HACETTEPE UNIVERSITY 21/22 FALL

BBM 415 FINAL PROJECT

SIMPLE IMAGE EDITOR

Students

Abdullah Atahan Türk

Yuşa Zorlu

Student IDs

21827943

21828054

T.A

Yunus Can Bilge

In our project we can use processes one by one. We can't do sequantial operations. For example if we click the "grayscale" button we get an grayscale image, than if we try to adjust brightness, we get the original image with different brightness. Only adjusting RGB can be done together. For instance, you can adjust both red and green values at the same time but you can't adjust brightness and saturation at the same time. If the user want to apply more than one process, he/she should apply a process, save the image and then load the processed image and apply another process.

1. Load Image

```
def load_img():
    global openedfile
    openedfile = filedialog.askopenfilename(title ='Choose The Image To Load')
    try:
        global myimage
        myimage = Image.open(openedfile)
        img = myimage.resize((355, 355), Image.ANTIALIAS)
        img = ImageTk.PhotoImage(img)
        inputCanvas.create_image(180, 180, image=img)
        inputCanvas.image=img
        outputCanvas.delete("all")
        outputCanvas.create_text(177,177,text="Edited Image", fill="aquamarine3",
font=("Showcard Gothic",30))
        rgbreset()
        adjustreset()
        cropreset()
    except AttributeError:
        pass
    except UnidentifiedImageError:
        messagebox.showerror("Inappropriate File Type", "To apply a filter, you should
upload a file which has an image file extension (.png, .jpg, .jpeg etc.)")
```

To open a file loading box, we used "filedialog.asopenfilename" method. Take the loaded image as "myimage" and then we put a resized copy of that picture on the canvas. The reset functions there for reset the all scales in our editor. In all functions we use those reset functions. So we will delete the reset functions section in other functions to save space and then put the code in the report. Of course, in the source code they remain. When we load an new image the old edited image on the output canvas will be deleted. If we open the load file window and close it without loading a file, it gives "Attribute Error" but the program working without effected so that's why I passed that error. If we try to upload a file which is not image, the program will show you an error box.

2. Reset Image

```
def reset_image():
    global myimage
    myimage = None
    outputCanvas.image = None
    inputCanvas.delete("all")
    inputCanvas.create_text(177,177,text="Input Image", fill="aquamarine3",font=("Showcard
Gothic",30))
    outputCanvas.delete("all")
    outputCanvas.create_text(177,177,text="Edited Image",fill="aquamarine3",font=("Showcard
Gothic",30))
```

This function resets both input and output images.

3. Save Image

```
def save_image():
    global myimage, openedfile, grayscaled, mirrored, blurred, deblurred, rotated90,
rotated180, rotated270, flipped, brighted, contred, satred, detected, reversedorg,\
        rgbedr, rgbimage, noised, noised1, cropped, cropped1, cropped2, original,
original1,\
        flipped1,mirrored1, blurred1, deblurred1, rotated90_1, rotated180_1, rotated270_1,
flipped1, brighted1, brighted2, contred1, contred2, satred1, satred2, detected1
    try:
        if myimage == None:
            messagebox.showerror("No Input Image", "To save an image, you should upload and
edit an image first")
        elif outputCanvas.image == None:
            messagebox.showerror("No Input Image", "To save an image, you should edit the
image first")
        else:
            file = filedialog.asksaveasfilename(defaultextension =".png",filetypes=[("All
Files","*.*"),("PNG file","*.png"),("jpg file","*.jpg")])
        if outputCanvas.image == grayscaled1:
            grayscaled.save(file)
        if outputCanvas.image == mirrored1:
            mirrored.save(file)
        if outputCanvas.image == blurred1:
            blurred.save(file)
        if outputCanvas.image == rotated90_1:
            rotated90.save(file)
        if outputCanvas.image == flipped1:
            flipped.save(file)
        if outputCanvas.image == brighted2:
            brighted1.save(file)
        if outputCanvas.image == rgbedr:
            rgbimage.save(file)
        if outputCanvas.image == cropped2:
            cropped.save(file)
                .
                .
                .
    except AttributeError:
        pass
    except ValueError:
        pass
    except UnboundLocalError:
        pass
```

In this function we use the edited image with using global variables. The image on the output canvas and the image we save are actually different. The output image is just a resized copy of the image that we save. For space saving I delete some “if statements” in here, because the logic is always the same. We used “asksaveasfilename” method for saving. Also if we try to save without loading or after loading but without editing, it will give an error. Also we excepted some errors, they don’t effect the program’s working but they write error in terminal.

