CS014AL: Software Engineering

Report for Individual Project——Image editor

李煜麟 1409853G-I011-0074

CONTENT

1. Introduction:

My individual project is concerning about the Image Editor which designed depend on our Group project. Comparing with our group project, furthermore, I added lots of functions in my project, such as filtering, zooming, and cropping and so on. My aim is to use it to help me to find out which kind of image used in deep learning having the good effects.

And I used the python language to finish the project.

Python version -3.6.3

Additionally, I also used lots of packages in python, such as Numpy, Pillow, OpenCV. And all the versions of the packages are shown in setup.py.

And the GUI, I designed, is completed by Tkinter in python.

Compared with our group project, I add lots of function in it, such as filters. And I will give some details in below.

Initially, I'd like to continue using tcl\tk, and use python to finish the command. But I find that, if I used Wish to operate my GUI, it will take lots of time, when I use a filter to process the image. For example, if I used the motion-blur to process the Image, it mostly cost 10-20 seconds to finish the blurring. The reason is when I write the command in python, I use "wish" to operate the GUI and click on the button to use the python command. The whole process will take lots of time. So I designed to use Tkinter to rewrite the GUI of the Image Editor. After using Tkinter, the time that we spend on processing apparently become less and less.

1.1 File Structure

Main.py|---------ImageOperations.py---------edit.py

|----------filters.py

1.2 Version of Package:

Package	Version
Pillow	4.3.0
cycler	0.10.0
llvmlite	0.20.0
matplotlib	2.1.0
numba	0.35.0
numpy	1.13.3
olefile	0.44
opencv-python	3.3.1

2. User Manual

First, you can use command “python3 setup.py” to install all the packages I used.

Second, the opencv you need to download from the website https://opencv.org/opencv-3-0.html and install it depends on the steps on the website.

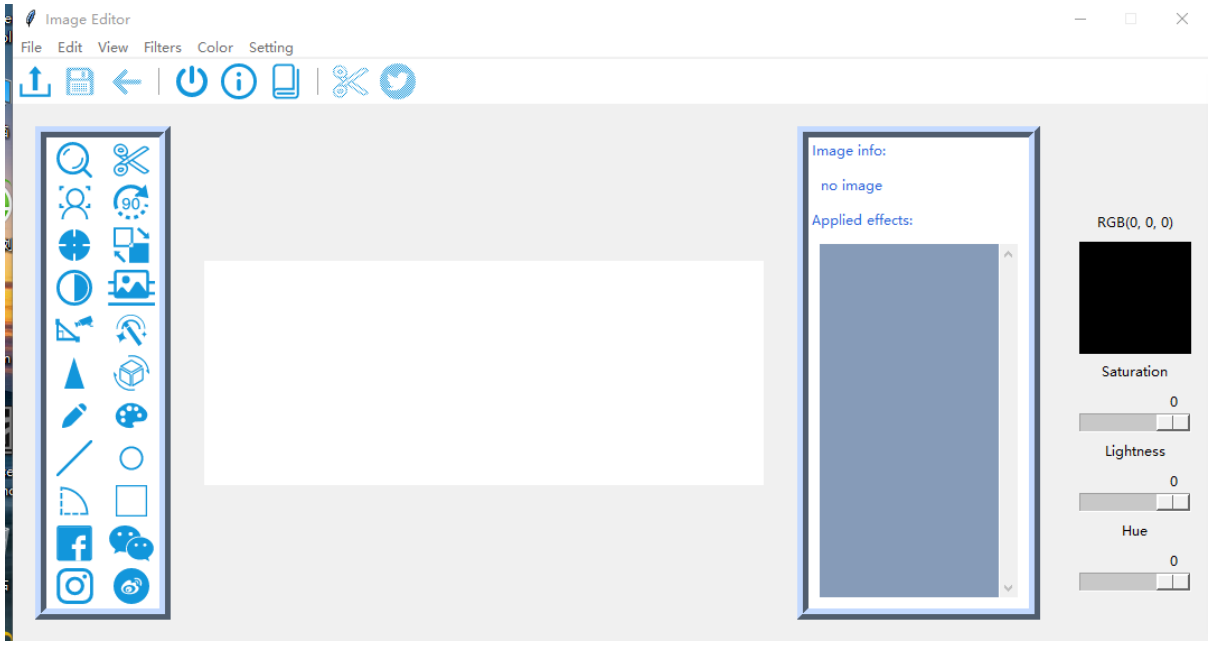
Third, after installing each package, we can use the command "python3 main.py" to run it.

