DECEMBER 22, 2020

DOCUMENTATION FOR OCR - EDITOR

AUTHOR : MOHAMED IRFAN

MOHAMED IRFAN
SAVEETHA ENGINEERING COLLEGE
Chennai

Contents

1. Problem statement:

To optically recognize a handwritten image or typed text and save the detected text to an external file.

2. My Solution:

I have used open CV in python for the backend code and Tkinter for the front end. Text can be recognized using tesseract module but at times, some images require preprocessing to be recognized properly.

So, the idea I came with is to design a desktop app that might help with the preprocessing and also the OCR part.

The application must be able to edit and tinker certain values of the image, like B/W, threshold limit of the image, median blur

3. Requirements:

Pillow (PIL)

Pytesseract

Opencv-python

Numpy

Docx2pdf

Python-docx

4. Documenatation

4.1 Constants

Main.py

Filename (str) : the default image in the editor. It can be found in the root folder of the application

Constants.py

BG (str) : hex color code for background

FG (str) : hex color code for foreground

TEXT (str) : hex color code for Text

SECONDARY (str) : hex color code for extra color

4.2 Variables

Black_and_white (IntVar) : to store the user choice, whether the image should be colourised or in black and white.

Img (PIL Image) : to save the default image in the executable file

Save_path (str): current path of the image to be saved

Export_path (str): current path of the text to be exported

Output (str): recognized text is stored

Text (str): dummy variable to save the copy of ouput

4.3 Functions

4.3.1 Remove_control_characters :

Args : S (str)

Returns : str : string with only ASCII characters

Usage : remove_control_characters(s : str) -> str :

```
def remove_control_characters(s):
    """this function is to remove non ASCII characters

    Args:
        s (str): string

    Returns:
        str: string with only ASCII characters
    """
    res = ''
    for i in s:
        try:
            if unicodedata.category(i)[0] != 'C' or i == '\n':
                res += i
        except:
            print(i)
    return res
```

4.3.2 Get_file

Args: path (str): [path of the file to be exported]

returns: None

Function : To export the file as docx or pdf

Usage : get_file(path):

```python
def get_file(path):
    """To export the file as docx or pdf file

    Args:
        path (str): [path of the file to be exported]
    """
    global text
    extension = path.split('.')[1]
    filename = path.split('.')[0].split('/')[-1]
    print(filename)
    doc = docx.Document()
    doc.add_heading('Output', 0)
    text = remove_control_characters(text)

    text_copy = text.split('\n')
    final = []

    for line in text_copy:
        if line.isspace() or line == '':
            continue
        final.append(line)


    if extension != 'pdf':
        for line in final:
            para = doc.add_paragraph(line)
            run = para.add_run()
            run.add_break()
        doc.save(path)
    else:
        filename = filename + '.docx'
        try:
            remove(filename)
        except:
            pass
        for line in final:
            para = doc.add_paragraph(line)
            run = para.add_run()
            run.add_break()
        doc.save(filename)
        convert(filename,path)
        remove(filename)
```

4.3.3 Export

Args : None

Returns : None

Function : wrapper function to call get_file()

Usage : export():

```
def export():
    """to export a file which is already saved
    """
    global export_path
    if export_path == None:
        export_as()
    else:
        get_file(export_path)
```

4.3.4 Export_as

Args : None

Returns : None

Function : wrapper function to call get_file()

Usage : export_as()

```
def export_as():
    """To export the file as pdf or docx
    """
    global export_path
    files = [("Text files","*.docx"),
             ("PDF files","*.pdf"),
             ("all files","*.*")]
    try:
        export_path = asksaveasfile(filetypes = files, defaultextension = files).name
    except:
        return

    get_file(export_path)
```

4.3.5 Save

Args : None

Returns : None

Function : wrapper function to call save_as with an alias

Usage : save()

```
def save():
    """To save the edited image
    """
    global save_path
    if save_path == None:
        save_as()
    else:
        copy(edit_name,save_path)
```

