Recognizing Text in Images

Write an optical character recognition application that identifies and recognizes printed text within an image.

By: Eric Herman

Professor: Dr. McVey-Pankratz

Table of Contents

1.1 Executable Installation Instructions Pg. 3

1.2 Open/Run in PyCharm Pg. 4

2.1 Structure Diagram Pg. 5

2.2 Data flow Diagram Pg. 6

3.1 Interface Pg. 7

3.2 How to open image Pg. 8

3.3 How to use live video Pg. 9

3.4 How to take a picture Pg. 10

3.5 How to improve results Pg. 11 - 12

3.6 How to search results Pg. 13

4.1 Code Pg. 14 - 25

1.1 Executable Installation Instructions

**Install Instructions**

1. Unzip OCR Program.zip on device designated device.

2. Open file names OCR Program

3. Run OCR Reader.exe

**Troubleshooting**

N**othing happens when selecting "Live video” -** A camera device is not detected on your device. Check drivers and test with windows camera application restart application.

**Nothing happens when the search button is pressed -** Ensure that chrome is installed at "C://Program Files (x86)//Google//Chrome//Application//chrome.exe". If not move chrome folder to this location and restart application.

1.2 Open/Run in PyCharm

**Install Instructions**

1. Install PyCharm

2. Create new project

3. Navigate to virtual environment and move main.py into directory. Mine was “C:\Users\[username]\PycharmProjects\pythonProject”.

4. Put image in “C:\Users\[username]\PycharmProjects\pythonProject\venv\images” named “sign.PNG”.

5. pip install OpenCV, Pillow, pytesseract, numpy, tkinter.

6. Install Tesseract from the tesseract website in the virtual environment directory.

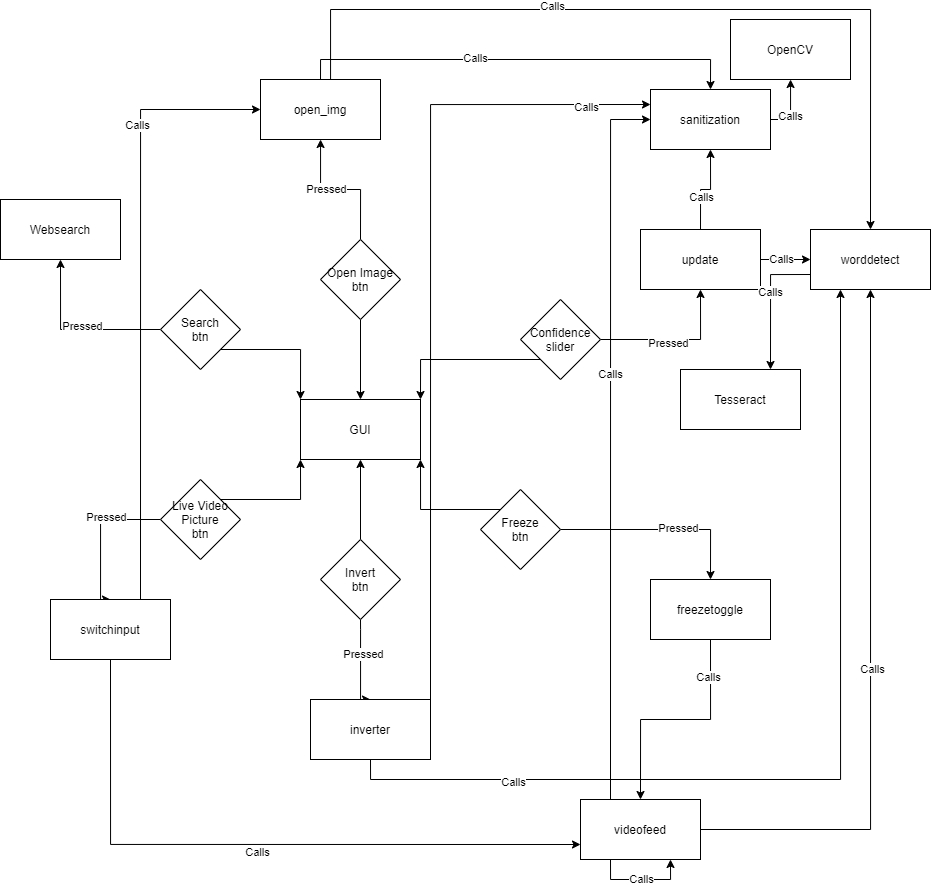
7. Run Program

**Troubleshooting**

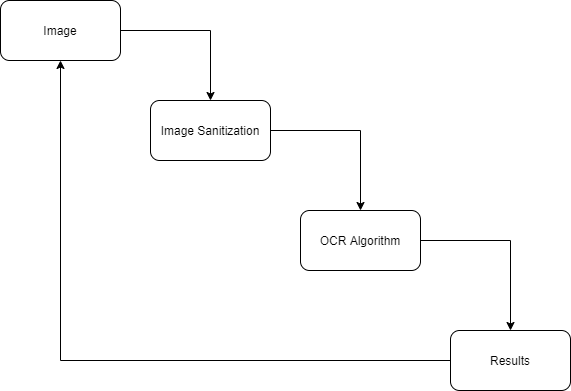
**Tesseract is not installed –** Ensure that tesseract is installed in correct directory.