3. Comparison

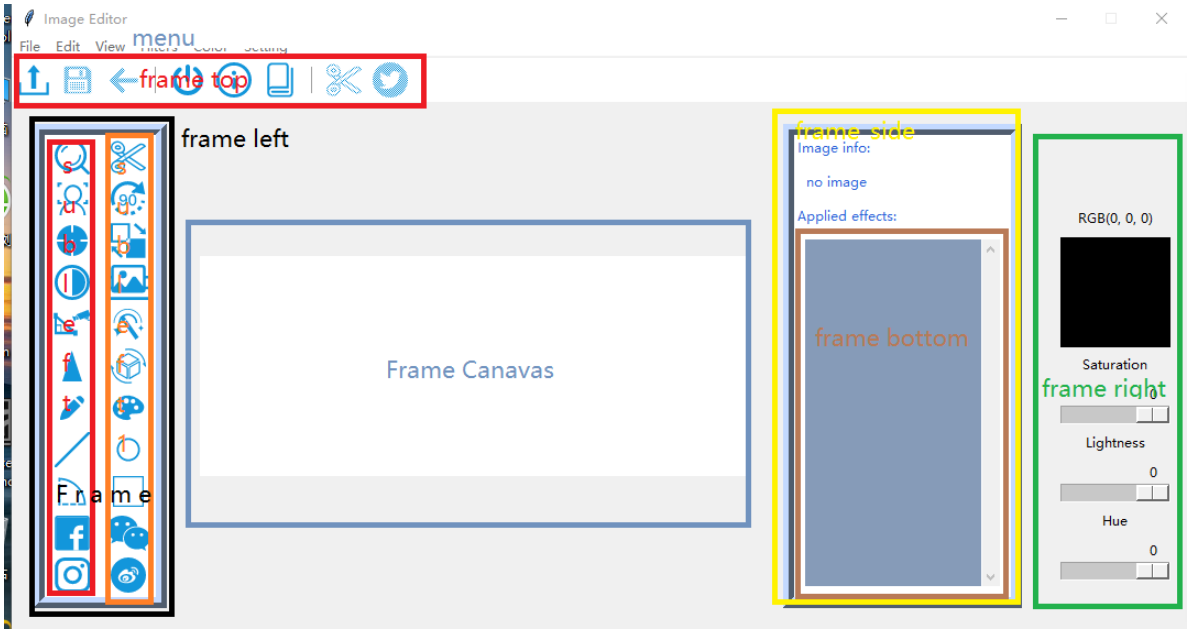
Comparing with our group project, excepts for the appearance of the GUI, there is nothing similar to the group project. And I use Tkinter to rewrite the UI of my individual project.

	Group Project	Individual Project
Fundamental function	Import, save, undo, redo and so on	Import, save, undo, get info of image, rotate, share and so on
Special function	No	Sky filter, human face recognition
Image process	No	Emboss, grayscale, inverse sharpen, edge detection and so on
Package	Base64img in Tcl\tk	Numpy, Matplotlib, OpenCV, Numba

4. User Interface



And I design it with the Tkinter. And because there are lots of codes here, I do not paste it into the report. And I also set some button on a DISABLE state, and set it only can work after importing an image. And you can find them in the frame top.



And to make the UI more beautiful, I add some option like relief = flat, image, which makes it attractive.

5. Function

5.1 human face recognition

For the function of human face recognition, I used the classifier in a cascade of OpenCV. And I use haarcascade_frontalface_alt2.xml, and this classifier is trained by boosting algorithm.

