```
!pip install opencv-python
!pip install matplotlib
!pip install numpy
!pip install -U pip jsoncomparison
```

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dis
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dis
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: pip in /usr/local/lib/python3.10/dist-packages
Collecting pip
  Downloading pip-23.2.1-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 19
Collecting jsoncomparison
  Downloading jsoncomparison-1.1.0-py3-none-any.whl (6.8 kB)
Installing collected packages: pip, jsoncomparison
  Attempting uninstall: pip
    Found existing installation: pip 23.1.2
    Uninstalling pip-23.1.2:
      Successfully uninstalled pip-23.1.2
Successfully installed jsoncomparison-1.1.0 pip-23.2.1
```

```
!pip install torch==1.7.1+cpu torchvision==0.8.2+cpu torchaudio===0.7.2 -f https:/
!pip install easyocr
```

```
Looking in links: https://download.pytorch.org/whl/torch_stable.html
ERROR: Could not find a version that satisfies the requirement torch==1.7.1+c
ERROR: No matching distribution found for torch==1.7.1+cpu
Requirement already satisfied: easyocr in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: torchvision>=0.5 in /usr/local/lib/python3.10/
Requirement already satisfied: opencv-python-headless in /usr/local/lib/pytho
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist
Requirement already satisfied: python-bidi in /usr/local/lib/python3.10/dist-
Requirement already satisfied: PyYAML in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: Shapely in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: pyclipper in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: ninja in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-pac
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dis
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/di
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/d
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.1
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist
```

1. Reading image files from Google Drive

```
from google.colab import drive
drive.mount('/gdrive', force_remount=True)
```

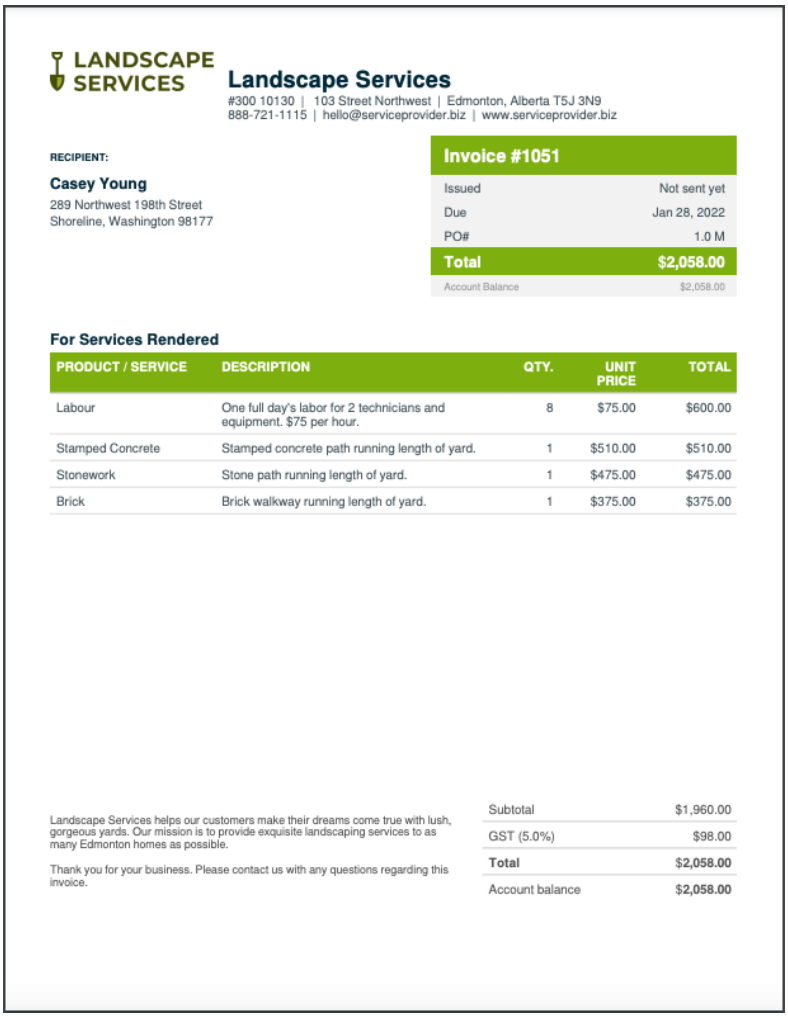
```
Mounted at /gdrive
```