4.3.6 Save_as

Args : None

Returns : None

Function : to save the edited image using file dialog box

Usage : save_as()

```
def save_as():
    """To edit the image as png or jpeg
    """
    global save_path,edit_name
    files = [("png files","*.png"),
             ("jpg files","*.jpg"),
             ("all files","*.*")]
    try:
        save_path = asksaveasfile(filetypes = files, defaultextension = files).name
    except:
        return
    copy(edit_name,save_path)
```

4.3.7 Open_edit

Args : None

Returns : None

Function : to show the preview of the edited image

Usage : open_edit()

```
def open_edit():
    """To open a new image
    """
    image = cv2.imread(edit_name)
    cv2.imshow("Edited", image)
```

4.3.8 Change_image

Args : None

Returns : None

Function : to change the image in the tkinter application

Usage : change_image()

```
def change_image(path):
    """To change the preview of the edited image

    Args:
        path ([str]): [the path of the edited image]
    """
    global img
    img = Image.open(path)
    img = img.resize((500,281) ,  Image.ANTIALIAS)
    img = ImageTk.PhotoImage(img)
    original.configure(image = img)
```

4.3.9 Refresh

Args : None

Returns : None

Function : to refresh all the edits happened to the image

```
def refresh():
    """Refreshing the edited image --- works on a seperate thread
    """
    global edit_name,text
    black = 0
    curr = None
    thresh = 0
    med_blur = 1
    should_change = 0
    t_start = 0
    t_end = 0
    alphanum = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#$%^&*()-_=+`~"\',.<>/?{}[]\\|'

    while True:
        global image,filename
        prev_file = curr
        curr = filename

        image = cv2.imread(filename)

        prev_black = black
        black = black_and_white.get()

        prev_thresh = thresh
        thresh = threshold.get()

        prev_med_blur = med_blur
        med_blur = median_blur.get()

        prev_t_start = t_start
        prev_t_end = t_end

        t_start = thresh_start.get()
        t_end = thresh_end.get()

        gray = image

        if black:
            gray = cv2.cvtColor(gray, cv2.COLOR_BGR2GRAY)

        if thresh:
            gray = cv2.threshold(gray,t_start,t_end,cv2.THRESH_BINARY)[1]

        if thresh and (t_start != prev_t_start or t_end != prev_t_end):
            should_change = 1
        else:
            should_change = 0
```

```
    # if prev_med_blur != med_blur:
    if not med_blur % 2:
        med_blur += 1
    gray = cv2.medianBlur(gray,med_blur)



    edit_name = 'edit.png'

    # print(pytesseract.image_to_boxes(Image.open(edit_name)))


    if prev_black != black or prev_file != curr or prev_thresh != thresh or prev_med_blur != med_blur or should_change:

        cv2.imwrite(edit_name, gray)

        img = Image.open(edit_name)
        img = img.resize((500,281) ,  Image.ANTIALIAS)
        img = ImageTk.PhotoImage(img)
        edit.configure(image = img)
        text = pytesseract.image_to_string(Image.open(edit_name))
        for i in alphanum:
            if i in text:
                output.delete('1.0',END)
                output.insert('1.0',text)
                break
        else:
            output.delete('1.0',END)
```

4.3.10 Open_file

Args : None

Returns : None

Function : Opens a dialog box to choose an image.wrapper for change_image() function

```
def open_file():
    """To open a photo"""
    global image
    filename_copy = filedialog.askopenfilename(initialdir = "/",title = "Select a File",filetypes = (("image files","*.png*"),
                                                                                                     ("image files","*.jpg*"),
                                                                                                     ("all files","*.*")))

    # Change label contents
    if filename_copy:
        global filename
        filename = filename_copy
        change_image(filename)
```

4.4 Threads

Main thread : to take care of all the UI widgets and to capture user input

Secondary thread : To take care of the refreshing of the image when edited

4.5 UI Elements

4.5.1 Buttons

Open edited : to open the preview of the edited image.