4. Grayscale

```
def grayscale():
    global myimage
    global grayscaled, grayscaled1
    if myimage is None:
        messagebox.showerror("No Input Image", "To edit an image, you should upload an
image first")
    elif myimage.mode == "L":
         messagebox.showerror("Wrong Input Image", "The image that you are trying
```

```
to apply grayscale filter is already a grayscale image")
    else:
        grayscaled = myimage.copy()
        grayscaled = ImageOps.grayscale(grayscaled)
        grayscaled1 = grayscaled.resize((355, 355), Image.ANTIALIAS)
        grayscaled1 = ImageTk.PhotoImage(grayscaled1)
        outputCanvas.create_image(180,180,image=grayscaled1)
        outputCanvas.image = grayscaled1
```

In this function, we have an error box, if we don't load an image and try to apply a process on the image, it states that we should load an image. In all my functions there is same error box. So we will delete those rows for other functions for space saving. There will be "elses" without "ifs" in other functions. Also if we try to grayscale filter on an image which is already a grayscale, it gives an error. We used grayscale method of PIL fort he filter.

5. Blur

```
def blur():
    global myimage
    global blurred, blurred1
    if myimage is None:
    else:
        blurred = myimage.copy()
        blurred = blurred.filter(ImageFilter.GaussianBlur(5))
        blurred1 = blurred.resize((355, 355), Image.ANTIALIAS)
        blurred1 = ImageTk.PhotoImage(blurred1)
        outputCanvas.create_image(180,180,image=blurred1)
        outputCanvas.image = blurred1
```

We used filter and Gaussian blur method of PIL for this filter. In all functions, we process the image and then resize it and put it in the output canvas. So we protect the size and quality of the input image. From now on, we will delete the rows that how we put the image on the output canvas in other functions, because the logic is always the same.

6. DeBlur

```
def deblur():
    global myimage
    global deblurred, deblurred1
    else:
        deblurred = myimage.copy()
        deblurred = deblurred.filter(ImageFilter.SHARPEN)
        deblurred = deblurred.filter(ImageFilter.SHARPEN)
        deblurred1 = deblurred.resize((355, 355), Image.ANTIALIAS)
        deblurred1 = ImageTk.PhotoImage(deblurred1)
```

It is simply a sharpening filter. If our image is blurry, we can use this filter. We used filter and SHARPEN methods of PIL twice for a better result.

7. Reverse Colors

```
def reverse_colors():
    global myimage
    global reversedorg, reversed1
    else:
        if myimage.mode != "L" or myimage.mode != "RGB":
            myimage = myimage.convert("RGB")
        reversedorg = myimage.copy()
        reversedorg = ImageOps.invert(reversedorg)
        reversed1 = reversedorg.resize((355, 355), Image.ANTIALIAS)
        reversed1 = ImageTk.PhotoImage(reversed1)
```

This filter gives us an negative image. We used invert method of PIL. If we try to upload an image which is not grayscale or RGB, it gives an error. So that's why we convert the image to RGB if it is not grayscale or RGB.

8. Add Noise

```
def addNoise():
    global myimage
    else:
        if myimage.mode != "L" or myimage.mode != "RGB":
            myimage = myimage.convert("RGB")
        global noised, noised1
        noised = myimage.copy()
        pixelnoised = np.array(noised)
        x_lim, y_lim = noised.size
        ran_val_y = random.randint(2,2)
        for x in range(0, y_lim, ran_val_y):
            ran_val_x = random.randint(2, 2)
            for y in range(0, x_lim, ran_val_x):
                r = round(random.random()*255)
                if r > 255:
                    r = 255
                g = round(random.random()*255)
                if g>255:
                    g=255
                b = round(random.random()*255)
                if b>255:
                    b=255
                pixelnoised[x,y] = r,g,b
        noised = pixelnoised
        noised = Image.fromarray(noised)
        noised1 = noised.resize((355, 355), Image.ANTIALIAS)
        noised1 = ImageTk.PhotoImage(noised1)
```

We add some random noise to image in this function. We created a for loop to change some pixels with random values in the image with specific intervals.

9. Detect Edges

```
def detect_edges():
    global myimage
    global detected, detected1
    else:
        detected = myimage.copy()
        detected = detected.filter(ImageFilter.FIND_EDGES)
        detected1 = detected.resize((355, 355), Image.ANTIALIAS)
        detected1 = ImageTk.PhotoImage(detected1)
```

We used filter and FIND_EDGES methods of PIL fort his function.

10. Mirror

```
def mirror():
    global myimage
    global mirrored, mirrored1
    else:
        mirrored = myimage.copy()
        mirrored = ImageOps.mirror(mirrored)
        mirrored1 = mirrored.resize((355, 355), Image.ANTIALIAS)
        mirrored1 = ImageTk.PhotoImage(mirrored1)
```

We used mirror method of PIL for this function.

