETERS : method - One of

Transpose.FLIP_LEFT_RIGHT
Transpose.FLIP_TOP_BOTTOM
Transpose.ROTATE_90
Transpose.ROTATE_180
Transpose.ROTATE_270
Transpose.TRANSPOSE
Transpose.TRANSVERSE

④ Paste another image into the image and diplay the image.

→ PYTHON CODE -

```
from PIL import Image
im = Image.open('Sample_Image_web.tiff')
face_box = (100, 100, 300, 300)
face = im.crop(face_box)

rotated_face = face.transpose(Image.Transpose.ROTATE_180)
im.paste(rotated_face, face_box)
im.show()
```

(h) Image Copy and Save :

→ ALGORITHM -

① Import Image from PIL, provides a class with the same name which is used to represent a PIL image.

② Opens and identifies the given image file.

③ Copies the image.

④ Saves the image under the given filename.

→ P.T.O

Figure 8. Image Crop and Paste

→ PYTHON CODE -

```
from PIL import Image
im = Image.open('Sample_Image_web.tiff')
im_temp = im.copy()
im_temp.save("Image_Save.tiff")
```

(i) Assignment : To implement Gaussian blur, emboss, and image rotation while having their sliders in the left, right and down positions respectively with respect to the image.

→ ALGORITHM -

① From PIL import Image, provides a class with the same name which is used to represent a PIL image.

② From PIL import ImageTk, contains support to create and modify Tkinter BitmapImage and PhotoImage objects from PIL images.

③ From PIL import ImageFilter, contains definitions for a pre-defined set of filters, which can be used with the Image.filter() method.

④ Import tkinter as tk, standard Python interfaces to the Tcl/Tk GUI toolkit.

⑤ Module blur_image :-
(i) Blurs the image with a sequence of extended box filters.
(ii) Filters the image using the given filter.
(iii) A Tkinter-compatible photo image.

⑥ Module rotate_image :-
(i) Return a rotated copy of the image
(ii) A Tkinter-compatible photo image

⑦ module emboss_image :-
(i) Apply emboss filter on the image
(ii) Filter the image using the given image
(iii) A Tkinter-compatible photo image

⑧ Construct a toplevel Tk widget and refer to the name provided to the window. Open and identify the given image file.

figure 9- Image copy and Save.

(9) A Tkinter - compatible photo image and widget that is used to implement display boxes where you can place text or images.

(10) Organize widgets in a table-like structure in the parent widget :-
(i) column - The column to put widget in ; default 0 (leftmost column)
(ii) row - The row to put widget in ; default the first row that is still empty.
(iii) padx - How many pixels to pad widget, horizontally and vertically, outside v's borders
(iv) sticky - what to do if the cell is larger than widget. By default, with sticky = '', widget is centered in its cell.

(11) Provide a graphical slider object that allows you to select values from a specific scale. Loop forever, waiting for events from the user, until the user exits the program.

→ PYTHON CODE -

```
from PIL import Image
from PIL import ImageTk
from PIL import ImageFilter
import tkinter as tk

def blur_image(blur_radius):
    print('Gaussian Blur Radius: ', blur_radius)
    custom_filter = ImageFilter.GaussianBlur(radius = float(blur_radius))
    img = im.filter(custom_filter)
    photo = ImageTk.PhotoImage(img)
    l['image'] = photo
    l.photo = photo
```

P.T.O


```
57 root.mainloop() # Loop forever, waiting for events from the user, until the user exits the progra
58
```

```
Shell
>>> %Run 19CCE381_Expt_6_Assignment.py
  Gaussian Blur Radius:  1
  Gaussian Blur Radius:  2
```

