

OPEN CV INTERNSHIP

TOPIC- OCR GUI Application Documentation

This is a simple GUI application, so it can be simply run over terminal/command prompt.

Installation-

1. The following code can be executed using any code editor(VS,Pycharm,etc). To run this application we are required to install libraries such as Open CV,Tkinter,PIL,Tesseract. We can do the above by simply typing “pip install name-python”.
2. But for installation of tesseract(library mainly used for ocr application) ,you have to download the tesseract library and then during the installation process you have to select the languages for which the above library tesseract should work for(if selected nothing then by default only English language is selected).Then finally go to the command prompt and write ‘pip install tesseract’ and tesseract is finally installed and you can start your coding.
3. OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in OpenCV/data/haarcascades/ folder. We need to download it
In this application like for accessing photos in the directory of your computer you have to change the directory name as per your need in the code since different system have different locations where images are stored.
4. For example (‘Users/Desktop/’ can be changed to ‘E:\’)

2. Working of GUI app: -

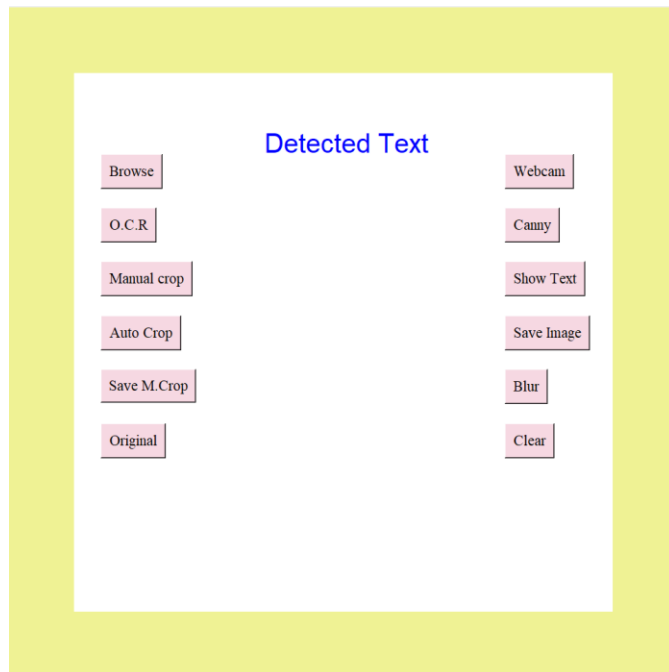
We have created button using Tkinter. which perform certain functions on the images selected. We have also created a textbox to display the recognized information in OCR. We can perform different operations on images using this app.

1. Browse – This button helps you select an image.

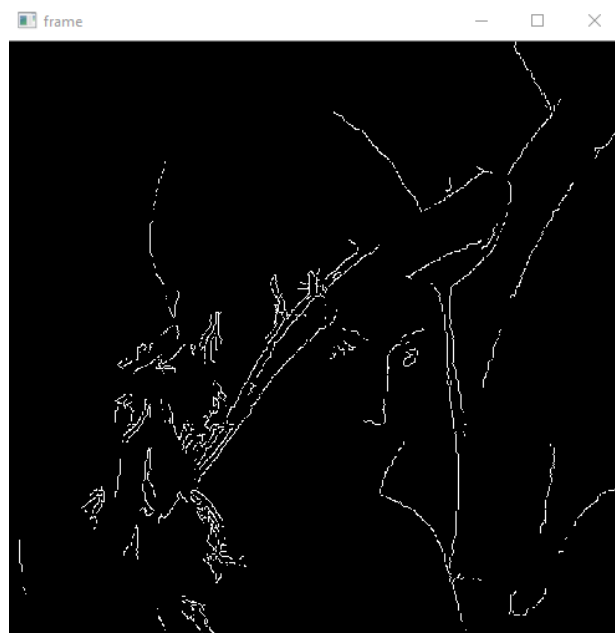
2. Blur image – This button helps you perform Gaussian blur on the image after converting it to gray scale.
 3. Auto Crop Image – This button finds four different edges which are most appropriate according to the large change in the color of the pixels which are adjacent to each other and forms a closed loop around those 4 points and crops the image automatically using contours.
 4. Manual Crop – We can select any 4 different points manually and the image will be cropped using array slicing.
 5. Canny– This button helps to canny an gray scale image which is used for edge detection.
 6. Webcam – This button helps to use the live web cam for face detection and to capture a image.
 7. OCR – This button helps us to find if there are any characters of English in the image by highlighting it in green colored rectangle and displays the text above it in blue color.
 8. Show Text – This button shows the text detected in the text box.
 9. Save Image – This button saves the image displayed after applying various function on it. imwrite function is used for the same
 10. Save M. Crop Image- This button is specially designed to save the manually cropped image.
 11. Original– This button helps you get the original image before any changes.
 12. Clear – This button is used to clear the window.
 13. Rotate- This is used to rotate the image by 90 degree clockwise
 14. Flip- It is used to flip the image vertically.
- If we use -1 -Vertically ; 1-Horizontally