```
from IPython.display import Image
Image('/gdrive/My Drive/kyc/hkidSample.png') ## HKID Information
```

파일 ×

..
OCR
Patches
sample_data
Detected_Text.png
Output.txt
Patches.zip
Prediction_Score.png
ocr_config


```
Image('/gdrive/My Drive/kyc/informationSample.jpg') ## Personal Information of the
```



LANDSCAPE SERVICES

Landscape Services

#300 10130 | 103 Street Northwest | Edmonton, Alberta T5J 3N9
888-721-1115 | hello@serviceprovider.biz | www.serviceprovider.biz

RECIPIENT:

Casey Young
289 Northwest 198th Street
Shoreline, Washington 98177

Invoice #1051	
Issued	Not sent yet
Due	Jan 28, 2022
PO#	1.0 M
Total	**$2,058.00**
Account Balance	$2,058.00

For Services Rendered

PRODUCT / SERVICE	DESCRIPTION	QTY.	UNIT PRICE	TOTAL
Labour	One full day's labor for 2 technicians and equipment. $75 per hour.	8	$75.00	$600.00
Stamped Concrete	Stamped concrete path running length of yard.	1	$510.00	$510.00
Stonework	Stone path running length of yard.	1	$475.00	$475.00
Brick	Brick walkway running length of yard.	1	$375.00	$375.00

Landscape Services helps our customers make their dreams come true with lush, gorgeous yards. Our mission is to provide exquisite landscaping services to as many Edmonton homes as possible.

Thank you for your business. Please contact us with any questions regarding this invoice.

Subtotal	$1,960.00
GST (5.0%)	$98.00
Total	**$2,058.00**
Account balance	**$2,058.00**

```
import cv2
import numpy as np
import pandas as pd
import json
import os
import easyocr
import urllib
import io
import matplotlib.pyplot as plt
%matplotlib inline


im_1_path = '/gdrive/My Drive/kyc/hkidSample.png'
im_2_path = '/gdrive/My Drive/kyc/informationSample.jpg'



def recognize_text(img_path):
    '''loads an image and recognizes text.'''
```

```
    reader = easyocr.Reader(['en'])
    return reader.readtext(img_path)