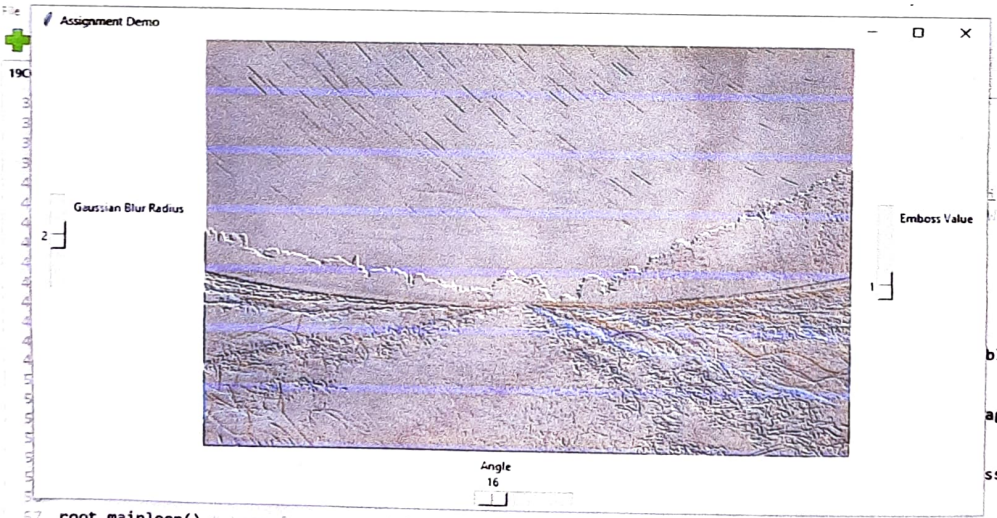
Figure 10 - Assignment : Gaussian Blur Radius

```
def rotate_image(angle):
    print('Angle: ', angle)
    img = im.rotate(float(angle))
    photo = ImageTk.PhotoImage(img)
    l['image'] = photo
    l.photo = photo

def emboss_image(emboss_value):
    print('Emboss Value: ', emboss_value)
    custom_filter = ImageFilter.EMBOSS()
    img = im.filter(custom_filter)
    photo = ImageTk.PhotoImage(img)
    l['image'] = photo
    l.photo = photo

root = tk.Tk()
root.title('Assignment Demo')
im = Image.open('Sample_Image_Web.tiff')
photo = ImageTk.PhotoImage(im)

l = tk.Label(root, image = photo)
l.grid(column=50, row=0, padx=5, sticky='n')
l.photo = photo

image_blur = (tk.Scale(root, label="Gaussian Blur Radius",
from_=0, to=5, resolution=1, command=blur_image,
orient=tk.VERTICAL))
image_blur.grid(column=0, row=0, padx=5, sticky='W')

rotate_image = (tk.Scale(root, label="Angle", from_=0, to=360,
resolution=1, command=rotate_image, orient=tk.HORIZONTAL))
rotate_image.grid(column=50, row=10, padx=5, sticky='s')
```


```
root.mainloop() # Loop forever, waiting for events from the user, until the user exits the program
```

```
Shell
>>> %Run 19CCE351_Expt_5_Assignment.py
Gaussian Blur Radius:  1
Gaussian Blur Radius:  2
Angle:  5
Angle:  11
Angle:  16
```

Figure 11 - Assignment : Angle




```
root.mainloop() # Loop forever, waiting for events from the user, until the user exits the program
```

```
Shell
Gaussian Blur Radius:  1
Gaussian Blur Radius:  2
Angle:  5
Angle:  11
Angle:  16
Emboss Value:  1
Emboss Value:  0
Emboss Value:  1
```

Figure 12 - Assignment : Emboss Value

```
image_emboss = tk.Scale(root, label="Emboss Value", from_=0,
to=1, resolution=1, command=emboss_image, orient=tk.VERTICAL))
image_emboss.grid(column=60, row=0, padx=5, sticky='e')

root.mainloop()
```

* **RESULT:**

Thus, introduced basic image processing techniques using Raspberry Pi. All the simulation results were verified successfully.