1) Instruction

- Pre-processing by visual studio 2010

1. Input All png files ( s1, s2…, e1, e2… png) into “ocr” folder.

2. Make the folders (s1,s2,s3…, e1,e2,e3…) in “ocr” folder.

2. Open OCR project (ocr/ocr.sln) by using visual studio 2010.

3. Execute the OCR solution.

4. Move folders (e1,e2, e3,… , s1, s2, …) in “ocr” folder to /pytesser/input in “pytesser.zip”.

- Recognition Processing

1. Input "input.txt" files into /pytesser/input folder.

2. Execute pytesser.py, then you can get output.txt in /pytesser/output

2) Method

When I tested it firstly, the recognition rate is too low. I can’t modify Pytesser or Tesseract core sourcse, because I doubt that works. Instead, I decided to do pre-processing before executing Pytesser.

Pre-processing is made by Opencv of C++. Firstly, I find each box including a letter by template matching, save each letter separately. In this process, I applied noise reduction by using fastNlMeansDenoising function (I tried making to binary image, but it didn’t work).

In recognition process, each letter is recognized by Pytesser. In this step, Python matched the id or number from “input.txt” to the recognized sentence. If id (or numbers) from “input.txt” has over 2 letters which are same letter and location that recognized sentence has, it is considered as it is matched. If not, it is considered as it is not matched.

3) Result and analysis

Among 6 samples, this program can recognize the 5 samples. But, this program has very strong assumption that is if only 2 letters are matched (letter itself and location of the letter), it is considered as matched. To overcome this limitation, we should improve recognition rate basically.