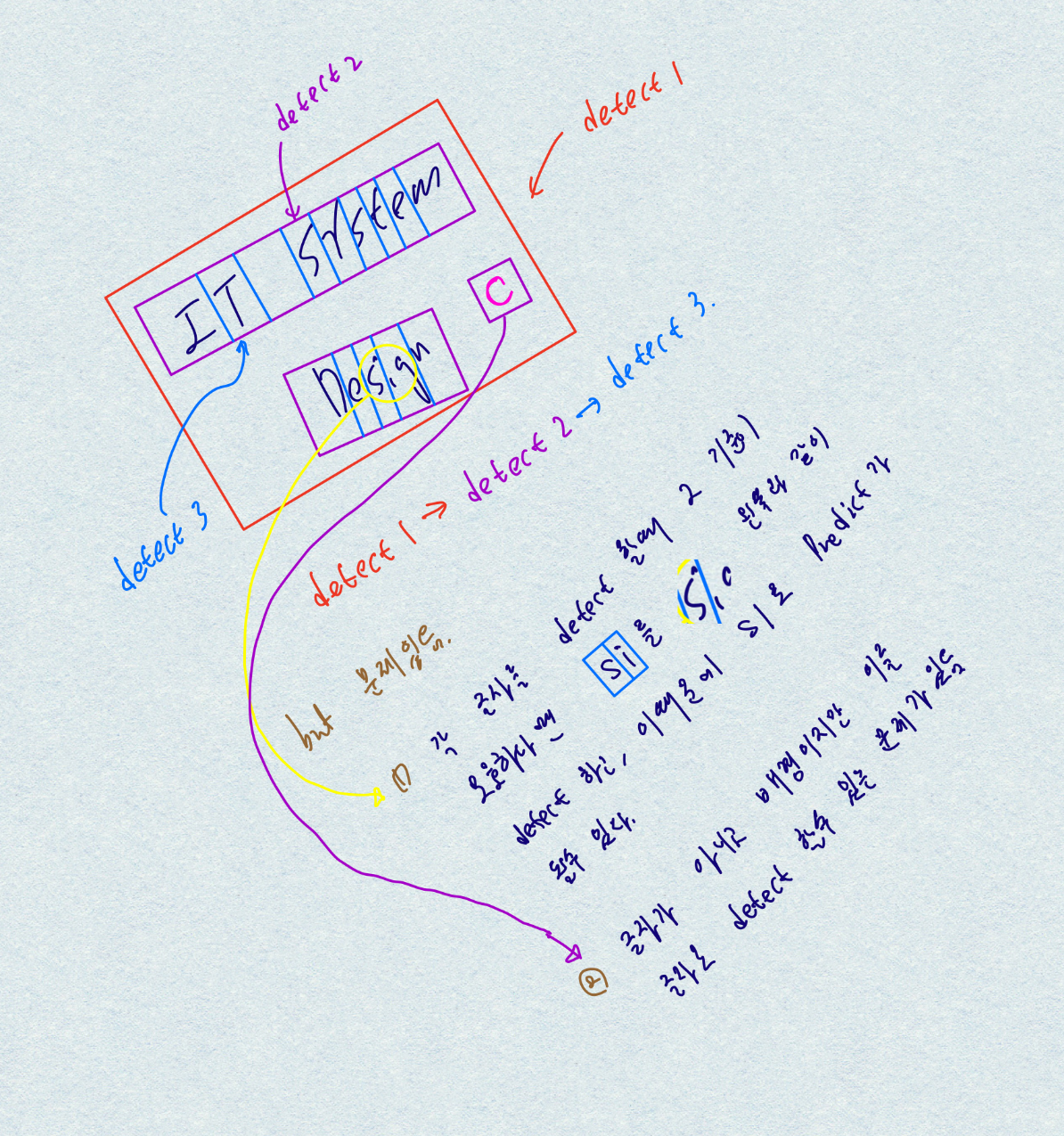
**HW #4 [글자인식]**

**B715141 이익범**

지금까지 배웠던 mnist training/test 는 각 28 by 28 의 박스안에 글자가 있고 이를 training 혹은 test 하는 방식이었다. 그러나 이번 과제는 그렇지 않다. 무작위의 사진이 있고 그 안에 글자가 있기때문에 기존의 방식대로 하면 글자 하나하나를 인식하는것이 아니고 글자가 몇개가 들어가있을수도 있고 없을수도 있고 엉뚱한 물체를 다른글자로 판별할것이다. 따라서, 글자가 있는 문장의 경계선을 detect 한다음(경계선은 아무래도 직사각형이 좋고 편할것같다.) 문장의 각 글자들의 경계를 또 나눈다음에 글자를 인식하는것이 좋을것이라고 판단된다. 다만 그 글자들의 경계로 나뉜 사각형의 크기는 모두 다를것이므로, 각 사각형은 나중에 scaling 을 통해 28 by 28 크기의 사각형을 각각 x\_test 행렬에 순서대로 저장한다음 순서대로 test 하면 될것이라고 구상을 하였다.

아래의 노트에 구상한것을 표현해보았다. ( 궁금증도 같이 필기가 되어있음)



이번 과제는 따라서 input data 에서 어떻게 알고리즘을 구현하여 글자만을 detect 할것이냐가 중요한 포인트가 되겠다. 저번 과제에서 각 회전된 글자 학습시키는 부분은 했으므로, 이번에는 input data를 적절히 뽑아내는것이 관건이 되겠고 이를 성공한다면 test 는 잘 이루어질 것이라고 생각한다.

저번 과제에서 tensorflow 와 numpy 만을 이용하여 data augmentation 을 하였는데, 나의것은 정확도는 높은대신 학습시간이 1epoch 당 40분이나 걸렸다. 반면 과제제출하신분들의 코드를 참조하니 한분빼고는 다 keras 로 하셨는데 이를 돌려본결과 keras 가 data augmentation 하기에도 간편하고, network 구조를 수정하는데 있어서(물론 세부적인 구조수정 제외) 편하기때문에 이번에는 pair\_review code 를 참고하여 keras 로 할것이다.

“””

모든 코드는 opencv 홈페이지와 github를 참고하였다.

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html#gaussian-filtering>

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#adaptive-thresholding>

<https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html#dilation>

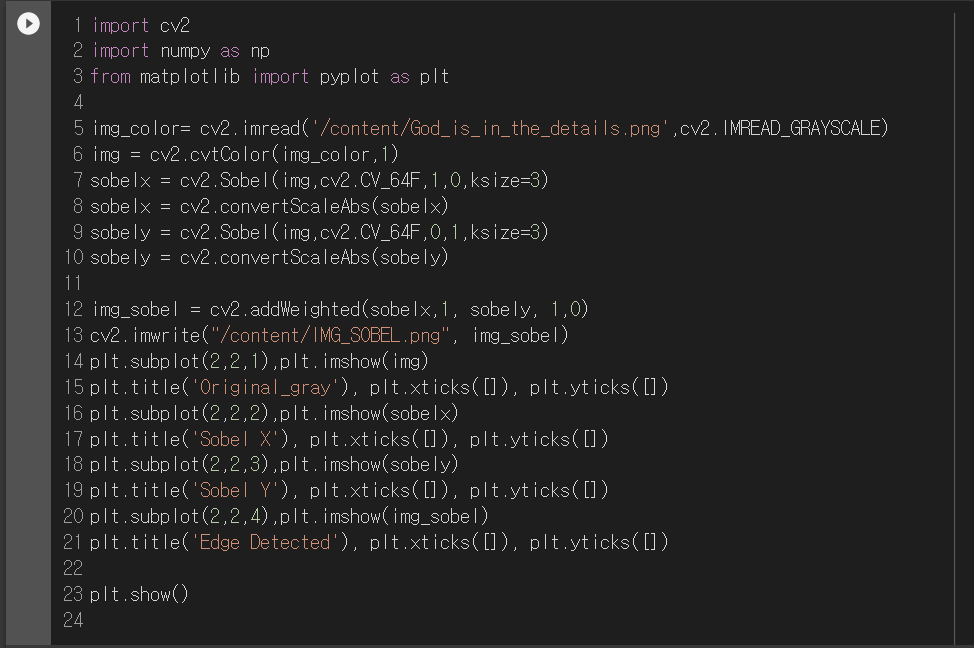
<https://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html#rectangle>

…등등 너무많아서 요기까지만 첨부자료 표현

“””

1. **사진 OpenCV 함수로 Binary로 변환 하기. + Box 만들기**

글자를 detect 하려면 가장자리를 강조하여 검출을 해야하기때문에, OpenCV 함수중에 sobel(1차 미분 기반)을 사용하여 image 의 글자부분을 강조시킬 필요가 있다. X 축 sobel 과 Y 축 sobel 을 각각 edge detect 한후 합하였다.