**sign.PNG not found –** Make sure sign.PNG is in “C:\Users\[username]\PycharmProjects\pythonProject\venv\images”.

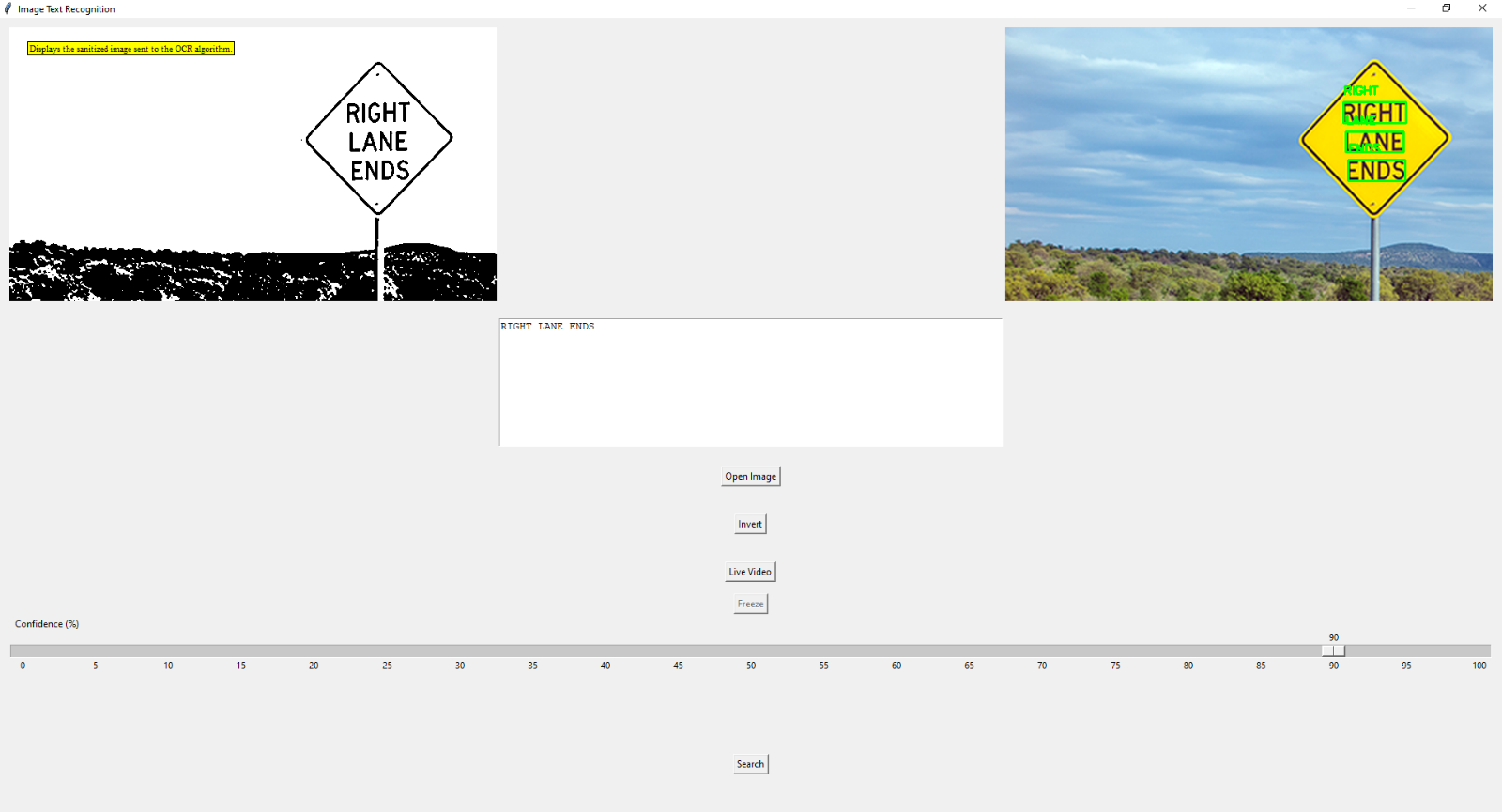
2.1 Structure Diagram



2.2 Data flow Diagram



3.1 Interface



Sanitized Image

Original Image with AR

Results Box

Opens new image

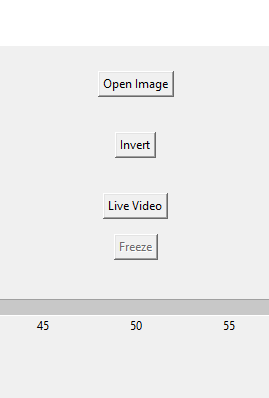
Inverts the black to white and white to black

