1, general architecture

1.1, computing features:

I choose HOG as the feature. I compute the HOG feature by cv2.HOGDescriptor.compute().

1.2, performing detection:

I implement OCL algorithm for the connected component labeling. The algorithm is from the paper “Optimizing Two-Pass Connected-Component Labeling Algorithms” by Kesheng Wu.
I have also referred to the paper “The connected-component labeling problem: A review of state-of-the-art algorithms” and webpage
“https://aishack.in/tutorials/connected-component-labelling/ ”.

OCL is a two-pass algorithm. It marks labels in the first pass and combines the connected labels in the second pass. It is based on decision tree and union-find data structure.

About the union-find data structure, I implement is with dictionary.

1.3, performing recognition:

I use L2 distance for recognition.
For every character block c waiting for recognition, for every template character t, I compute score_c=L2(c,t).
If min(score_c) >3, I think no template character t can match c, so c is “UNKNOWN”.
Otherwise, I choose the template character t with the minimum score.

2, Output

features folder is “./feature”
“results.json”

3, Code structure:

extractor: features extractor for enrollment step.
ccl: connected component labeling for detection step.
tools: several tools like encoder.py for json encoding, data_print.py for image print function and several config variables.
data: all data

In order to match the requirement, I also add other directories.
feature: feature for enrollment step
character: input character templates