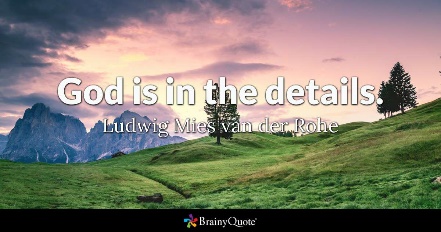
위와같이 grayscale 로 읽어드린다음 x 축, y축 각각 edge detection 을 한다음, x 축 y 축을 합친 img\_sobel 을 만들었다.

1. God is in the details 첫번째 버전



*[Original Image] [Original\_gray/ SobelX/ sobelY/ EdgeDetected]*

1. God is in the details 두번째 버전



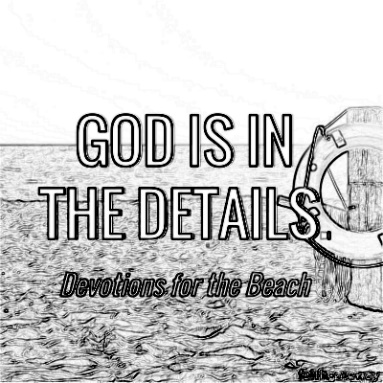
*[Original Image] [Original\_gray/ SobelX/ sobelY/ EdgeDetected]*

③ God is in the details 세번째 버전

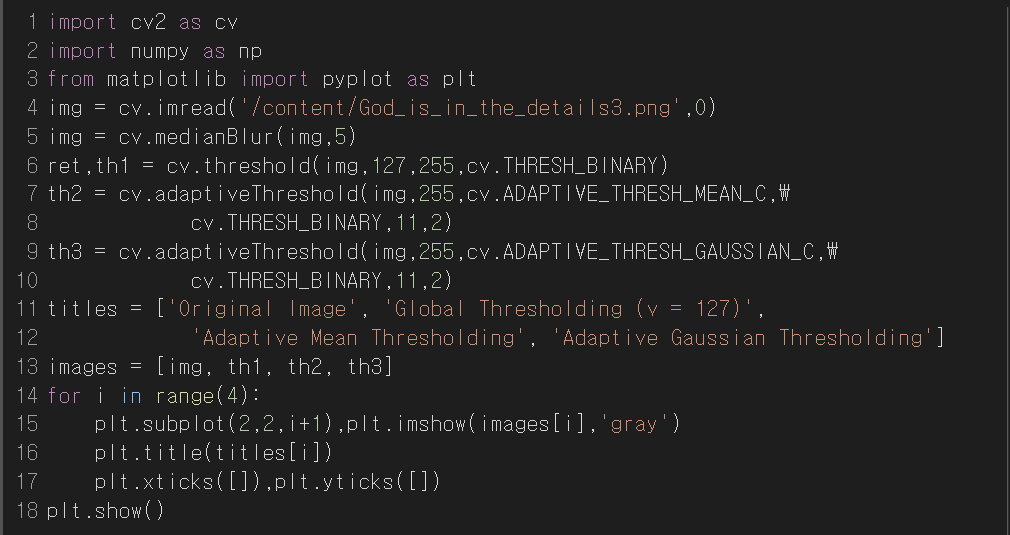


*[Original Image] [Original\_gray/ SobelX/ sobelY/ EdgeDetected]*

세가지의 DATA로 OpenCV 를 이용하여 gray color 로 변환한다음 edge detect 를 한 모습이다. Sobel 함수 특성상 경계선의 변화가 크므로 sobel 함수를 적용시켜 글씨를 edge detect 를 시키면 경계가 모두 하얀색으로 테두리가 생기는것을 볼 수 있을것이다. 따라서 색상 반전 코드 img\_sobel = 255 - img\_sobel를 통해 글씨부분 을 반전시켜보겠다.

 왼쪽과 같이 글씨의 윤곽이 검정색으로 나타나는것을 볼 수 있다. 따라서 만약 box 가 detection 된다면 각각의 글자를 나누는것은 더 쉬울것으로 보인다. 또한 모든 글자의 테두리에 색상반전을 입혀줄경우 위와같이 테두리가 검정색으로 변할것이다. 따라서 위의 방식으로 test 를 진행하기 위해서는, word dataset 도 edge detection 을 시킨 다음 색상반전을 시키고서 image 를 새로 저장한 후에 training 을 시킬 필요가 있겠다. ->위와같은 방식으로 똑같이 적용하면 되므로 word data augentation 하는것은 마지막에 진행.

근데 위와같이 했더니 아무래도 기존의 word train data 와 테두리만 같지 그 안쪽의 색상이 binary 하게 반대이기 때문에 문제가 있다고 생각해서 내방법은 틀렸을것이 확실하기 때문에 조사를 더 해봤더니, 나와는 다르게 Gaussian blur, Adaptive Threshold 라는 기법을 더 적용을 해주었다. 따라서 아래와 같이 적용해보았다.



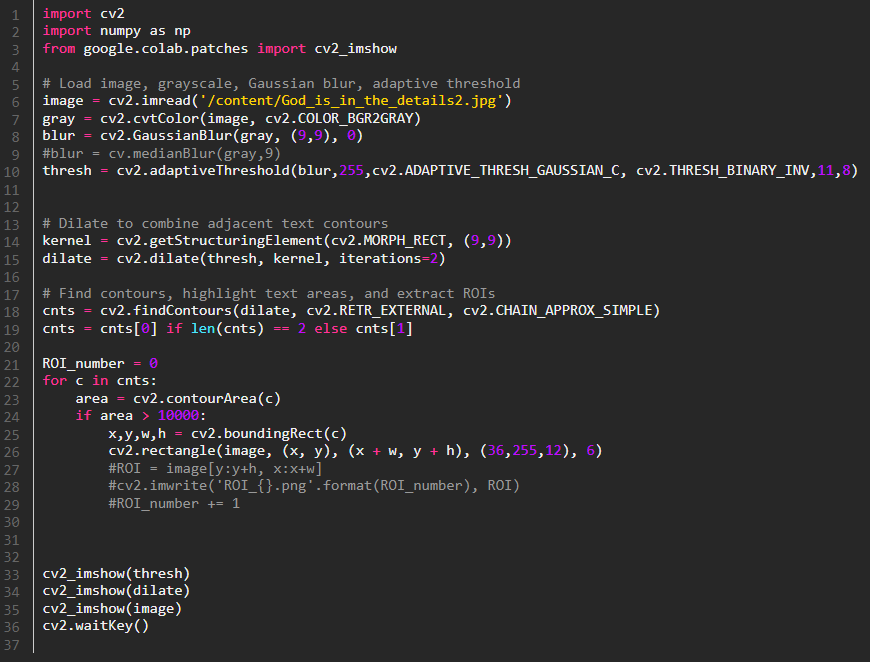
위와 같은 코드를 실행시켰더니 아래와 같은 결과가 나왔다.

Color scriptor 가 자꾸 오류가 나서 screenshot 으로 올리고있다. 코드는 github 와 txt 파일로 따로 꼭 첨부해두겠다.



*[Original Image] [Original\_gray/ Global / Adaptive Mean/ Adaptive Gaussian Thresholding]*

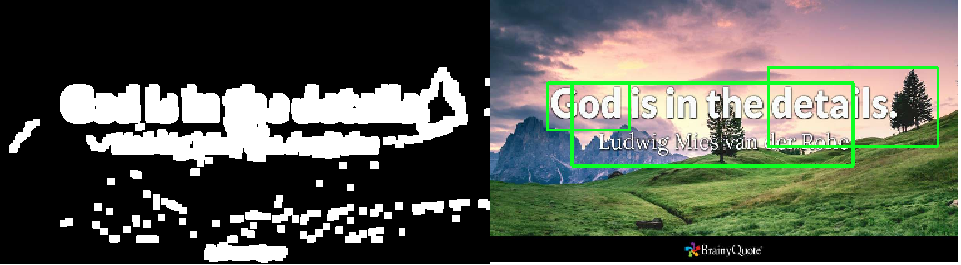
결과를 보았을때 Global Thresholding 이 좋아보이지만, 이미지가 바뀌어도 글자를 잘 detect 해야하므로 adaptive Gaussian Threholding 을 써야한다는것을 깨달았다. 이제 deliation 을 통해 윤곽을 만들고 그다음 사각형 box 를 씌우도록 하겠다. 코드는 다음과 같다.