Toggles between Live video and pictures

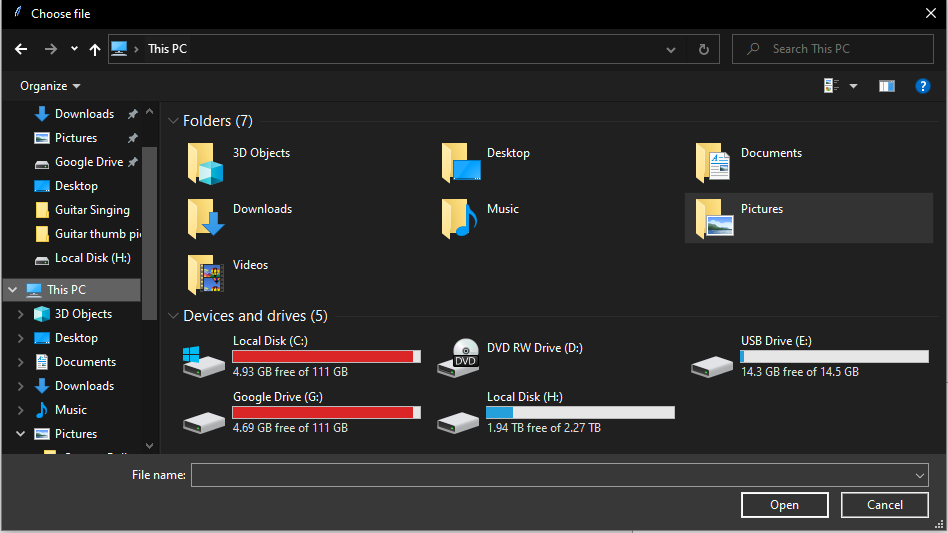
Takes a picture

Adjusts confidence threshold

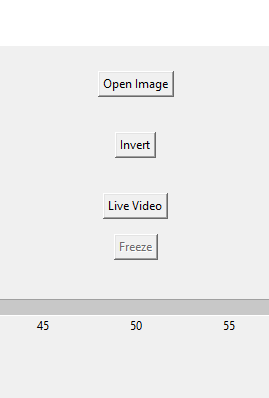
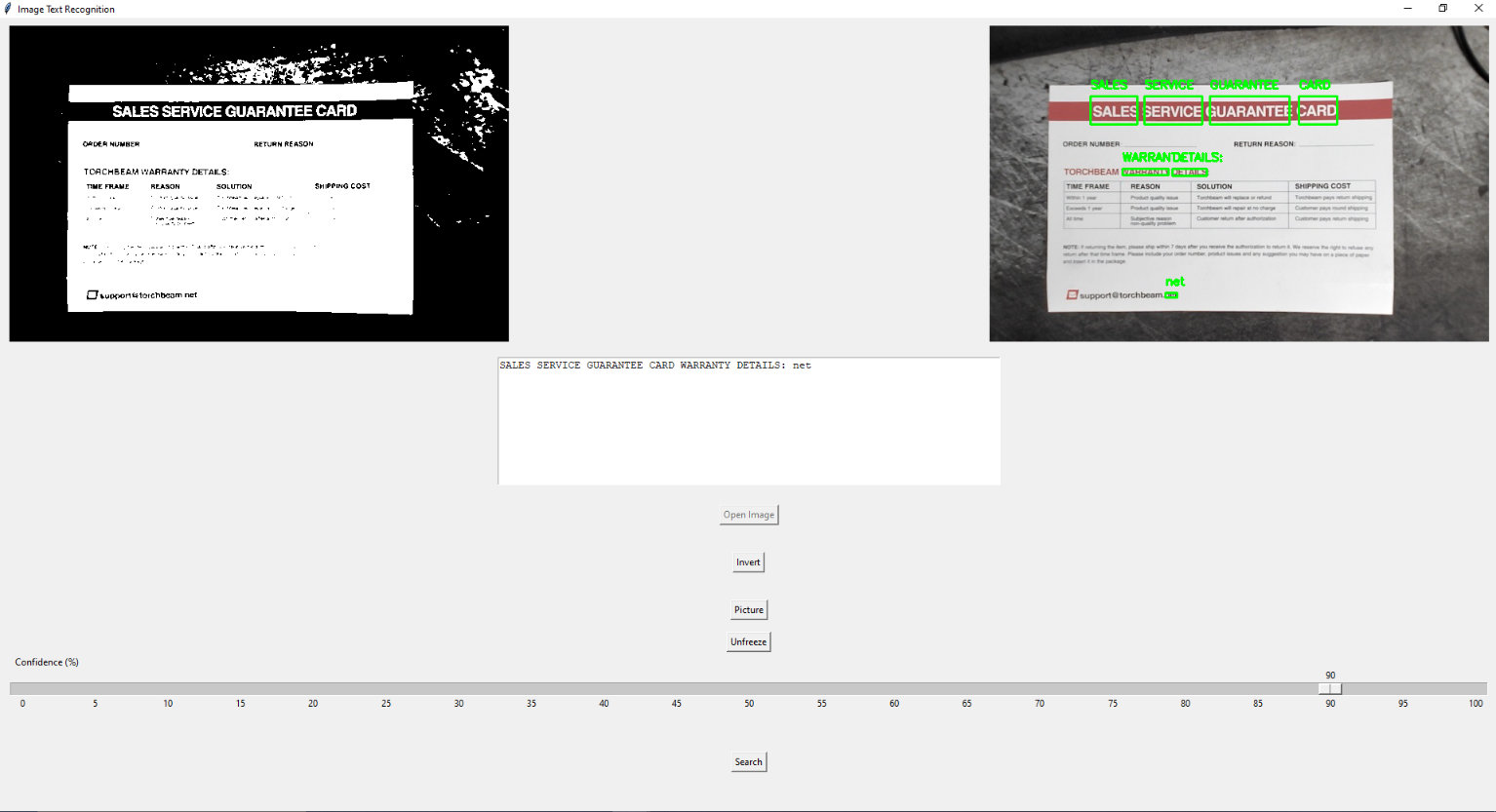
Does a Google search on results

