#  
#-------------------------------------------------------------------------------  
from \_\_future\_\_ import division  
from tkinter import \*  
from tkinter import messagebox  
from PIL import Image, ImageTk  
import os  
import glob  
import random  
  
  
#keras code  
from [skimage.io](http://skimage.io/" \t "https://mail.google.com/mail/u/1/" \l "search/hybrid+yolov2/_blank) import imread  
from skimage.filters import threshold\_otsu  
import matplotlib.pyplot as plt  
car\_image = imread("C:\\Users\\Anup Rao\\Desktop\\darknet-master\\BBox-Label-Tool-master\\Images\\001\\001.jpg", as\_grey=True)  
print(car\_image.shape)  
  
gray\_car\_image = car\_image \* 255  
fig, (ax1, ax2) = plt.subplots(1, 2)  
ax1.imshow(gray\_car\_image, cmap="gray")  
threshold\_value = threshold\_otsu(gray\_car\_image)  
binary\_car\_image = gray\_car\_image > threshold\_value  
ax2.imshow(binary\_car\_image, cmap="gray")  
plt.show()  
#keras code  
  
# colors for the bboxes  
COLORS = ['red', 'blue', 'yellow', 'pink', 'cyan', 'green', 'black']  
# image sizes for the examples  
SIZE = 256, 256  
  
class LabelTool():   
    def \_\_init\_\_(self, master):  
        # set up the main frame  
        self.parent = master  
        self.parent.title("LabelTool")  
        self.frame = Frame(self.parent)  
        self.frame.pack(fill=BOTH, expand=1)  
        self.parent.resizable(width = FALSE, height = FALSE)  
  
        # initialize global state  
        self.imageDir = ''  
        self.imageList= []  
        self.egDir = ''  
        self.egList = []  
        self.outDir = ''  
        self.cur = 0  
        self.total = 0  
        self.category = 0  
        self.imagename = ''  
        self.labelfilename = ''  
        self.tkimg = None  
  
        # initialize mouse state  
        self.STATE = {}  
        self.STATE['click'] = 0  
        self.STATE['x'], self.STATE['y'] = 0, 0  
  
        # reference to bbox  
        self.bboxIdList = []  
        self.bboxId = None  
        self.bboxList = []  
        self.hl = None  
        self.vl = None  
  
        # ----------------- GUI stuff ---------------------  
        # dir entry & load  
        self.label = Label(self.frame, text = "Image Dir:")  
        self.label.grid(row = 0, column = 0, sticky = E)  
        self.entry = Entry(self.frame)  
        self.entry.grid(row = 0, column = 1, sticky = W+E)  
        self.ldBtn = Button(self.frame, text = "Load", command = self.loadDir)  
        self.ldBtn.grid(row = 0, column = 2, sticky = W+E)  
  
        # main panel for labeling  
        self.mainPanel = Canvas(self.frame, cursor='tcross')  
        self.mainPanel.bind("<Button-1>", self.mouseClick)  
        self.mainPanel.bind("<Motion>", self.mouseMove)  
        self.parent.bind("<Escape>", self.cancelBBox)  # press <Espace> to cancel current bbox  
        self.parent.bind("s", self.cancelBBox)  
        self.parent.bind("a", self.prevImage) # press 'a' to go backforward  
        self.parent.bind("d", self.nextImage) # press 'd' to go forward  
        self.mainPanel.grid(row = 1, column = 1, rowspan = 4, sticky = W+N)  
  
        # showing bbox info & delete bbox  
        self.lb1 = Label(self.frame, text = 'Bounding boxes:')  
        self.lb1.grid(row = 1, column = 2,  sticky = W+N)  
        self.listbox = Listbox(self.frame, width = 22, height = 12)  
        self.listbox.grid(row = 2, column = 2, sticky = N)  
        self.btnDel = Button(self.frame, text = 'Delete', command = self.delBBox)  
        self.btnDel.grid(row = 3, column = 2, sticky = W+E+N)  
        self.btnClear = Button(self.frame, text = 'ClearAll', command = self.clearBBox)  
        self.btnClear.grid(row = 4, column = 2, sticky = W+E+N)  
  