위와같이 코드를 짜고, threshold 와 deliation 등 argument 를 조절해가면서 실험을 해봤다. 결과는 역시나 배경에 나무나 무늬가 있으면 이것에 박스를 쳐버리는 사태가 발생하였고 잔디와같이 deliation 을하면 퍼지면서 만나기때문에 이것을 사각형 box 로 선택해버리는 일이 생겨버리고야 말았다. 다음 실험에서 나오지만 배경이 부드러운 모래나 이런것은 괜찮다. 하지만 아래와같은 나무같은 것은 걸림돌이 된다는것을 알 수 있다. 지금으로서는 첫번쨰 케이스가 GOD 가 중첩되지만 제일 나은것으로 보인다.



Thresh=(11, 6) , Deliation = 1 box image



Thresh=(11, 13) , Deliation = 2 box image



Thresh=(11, 8) , Deliation = 2 box image



Thresh=(11, 6) , Deliation = 2 box image



Thresh=(15, 1) , Deliation = 2 box image

처음에 목표하였던것은 각 단어를 인식하는것이었는데, 단어단위는 안되고 같은크기의 문장단위로 인식이 되는것을 볼 수 있다. 배경이 깨끗한 이미지도 한번 실험해 보도록 하겠다.



Thresh=(11, 6) , Deliation = 2 box image

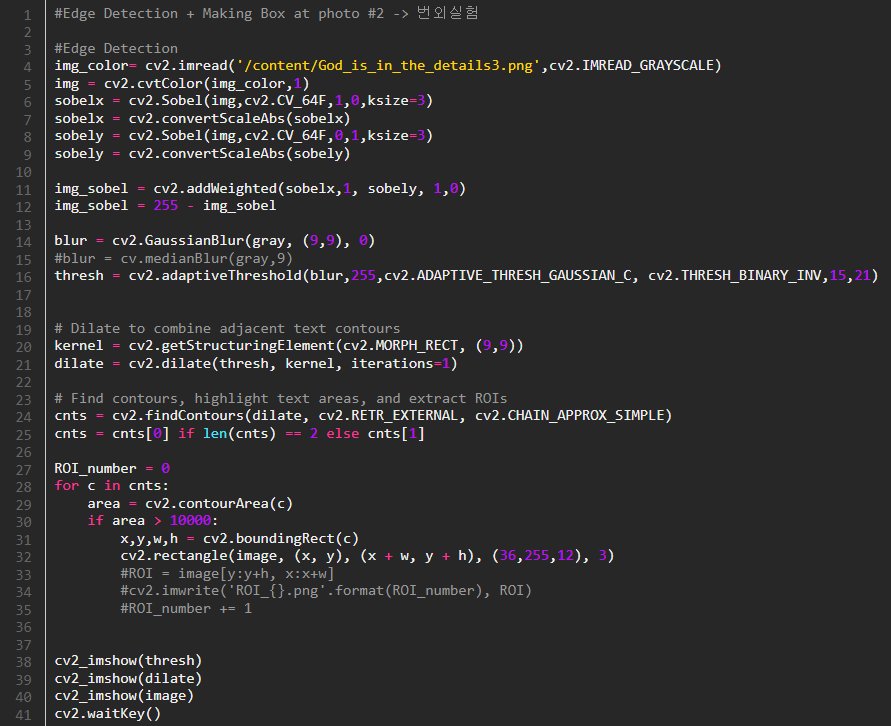
위와 같이 배경에 무늬가 있어도 잘 인식되는것을 확인할 수 있다. 다만 이제 고민거리가 생겼다. 위와같이 적절한 text 를 인식하려면 모든 물체마다 각각 tresh 값, deliation 값 등등 argument를 정해줘야하는데 그것이 가능할까? 그것에 대한 문제점도 같이 해결해야 한다. 즉 이제 해결해야 할 문제가 2가지가 생겼다.

▶ 각 사진들마다 모두 밝기, 배경에 글자이외의 장애물, 글자의 크기 등등이 모두 다르기 때문에 위처럼 코드를 짜면 일일이 argument 를 바꿔주어야 한다는 문제점

▶ 뒤에 배경에 무늬 혹은 장애물이 있으면(대표적으로 나무가있는 사진) 이것이 deliation 되면서 box 가 같이 쳐질 가능성 있음.

--------------------------------------------------------------------------------------------------------------

**[번외실험]**

아이디어를 하나더 생각해 보았다. 글자가 deliation 되면서 크기가 다른 문장들이 서로 붙어 한박스로 인식될수 있는 문제가 있고, 문자가 보통 배경의 물체보다 강조되고 두꺼우므로, 아까 처음에 실험하였던 edge detection(sobel) 을 box making code 앞부분에 대신 넣고 deliation 을 하면 어떨까 하는 생각이다. 따라서 한번앞부분에 추가를 해보았다.

위와같이 Thresh의 마지막 argument 2개의 값과 Deliation의 iteration 을 최선으로 조절하였을때, 최선의 결과는 다음과 같이 나오게 되었다.





같은 단어의 중첩이 너무많이 생기는 문제가 발생하였으므로 기존의 방법을 쓰도록 하겠다. 그렇지만 이 방법은 글자의 특성별로 더 세부적으로 나누어져 나중에 글자를 recognition 할때 더 유용할 것같기도 하므로 이방법이 아주 쓸모없지는 않아보인다. 또한 두번쨰 사진의 단어 인식은 아주 어려운 난이도같다. 나무와 배경이 아주 진하여 뒤의 배경을 없애지 않는 이상 box 가 쳐지게 될것이고(두번째사진에, 나무때문에 중간에 두개의 문장이 합쳐져서 box가 같이 쳐지 는 것을 볼 수 있다.) 이때문에 나중에 박스를 세분화해서 나눈다음 글자인식을 시키면 나무가 단어로 인식되고 풀이 나무로 인식되는등 문제가 발생하는 어려운 문제이다. 하지만 여전히 이방법은 다른 면에서 유용할것같다. -> 기존의 박스로 큰 박스를 만든다음에 그 각각의 박스를 divide 할때 이방법을 써보면 각 글자가 divide가 잘될것같은 생각이 든다.

-----------------------------------------------------------------------------------------------------------

일단 아까 위에서 했던 코드를 손봐서 좀더 대중적인 코드를 중간중간 적용하여 보았다. 아까의 경우는 뒤에 물체가 있기때문에 dilation 하는방식을 써볼까 했었지만

배경에 무엇이 있는것은 너무 어려워 이제 백지상태로 해볼것이기 때문에 바탕의 noise 를 제거하기 위하여 dilation 대신에cv2.morphologyEx(bin\_img, cv2.MORPH\_CLOSE, kernel) 를 써주었다.

|  |  |  |
| --- | --- | --- |
| src\_img= cv2.imread('/content/img\_3.jpg', 1) # Image Read, Color Read  copy = src\_img.copy() # Shallow Copy  height = src\_img.shape[0] # 행의 개수(length)  width = src\_img.shape[1] # 열의 개수(length)    print("\n Resizing Image........")  src\_img = cv2.resize(copy, dsize =(1320, int(1320\*height/width)), interpolation = cv2.INTER\_AREA) # resizing    height = src\_img.shape[0] # Resized 행의 개수(length)  width = src\_img.shape[1]  # Resized 열의 개수(length)    print("#---------Resized Image Info:--------#")  print("\tHeight =",height,"\n\tWidth =",width)  print("#----------------------------#")    grey\_img = cv2.cvtColor(src\_img, cv2.COLOR\_BGR2GRAY) # BGR to Gray    print("Applying Adaptive Threshold with kernel :- 21 X 21")  bin\_img = cv2.adaptiveThreshold(grey\_img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,cv2.THRESH\_BINARY\_INV,21,20) # Adaptive Threshold 적용  coords = np.column\_stack(np.where(bin\_img > 0)) # bin\_img > 0 인값만 반환하여 column 방향으로 stack 하여 2차원 array 형성  angle = cv2.minAreaRect(coords)[-1] # 물체의 회전을 고려한 rectangular 사각형  if angle < -45:      angle = -(90 + angle)  else:      angle = -angle  h = bin\_img.shape[0]  w = bin\_img.shape[1]  center = (w//2,h//2) # Centor 정중앙  angle = 0  M = cv2.getRotationMatrix2D(center,angle,1.0)  bin\_img = cv2.warpAffine(bin\_img,M,(w,h),                           flags=cv2.INTER\_CUBIC, borderMode=cv2.BORDER\_REPLICATE) # bin\_img : 변경될 이미지 M: 변경 matrix, 출력될 이미지 (w,h)    bin\_img1 = bin\_img.copy()  bin\_img2 = bin\_img.copy() # shallow copy    kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(3,3)) # 타원형 kernel 생성  kernel1 = np.array([[1,0,1],[0,1,0],[1,0,1]], dtype = np.uint8)  # final\_thr = cv2.morphologyEx(bin\_img, cv2.MORPH\_OPEN, kernel)  # final\_thr = cv2.dilate(bin\_img,kernel1,iterations = 1)  print("Noise Removal From Image.........")  final\_thr = cv2.morphologyEx(bin\_img, cv2.MORPH\_CLOSE, kernel) # noise 제거  contr\_retrival = final\_thr.copy() # shallow copy  [*Colored by Color Scripter*](http://colorscripter.com/info#e) |  | [cs](http://colorscripter.com/info#e) |