3.2 How to open image

Click Open Image

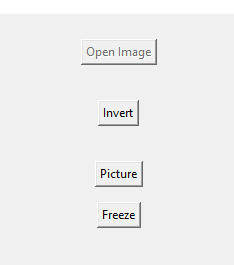
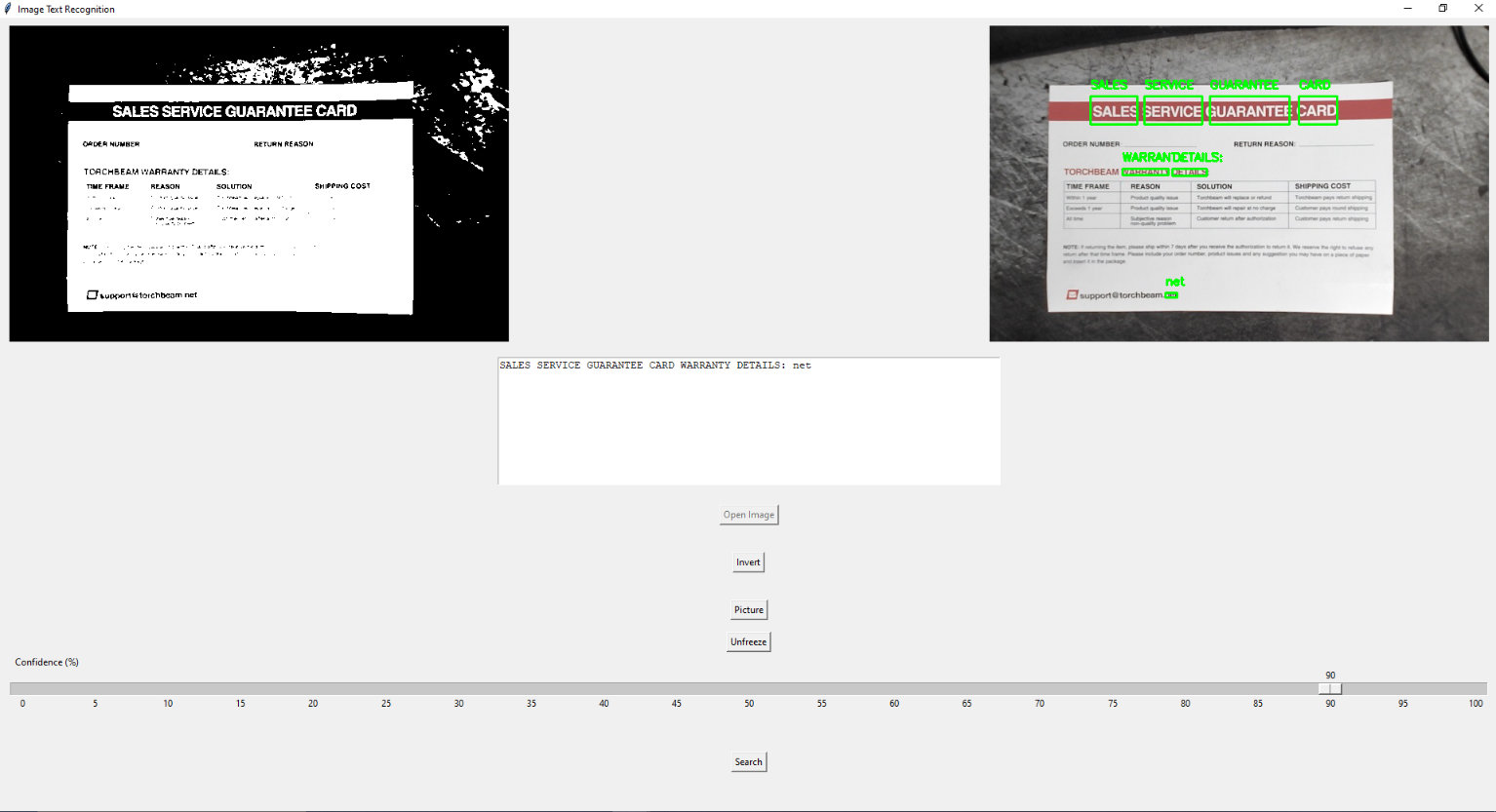


Find the Image you want to open but it must be a .png or .jpeg otherwise it will error.

3.3 How to use live video

Click Live Video

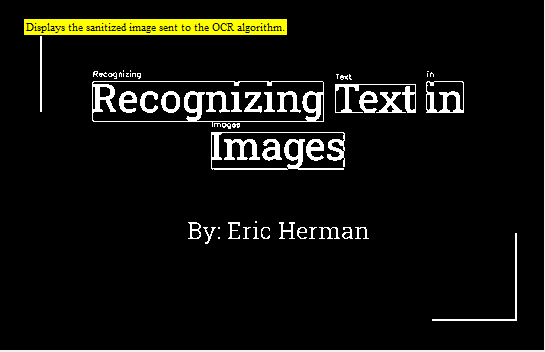
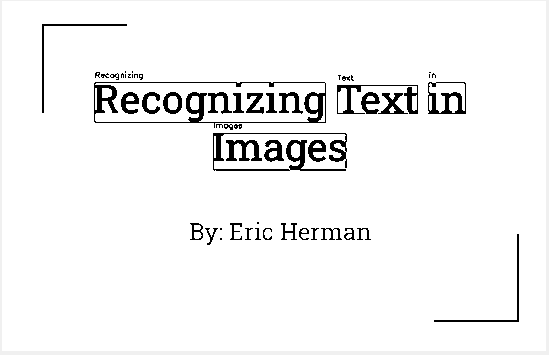
If video does not appear, ensure drivers and camera are plugged in and restart application.

3.4 How to take a picture

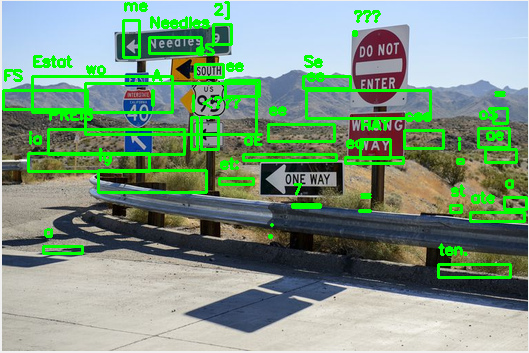
Click Freeze

The image cannot be saved but you can go back to live video by pressing the unfreeze button.

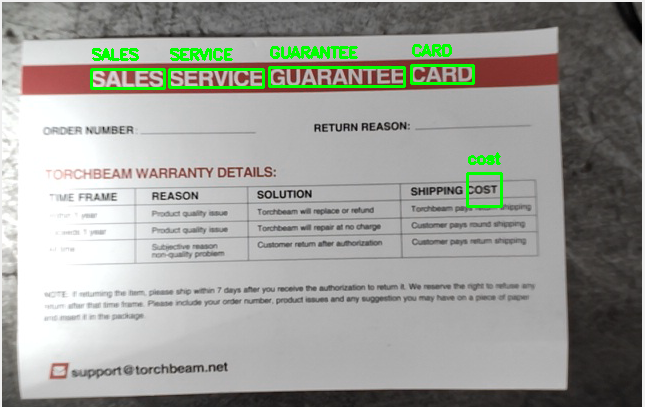
3.5 How to improve results

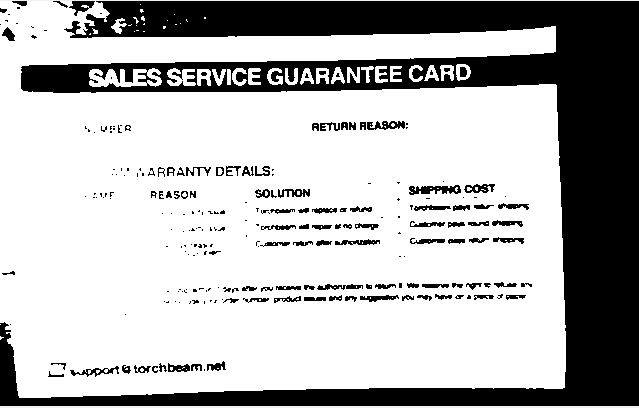
**Invert colors**

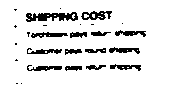
The OCR algorithm prefers the use of black text on a white background.