Screenshots of the output: -

1.Ocr App



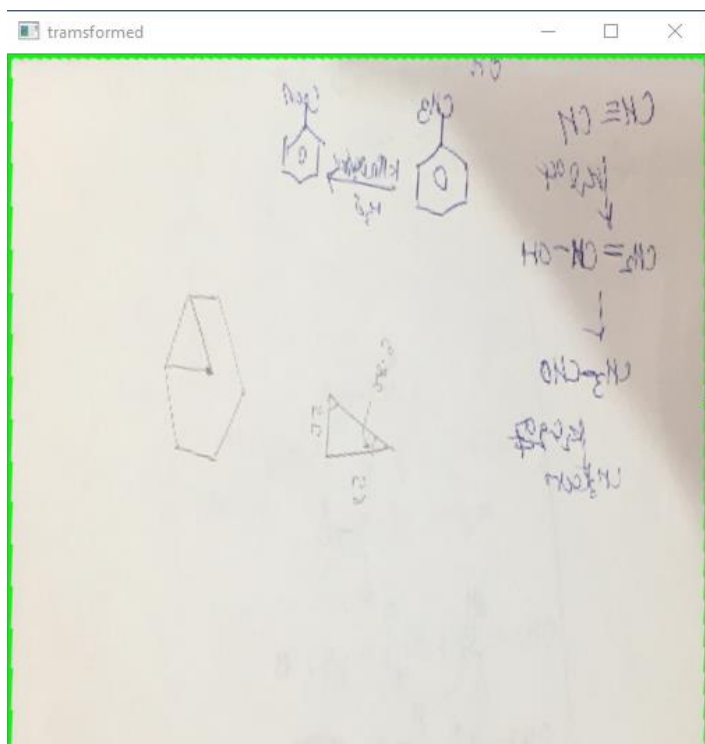
2.Canny



3.Glaussian Blur



4.Auto Crop



5. Manually Cropped Image

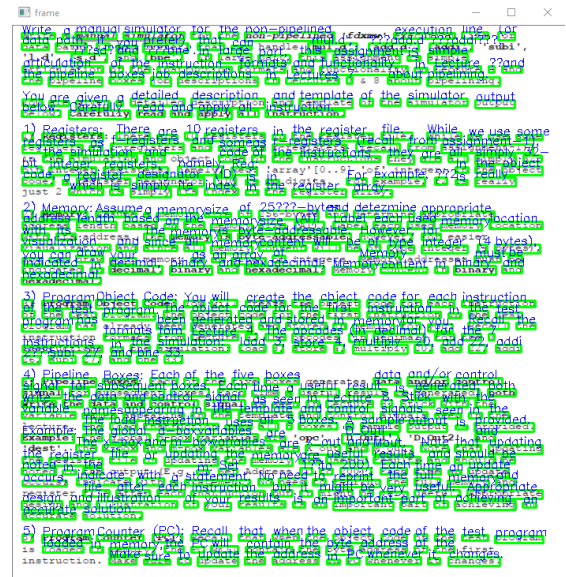
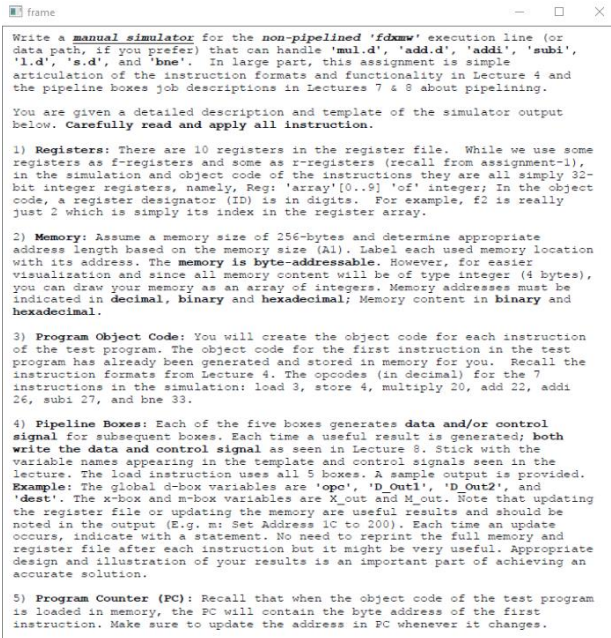


Original Image



Cropped Image

6. Optical Character Recognition



7. Detected Text

Browse

O.C.R

Manual crop

Auto Crop

Save M.Crop

Original

Webcam

Canny

Show Text

Save Image

Blur

Clear

Write a manual simulator for the non-pipelined 'fdamw' execution line (or data path, if you prefer) that can handle 'mul.d', 'add.d', 'addi', 'subi', 'ld', 'sd', and 'bne'. In large part, this assignment is simple articulation of the instruction formats and functionality in Lecture 6 and the pipeline boxes job descriptions in Lectures 7 & 8 about pipelining.

You are given a detailed description and template of the simulator output below. Carefully read and apply all instruction.

1) **Registers:** There are 10 registers in the register file. While we use some registers as f-registers and some as r-registers (recall from assignment-1), in the simulation and object code of the instructions they are all simply 32-bit integer registers, namely, Reg: 'array'[0..9] 'of' integer; In the object code, a register designator (ID) is in digits. For example, f2 is really just 2 which is simply its index in the register array.

2) **Memory:** Assume a memory size of 256-bytes and determine appropriate address length based on the memory size (A1). Label each used memory location with its address. The memory is byte-addressable. However, for easier visualization and since all memory content will be of type integer (4 bytes), you can draw your memory as an array of integers. Memory addresses must be indicated in decimal, binary and hexadecimal; Memory content in binary and hexadecimal.