        # control panel for image navigation  
        self.ctrPanel = Frame(self.frame)  
        self.ctrPanel.grid(row = 5, column = 1, columnspan = 2, sticky = W+E)  
        self.prevBtn = Button(self.ctrPanel, text='<< Prev', width = 10, command = self.prevImage)  
        self.prevBtn.pack(side = LEFT, padx = 5, pady = 3)  
        self.nextBtn = Button(self.ctrPanel, text='Next >>', width = 10, command = self.nextImage)  
        self.nextBtn.pack(side = LEFT, padx = 5, pady = 3)  
        self.progLabel = Label(self.ctrPanel, text = "Progress:     /    ")  
        self.progLabel.pack(side = LEFT, padx = 5)  
        self.tmpLabel = Label(self.ctrPanel, text = "Go to Image No.")  
        self.tmpLabel.pack(side = LEFT, padx = 5)  
        self.idxEntry = Entry(self.ctrPanel, width = 5)  
        self.idxEntry.pack(side = LEFT)  
        self.goBtn = Button(self.ctrPanel, text = 'Go', command = self.gotoImage)  
        self.goBtn.pack(side = LEFT)  
  
        # example pannel for illustration  
        self.egPanel = Frame(self.frame, border = 10)  
        self.egPanel.grid(row = 1, column = 0, rowspan = 5, sticky = N)  
        self.tmpLabel2 = Label(self.egPanel, text = "Examples:")  
        self.tmpLabel2.pack(side = TOP, pady = 5)  
        self.egLabels = []  
        for i in range(3):  
            self.egLabels.append(Label(self.egPanel))  
            self.egLabels[-1].pack(side = TOP)  
  
        # display mouse position  
        self.disp = Label(self.ctrPanel, text='')  
        self.disp.pack(side = RIGHT)  
  
        self.frame.columnconfigure(1, weight = 1)  
        self.frame.rowconfigure(4, weight = 1)  
  
        # for debugging  
##        self.setImage()  
##        self.loadDir()  
  
    def loadDir(self, dbg = False):  
        if not dbg:  
            s = self.entry.get()  
            self.parent.focus()  
            self.category = int(s)  
        else:  
            #s = r'D:\workspace\python\labelGUI'  
             s = r'.'  
##        if not os.path.isdir(s):  
##            tkMessageBox.showerror("Error!", message = "The specified dir doesn't exist!")  
##            return  
        # get image list  
        self.imageDir = os.path.join(r'./Images', '%03d' %(self.category))  
        self.imageList = glob.glob(os.path.join(self.imageDir, '\*.jpg'))  
        if len(self.imageList) == 0:  
            print ('No .JPG images found in the specified dir!')  
            return  
  
        # default to the 1st image in the collection  
        self.cur = 1  
        self.total = len(self.imageList)  
  
         # set up output dir  
        self.outDir = os.path.join(r'./Labels', '%03d' %(self.category))  
        if not os.path.exists(self.outDir):  
            os.mkdir(self.outDir)  
  
        # load example bboxes  
        self.egDir = os.path.join(r'./Examples', '%03d' %(self.category))  
        if not os.path.exists(self.egDir):  
            return  
        filelist = glob.glob(os.path.join(self.egDir, '\*.jpg'))  
        self.tmp = []  
        self.egList = []  
        random.shuffle(filelist)  
        for (i, f) in enumerate(filelist):  
            if i == 3:  
                break  
            im = Image.open(f)  
            r = min(SIZE[0] / im.size[0], SIZE[1] / im.size[1])  
            new\_size = int(r \* im.size[0]), int(r \* im.size[1])  
            self.tmp.append(im.resize(new\_size, Image.ANTIALIAS))  
            self.egList.append(ImageTk.PhotoImage(self.tmp[-1]))  
            self.egLabels[i].config(image = self.egList[-1], width = SIZE[0], height = SIZE[1])  
  
        self.loadImage()  
        print ('%d images loaded from %s' %(self.total, s))  
  
    def loadImage(self):  
        # load image  
        imagepath = self.imageList[self.cur - 1]  
        self.img = Image.open(imagepath)  
        self.tkimg = ImageTk.PhotoImage(self.img)  
        self.mainPanel.config(width = max(self.tkimg.width(), 400), height = max(self.tkimg.height(), 400))  
        self.mainPanel.create\_image(0, 0, image = self.tkimg, anchor=NW)  
        self.progLabel.config(text = "%04d/%04d" %(self.cur, self.total))  
  