4.5.2 Check boxes

Black and white : To make the image black and white

Threshold : to turn on the threshold for the image

4.5.3 Sliders

Threshold start : to set the start of the threshold value

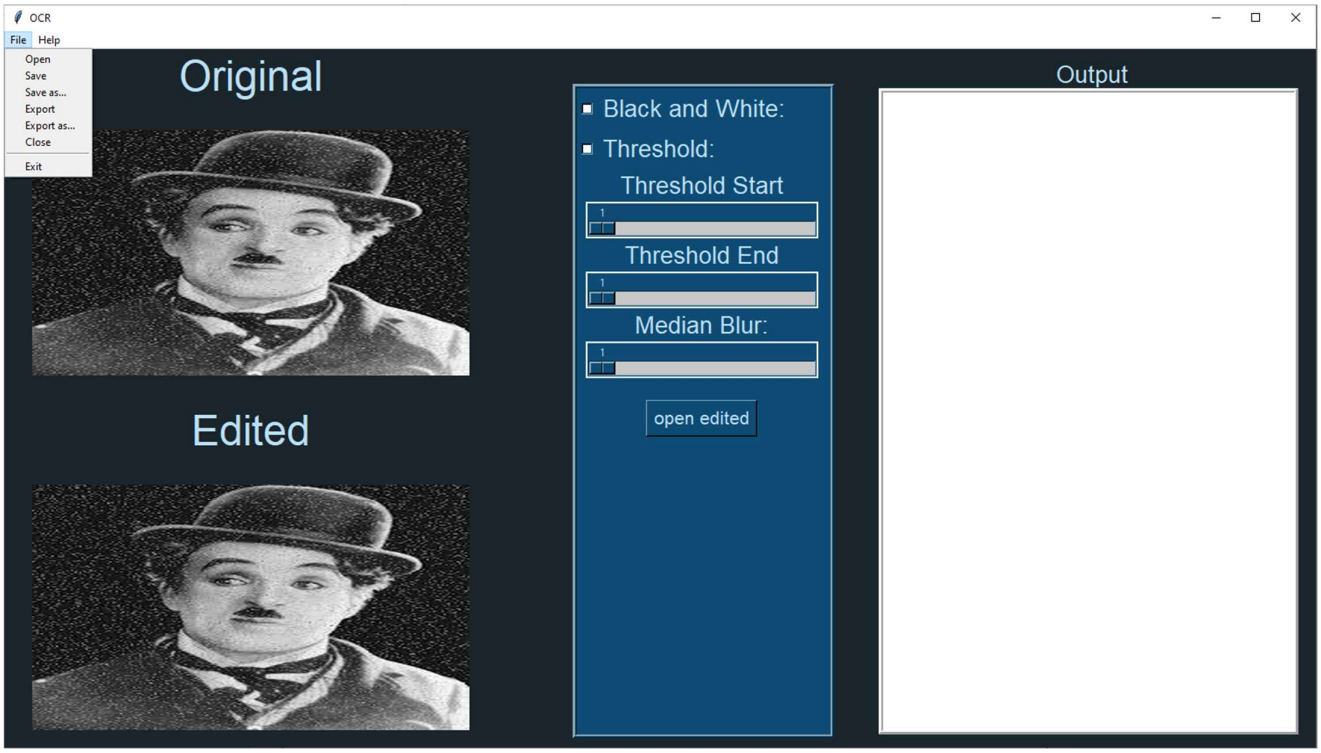
Threshold end : to set the end of threshold value