First I need to convert the numpy image into OpenCV image. Then turn it into the grey image. Using the classifier which has been trained by

opencv. And then use it to detect the face, if the classifier recognizes the face, use the green frame to square it.

```
def humanRecognition(self):
    color = (0, 255, 0)
    cv_image = self.np_image[:, :, ::-1].copy()
    grey = cv2.cvtColor(cv_image.astype(np.uint8), cv2.COLOR_BGR2GRAY)
    classfier = cv2.CascadeClassifier("G:\opencv\sources\data\haarcascades\haarcascade_frontalface_alt2.xml")
    faceRects = classfier.detectMultiScale(grey, scaleFactor=1.2, minNeighbors=3, minSize=(32, 32))
    if len(faceRects) > 0:  # >0 can detect the face
        for faceRect in faceRects:  # mark the face
            x, y, w, h = faceRect
            cv2.rectangle(cv_image, (x - 10, y - 10), (x + w + 10, y + h + 10), color, 3)  # control the thickness
    cv2.imwrite('./2.jpg', cv_image)
    self.update_info()
    self.update_app()
```

5.2 Sky filter

For the function of the sky filter, we can use it to make a scenery much more beautiful. The idea of the filter comes from a Japanese cartoon movie called ‘Your Name', because I like the scenery in this movie. For example, the scenery this movie like that:

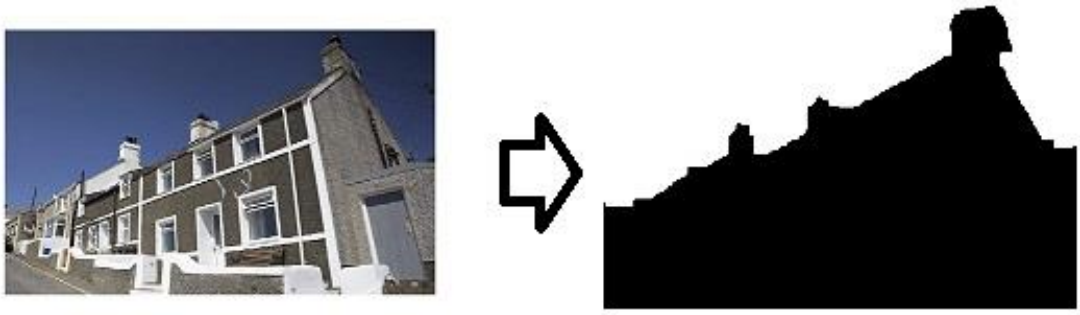
Then I'd like to design a kind of filter that can make the photo like that. So I think I can combine a picture with a sky photo.

And I try to find some similar functions from the internet, and I find most of the combine functions are designed in a process ‘get the region, use the photo to replace it, add some filter on it'. So I begin to develop a function to get the sky region.

First, I try three kinds of functions to get the sky region. Function A, because most of the sky is blue if try to use the HSV model to split the color. First, I convert the OpenCV image, BGR into HSV. And then use the split function to get three tunnels of HSV. Using inrange() to get the blue threshold we want to

use and using median blur to do some denoising. To make the mask more complete, I also add some open and erosion operation in it. And then save the region into the mask to make it become a black&white picture.

Like that:



Function B, first, I get the gradient information, then I try to use gradient energy threshold method to estimate the region of the sky. Finally, it's to examine the part of the sky, and use some open, close, erosion operation to make the region complete. The results are not as good as A.

Function C, I try to use CNN model in Tensorflow to get the region of the sky, but maybe there is less data, I only use 1000 sky photos to train the model, and the structure I used is the alexnet. But the result is a little bit worse. Maybe some of the parameters need to be altered.

Finally, I decide to use the most natural function, Function A, which have the best effect.

```
def skyRegion1(self):
    iLow = np.array([100, 43, 46])
    iHigh = np.array([124, 255, 255])
    img = cv2.imread(self.pillow_image)
    imgOri = cv2.imread(self.pillow_image)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    h, s, v = cv2.split(img)
    v = cv2.equalizeHist(v)
    hsv = cv2.merge((h, s, v))

    imgThresholded = cv2.inRange(hsv, iLow, iHigh)

    imgThresholded = cv2.medianBlur(imgThresholded, 9)

    kernel = np.ones((5, 5), np.uint8)
    imgThresholded = cv2.morphologyEx(imgThresholded, cv2.MORPH_OPEN, kernel, iterations=10)
    imgThresholded = cv2.medianBlur(imgThresholded, 9)
    pic_name = self.pillow_image.split('/')[-1].split('.')[0]
    tmp = "tmp/" + pic_name + "-mask.jpg"
    print(tmp)
    cv2.imwrite("./image/tmp.jpg", imgThresholded)
    return tmp
```