        # load labels  
        self.clearBBox()  
        self.imagename = os.path.split(imagepath)[-1].split('.')[0]  
        labelname = self.imagename + '.txt'  
        self.labelfilename = os.path.join(self.outDir, labelname)  
        bbox\_cnt = 0  
        if os.path.exists(self.labelfilename):  
            with open(self.labelfilename) as f:  
                for (i, line) in enumerate(f):  
                    if i == 0:  
                        bbox\_cnt = int(line.strip())  
                        continue  
                    tmp = [int(t.strip()) for t in line.split()]  
##                    print tmp  
                    self.bboxList.append(tuple(tmp))  
                    tmpId = self.mainPanel.create\_rectangle(tmp[0], tmp[1], \  
                                                            tmp[2], tmp[3], \  
                                                            width = 2, \  
                                                            outline = COLORS[(len(self.bboxList)-1) % len(COLORS)])  
                    self.bboxIdList.append(tmpId)  
                    self.listbox.insert(END, '(%d, %d) -> (%d, %d)' %(tmp[0], tmp[1], tmp[2], tmp[3]))  
                    self.listbox.itemconfig(len(self.bboxIdList) - 1, fg = COLORS[(len(self.bboxIdList) - 1) % len(COLORS)])  
  
    def saveImage(self):  
        with open(self.labelfilename, 'w') as f:  
            f.write('%d\n' %len(self.bboxList))  
            for bbox in self.bboxList:  
                f.write(' '.join(map(str, bbox)) + '\n')  
        print ('Image No. %d saved' %(self.cur))  
  
  
    def mouseClick(self, event):  
        if self.STATE['click'] == 0:  
            self.STATE['x'], self.STATE['y'] = event.x, event.y  
        else:  
            x1, x2 = min(self.STATE['x'], event.x), max(self.STATE['x'], event.x)  
            y1, y2 = min(self.STATE['y'], event.y), max(self.STATE['y'], event.y)  
            self.bboxList.append((x1, y1, x2, y2))  
            self.bboxIdList.append(self.bboxId)  
            self.bboxId = None  
            self.listbox.insert(END, '(%d, %d) -> (%d, %d)' %(x1, y1, x2, y2))  
            self.listbox.itemconfig(len(self.bboxIdList) - 1, fg = COLORS[(len(self.bboxIdList) - 1) % len(COLORS)])  
        self.STATE['click'] = 1 - self.STATE['click']  
  
    def mouseMove(self, event):  
        self.disp.config(text = 'x: %d, y: %d' %(event.x, event.y))  
        if self.tkimg:  
            if self.hl:  
                self.mainPanel.delete(self.hl)  
            self.hl = self.mainPanel.create\_line(0, event.y, self.tkimg.width(), event.y, width = 2)  
            if self.vl:  
                self.mainPanel.delete(self.vl)  
            self.vl = self.mainPanel.create\_line(event.x, 0, event.x, self.tkimg.height(), width = 2)  
        if 1 == self.STATE['click']:  
            if self.bboxId:  
                self.mainPanel.delete(self.bboxId)  
            self.bboxId = self.mainPanel.create\_rectangle(self.STATE['x'], self.STATE['y'], \  
                                                            event.x, event.y, \  
                                                            width = 2, \  
                                                            outline = COLORS[len(self.bboxList) % len(COLORS)])  
  
    def cancelBBox(self, event):  
        if 1 == self.STATE['click']:  
            if self.bboxId:  
                self.mainPanel.delete(self.bboxId)  
                self.bboxId = None  
                self.STATE['click'] = 0  
  
    def delBBox(self):  
        sel = self.listbox.curselection()  
        if len(sel) != 1 :  
            return  
        idx = int(sel[0])  
        self.mainPanel.delete(self.bboxIdList[idx])  
        self.bboxIdList.pop(idx)  
        self.bboxList.pop(idx)  
        self.listbox.delete(idx)  
  
    def clearBBox(self):  
        for idx in range(len(self.bboxIdList)):  
            self.mainPanel.delete(self.bboxIdList[idx])  
        self.listbox.delete(0, len(self.bboxList))  
        self.bboxIdList = []  
        self.bboxList = []  
  
    def prevImage(self, event = None):  
        self.saveImage()  
        if self.cur > 1:  
            self.cur -= 1  
            self.loadImage()  
  
    def nextImage(self, event = None):  
        self.saveImage()  
        if self.cur < self.total:  
            self.cur += 1  
            self.loadImage()  
  
    def gotoImage(self):  
        idx = int(self.idxEntry.get())  
        if 1 <= idx and idx <= self.total:  
            self.saveImage()  
            self.cur = idx  
            self.loadImage()  
  
##    def setImage(self, imagepath = r'test2.png'):  
##        self.img = Image.open(imagepath)  
##        self.tkimg = ImageTk.PhotoImage(self.img)  
##        self.mainPanel.config(width = self.tkimg.width())  
##        self.mainPanel.config(height = self.tkimg.height())  
##        self.mainPanel.create\_image(0, 0, image = self.tkimg, anchor=NW)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
    root = Tk()  
    tool = LabelTool(root)  
    root.resizable(width =  True, height = True)  
    root.mainloop()