11. Rotate 90, 180 and 270

```
def rotate_90():
    global myimage
    global rotated90, rotated90_1
    else:
        rotated90 = myimage.copy()
        rotated90 = rotated90.rotate(90,expand=True)
        rotated90_1 = rotated90.resize((355, 355), Image.ANTIALIAS)
        rotated90_1 = ImageTk.PhotoImage(rotated90_1)
```

We just pasted the rotate_90 function in the report because the logic is same with 180 and 270. We used rotate method of PIL. Also we write "expand=True" for protecting the original height and width values of the image.

12. Flip

```
def flip():
    global myimage
    global flipped, flipped1
    else:
        flipped = myimage.copy()
        flipped = ImageOps.mirror(flipped)
        flipped = flipped.rotate(180)
        flipped1 = flipped.resize((355, 355), Image.ANTIALIAS)
        flipped1 = ImageTk.PhotoImage(flipped1)
```

In here, we used both mirror and rotate functions of PIL. So we can flip the image both vertically and horizontally.

13. Original

```
def originalimage():
    global myimage
    else:
        global original, original1
        original = myimage.copy()
        original1 = original.resize((355, 355), Image.ANTIALIAS)
        original1 = ImageTk.PhotoImage(original1)
```

It shows the original image on the output canvas, if you want to save original image or see the image without processing on the output canvas.

14. Crop

```
def crop(hebelehubele):
    global myimage
    if myimage != None:
        global cropped, cropped1, cropped2
        cropped = myimage.copy()
        (originalx,originaly) = cropped.size
        ratiox, ratioy = originalx/355, originaly/355
        cropped1 = cropped.resize((355, 355), Image.ANTIALIAS)
        (left,upper,right,bottom) =
(scaleX_start.get(),scaleY_start.get(),scaleX_end.get(),scaleY_end.get())
        cropped1 = cropped1.crop((left,upper,right,bottom))
        cropped2 = ImageTk.PhotoImage(cropped1)
        outputCanvas.create_image(180,180,image=cropped2)
        outputCanvas.image = cropped2
        cropped = cropped.crop((left*ratiox,upper*ratioy,right*ratiox,bottom*ratioy))
```

In this function, firstly we get the size informations of the original image. Then we calculate the ratio between the size of the original image and the image on the canvas. After that we take a resized(355,355) copy and put it on the canvas. With getting values from scale and

using crop method of PIL, we crop the image and show it on the output canvas. At the end to get the original sized cropped image, we multiply the coordinat with ratios and crop the original sized copy. Then we can save the cropped image without losing any data.

15. RGB

```
def rgb(hebelehubele):
    global myimage
    if myimage != None:
        if myimage.mode != "L" or myimage.mode != "RGB":
            myimage = myimage.convert("RGB")
        global rgbimage, rgbedr
        rgbimage = myimage.copy()
        pixel = np.array(rgbimage)
        x_lim, y_lim = rgbimage.size
        redscale = varRed.get()
        greenscale = varGreen.get()
        bluescale = varBlue.get()
        for x in range(0,y_lim):
            for y in range(0,x_lim):
                r,g,b = pixel[x,y]
                r = round(r * (redscale/5))
                if r>255:
                    r =255
                g = round(g * (greenscale/5))
                if g>255:
                    g =255
                b = round(b * (bluescale/5))
                if b>255:
                    b=255

                pixel[x,y] = r,g,b
        rgbimage = pixel
        rgbimage = Image.fromarray(rgbimage)
        rgbedr = rgbimage.resize((355, 355), Image.ANTIALIAS)
        rgbedr = ImageTk.PhotoImage(rgbedr)
```

Thanks to this function, we can change the RGB values. If the image is not RGB, we convert it to RGB image to prevent errors. Then we take the image as an array, reach every RGB values and change them according to values which we get from the RGB scales. After that we use fromarray method to get the image from the array form. Otherwise, PIL gives an error. It work a little bit slowly especially if the input image has high resolution but it works without any problem.

