#The interface pf the python image processor is here

from Tkinter import \* #Importing the Tkinter module

import tkMessageBox

def display():

import tkMessageBox #Module to display the image so chosen

from skimage import io

#Importing skimage module

try:

io.imread(e1.get())

except IOError:

tkMessageBox.showinfo("File not found")

img=io.imread(e1.get()) #Read or scan the image

io.imshow(img) #Display the image

print io.imshow(img) #Show the attributes

io.show()

def dimen(): #Module to display the dimensions

from skimage import io #Importing the skimage module

try:

io.imread(e1.get())

except IOError:

tkMessageBox.showinfo("File not found")

img = io.imread(e1.get()) #Read or scan the image

print io.shape #Display the dimensional parameters of image in form of(height,width,spatial\_dimension)

def greyscale(): #Module to perform the greyscaling of the image

import skimage.io as io #Import skimage module

from skimage.color import rgb2gray #Importing the skimage module utility to perform greyscaling

try:

io.imread(e1.get())

except IOError:

tkMessageBox.showinfo("File not found")

img = io.imread(e1.get()) #Read or scan the image

img\_grayscale = rgb2gray(img) #Greyscaling the image

io.imsave('gs.png',img\_grayscale) #Method to save the greyscale image

show\_grayscale = io.imshow(img\_grayscale) #Method to display the image

io.show()

def filt(): #Method to filter the image

from skimage import data, io, filters #Importing the io,data and filters of the image

try:

io.imread(e1.get())

except IOError:

tkMessageBox.showinfo("File not found")

img = io.imread('gs.png') #Read or scan the image

edges = filters.sobel(img) #Performing the sobel filtering of image (Using the Sobel algorithm)

io.imshow(edges) #Filtering the display of the edges only

io.show()

def sharp():

"""

IMAGE DECONVOLUTION:

It increases sharp corners and enhances restoration

"""

import numpy as np #Performing the restoration of the image in 2-d or greyscale format

import matplotlib.pyplot as plt #Importing matplotlib module

from skimage import color, data, restoration,io

try:

io.imread(e1.get())

except IOError:

tkMessageBox.showinfo("File not found")

astro = color.rgb2gray(io.imread(e1.get()))

from scipy.signal import convolve2d as conv2

psf = np.ones((5, 5)) / 25

astro = conv2(astro, psf, 'same') #Function to perform the deconvolution of the image

astro += 0.1 \* astro.std() \* np.random.standard\_normal(astro.shape)

deconvolved, \_ = restoration.unsupervised\_wiener(astro, psf)

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(8, 5), #Arranging the display grids

sharex=True, sharey=True,

subplot\_kw={'adjustable': 'box-forced'})

plt.gray()

ax[0].imshow(astro, vmin=deconvolved.min(), vmax=deconvolved.max()) #Axial representation to display the original image

ax[0].axis('off')

ax[0].set\_title('Data')

ax[1].imshow(deconvolved) #Axial representation to display the restored image

ax[1].axis('off')

ax[1].set\_title('Self tuned restoration')

fig.tight\_layout()

plt.show()

def area(): #Module to perform the area of shadow an object will form on the floor(Object is to consist of value 0 and 1)

import matplotlib.pyplot as plt

from skimage.morphology import convex\_hull\_image

from skimage import data, img\_as\_float #Here we import the morphology utility module of skimage package

from skimage.util import invert

# The original image is inverted as the object must be white.

tkMessageBox.showinfo("This only works for 2-D image. Inbuilt is a horse in black and white")

image = invert(data.horse())

chull = convex\_hull\_image(image)

# The arrangement of plots

fig, axes = plt.subplots(1, 2, figsize=(8, 4))

ax = axes.ravel()

#Displaying the plots

ax[0].set\_title('Original picture')

ax[0].imshow(image, cmap=plt.cm.gray, interpolation='nearest')

ax[0].set\_axis\_off()

ax[1].set\_title('Transformed picture')

ax[1].imshow(chull, cmap=plt.cm.gray, interpolation='nearest')

ax[1].set\_axis\_off()

plt.tight\_layout()

plt.show()

def skel():

"""

Image skeletonize:

To figure out the outline of the image in 2-d (black and white format), image will consist of values 0 and 1 in pixel sense

"""

from skimage.morphology import skeletonize #Importing the skeletonize utility of the skimage package

from skimage import data #Importing the data utility of skimage package

import matplotlib.pyplot as plt

from skimage.util import invert

tkMessageBox.showinfo("This only works for 2-D image. Inbuilt is a horse in black and white")

# Invert the horse image

image = invert(data.horse())

# perform skeletonization

skeleton = skeletonize(image)

# display results

fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(8, 4),

sharex=True, sharey=True,

subplot\_kw={'adjustable': 'box-forced'})

ax = axes.ravel()

# Arrangement of axes in grid display

ax[0].imshow(image, cmap=plt.cm.gray)

ax[0].axis('off')

ax[0].set\_title('original', fontsize=20)

ax[1].imshow(skeleton, cmap=plt.cm.gray)

ax[1].axis('off')

ax[1].set\_title('skeleton', fontsize=20)

fig.tight\_layout()

plt.show()

#The main ui framework class starts here

#Importing the Tkinter module to perform tasks

root=Tk()

root.title("Photo Editor ver:1.0")

Label(root,text="Photo Editor",bg='black',fg='red',font=("bold",24),height=10,width=50,relief=RIDGE).pack(fill=X)

Label(root,text="Enter the name of the file in png extension",fg='red',font='bold').pack(fill=X)

label=Label(root)

from skimage import io

import tkMessageBox

e1=Entry(root)

e1.pack()

label=Label(root)

Button(text="Display the image chosen",command=display).pack()

Button(text="Get the dimensional parameters",command=dimen).pack()

Button(text="Increase the sharpness of the image",command=sharp).pack()

Button(text="Apply filter to the image",command=filt).pack()

Button(text="Get the area of a negative filter",command=area).pack()

Button(text="Get the skeleton of the image",command=skel).pack()

Button(text="Gray-scale of the image",command=greyscale).pack()

Button(text="Exit the application",command=exit).pack()

Label(root,text="Note:that to get filter first press the greyscale",bg='black',fg='white').pack(fill=X)

root.mainloop()