Second, we need to combine the sky with the mask

In Function A. I tried two functions here. Function C, I use the copyto(), but the result is worse, the edge seems like we insert something on the photo. Function B, I use the function seamlessClone(), and this function is not included in opencv2, so I have to learn to use the opencv3.Before using the Seamlessclone(), we also need to do some edge detection and localization to find out where we need to paste the sky And I set it to combine the mask with the following sky region:

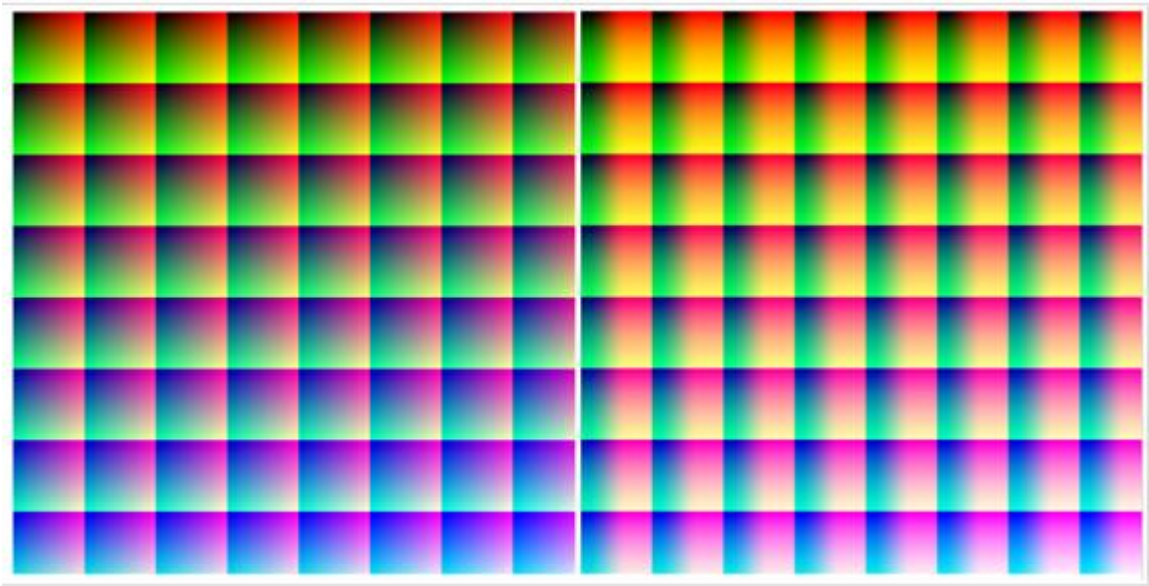
Here is an example of using Seamlessclone():

Part of the code:

```
def seamClone(skyname, picname, maskname):
    src = cv2.imread(skyname)
    dst = cv2.imread(picname)

    src_mask = cv2.imread(maskname, 0)
    src_mask0 = cv2.imread(maskname, 0)
    contours = cv2.findContours(src_mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cnt = contours[0]

    x, y, w, h = cv2.boundingRect(cnt)
    print(x, y, w, h)
    if w == 0 or h == 0:
        return dst
    dst_x = len(dst[0])
    dst_y = len(dst[1])
    src_x = len(src[0])
    src_y = len(src[1])
    scale_x = w * 1.0 / src_x
    src = cv2.resize(src, (dshape, dshape), interpolation=cv2.INTER_CUBIC)

    cv2.imwrite("src_sky.jpg", src)
    center = ((x + w) / 2, (y + h) / 2)
    print(center)

    output = cv2.seamlessClone(src, dst, src_mask0, center, cv2.NORMAL_CLONE)

    return output
```

Third, I want to design a filter to make the image beautiful, and I think this is the most challenging part when I design this filter. So I find lots of information on the internet, about how to create a filter. I find that lots of the filters use the new color to replace the old color on the image. For example, axis (0, 0), his color is (123,213,312), I need to find out where is this color in color table1

The following is the color table:



And we need to use the new color we'd like to use to replace the color table2.

Then I need to know how to localize the pixel depending on the BGR value.
And I find a format from the internet.

A&B is the 2D-coordinates.X, Y, Z is three tunnels of the color.