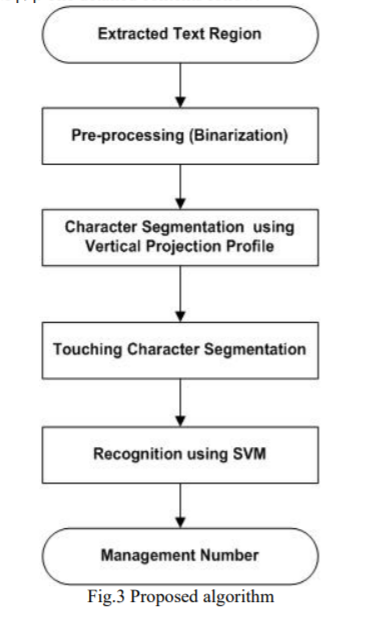
-----------------------------------------------------------------------------------------------------------

1. **각 인식된 박스의 구획을 나누기. ( character Segmentation )**

[**Character segmentation and recognition algorithm of text region in steel images**](https://www.researchgate.net/publication/228896273_Character_segmentation_and_recognition_algorithm_of_text_region_in_steel_images) 논문을 보면

Character segmentation 을 위해 아래와 같은 algorithm 으로 구역을 나누게 된다.

나는 Text Region 까지 추출했으므로 그 다음인 Binarization, Character Segmentation, Touching Character Segmentation 순으로 진행을 하면 될것 같다.



low level 로 짜여진 코드를 참고하여 수정을 하였다. 최종 코드는 다음과 같다. 많은 부분 인터넷을 참고하여 수정하였다. Image Processing + Character Segmentation code 는 다음과 같다.