**Confidence Threshold**

The image on the left displays the text that it is 65% or more confident that it is correct the one on the right is 0% to 100%.

**Lighting**

Gloss or semi-gloss surfaces can cause glare and will reduce the ability to read text. Avoid hot spots with too much light.

**Text Size**

If the text is readable then it most likely will be able to be read by the program. However, if the text is too small then it will no longer be able to be read.

3.6 How to search results

Search Box

Click Search

Clicking Search will do a Google search of what ever is in the search box at the top.

4.1 Code

import cv2 as cv

import re

import PIL

from PIL import Image, ImageTk, ImageShow

import pytesseract

import numpy as np

from pytesseract import Output

import tkinter as tk

from tkinter import \*

from tkinter import filedialog

from tkinter import messagebox

#from tkinter.tix import \* #for balloon

import webbrowser

#print("Your opencv version is: " + cv2.\_\_version\_\_)

rectwidth = 20;

rectheight = 20;

class CreateToolTip(object): # makes the balloon messeges

def \_\_init\_\_(self, widget, text='widget info'):

self.widget = widget

self.text = text

self.widget.bind("<Enter>", self.enter)

self.widget.bind("<Leave>", self.close)

def enter(self, event=None):

x = y = 0

x, y, cx, cy = self.widget.bbox("insert")

x += self.widget.winfo\_rootx() + 25

y += self.widget.winfo\_rooty() + 20

# creates a toplevel window

self.tw = tk.Toplevel(self.widget)

# Leaves only the label and removes the app window

self.tw.wm\_overrideredirect(True)

self.tw.wm\_geometry("+%d+%d" % (x, y))

label = tk.Label(self.tw, text=self.text, justify='left',

background='yellow', relief='solid', borderwidth=1,

font=("times", "8", "normal"))

label.pack(ipadx=1)

def close(self, event=None):

if self.tw:

self.tw.destroy()

############################################################################################

# Description: Takes the original image and grayscales the image and threasholds and returns

# the cleaned image.

# Input: Image

# Output: Image

############################################################################################

def sanitization(img):

# displays original

#cv2.imshow('Original', frame)

# turns image to greyscale

gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY) # turns image to greyscale

# displays greyscaled image

#cv.imshow('Greyscale', gray)

# blurs the image

#blur = cv.GaussianBlur(gray, (5, 5), 0)

#denoise = cv.fastNlMeansDenoising(gray) #improves acuracy sligtly

# displays the blured and greyscaled images

#cv.imshow('Greyscale+Blur', blur)

# thresholds the image

thresholded = cv.threshold(gray, 0, 255, cv.THRESH\_BINARY + cv.THRESH\_OTSU)[1] #Thresholds image

#thresholded = cv.adaptiveThreshold(gray, 255, cv.ADAPTIVE\_THRESH\_MEAN\_C,cv.THRESH\_BINARY,11,2)

# displays the blured, greyscaled and thresholds image

#cv.imshow('Greyscale+Threshold', thresholded)

return thresholded

# def boxingcontours(orig, contours): #opencv testing code of contours

# for cnt in contours:

# x, y, w, h = cv.boundingRect(cnt)

# if w > rectwidth and h > rectheight and w<orig.shape[1] and h<orig.shape[0]: # size of the possible contour rectangles

# rect = cv.rectangle(orig, (x, y), (x + w, y + h), (0, 0, 255), 2)

# cv.imshow('word detect', rect)

# return orig[y:y+h, x:x+w]

#detects individual chars in image

# def chardetect(frame,orig):

# height = orig.shape[0]

# width = orig.shape[1]

# custom\_config = r'-l eng --psm 6'

# d = pytesseract.image\_to\_boxes(frame, output\_type=Output.DICT, config=custom\_config)

# n\_boxes = len(d['char'])

# for i in range(n\_boxes):

# #if int(d['conf'][i]) > accuracythresh:

# #(x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])

# (text, x1, y2, x2, y1) = (d['char'][i], d['left'][i], d['top'][i], d['right'][i], d['bottom'][i])

# cv.rectangle(orig, (x1, height - y1), (x2, height - y2), (0, 255, 0), 2)

# # wordlength = len(d['text'][i])

# # print(wordlength)

# # letterwidth = w/wordlength

# # print(letterwidth)

# # for j in range(wordlength):

# # startx = int(x + (letterwidth \* j))

# # width = int(startx + letterwidth)

# # orig = cv.rectangle(orig, (startx, y), (width, y + h), (0, 255, 0), 2)

# #orig = cv.rectangle(orig, (x, y), (x + w, y + h), (0, 255, 0), 2)

# print(text)

# #cv.waitKey(0)

#

# cv.imshow('Boxed', orig)

############################################################################################

# Description: Reads the text in the image and puts a box around the text along with the

# text that was found.

# Input: sanitized image, original image, confidence threshold

# Output: original image

############################################################################################

def worddetect(frame,orig, confidence):

global result

custom\_config = r'--oem 3 -l eng --psm 6 ' # configuration of tesseract

d = pytesseract.image\_to\_data(frame,lang='eng', output\_type=Output.DICT, config=custom\_config) # Runs OCR on image

n\_boxes = len(d['text'])

result.delete('1.0',END) # clears textbox

for i in range(n\_boxes):

if int(d['conf'][i]) > confidence: # Only displays words with certain confidence threshold

(x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i]) # coordinates of the text