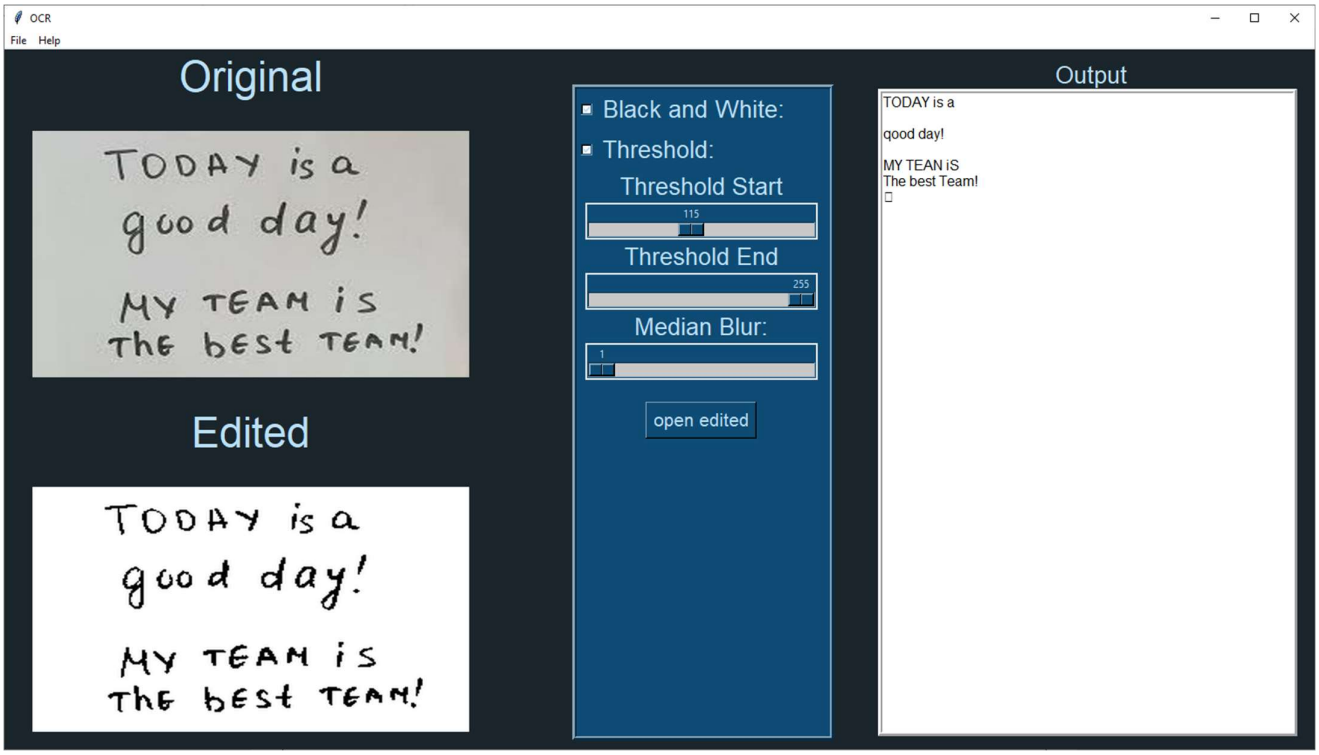
Medan blur : To set the amount of median blur