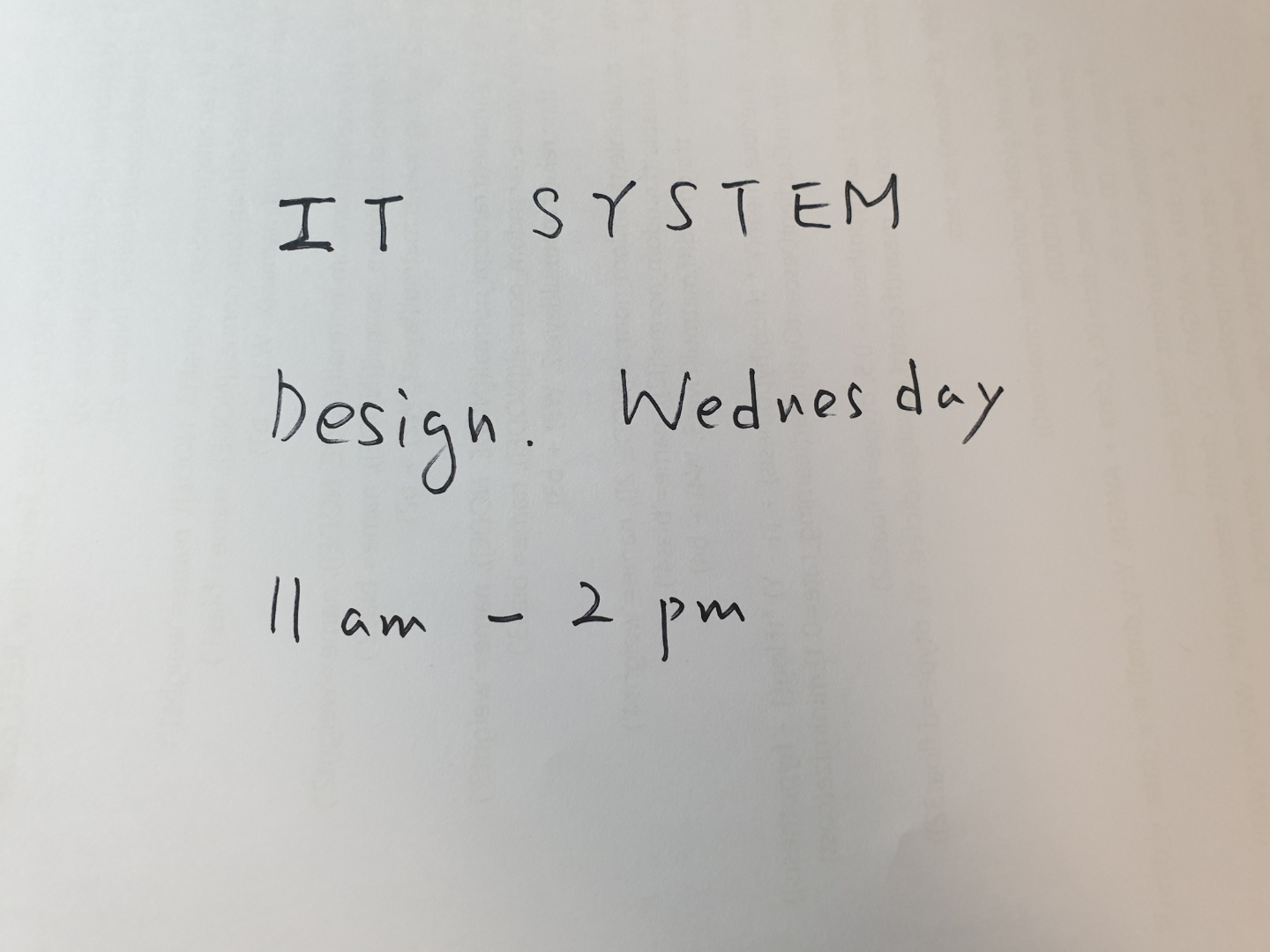
|  |  |
| --- | --- |
| import cv2  import numpy as np  import matplotlib.pyplot as plt  import os  from google.colab.patches import cv2\_imshow  import sys    np.set\_printoptions(threshold=sys.maxsize) # colab 버전 spyder 혹은 pycharm 에서는 np.nan 도 사용가능      def showimages(): # 지금까지의 이미지 과정들 출력      cv2\_imshow(bin\_img)  # Binary Image      cv2\_imshow(final\_thr) # Threshold Image      cv2\_imshow(src\_img)  # Source Image      cv2.waitKey(0)      cv2.destroyAllWindows()      def line\_array(array): # list x 의 최소값 상한값 반환      list\_x\_upper = []      list\_x\_lower = []      for y in range(5, len(array)-5):          s\_a, s\_p = strtline(y, array)          e\_a, e\_p = endline(y, array)          print(str(s\_a) + ',' + str(s\_p) + ',' + str(e\_a) + ',' + str(e\_p) + ',' + str(y))          if s\_a>=7 and s\_p>=5:              list\_x\_upper.append(y)          # bin\_img[y][:] = 255          if e\_a>=5 and e\_p>=7:              list\_x\_lower.append(y)              # bin\_img[y][:] = 255      return list\_x\_upper, list\_x\_lower    def strtline(y, array):      count\_ahead = 0      count\_prev = 0      for i in array[y:y+10]:          if i > 3:              count\_ahead+= 1      for i in array[y-10:y]:          if i == 0:              count\_prev += 1      return count\_ahead, count\_prev    def endline(y, array):      count\_ahead = 0      count\_prev = 0      for i in array[y:y+10]:          if i==0:              count\_ahead+= 1      for i in array[y-10:y]:          if i >3:              count\_prev += 1      return count\_ahead, count\_prev    def endline\_word(y, array, a):      count\_ahead = 0      count\_prev = 0      for i in array[y:y+2\*a]:          if i < 2:              count\_ahead+= 1      for i in array[y-a:y]:          if i > 2:              count\_prev += 1      return count\_prev ,count\_ahead    def end\_line\_array(array, a):      list\_endlines = []      for y in range(len(array)):          e\_p, e\_a = endline\_word(y, array, a)          #print(e\_p, e\_a)          if e\_a >= int(0.8\*a) and e\_p >= int(0.7\*a):              list\_endlines.append(y)      return list\_endlines    def refine\_endword(array):      refine\_list = []      for y in range(len(array)-1):          if array[y]+1 < array[y+1]:              refine\_list.append(array[y])      refine\_list.append(array[-1])      return refine\_list      def refine\_array(array\_upper, array\_lower):      upperlines = []      lowerlines = []      for y in range(len(array\_upper)-1):          if array\_upper[y] + 5 < array\_upper[y+1]:              upperlines.append(array\_upper[y]-10)      for y in range(len(array\_lower)-1):          if array\_lower[y] + 5 < array\_lower[y+1]:              lowerlines.append(array\_lower[y]+10)        upperlines.append(array\_upper[-1]-10)      lowerlines.append(array\_lower[-1]+10)        return upperlines, lowerlines    def letter\_width(contours):      letter\_width\_sum = 0      count = 0      for cnt in contours:          if cv2.contourArea(cnt) > 20:              x,y,w,h = cv2.boundingRect(cnt)              letter\_width\_sum += w              count += 1        return letter\_width\_sum/count      def end\_wrd\_dtct(final\_local, i, bin\_img, mean\_lttr\_width):      count\_y = np.zeros(shape = width)      for x in range(width):          for y in range(final\_local[i],final\_local[i+1]):              if bin\_img[y][x] == 255:                  count\_y[x] += 1      #end\_lines = end\_line\_array(count\_y, int(mean\_lttr\_width))      #endlines = refine\_endword(end\_lines)      #print(i)      '''for x in range(len(count\_y)):          if max(count\_y[0:x+1]) >= 3 and max(count\_y[x:]) >= 3 and (20-np.count\_nonzero(count\_y[x-10:x+10])) > 6:              print(x)'''        contours, hierarchy = cv2.findContours(lines\_img[i], cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)      letter\_width\_sum = 0      count = 0      for cnt in contours:          if cv2.contourArea(cnt) > 20:              x, y, w, h = cv2.boundingRect(cnt)              letter\_width\_sum += w              count += 1      if count != 0 :          mean\_width = letter\_width\_sum / count      else:          mean\_width = 0      #print(mean\_width)      spaces = []      line\_end = []      for x in range(len(count\_y)):          number = int(0.5\*int(mean\_width)) - np.count\_nonzero(count\_y[x-int(0.25\*int(mean\_width)):x+int(0.25\*int(mean\_width))])          if max(count\_y[0:x + 1]) >= 3 and number >= 0.4\*int(mean\_width):              spaces.append(x)          if max(count\_y[x:]) <= 2:              line\_end.append(x)      true\_line\_end = min(line\_end) + 10      #spaces = refine\_endword(spaces)      #print(spaces)      #print(true\_line\_end)      reti = []      final\_spaces = []      for j in range(len(spaces)):          if spaces[j] < true\_line\_end:              if spaces[j] == spaces[j-1] + 1:                  reti.append(spaces[j-1])              elif spaces[j] != spaces[j-1] + 1 and spaces[j-1] == spaces[j-2] +1:                  reti.append(spaces[j-1])                  retiavg = int(sum(reti)/len(reti))                  final\_spaces.append(retiavg)                  reti = []              elif spaces[j] != spaces[j-1] + 1 and spaces[j-1] != spaces[j-2] +1 and spaces[j] != spaces[j+1] -1:                  final\_spaces.append(spaces[j])          elif spaces[j] == true\_line\_end:              final\_spaces.append(true\_line\_end)      #print(final\_spaces)      for x in final\_spaces:          final\_thr[final\_local[i]:final\_local[i+1], x] = 255      return final\_spaces      def letter\_seg(lines\_img, x\_lines, i):      copy\_img = lines\_img[i].copy()      x\_linescopy = x\_lines[i].copy()        letter\_img = []      letter\_k = []        contours, hierarchy = cv2.findContours(copy\_img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)      for cnt in contours:          if cv2.contourArea(cnt) > 5:              x,y,w,h = cv2.boundingRect(cnt)              # letter\_img.append(lines\_img[i][y:y+h, x:x+w])              letter\_k.append((x,y,w,h))        letter\_width\_sum = 0      count = 0      for cnt in contours:          if cv2.contourArea(cnt) > 20:              x, y, w, h = cv2.boundingRect(cnt)              letter\_width\_sum += h              count += 1        #mean\_height = letter\_width\_sum/count        letter = sorted(letter\_k, key=lambda student: student[0])        for e in range(len(letter)):          if e < len(letter)-1:              if abs(letter[e][0] - letter[e+1][0]) <= 2:                  x,y,w,h = letter[e]                  x2,y2,w2,h2 = letter[e+1]                  if h >= h2:                      letter[e] = (x,y2,w,h+h2)                      letter.pop(e+1)                  elif h < h2:                      letter[e+1] = (x2,y,w2,h+h2)                      letter.pop(e)        for e in range(len(letter)):          letter\_img\_tmp = lines\_img[i][letter[e][1]-0:letter[e][1]+letter[e][3]+0,letter[e][0]-0:letter[e][0]+letter[e][2]+0]          letter\_img\_tmp = cv2.resize(letter\_img\_tmp, dsize=(28, 28), interpolation=cv2.INTER\_AREA)          width = letter\_img\_tmp.shape[1]          height = letter\_img\_tmp.shape[0]          count\_y = np.zeros(shape=(width))          for x in range(width):              for y in range(height):                  if letter\_img\_tmp[y][x] == 255:                      count\_y[x] = count\_y[x] +1          print(count\_y)          max\_list = []          for z in range(len(count\_y)):              if z>=5 and z<= len(count\_y)-6:                  if max(count\_y[z-5:z+6]) == count\_y[z] and count\_y[z] >= 2:                      max\_list.append(z)              elif z<5:                  if max(count\_y[0:z+6]) == count\_y[z] and count\_y[z] >= 2:                      max\_list.append(z)              elif z > len(count\_y)-6:                  if max(count\_y[z-5:len(count\_y)-1]) == count\_y[z] and count\_y[z] >= 2:                      max\_list.append(z)          print(max\_list)          rem\_list = []          final\_max\_list = []          for z in range(len(max\_list)):              if z > 0:                  if max\_list[z]-max\_list[z-1] <= 3:                      rem\_list.append(z-1)          for z in range(len(max\_list)):              if z not in rem\_list:                  final\_max\_list.append(max\_list[z])          print(final\_max\_list)          if len(final\_max\_list) <= 1:              print(False)          else:              max\_len = len(final\_max\_list) - 1              for j in range(max\_len):                  list = count\_y[final\_max\_list[j]:final\_max\_list[j+1]]                  min\_list = sorted(list)[:3]                  avg = sum(min\_list)/len(min\_list)                  print(avg)            x\_linescopy.pop(0)      word = 1      letter\_index = 0      for e in range(len(letter)):          #print(str(letter[e][0]) + ',' + str(letter[e][1]) + ',' + str(letter[e][2]) + ',' + str(letter[e][3]) + ',' + str(e))          if(letter[e][0]<x\_linescopy[0]):              letter\_index += 1              letter\_img\_tmp = lines\_img[i][letter[e][1]-0:letter[e][1]+letter[e][3]+5,letter[e][0]-2:letter[e][0]+letter[e][2]+2]              letter\_img = cv2.resize(letter\_img\_tmp, dsize =(28, 28), interpolation = cv2.INTER\_AREA)              cv2.imwrite('img1/'+str(i+1)+'\_'+str(word)+'\_'+str(letter\_index)+'.jpg', 255-letter\_img)          else:              x\_linescopy.pop(0)              word += 1              letter\_index = 1              letter\_img\_tmp = lines\_img[i][letter[e][1]-0:letter[e][1]+letter[e][3]+5,letter[e][0]-2:letter[e][0]+letter[e][2]+2]              letter\_img = cv2.resize(letter\_img\_tmp, dsize =(28, 28), interpolation = cv2.INTER\_AREA)              cv2.imwrite('img1/'+str(i+1)+'\_'+str(word)+'\_'+str(letter\_index)+'.jpg', 255-letter\_img)              # print(letter[e][0],x\_linescopy[0], word)      print("\n........Program Initiated.......\n")  src\_img= cv2.imread('/content/img\_3.jpg', 1) # Image Read, Color Read  copy = src\_img.copy() # Shallow Copy  height = src\_img.shape[0] # 행의 개수(length)  width = src\_img.shape[1] # 열의 개수(length)    print("\n Resizing Image........")  src\_img = cv2.resize(copy, dsize =(1320, int(1320\*height/width)), interpolation = cv2.INTER\_AREA) # resizing    height = src\_img.shape[0] # Resized 행의 개수(length)  width = src\_img.shape[1]  # Resized 열의 개수(length)    print("#---------Resized Image Info:--------#")  print("\tHeight =",height,"\n\tWidth =",width)  print("#----------------------------#")    grey\_img = cv2.cvtColor(src\_img, cv2.COLOR\_BGR2GRAY) # BGR to Gray    print("Applying Adaptive Threshold with kernel :- 21 X 21")  bin\_img = cv2.adaptiveThreshold(grey\_img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,cv2.THRESH\_BINARY\_INV,21,20) # Adaptive Threshold 적용  coords = np.column\_stack(np.where(bin\_img > 0)) # bin\_img > 0 인값만 반환하여 column 방향으로 stack 하여 2차원 array 형성  angle = cv2.minAreaRect(coords)[-1] # 물체의 회전을 고려한 rectangular 사각형  if angle < -45:      angle = -(90 + angle)  else:      angle = -angle  h = bin\_img.shape[0]  w = bin\_img.shape[1]  center = (w//2,h//2) # Centor 정중앙  angle = 0  M = cv2.getRotationMatrix2D(center,angle,1.0)  bin\_img = cv2.warpAffine(bin\_img,M,(w,h),                           flags=cv2.INTER\_CUBIC, borderMode=cv2.BORDER\_REPLICATE) # bin\_img : 변경될 이미지 M: 변경 matrix, 출력될 이미지 (w,h)    bin\_img1 = bin\_img.copy()  bin\_img2 = bin\_img.copy() # shallow copy    kernel = cv2.getStructuringElement(cv2.MORPH\_ELLIPSE,(3,3)) # 타원형 kernel 생성  kernel1 = np.array([[1,0,1],[0,1,0],[1,0,1]], dtype = np.uint8)  # final\_thr = cv2.morphologyEx(bin\_img, cv2.MORPH\_OPEN, kernel)  # final\_thr = cv2.dilate(bin\_img,kernel1,iterations = 1)  print("Noise Removal From Image.........")  final\_thr = cv2.morphologyEx(bin\_img, cv2.MORPH\_CLOSE, kernel) # noise 제거  contr\_retrival = final\_thr.copy() # shallow copy      print("Beginning Character Segmentation..............") ## Character Segmentation  count\_x = np.zeros(shape= (height))  for y in range(height):      for x in range(width):          if bin\_img[y][x] == 255 :              count\_x[y] = count\_x[y]+1    local\_minima = []  for y in range(len(count\_x)):      if y >= 10 and y <= len(count\_x)-11:          arr1 = count\_x[y-10:y+10]      elif y < 10:          arr1 = count\_x[0:y+10]      else:          arr1 = count\_x[y-10:len(count\_x)-1]      if min(arr1) == count\_x[y]:          local\_minima.append(y)    final\_local = []  init = []  end = []  for z in range(len(local\_minima)):      if z != 0 and z!= len(local\_minima)-1:          if local\_minima[z] != (local\_minima[z-1] +1) and local\_minima[z] != (local\_minima[z+1] -1):              final\_local.append(local\_minima[z])          elif local\_minima[z] != (local\_minima[z-1] + 1) and local\_minima[z] == (local\_minima[z+1] -1):              init.append(local\_minima[z])          elif local\_minima[z] == (local\_minima[z-1] + 1) and local\_minima[z] != (local\_minima[z+1] -1):              end.append(local\_minima[z])      elif z == 0:          if local\_minima[z] != (local\_minima[z+1]-1):              final\_local.append(local\_minima[z])          elif local\_minima[z] == (local\_minima[z+1]-1):              init.append(local\_minima[z])      elif z == len(local\_minima)-1:          if local\_minima[z] != (local\_minima[z-1]+1):              final\_local.append(local\_minima[z])          elif local\_minima[z] == (local\_minima[z-1]+1):              end.append(local\_minima[z])  for j in range(len(init)):      mid = (init[j] + end[j])/2      if (mid % 1) != 0:          mid = mid+0.5      final\_local.append(int(mid))    final\_local = sorted(final\_local)    no\_of\_lines = len(final\_local) - 1    print("\nGiven Text has   # ",no\_of\_lines, " #   no. of lines")    lines\_img = []    for i in range(no\_of\_lines):      lines\_img.append(bin\_img2[final\_local[i]:final\_local[i+1], :])    contours, hierarchy = cv2.findContours(contr\_retrival,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)  final\_contr = np.zeros((final\_thr.shape[0],final\_thr.shape[1],3), dtype = np.uint8)  cv2.drawContours(src\_img, contours, -1, (0,255,0), 1)    mean\_lttr\_width = letter\_width(contours)  print("\nAverage Width of Each Letter:- ", mean\_lttr\_width)      x\_lines = []    for i in range(len(lines\_img)):      x\_lines.append(end\_wrd\_dtct(final\_local, i, bin\_img, mean\_lttr\_width))    for i in range(len(x\_lines)):      x\_lines[i].append(width)    print(x\_lines)  #-------------/Word Detection-----------------#    #-------------Letter Segmentation-------------#    cv2.waitKey(0)  for i in range(no\_of\_lines):      letter\_seg(lines\_img, x\_lines, i)      chr\_img = bin\_img1.copy()    contours, hierarchy = cv2.findContours(chr\_img,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)  # print(len(contours))  # final\_contr = np.zeros((final\_thr.shape[0],final\_thr.shape[1],3), dtype = np.uint8)  # cv2.drawContours(src\_img, contours, -1, (0,255,0), 1)    for cnt in contours:      if cv2.contourArea(cnt) > 20:          x,y,w,h = cv2.boundingRect(cnt)          cv2.rectangle(src\_img,(x,y),(x+w,y+h),(0,255,0),2)      showimages()  [*Colored by Color Scripter*](http://colorscripter.com/info#e) | [cs](http://colorscripter.com/info#e) |