# cv.rectangle(orig, (x, y), (x+w, y+h), (255, 255, 255), -1)#covers text and overlays with what it reads

# cv.putText(orig, d['text'][i], (x, y+12), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,0), 2)

cv.rectangle(orig, (x, y), (x + w, y + h), (0, 255, 0), 2)#boxes text

cv.putText(orig, d['text'][i], (x, y - 10), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)# writes text in top left of the box

result.insert(END, d['text'][i]) # writes in textbox

result.insert(END, ' ') # inserts space between words

#print(d['text'][i])

#print(d['conf'][i])

#cv.waitKey(0)

#cv.imshow('Boxed', orig)

# #Main Picture does the OCR on images only no interface

# frame = cv.imread('venv/Pictures/sign.PNG',1)

# #print(frame)

# sanitized = sanitization(frame)

# # contours, \_ = cv.findContours(sanitized, cv.RETR\_TREE, cv.CHAIN\_APPROX\_NONE)

# # cropped = boxingcontours(frame, contours)

# # sanitizedagain = sanitization(cropped)

# # worddetect(sanitizedagain, cropped)

# worddetect(sanitized,frame)

# # When everything done, release the capture

# cv.waitKey(0)

# cv.destroyAllWindows()

# def worddetect2(frame,orig):

# global result

# custom\_config = r'--oem 3 -l eng --psm 6 '

# d = pytesseract.image\_to\_data(frame,lang='eng', output\_type=Output.DICT, config=custom\_config)

# n\_boxes = len(d['text'])

# #result.delete('1.0',END)

# for i in range(n\_boxes):

# if int(d['conf'][i]) > 80:

# (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])

# # cv.rectangle(orig, (x, y), (x+w, y+h), (255, 255, 255), -1)#covers text and overlays with what it reads

# # cv.putText(orig, d['text'][i], (x, y+12), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0,0,0), 2)

# cv.rectangle(orig, (x, y), (x + w, y + h), (0, 255, 0), 2)#boxes text and writes what it reads on top

# cv.putText(orig, d['text'][i], (x, y - 10), cv.FONT\_HERSHEY\_SIMPLEX, 0.5, (0, 255, 0), 2)

# #result.insert(END, d['text'][i])

# #result.insert(END, ' ')

# #print(d['text'][i])

# #print(d['conf'][i])

# #cv.waitKey(0)

#

# cv.imshow('Boxed', orig)

#

# #Main Camera does camera feed without an interface

# cap = cv.VideoCapture(0)

# while (True):# Capture frame-by-frame

# ret, frame = cap.read()

# sanitized = sanitization(frame)

# worddetect2(sanitized, frame)

# if cv.waitKey(1) & 0xFF == ord('q'):

# break

# # When everything done, release the capture

# cap.release()

# cv.destroyAllWindows()

#gui interface displays camera feed

# width, height = 1080, 720

# cap = cv.VideoCapture(0)

# cap.set(cv.CAP\_PROP\_FRAME\_WIDTH, width)

# cap.set(cv.CAP\_PROP\_FRAME\_HEIGHT, height)

#

# root = Tk()

# root.bind('<Escape>', lambda e: root.quit())

# lmain = Label(root)

# lmain.pack()

#

# def show\_frame():

# ret, frame = cap.read()

# if ret:

# sanitized = sanitization(frame)

# worddetect(sanitized, frame)

# cvimage = cv.cvtColor(frame, cv.COLOR\_BGR2RGBA)

# img = PIL.Image.fromarray(cvimage)

# imgtk = ImageTk.PhotoImage(image=img)

# lmain.imgtk = imgtk

# lmain.configure(image=imgtk)

# lmain.after(10, show\_frame)

#

# show\_frame()

# root.mainloop()

############################################################################################

# Description: Opens image and then resizes converts to array and displays the original and

# detected word image on the screen when the Open Image button is pressed.

# Input: Nothing

# Output: Nothing

############################################################################################

def open\_img():

global panelA, panelB, confslider, x, result, invert, freash #global variables

x = filedialog.askopenfilename(title='Choose file') #opens a explorer window to search for images

try:

freash = PIL.Image.open(x) #opens the image and sets it to the variable

except IOError:

messagebox.showerror("ERROR", "INVALID DATA TYPE MUST BE JPG OR PNG") #displays error message if file type is wrong

#print("invalid image type")

hsize = freash.size[1] # gets the height of the image

wsize = freash.size[0] # gets width of the image

freash = freash.resize((int((350 / hsize) \* wsize), 350), PIL.Image.ANTIALIAS) # uniformly scales the image

orig = freash

img = np.asarray(orig) # converts image to array

#orig = ImageTk.PhotoImage(orig)

img = cv.cvtColor(img, cv.COLOR\_BGR2RGB) # changes BGR to RGB

sanitized = sanitization(img) # runs sanitization on image

if invert['text'] == "Revert": # inverts black and white if invert is selected

sanitized = cv.bitwise\_not(sanitized)

worddetect(sanitized, img, confslider.get()) # runs OCR on image

sanitized = cv.cvtColor(sanitized, cv.COLOR\_RGB2BGR) # changes RGB to BGR

orig = PIL.Image.fromarray(sanitized) # converts the image back

orig = ImageTk.PhotoImage(orig) # changes format so that it works for tkinter

if panelA is None: # creates an image in panelA if not created

panelA = Label(root, image=orig)

panelA.image = orig

panelA.pack(side="left", padx=10, pady=10)

else:

panelA.configure(image=orig) # changes image to the new original

panelA.image = orig

img = cv.cvtColor(img, cv.COLOR\_RGB2BGR) # changes RGB to BGR

frame = PIL.Image.fromarray(img) # converts the image back

frame = ImageTk.PhotoImage(frame) # changes format so that it works for tkinter

if panelB is None: # creates and image in panelB if not created yet

panelB = Label(image=frame)

panelB.image = frame

panelB.pack(side="right", padx=10, pady=10)

else:

panelB.configure(image=frame)# changes image to new modified image

panelB.image = frame

############################################################################################