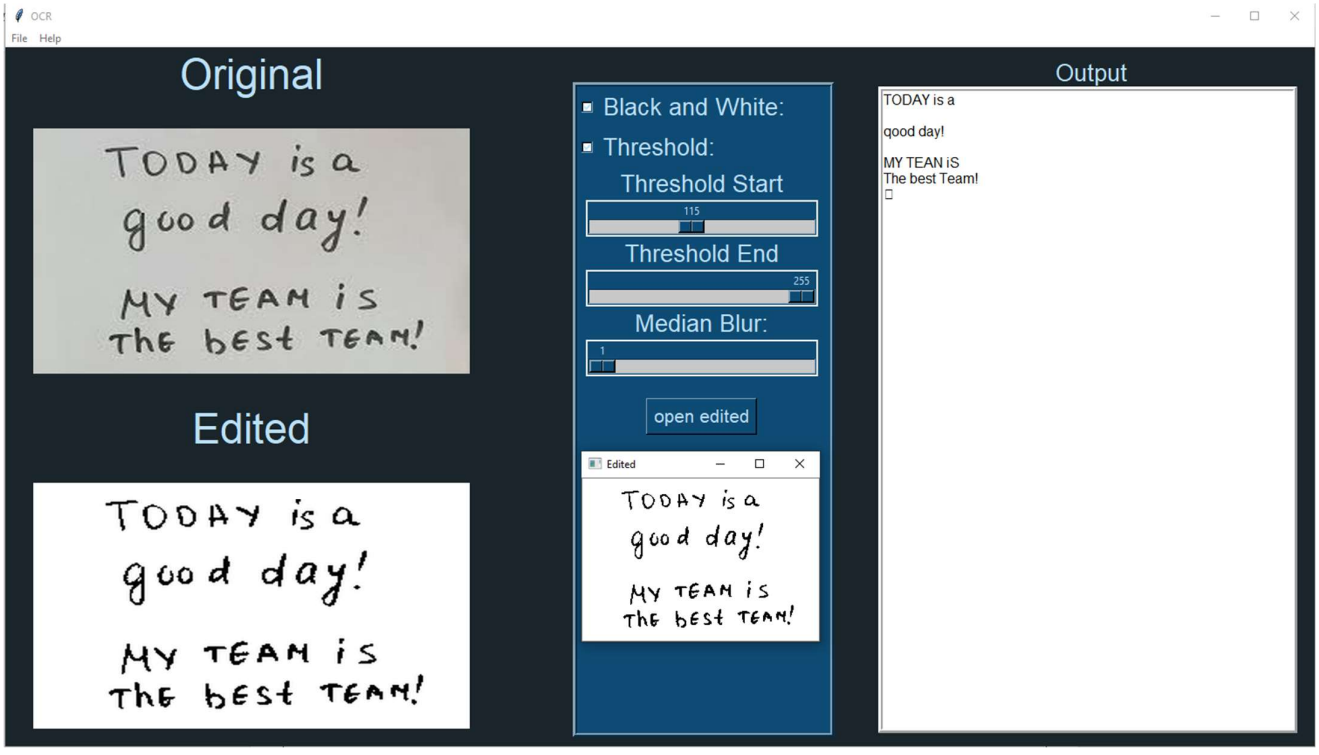
4.5.4 Text box

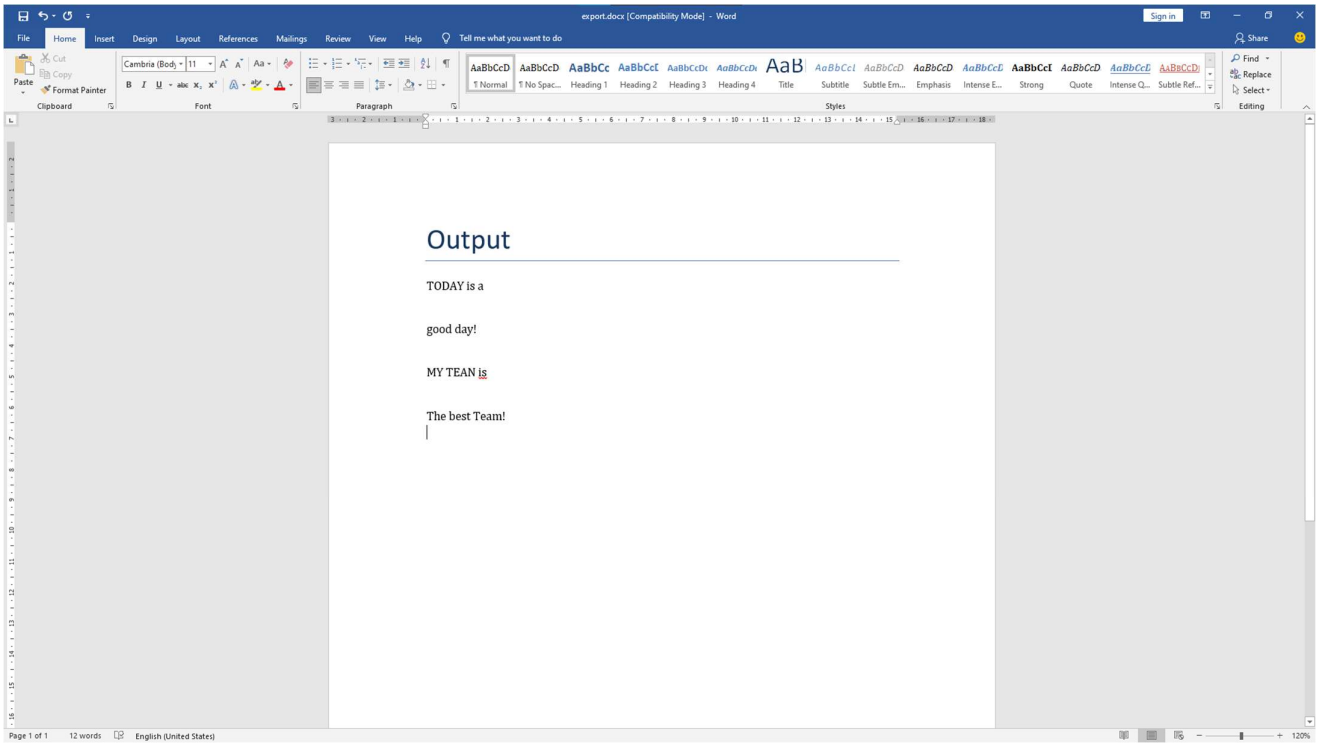
Output : To display the output of the recognized text.