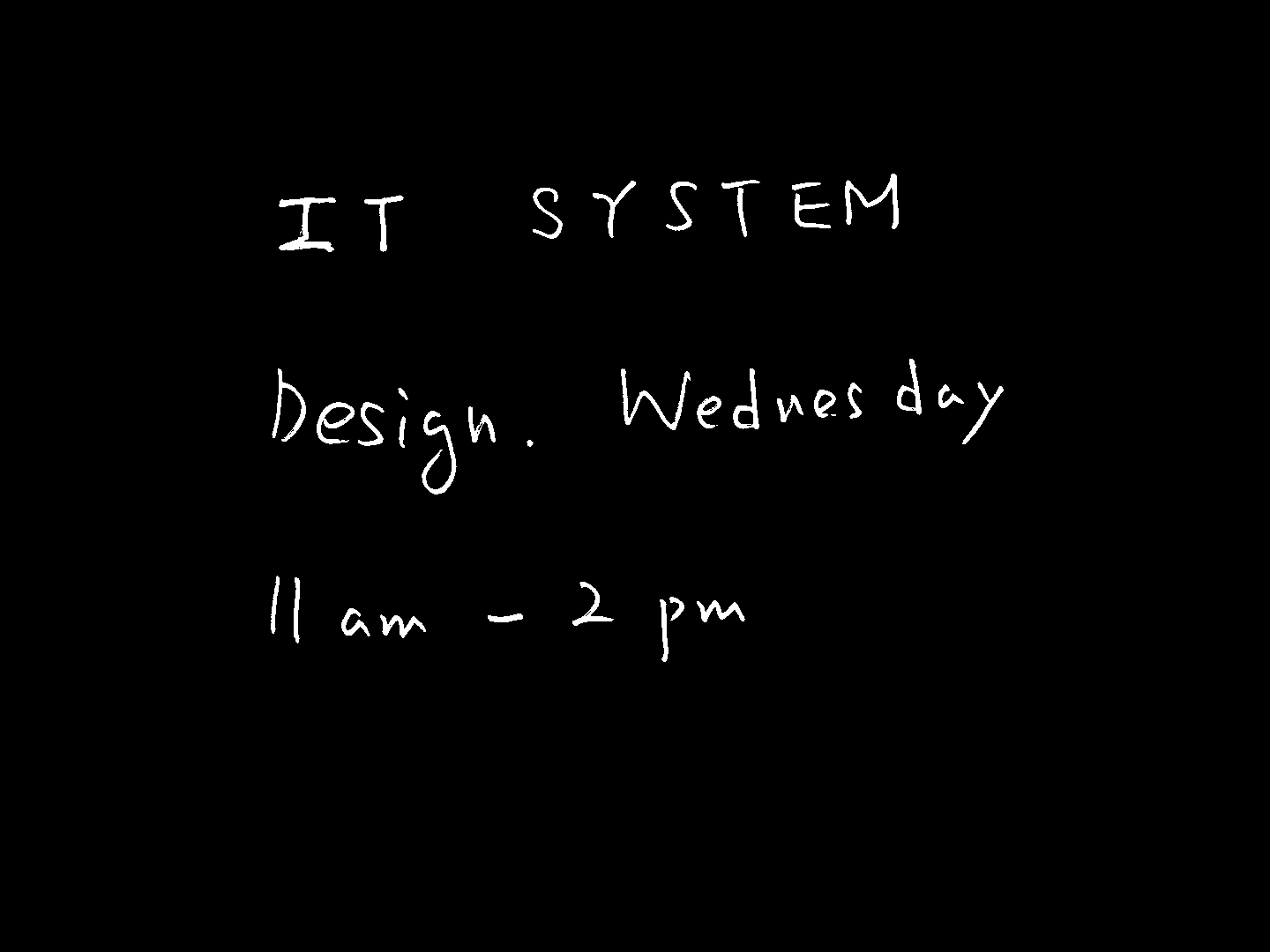
--------------------------------------------------------------------------------------------------------------

Character Segmentation 결과는 다음과 같다.