A = y/4 + (x/32)*64

B = z/4 + ((x%32)/4)*64

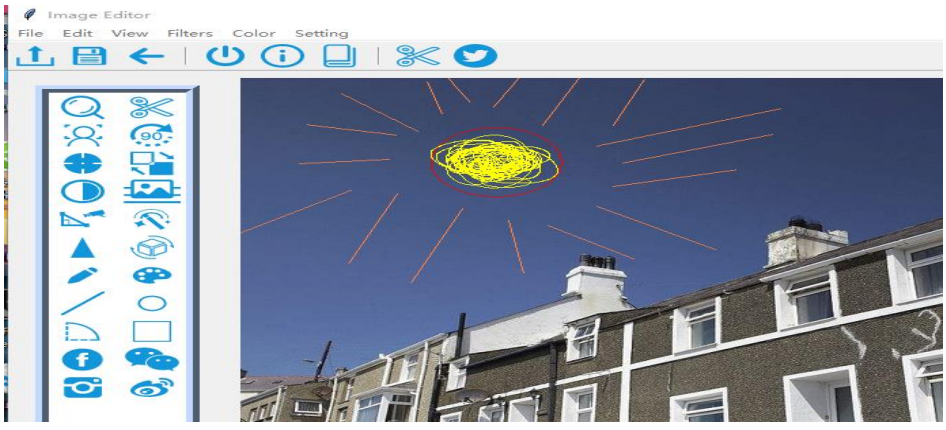
And the color table I used in sky filter, is I find from the internet.

Here is the core function:

```
def myFilter(orimap, newmap, picname):
    ori = cv2.imread(orimap)
    new = cv2.imread(newmap)
    my = cv2.imread(picname)

    pic_name = picname.split('/')[-1].split('.')[0]
    style_name = newmap.split('/')[-1].split('.')[0]

    tmp = "tmp/" + pic_name + "-" + style_name + ".jpg"

    for i in range(len(my)):
        for j in range(len(my[0])):
            pos = cv2.findNonZero(my[i][j], ori)
            my[i][j] = new[pos[0], pos[1]]

    cv2.imwrite(tmp, my)

    return tmp
```

5.3 Drawing tools

For the drawing tools, I used the features in Tkinter.canvas, such as create line, acreate_arc, create_rectangle and so on. And I bind the left click with draw and bind the right click with the press.



And I define the press and draw funcntion

```
def Press(self,event):
    self.i += 1
    self.x = event.x
    self.y = event.y
```

```
def setStates(self, state):
        self.state = state

def pan(self):
    self.add_history(self.pillow_image)
    self.setStates("pan")
```

I use I to record the times of the users to click the button. And drawing lines, circles, arc, and rectangle are the same. I use the functions in canvas and set a state init() to use it. In addition, also set a tag to check whether it can find the tools like line, circles….If it cannot find, it will create the tag with the parameter, and if it can find, it will delete the original one, and remake again.

Here is the example code of creating the line.

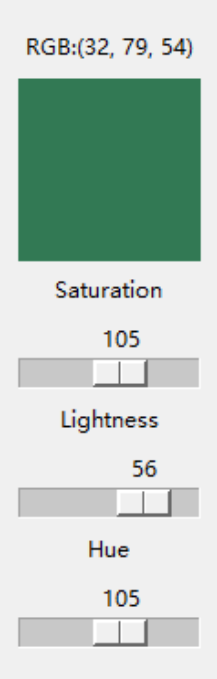
```
ef Draw(self,event):
  if self.state == 'pan':
      self.canvas.create_line(self.x, self.y, event.x, event.y, fill=self.color)
      self.x = event.x
      self.y = event.y

  if self.state == 'line':
      if not self.canvas.find_withtag('line' + str(self.i)):
          self.canvas.create_line(self.x, self.y, event.x, event.y, fill=self.color, tags='line' + str(self.i)
      else:
          self.canvas.delete('line' + str(self.i))
          self.canvas.create_line(self.x, self.y, event.x, event.y, fill=self.color, tags='line' + str(self.i)
```

5.4 share

It's one of the easiest function in my project, I only set and URL on the button, and when we click it, it will use the browser to open the URL. And you can use it to log into the social media network and share it.

```
def ins(self):
    sys.path.append("libs")
    url='https://www.instagram.com/'
    webbrowser.open(url)
    print
    webbrowser.get
```

5.5 color board



This color board is one of the useful function in my project. And we can change the saturation, lightness, and hue to get the color you want.