result = recognize_text(im_1_path)
```

```
    WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CP
```

```
result
```

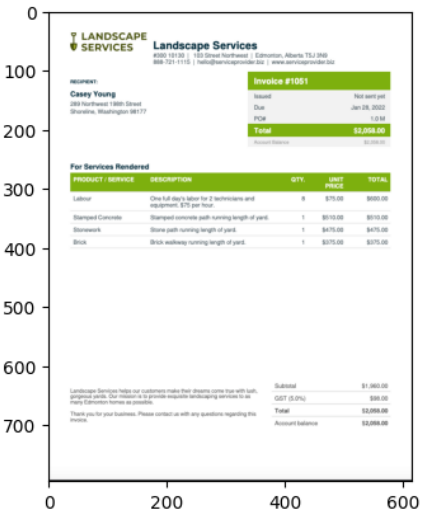
```
    [([[576, 14], [708, 14], [708, 44], [576, 44]], 'SAMPLE',
    0.7566931944652333),
     ([[224, 24], [549, 24], [549, 64], [224, 64]],
      '#1x71ER461',
      0.0325319032779643),
     ([[94, 58], [678, 58], [678, 84], [94, 84]],
      'HONG KONG PERMANENT IDENTITY CARD',
      0.7577993583498864),
     ([[31, 81], [165, 81], [165, 123], [31, 123]], '# % @',
    0.5022772153740567),
     ([[34, 124], [124, 124], [124, 156], [34, 156]], 'CHIU ,',
    0.788580575214395),
     ([[137, 123], [214, 123], [214, 156], [137, 156]],
      'Fung',
      0.9998358488082886),
     ([[226, 122], [300, 122], [300, 152], [226, 152]], 'Shek',
    0.999473512172699),
     ([[224, 166], [300, 166], [300, 196], [224, 196]],
      '2156',
      0.9269122621690242),
     ([[312, 164], [388, 164], [388, 196], [312, 196]], '6265',
    0.990030848149623),
     ([[402, 164], [476, 164], [476, 196], [402, 196]],
      '4311',
      0.9992089867591858),
     ([[222, 230], [428, 230], [428, 260], [222, 260]],
      '#KEA#tg Date of Birth',
      0.12048879441395849),
     ([[730, 151], [757, 151], [757, 256], [730, 256]], '1',
    0.5437483803760337),
     ([[226, 260], [408, 260], [408, 292], [226, 292]],
      '11-06-1965',
      0.9999756096822452),
     ([[464, 258], [526, 258], [526, 290], [464, 290]],
      '9} M',
      0.5428662896156311),
     ([[228, 300], [266, 300], [266, 326], [228, 326]], 'Az',
    0.866616605583141),
     ([[735, 263], [755, 263], [755, 339], [735, 339]], '1',
    0.5098538610700984),
     ([[224, 328], [436, 328], [436, 356], [224, 356]],
      'S#HI Date of Issue',
      0.3830257048817079),
     ([[51, 339], [93, 339], [93, 375], [51, 375]], 'K', 0.44465438354387743),
     ([[228, 362], [352, 362], [352, 392], [228, 392]],
      '(07-76)',
      0.9510012163874194),
     ([[225, 409], [409, 409], [409, 445], [225, 445]],
      '09-08-03',
      0.6616641670714657),
     ([[489, 407], [709, 407], [709, 445], [489, 445]],
      'D788888 ( 5)',
      0.6157737233965423)]
```

```
img_1 = cv2.imread(im_1_path)
img_1 = cv2.cvtColor(img_1, cv2.COLOR_BGR2RGB)
plt.imshow(img_1)
```

```
    <matplotlib.image.AxesImage at 0x7fc9e2d3ebf0>
```

```
img_2 = cv2.imread(im_2_path)
img_2 = cv2.cvtColor(img_2, cv2.COLOR_BGR2RGB)
plt.imshow(img_2)
```

```
<matplotlib.image.AxesImage at 0x7fc9e1547760>
```



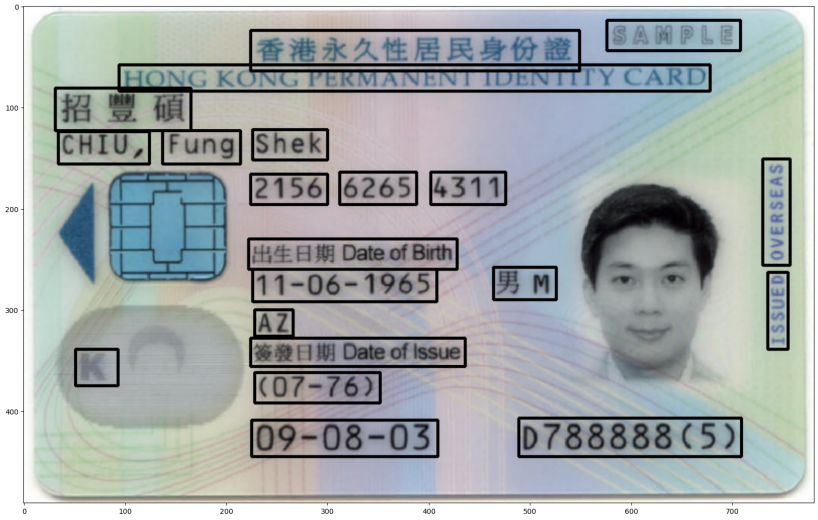

```
def overlay_ocr_text(img_path, save_name):
    '''loads an image, recognizes text, and overlays the text on the image.'''