4.6 Execution




OCR
File Help
Original
TODAY is a
good day!
MY TEAM is
the best TEAM!
Edited
TODAY is a
good day!
MY TEAM is
the best TEAM!
Black and White:
Threshold:
Threshold Start
115
Threshold End
255
Median Blur:
1
open edited
Output
TODAY is a
qood day!
MY TEAN iS
The best Team!


OCR
File Help
Original
TODAY is a
good day!
MY TEAM is
the best TEAM!
Edited
TODAY is a
good day!
MY TEAM is
the best TEAM!
Black and White:
Threshold:
Threshold Start
115
Threshold End
255
Median Blur:
1
open edited
Edited
TODAY is a
good day!
MY TEAM is
the best TEAM!
Output
TODAY is a
qood day!
MY TEAN iS
The best Team!



5. References

Tkinter :

Tkinter documentation

Pytesseract :

Pytesseract documentation

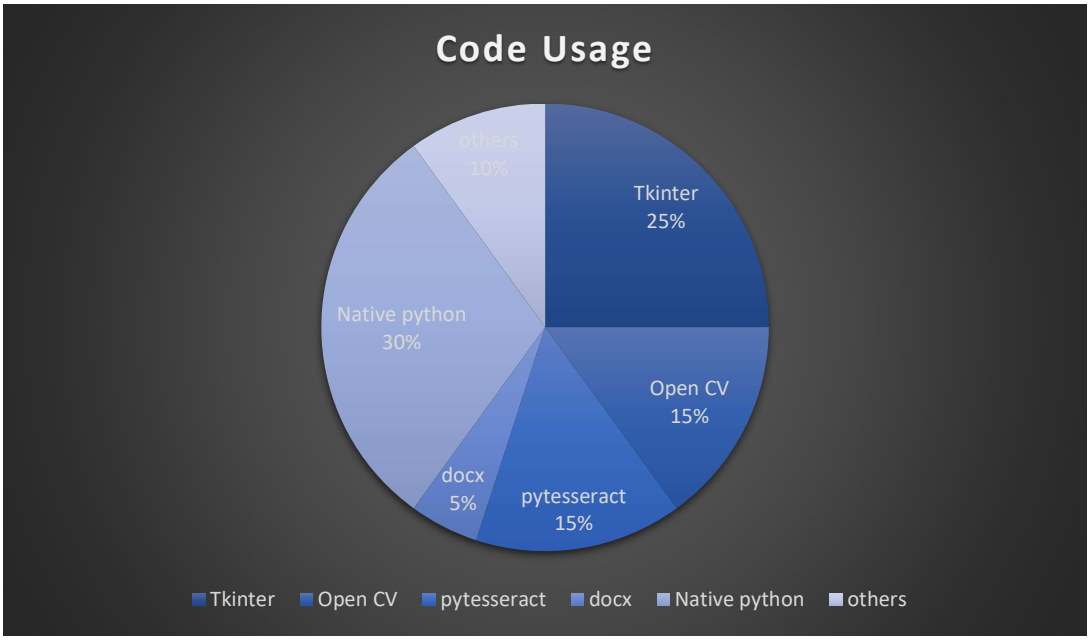
Youtube video for pytesseract

Open CV :

Open CV docs

Youtube playlist on open CV

6. Code Usage



7. Contact details

Name : Mohamed Irfan P

College : Saveetha Engineering College

Phone : 8778942543

Email : mohamedirfanp3@gmail.com

Linkedin : www.linkedin.com/in/mohamed-irfan-mdi