16. Brightness, Saturation and Contrast

```
def brightness(hebelehubele):
    global myimage
    global brighted, brighted1, brighted2
    if myimage != None:
        if myimage.mode != "L" or myimage.mode != "RGB":
            myimage = myimage.convert("RGB")
        brighted = myimage.copy()
        myval = (varBright.get()/5)
        if myval == 0:
            myval = 0.1 #I think getting a completely dark image doesn't make any sense, so
that's why I did this if statement
        brighted = ImageEnhance.Brightness(brighted)
        brighted1 = brighted.enhance(myval)
        brighted2 = brighted1.resize((355, 355), Image.ANTIALIAS)
        brighted2 = ImageTk.PhotoImage(brighted2)
```

We pasted here just brightness function, because both saturation and contrast have the same logic. Getting our brightness/saturation/contrast value from the scale, do the calculations to get myval and then use ImageEnhance.Brightness method of PIL. (For saturation ImageEnhance.Color and for contrast ImageEnhance.Contrast). “myval” will always be between 0 and 2. But getting a completely dark image doesn’t make any sense, so we changed 0 with 0.1 in brightness and contrast functions but not in saturation.

INPUT and OUTPUT IMAGE EXAMPLES


Original Image

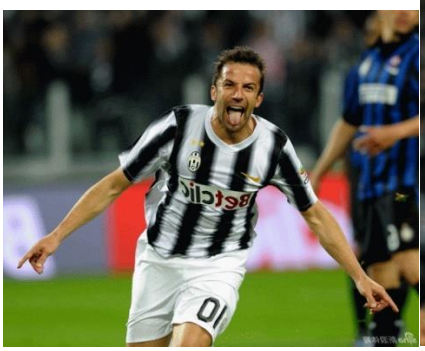

Grayscale Image

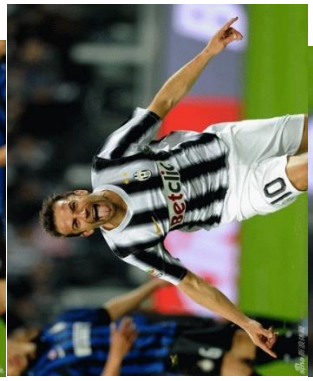

Blurred Image

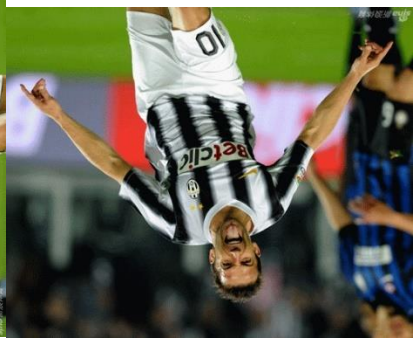

Reverse Color Image


Add Noise Image


Detect Edges Image

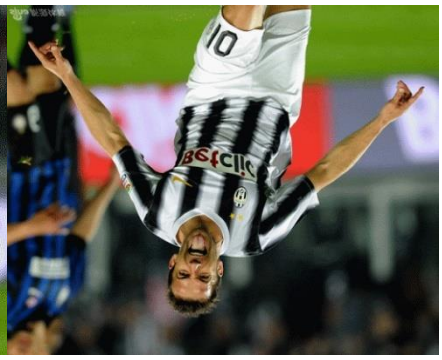

Mirror Image


Rotate 90 Image


Rotate 180 Image

Rotate 270 Image

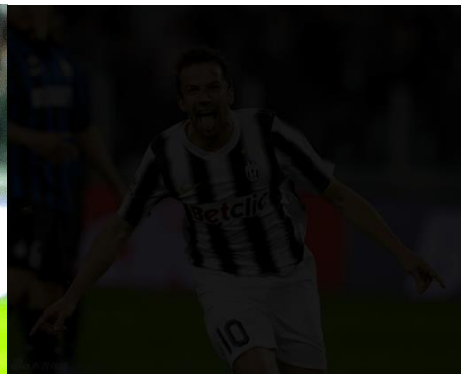
DeBlur Image



Flip Image

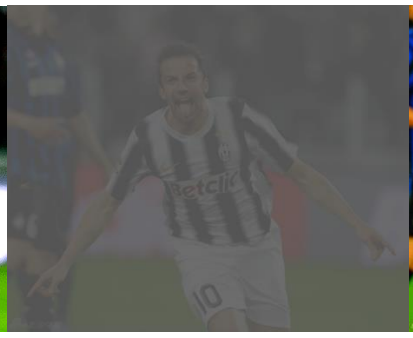
Crop Image

Max Bright



Min Bright

Max Contrast



Min Contrast

Max Saturation

Min Saturation

Max R, Min G-B Image

Max G, Min R-B Image

Max B, Min R-G Image

Max G-B, Min R Image

Max R-G, Min B Image

Max R-B, Min G Image

Max R-G-B Image

Crop Image 2

DeBlurring A Blurry Image:

7x7 Gaussian Filtered Image

DeBlurred Image

That's all for our Photo Editor. You can load and show the image with "File → Load and Show Image", reset the image with "File → Reset Image" and exit the program with "File → Exit".