I set three variable R, G, B, and h, l, s. Then there is a function get(), and we can use it to get the value of the scale number. Then use configure to change the color of the background, and the number over the scale.

```
def update(self,*args):
    'color'
    r, g, b = colorsys.hls_to_rgb(self.h.get() / 255.0, self.l.get() / 255.0, self.s.get() / 255.0)
    r, g, b = r * 255, g * 255, b * 255
    self.rgb.configure(text='RGB: (%d, %d, %d)' % (r, g, b))
    self.c.configure(bg='#%02d%02d%02d' % (r, g, b))
```

5.6 filter

For the filter function, first, I define a function called apply_effect to apply for the filter like embossing, inverse, grayscale and so on.

```
def apply_effect(self, function, name):
    self.add_history(self.pillow_image)
    self.pillow_image, self.pillow_preview_image, self.np_image = function(self.np_image, self.mode)
    self.add_list(name)
    self.update_info()
    self.update_app()
```

And we will use the apply_effect function to apply for the effect like below, and I have an ImageOperation file. The imageOpearation included some if, else to examine what the mode of the image is, RGB or RGBA or P and so on. Then we use the numpy_image which we import, to do the effect. And the effect matrix is contained in a file called filters.py. Some functions of this file are copied from the internet, such as the matrix of doing blurring, emboss and so on. Besides, we have a record called edit.py, and in this file, we define the operation we use to apply for the effect of the filter.

Additionally, when I use the matrix of the filter and the numpy image to do the calculating. I use the numba package, @jit to increase the efficiency to do the calculating.

And here is an example to use the effect of edge detection:

```
def edges_detection(self):
    self.apply_effect(ImageOperations.edges_detection, "edges detection")
```

```
def edges_detection(np_image, mode):
    out = edit.filter(filters.edges_detection, np_image, mode)
    return output(out, mode)
```

```
edges_detection = (1, 0, np.array([
    [-1, -1, -1],
    [-1,  8, -1],
    [-1, -1, -1]]),
    "edges detection")
```

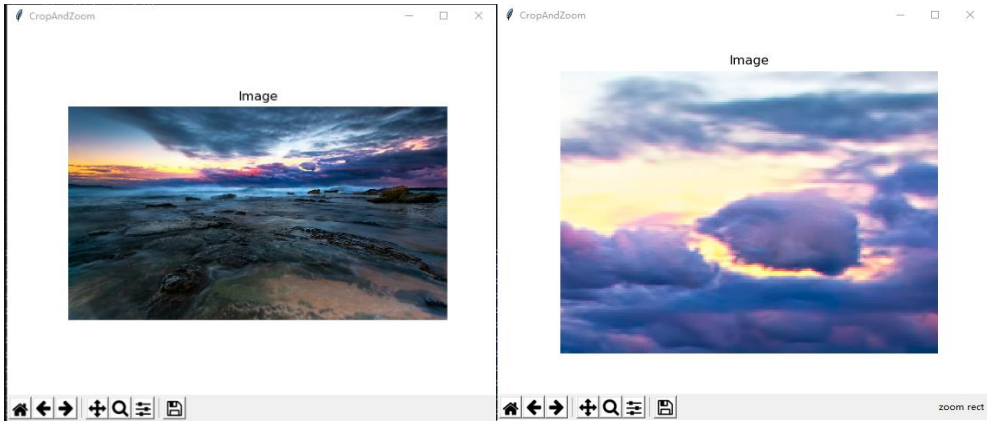
This is the way we try to apply for the impact of the filter. And other kinds of the filters are also the same.

And the following are the filters we have, and we can also use many filters at the same time:

And we can also combine lots of filters here.

5.7 crop and zoom

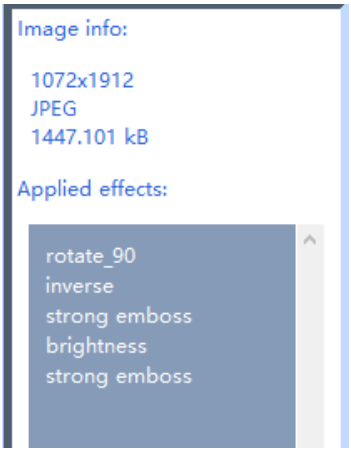
For the crop and zoom function, I use the package matplotlib. And he has a function pyplot; we use it to do the crop and zoom at the same time.