3) **Program Object Code:** You will create the object code for each instruction of the test program. The object code for the first instruction in the test program has already been generated and stored in memory for you. Recall the instruction formats from Lecture 4. The opcodes (in decimal) for the 7 instructions in the simulation: load 3, store 4, multiply 20, add 22, addi 26, subi 27, and bne 33.

4) **Pipeline Boxes:** Each of the five boxes generates data and/or control signal for subsequent boxes. Each time a useful result is generated, both write the data and control signal as seen in Lecture 8. Stick with the variable names appearing in the template and control signals seen in the lecture. The load instruction uses all 5 boxes. A sample output is provided. **Example:** The global d-box variables are 'd_out', 'D_Out1', 'D_Out2', and 'dest'. The x-box and m-box variables are X_out and M_out. Note that updating the register file or updating the memory are useful results and should be noted in the output (E.g. m: Set Address 10 to 200). Each time an update occurs, indicate with a statement. No need to reprint the full memory and register file after each instruction but it might be very useful. Appropriate design and illustration of your results is an important part of achieving an accurate solution.

5) **Program Counter (PC):** Recall that when the object code of the test program is loaded in memory, the PC will contain the byte address of the first instruction. Make sure to update the address in PC whenever it changes.

8. Vertically Flipped



9. Horizontally rotate 90 degree clockwize