# Description: Similar to open\_img but is called when variables of the OCR or sanitization

# process are changed and then updates the graphical interface.

# Input: threshold value

# Output: Nothing

############################################################################################

def update(num):

global panelA, panelB, confslider, x, result, invert, freash

orig = freash

# hsize = orig.size[1]

# wsize = orig.size[0]

# orig = orig.resize((int((350 / hsize) \* wsize), 350), PIL.Image.ANTIALIAS)

img = np.asarray(orig) # converts image to array

img = cv.cvtColor(img, cv.COLOR\_BGR2RGB) # changes BGR to RGB

sanitized = sanitization(img) # sanitizes image

if invert['text'] == "Revert": # reverts if selected

sanitized = cv.bitwise\_not(sanitized)

worddetect(sanitized, img, confslider.get()) # runs OCR

sanitized = cv.cvtColor(sanitized, cv.COLOR\_RGB2BGR) # changes RGB to BGR

orig = PIL.Image.fromarray(sanitized) # converts image back

orig = ImageTk.PhotoImage(orig) # modifies image for Tkinter

if panelA is None: # updates panelA

panelA = Label(root, image=orig)

panelA.image = orig

panelA.pack(side="left", padx=10, pady=10)

#tip.bind\_widget(panelA, balloonmsg="Displays the sanitized image sent to the OCR algorithm.")

panelA\_ttp = CreateToolTip(panelA,"Displays the sanitized image sent to the OCR algorithm.")

else:

panelA.configure(image=orig)

panelA.image = orig

img = cv.cvtColor(img, cv.COLOR\_RGB2BGR) # changes RGB to BGR

frame = PIL.Image.fromarray(img) # converts the image back

frame = ImageTk.PhotoImage(frame) # modifies the image for Tkinter

if panelB is None: # updates panelB

panelB = Label(image=frame)

panelB.image = frame

panelB.pack(side="right", padx=10, pady=10)

#tip.bind\_widget(panelB, balloonmsg="Displays the output of what is found using the OCR algorithm with the results boxed on the original image.")

panelB\_ttp = CreateToolTip(panelB, "Displays the output of what is found using the OCR algorithm with the results boxed on the original image.")

else:

panelB.configure(image=frame)

panelB.image = frame

############################################################################################

# Description: Does the logic of the invert and revert button in GUI along with updating the

# images.

# Input: Nothing

# Output: Nothing

############################################################################################

def inverter():

global panelA, panelB, confslider, x, result, invert, root, inversion, freash

orig = freash

# hsize = orig.size[1]

# wsize = orig.size[0]

# orig = orig.resize((int((350 / hsize) \* wsize), 350), PIL.Image.ANTIALIAS)

img = np.asarray(orig) # converts to array

img = cv.cvtColor(img, cv.COLOR\_BGR2RGB) # changes BGR to RGB

sanitized = sanitization(img) # sanitizes image

if invert['text'] == "Invert": # If inverts the it wil invert the image

sanitized = cv.bitwise\_not(sanitized)

invert['text'] = "Revert"

else: # if revert then change to invert

invert['text'] = "Invert"

worddetect(sanitized, img, confslider.get()) # runs OCR

sanitized = cv.cvtColor(sanitized, cv.COLOR\_RGB2BGR) # changes RGB to BGR

orig = PIL.Image.fromarray(sanitized) # converts image back

orig = ImageTk.PhotoImage(orig) # convert for Tkinter

if panelA is None:# update panelA

panelA = Label(root, image=orig)

panelA.image = orig

panelA.pack(side="left", padx=10, pady=10)

else:

panelA.configure(image=orig)

panelA.image = orig

img = cv.cvtColor(img, cv.COLOR\_RGB2BGR) # changes RGB to BGR

frame = PIL.Image.fromarray(img) # converts the image back

frame = ImageTk.PhotoImage(frame) # converts for Tkinter

if panelB is None: # update panelB

panelB = Label(image=frame)

panelB.image = frame

panelB.pack(side="right", padx=10, pady=10)

else:

panelB.configure(image=frame)

panelB.image = frame

############################################################################################

# Description: Grabs a new frame from the camera every 200 ms and runs the worddetect function

# every new frame that is imputed.

# Input: Nothing

# Output: Nothing

############################################################################################

def videofeed():

global panelA, panelB, confslider, x, result, invert, root, inversion, freash, cameratoggle, btn, cap, vidcon

if vidcon: # if bool is true how to stop video feed

ret, frame = cap.read() # reads image

img = np.asarray(frame) # converts to array

img = cv.cvtColor(img, cv.COLOR\_BGR2RGB) # to RGB

freash = img

if ret: # if there is an image from camera

sanitized = sanitization(frame) # sanitize image

if invert['text'] == "Revert": # keep inverted if inverted

sanitized = cv.bitwise\_not(sanitized)

cvimage = cv.cvtColor(sanitized, cv.COLOR\_BGR2RGBA) # to RGB

img = PIL.Image.fromarray(cvimage) # converts image back

imga = ImageTk.PhotoImage(image=img) # sanitized image in panelA

panelA.configure(image=imga)

panelA.image = imga

worddetect(sanitized, frame, confslider.get()) # runs OCR

cvimage2 = cv.cvtColor(frame, cv.COLOR\_BGR2RGBA) # to RGB

img2 = PIL.Image.fromarray(cvimage2)

imgb = ImageTk.PhotoImage(image=img2) # applies OCR image to panelB

panelB.configure(image=imgb)

panelB.image = imgb

panelB.after(200, videofeed) # event timer for 200 ms

#videofeed();

# panelB.after(10, videofeed)

# worddetect(sanitized, frame)

# cvimage = cv.cvtColor(frame, cv.COLOR\_BGR2RGBA)

# img = PIL.Image.fromarray(cvimage)

# imgtk = ImageTk.PhotoImage(image=img)

# lmain.imgtk = imgtk

# lmain.configure(image=imgtk)

# lmain.after(10, show\_frame)

############################################################################################