And we can use the left click to choose the area we'd like to zoom in, or use right-click to zoom out. And then use the save to crop the image you want.

5.8 history and information of the image

As you can see, there is a frame including the data of the image. And we define a function called undate_app()&update_info to save the image into a variable called pillow_review_image, and use .size to get the width, height of the image. Of course, we need to use the update the information and the image on the canvas.



And we have the function add_list() to add the operation to the list box, also a function remove_list() to delete the action in the listbox().When we use the undo function.

Also, there is a function called add_history it uses to mark down the operation we have done, and it's always useful when we design the undo function. So these several functions. Add_list(), remove_list(), update_info(), add_history, undo_history, update_app have the close relation.

```
def add_list(self, value):
    self.listbox.insert(END, value)
    self.listbox.yview(END)

def remove_list(self):
    self.listbox.delete(END)

def update_info(self):
    info, self.info_raw = ImageOperations.update_size_info(self.pillow_image, self.info_raw)
    self.label2.config(text=info)

def add_history(self, pic):
    self.history.append(pic)
    self.submenu2.entryconfig("Undo", state=NORMAL)

def undo_history(self):
    try:
        pic = self.history[-1]
        del self.history[-1]
    except:
        return

    try:
        self.history[-1]
    except:
        self.submenu2.entryconfig("Undo", state=DISABLED)

    self.pillow_image, self.pillow_preview_image, self.np_image = ImageOperations.update_image(pic, self.info_raw)
    self.update_info()
    self.remove_list()
    self.update_app()
```

5.9 Always on top

This is also an easier function that only needs to set the root to be the top level with using the Toplevel().

5.10 User manual & about us

This is the most straightforward function that we only need to use the showinfo() to print the contents in it.

5.11 load&save

This function is that I modify the code from the internet, and I add some logic to it, making it more complete. And I also add a load the main image to put the picture on the canvas and add the info.

Here is some of the code.

```
def load_main_image(self, directory):
    # add image and info
    self.pillow_image, self.pillow_preview_image, self.np_image, info, self.info_raw = ImageOperations.load_image(
        directory)

    try:
        self.canvas.delete(self.main_image)
    except:
        pass

    width, height = self.pillow_preview_image.size
    self.mode = self.pillow_image.mode
    self.data = ImageTk.PhotoImage(self.pillow_preview_image)
    self.main_image = self.canvas.create_image(0, 0, image=self.data, anchor=NW)
    self.canvas.config(width=width, height=height, bg='white')
    self.label2.config(text=info)
    self.listbox.delete(0, END)
```

6. Conclusion

6.1 challenge

1. The first time to use opencv3.3.1, and there are lots of differences from the opencv2.

2. The way to design the sky filter and I have tried lots of ways to finish it to find which one has the best effect.

3. The way to mark down the operation we have because there are lots of logic here.

4. The share button of wechat, I have found that there is a way to get the permission from the wechat and then we can log in it with one button and send it. But it's difficult to learn. I haven't finished it yet.

6.2 further improvement

1. The model of human face recognition we can try to add some deep learning structure to replace the boosting.

2. After recognizing the face, try to do something to make your face beauty.

6.3 attention

Comparing with the Group project, except for the appearance of the UI, there is nothing the same as Group project, included the way to import, save, undo and so on.

And the demo can be found in the slide of presentation from p10 to p15

7. Reference

1. OpenCV change ogs: http://code.opencv.org/projects/opencv/wiki/ChangeLog
2. OpenCV User Site: http://opencv.org/
3. ChOpenCV: http://www.softintegration.com/products/thirdparty/opencv/
4. "Python Imaging Library". *Secret Labs AB*. Retrieved December 8, 2013.
5. *"Details of package python-imaging in sid". packages.debian.org. Software in the Public Interest. Retrieved December 8, 2013.*
6. "NumPy Sourceforge Files". Retrieved 2008-03-24.
7. *"Numarray Homepage". Retrieved 2006-06-24.*
8. *"NumPy 1.5.0 Release Notes". Retrieved 2011-04-29.*
9. "Numba vs. Cython: Take 2".
10. *"numba/numba: NumPy aware dynamic Python compiler using LLVM". GitHub.*