对每个切分目标进行去噪处理，并将其做二值图像的转化，切分以垂直投影法为基础，这能将在垂直空间上没有笔画重叠的目标分开，由于汉字的左右部首偏旁特性，设置最小切分阈值，避免将偏旁部首作为单独的切分结果，同时设置最长阈值，认为超过此阈值长度的目标则为未能成功切分的目标，对于这些不正常的切分结果，采取个性化长度切分。性别填写为单字，所以以每份简历的性别长度为基础，对每份简历的汉字填写部分进行切分，若出现长度小于阈值的切分结果，则将此结果与上一个切分结果粘合起来继续做个性化切分。由于数字和字母填写部分的书写长度大概一致，所以对数字和字母的切分目标分别采用固定的切分长度进行切分。

代码：

import os

import cv2

import numpy as np

dst\_dir = "./result/"

top\_dir='cut\_image/'

min\_val =1000

write\_width={}

def extract\_peek(array\_vals, minimun\_val, minimun\_range):

start\_i = None

end\_i = None

peek\_ranges = []

for i, val in enumerate(array\_vals):

if val > minimun\_val and start\_i is None:

start\_i = i

elif val > minimun\_val and start\_i is not None:

pass

elif val < minimun\_val and start\_i is not None:

if i - start\_i < 4:

start\_i = None

end\_i = None

elif i - start\_i >= minimun\_range:

end\_i = i

peek\_ranges.append((start\_i, end\_i))

start\_i = None

end\_i = None

elif val < minimun\_val and start\_i is None:

pass

else:

raise ValueError("cannot parse this case...")

return peek\_ranges

def cutImage(img, peek\_range):

count=0

os.makedirs('result/' + single\_dir + '/' + fileName + '/')

for i, peek\_range in enumerate(peek\_ranges):

for vertical\_range in vertical\_peek\_ranges2d[i]:

x = vertical\_range[0]

label=x

y = peek\_range[0]

img1 = img[y:100, x:vertical\_range[1]]

width=img1.shape[1]

if single\_dir == 'nation':

if width>=60:

num=round(width/33)

temp=x

for times in range(0,num+1):

if temp>=(vertical\_range[1]-3):

break

new\_img=img[y:100,temp:(temp+33)]

temp=temp+33

new\_width=new\_img.shape[1]

if new\_width<=15:

continue

gray = cv2.cvtColor(new\_img, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 35000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", new\_img)

else:

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum<35000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", img1)

elif single\_dir=='sex':

if count>=2:

continue

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if 22<=width<40:

if width>=34:

width=32

write\_width[fileName]=width

elif width<22:

write\_width[fileName]=24

else:

write\_width[fileName]=int(width/2)

if img\_sum<35000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", img1)

elif single\_dir=='blood' :

if width >= 25:

num = round(width /17)

temp = x

for times in range(0, num + 1):

if temp >= (vertical\_range[1] - 3):

break

new\_img = img[y:100, temp:(temp + 17)]

temp = temp + 17

gray = cv2.cvtColor(new\_img, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 30000:

pass

else:

count += 1

if count>=3:

break

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", new\_img)

else:

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 30000:

pass

else:

count += 1

if count >= 3:

break

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", img1)

elif 'time' in single\_dir:

if width >= 20:

num = round(width /11)

temp = x

for times in range(0, num + 1):

if temp >= (vertical\_range[1] - 2):

break

new\_img = img[y:100,temp:((temp + 11) if (temp + 11) < vertical\_range[1] else vertical\_range[1])]

if (vertical\_range[1] - temp) < 3:

continue

temp = temp + 11

gray = cv2.cvtColor(new\_img, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 10000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", new\_img)

else:

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 10000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", img1)

elif single\_dir=='weight':

if width >= 25:

num = round(width /17)

temp = x

print(temp)

for times in range(0, num + 1):

if temp >= (vertical\_range[1] - 2):

break

new\_img = img[y:100,temp:(temp + 17)]

if (vertical\_range[1] - temp) < 3:

continue

temp = temp + 17

gray = cv2.cvtColor(new\_img, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 10000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", new\_img)

else:

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 10000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", img1)

else:

if width >=34:

single\_write\_width=26

num = round(width /26)

temp = x

for times in range(0, num + 1):

if temp >= (vertical\_range[1] - 2):

break

new\_img = img[y:100, temp:((temp + single\_write\_width) if (temp+single\_write\_width)<vertical\_range[1] else vertical\_range[1])]

if new\_img.shape[1]<16:

os.remove('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png")

count=count-1

new\_img=img[y:100,label:((temp + single\_write\_width) if (temp+single\_write\_width)<vertical\_range[1] else vertical\_range[1])]

if new\_img.shape[1]<38:

pass

else:

n=int(label+new\_img.shape[1]/2)

count=count+1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png",img[y:100,label:n])

count=count+1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png",img[y:100,n:((temp + single\_write\_width) if (temp+single\_write\_width)<vertical\_range[1] else vertical\_range[1])])

label = temp

temp = (

(temp + single\_write\_width) if (temp + single\_write\_width) < vertical\_range[1] else

vertical\_range[1])

continue

gray = cv2.cvtColor(new\_img, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 20000:

pass

else:

count += 1

label = temp

temp = (

(temp + single\_write\_width) if (temp + single\_write\_width) < vertical\_range[1] else

vertical\_range[1])

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", new\_img)

else:

gray = cv2.cvtColor(img1, cv2.COLOR\_BGR2GRAY)

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

img\_sum = adaptive\_threshold.sum()

if img\_sum < 20000:

pass

else:

count += 1

cv2.imwrite('result/' + single\_dir + '/' + fileName + '/' + str(count) + ".png", img1)

change = 0

dir\_list = os.listdir(top\_dir)

for location in range(0, len(dir\_list)):

if (dir\_list[location] == 'sex'):

change = location

break

h = dir\_list[0]

dir\_list[0] = dir\_list[change]

dir\_list[change] = h

for single\_dir in dir\_list:

base\_dir=top\_dir+single\_dir

for fileName in os.listdir(base\_dir):

img\_temp = cv2.imread(base\_dir +'/'+ fileName)

img = cv2.fastNlMeansDenoisingColored(img\_temp, None, 10, 10, 7, 21)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

row=gray.shape[0]

column=gray.shape[1]

for border\_row in range(0,column):

gray[0][border\_row]=255

gray[row-1][border\_row]=255

for border\_column in range(0,row):

gray[border\_column][0]=255

gray[border\_column][column-1]=255

gray[border\_column][1] = 255

gray[border\_column][2] = 255

gray[border\_column][3] = 255

gray[border\_column][4] = 255

gray[border\_column][5] = 255

gray[border\_column][6] = 255

gray[border\_column][7] = 255

gray[border\_column][column-3] = 255

gray[border\_column][column-2] = 255

adaptive\_threshold = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, \

cv2.THRESH\_BINARY\_INV, 11, 2)

horizontal\_sum = np.sum(adaptive\_threshold, axis=1)

if single\_dir=='blood' :

min\_range=12

elif single\_dir=='sex':

min\_range=20

elif 'time' in single\_dir:

min\_range=6

elif single\_dir=='weight':

min\_range=6

else:

min\_range=20

peek\_ranges = extract\_peek(horizontal\_sum, min\_val, min\_range)

line\_seg\_adaptive\_threshold = np.copy(adaptive\_threshold)

for i, peek\_range in enumerate(peek\_ranges):

x = 0

y = peek\_range[0]

w = line\_seg\_adaptive\_threshold.shape[1]

h = peek\_range[1] - y

pt1 = (x, y)

pt2 = (x + w, y + h)

cv2.rectangle(line\_seg\_adaptive\_threshold, pt1, pt2, 255)

vertical\_peek\_ranges2d = []

peek\_ranges=[(0,100)]

for peek\_range in peek\_ranges:

start\_y = peek\_range[0]

end\_y = peek\_range[1]

line\_img = adaptive\_threshold[start\_y:end\_y, :]

vertical\_sum = np.sum(line\_img, axis=0)

vertical\_peek\_ranges = extract\_peek(

vertical\_sum, min\_val, min\_range)

vertical\_peek\_ranges2d.append(vertical\_peek\_ranges)

if vertical\_peek\_ranges2d==[[]]:

write\_width[fileName]=26

cutImage(img, 100)