# Description: Does the logic for toggling between live video and pictures.

# Input: Nothing

# Output: Nothing

############################################################################################

def switchinput():

global panelA, panelB, confslider, x, result, invert, root, inversion, freash, cameratoggle, btn, vidcon, camerafreeze

if cameratoggle['text'] == "Live Video":

btn["state"] = DISABLED

camerafreeze["state"] = NORMAL

cameratoggle['text'] = "Picture"

vidcon = True

videofeed()

else:

btn["state"] = NORMAL

camerafreeze["state"] = DISABLED

cameratoggle['text'] = "Live Video"

camerafreeze['text'] = "Freeze"

vidcon = False

open\_img()

############################################################################################

# Description: Toggles the ability to freeze the live video feed and unfreeze.

# Input: Nothing

# Output: Nothing

############################################################################################

def freezetoggle():

global panelA, panelB, confslider, x, result, invert, root, inversion, freash, cameratoggle, btn, vidcon, camerafreeze

if camerafreeze['text'] == "Freeze":

camerafreeze['text'] = "Unfreeze"

vidcon = False

else:

camerafreeze['text'] = "Freeze"

vidcon = True

videofeed()

############################################################################################

# Description: Does a google search of what ever text is found in the textbox.

# Input: Nothing

# Output: Nothing

############################################################################################

def websearch():

global panelA, panelB, confslider, x, result, invert, root, inversion, freash, cameratoggle, btn, vidcon, camerafreeze, search

url = "https://www.google.com.tr/search?q={}".format(result.get(1.0, "end"))

webbrowser.register('chrome',None,webbrowser.BackgroundBrowser("C://Program Files (x86)//Google//Chrome//Application//chrome.exe"))

webbrowser.get('chrome').open(url)

root = Tk() # creates tkinter window called root

width, height = 525, 350 # resolution of video feed

cap = cv.VideoCapture(0) # handle for camera

cap.set(cv.CAP\_PROP\_FRAME\_WIDTH, width) # applies width

cap.set(cv.CAP\_PROP\_FRAME\_HEIGHT, height) # applies height

vidcon = True # video feed continue bool

x = 'venv/Pictures/sign.PNG' # default image on start up

freash = PIL.Image.open(x) # opens image

hsize = freash.size[1] # gets height

wsize = freash.size[0] # gets width

freash = freash.resize((int((350 / hsize) \* wsize), 350), PIL.Image.ANTIALIAS) # uniform scale

inversion = False

panelA = None

panelB = None

root.title("Image Text Recognition") # sets title of window

# w, h = root.winfo\_screenwidth(), root.winfo\_screenheight()

# root.geometry("%dx%d+0+0" % (w,h))

#root.attributes('-zoomed', True)

root.state("zoomed") # opens application in full screen

#root.geometry("1000x600")

#root.resizable(width = True, height = True)

#tip = Balloon(root) # applies balloon pop up messages on root

search = Button (root, text = "Search", command = websearch ) # initalizes the search button

search.pack(side="bottom", padx="10", pady="50")

#tip.bind\_widget(search, balloonmsg = "Does a google search on the results.") # creates pop-up text

search\_ttp = CreateToolTip(search, "Does a google search on the results.")

confslider = Scale(root, from\_=0, to=100,tickinterval=5, length=600, resolution=5, label = 'Confidence (%)', orient=HORIZONTAL, command = update) # initalizes the confidence slider

confslider.pack(side="bottom", fill="both", expand="yes", padx="10", pady="0")

#tip.bind\_widget(confslider, balloonmsg = "Adjusts the confidence level threshold to improve or decrease the accuracy of the OCR algorithm. ") # creates pop-up text

confslider\_ttp = CreateToolTip(confslider, "Adjusts the confidence level threshold to improve or decrease the accuracy of the OCR algorithm. ")

camerafreeze = Button (root, text = "Freeze", command = freezetoggle, state = DISABLED) # initalizes the freeze button

camerafreeze.pack(side="bottom", padx="10", pady="0")

#tip.bind\_widget(camerafreeze, balloonmsg = "Allows for pictures to be taken when in live video feed mode.") # creates pop-up text

camerafreeze\_ttp = CreateToolTip(camerafreeze, "Allows for pictures to be taken when in live video feed mode.")

cameratoggle = Button (root, text = "Live Video", command = switchinput) # initalizes the toggle button between live video and pictures

cameratoggle.pack(side="bottom", padx="10", pady="15")

#tip.bind\_widget(cameratoggle, balloonmsg = "Toggles between live video feed and opening pictures already saved on your device.") # creates pop-up text

cameratoggle\_ttp = CreateToolTip(cameratoggle, "Toggles between live video feed and opening pictures already saved on your device.")

invert = Button(root, text = "Invert", command = inverter) # initalizes the invert button

invert.pack(side="bottom", padx="10", pady="20")

#tip.bind\_widget(invert, balloonmsg = "Changes all black to white and white to black in the image sent to the OCR algorithm.") # creates pop-up text

invert\_ttp = CreateToolTip(invert, "Changes all black to white and white to black in the image sent to the OCR algorithm.")

btn = Button(root, text = 'Open Image', command = open\_img ) # initalizes the open image button

btn.pack(side="bottom", padx="10", pady="15")

#tip.bind\_widget(btn, balloonmsg = "Allows for the picture to be changed when in picture mode.") # creates pop-up text

btn\_ttp = CreateToolTip(btn, "Allows for the picture to be changed when in picture mode.")

confslider.set(90)

result = Text(root, height = "10") # initalizes the textbox with results

result.pack(side="bottom",padx="10", pady="10")

#tip.bind\_widget(result, balloonmsg = "Google search bar of results from the image.") # creates pop-up text

result\_ttp = CreateToolTip(result, "Google search bar of results from the image.")

root.mainloop() # game loop esencially to make the GUI reactive