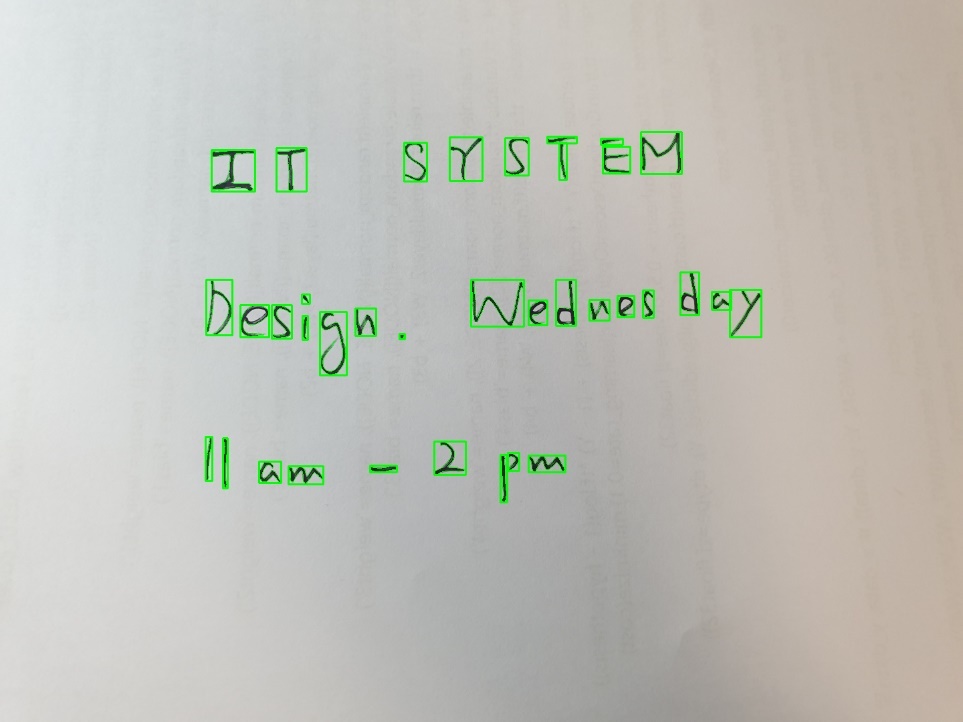
(다음장 계속)

① Image 1

 [Original Image]

 [Binary Image]

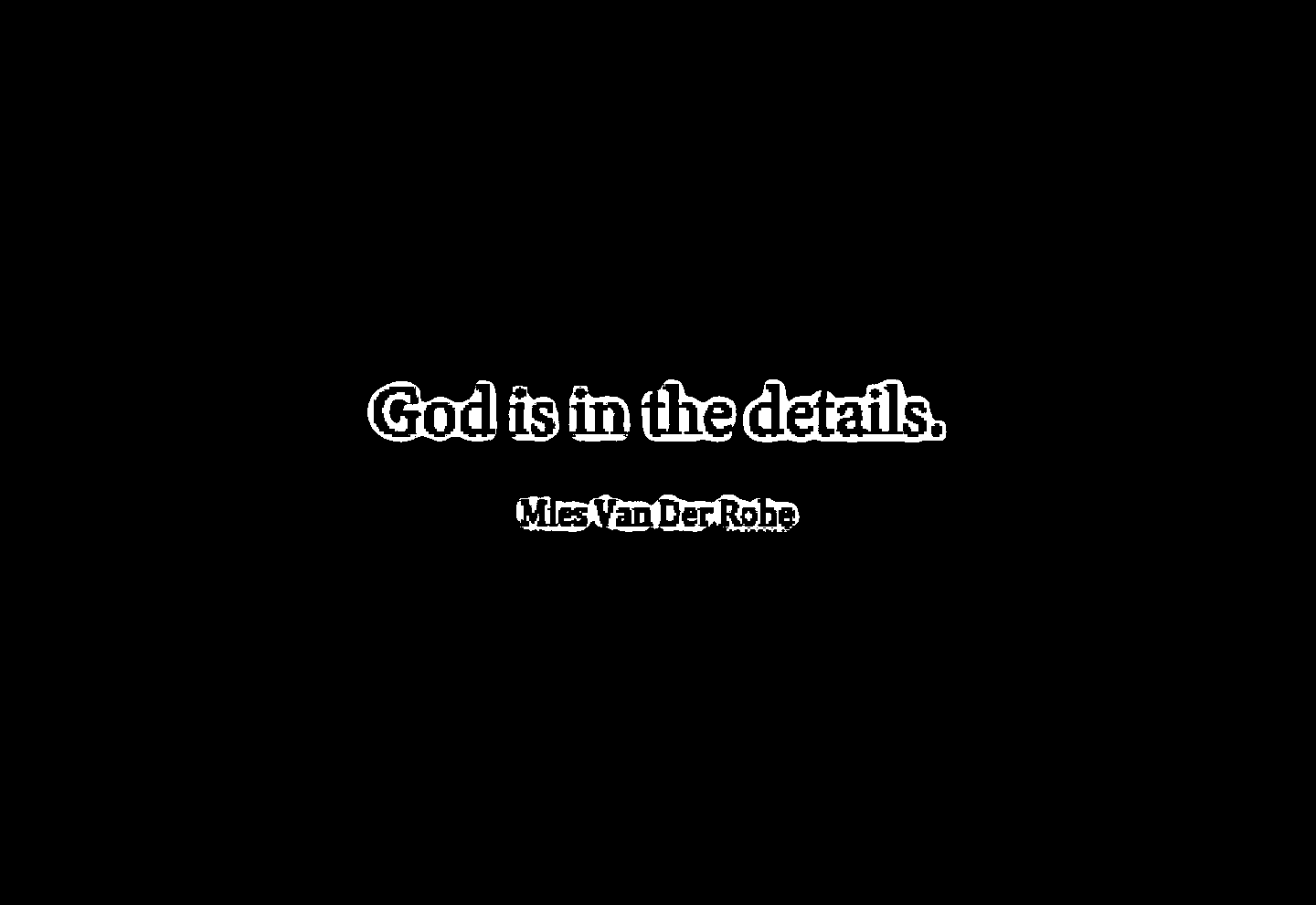
 [Threshold Image]

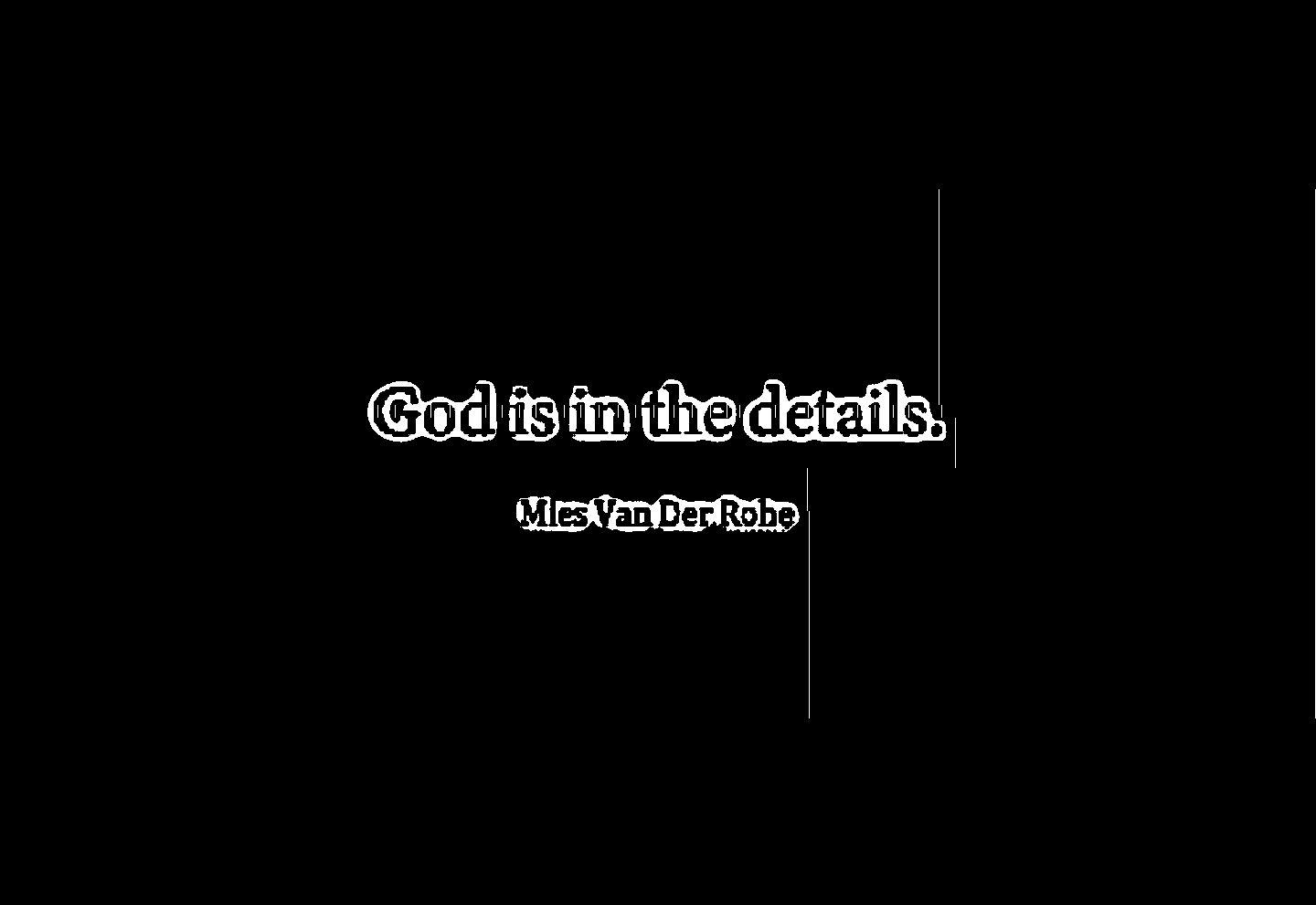
 [Character Segmented Image]

--------------------------------------------------------------------------------------------------------------

② Image 2

 [Original Image]

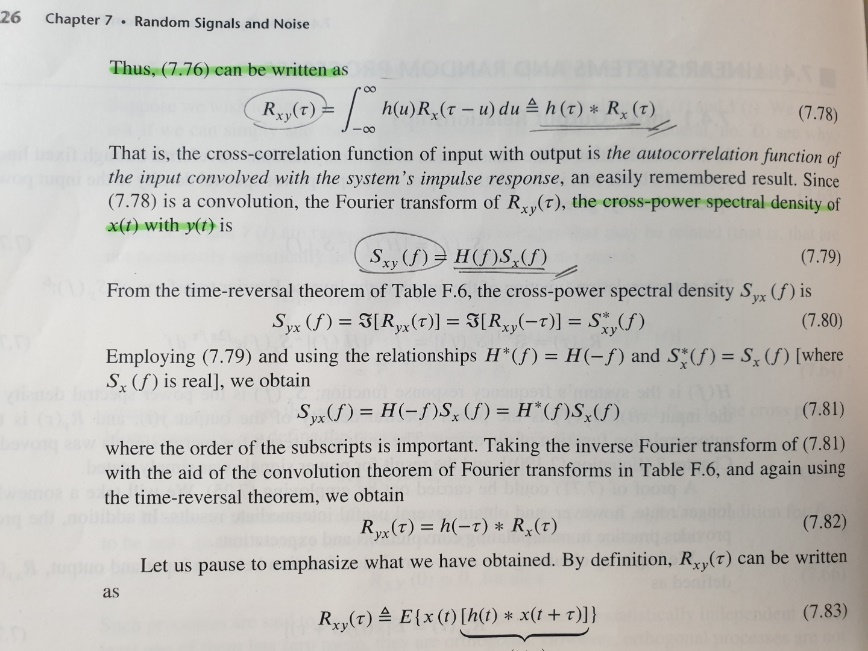
 [Binary Image]

 [Threshold Image]

 [Character Segmented Image]

Image 1 의경우 배경이 흰색이라 완벽히 detect 가 되는반면 배경이 조금이라도 있으면 확실히 부정확해 지는것을 볼 수 있다. 이제 배경이 없는 Image 1으로만 실험을 진행 하도록 하겠다.

[번외실험]



통신이론 책을 재미로 test 해봤는데, 위와 같은 사진은 size 를 조절해 줘도 문제가 자꾸 생긴다. Size 문제는 아닌데, 오류 메세지중 “ValueError: min() arg is an empty sequence” 가 뜨는데, x\_line 의 end 값이 없다는것은 x\_line을 append 할때 문제가 생겼다는것인데 구체적으로 어떠한것인지 밝혀내지 못하였다.

--------------------------------------------------------------------------------------------------------------

**3. 일반 alphabet, number training 시키기.**

Train 시킬때 keras model 은 pair review 할때 다른분들도 다 가져와서 쓰셨기때문에, 이번에 keras network 의 경우 알파벳 숫자 training 시키는 자료 인터넷에서 가져옴. (저번 과제에서 나의 코드는 정확도는 좋지만 1epoch 당 40분이 걸려서 쓰기가 두렵다. 또한 keras 는 간편하고 좋으므로 keras 사용) 다만 여기서 optimizer 을 rmsprop 을 사용하고있는데 수학적으로 정확히 어떤 방식으로 train 을 시키는지 잘 모르겠음.

근데 문제가 생겼다. 알파벳, 숫자 training 파일을 colab 에 업로드를 시켜야 하는데 database 자체를 한꺼번에 upload 하지 못하고 개별파일을 따로따로 업로드를 해야하는 문제가 생겨서 트레이닝은 불가해졌다.

다만 이부분은 코드를 긁어왔고 training 이 실패하지는 않을것이므로, training 을 시켜서 parameter 을 저장하고 그것을 기반으로 아까 2번에서 segment 단위로 분리한 사진을 저장한다음 resize 하여 test 하면 잘 나올것이라고 확신한다.

다만 시험기간이기때문에 마지막 부분을 구현을 못해서 아쉽긴 하지만 이것도 최선을 다한것이기 때문에 후회는 없다.

Training 부분 코드는 아래에 첨부하였다.

|  |  |
| --- | --- |
| # Alphabet, Number Training Code -> 인터넷에서 퍼옴.    from keras.preprocessing.image import ImageDataGenerator    from keras.models import Sequential    from keras.layers import Conv2D, MaxPooling2D    from keras.layers import Activation, Dropout, Flatten, Dense    from keras import backend as K            # dimensions of our images.    img\_width, img\_height = 150, 150        train\_data\_dir = 'database'    validation\_data\_dir = 'data/validation'    nb\_train\_samples = 7705    nb\_validation\_samples = 800    epochs = 50    batch\_size = 16        if K.image\_data\_format() == 'channels\_first':        input\_shape = (3, img\_width, img\_height)    else:        input\_shape = (img\_width, img\_height, 3)        model = Sequential()    model.add(Conv2D(32, (3, 3), input\_shape=input\_shape))    model.add(Activation('relu'))    model.add(MaxPooling2D(pool\_size=(2, 2)))        model.add(Conv2D(32, (3, 3)))    model.add(Activation('relu'))    model.add(MaxPooling2D(pool\_size=(2, 2)))        model.add(Conv2D(64, (3, 3)))    model.add(Activation('relu'))    model.add(MaxPooling2D(pool\_size=(2, 2)))        model.add(Flatten())    model.add(Dense(64))    model.add(Activation('relu'))    model.add(Dropout(0.5))    model.add(Dense(1))    model.add(Activation('sigmoid'))        model.compile(loss='binary\_crossentropy',                  optimizer='rmsprop',                  metrics=['accuracy'])        # this is the augmentation configuration we will use for training    train\_datagen = ImageDataGenerator(        rescale=1. / 255,        shear\_range=0.2,        zoom\_range=0.2,        horizontal\_flip=True)        # this is the augmentation configuration we will use for testing:    # only rescaling    test\_datagen = ImageDataGenerator(rescale=1. / 255)        train\_generator = train\_datagen.flow\_from\_directory(        train\_data\_dir,        target\_size=(img\_width, img\_height),        batch\_size=batch\_size,        class\_mode='binary')        validation\_generator = test\_datagen.flow\_from\_directory(        validation\_data\_dir,        target\_size=(img\_width, img\_height),        batch\_size=batch\_size,        class\_mode='binary')        model.fit\_generator(        train\_generator,        steps\_per\_epoch=nb\_train\_samples // batch\_size,        epochs=epochs,        validation\_data=validation\_generator,        validation\_steps=nb\_validation\_samples // batch\_size)        model.save\_weights('Parameter\_Training\_Data.h5') | [cs](http://colorscripter.com/info#e) |