    # loads image
    img = cv2.imread(img_path)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    dpi = 80
    fig_width, fig_height = int(img.shape[0]/dpi), int(img.shape[1]/dpi)
    plt.figure()
    f, axarr = plt.subplots(1,2, figsize=(fig_width, fig_height))
    axarr[0].imshow(img)

    # recognize text
    result = recognize_text(img_path)

    # if OCR prob is over 0.5, overlay bounding box and text
    for (bbox, text, prob) in result:
        if prob >= 0.5:
            # display
            print(f'Detected text: {text} (Probability: {prob:.2f})')

            # get top-left and bottom-right bbox vertices
            (top_left, top_right, bottom_right, bottom_left) = bbox
            top_left = (int(top_left[0]), int(top_left[1]))
            bottom_right = (int(bottom_right[0]), int(bottom_right[1]))

            # create a rectangle for bbox display
            cv2.rectangle(img=img, pt1=top_left, pt2=bottom_right, color=(255, 0,

            # put recognized text
            cv2.putText(img=img, text=text, org=(top_left[0], top_left[1] - 10), f

    # show and save image
    axarr[1].imshow(img)
    plt.savefig(f'./output/{save_name}_overlay.jpg', bbox_inches='tight')
```

3. Draw Bounding Boxes in Multiple Lines

```
img = cv2.imread(im_1_path)
# Image size with DPI
plt.figure(figsize=(20, 20), dpi=100)
i = 0
for detection in result:
  # Top left coordinate
    top_left = tuple([int (val) for val in detection[0][0]])
  # Bottom right coordinate
    bottom_right = tuple([int (val) for val in detection[0][2]])
  # Text extraction
    text = detection[1]
    print(str(i)+' '+text)
  # Draw Rectangle
    img = cv2.rectangle(img,top_left,bottom_right,(0,0,0),2)
    i+=1
```

```
plt.imshow(img)
plt.savefig('Detected_Text.png')
plt.show()
```

```
0 SAMPLE
1 #1x71ER461
2 HONG KONG PERMANENT IDENTITY CARD
3 # % @
4 CHIU ,
5 Fung
6 Shek
7 2156
8 6265
9 4311
10 #KEA#tg Date of Birth
11 1
12 11-06-1965
13 9} M
14 Az
15 1
16 S#HI Date of Issue
17 K
18 (07-76)
19 09-08-03
20 D788888 ( 5)
```




```
text = [result[res][1] for res in range(len(result))]
text
```

```
['SAMPLE',
 '#1x71ER461',
 'HONG KONG PERMANENT IDENTITY CARD',
 '# % @',
 'CHIU ,',
 'Fung',
 'Shek',
 '2156',
 '6265',
 '4311',
 '#KEA#tg Date of Birth',
 '1',
 '11-06-1965',
 '9} M',
 'Az',
 '1',
 'S#HI Date of Issue',
 'K',
 '(07-76)',
 '09-08-03',
 'D788888 ( 5)']
```

```
result = recognize_text(im_2_path)
```

```
WARNING:easyocr.easyocr:Neither CUDA nor MPS are available - defaulting to CP
```

```
img = cv2.imread(im_2_path)
# Image size with DPI
plt.figure(figsize=(20, 20), dpi=100)
i = 0
for detection in result:
  # Top left coordinate
    top_left = tuple([int (val) for val in detection[0][0]])
```

```
    # Bottom right coordinate
      bottom_right = tuple([int (val) for val in detection[0][2]])
    # Text extraction
      text = detection[1]
      print(str(i)+' '+text)
    # Draw Rectangle
      img = cv2.rectangle(img,top_left,bottom_right,(0,0,0),2)
      i+=1

plt.imshow(img)
plt.savefig('Detected_Text.png')
plt.show()
```

```
0 LANDSCAPE
1 SERVICES
2 Landscape Services
3 #300 10130
4 03 Sireet Norhiwest
5 Edrionton; Alberta T5J 3N9
6 888-721-1115
7 nellb@serviceprovider_Diz
8 WW/A.serviceprovider biz
9 AECIPIENT:
10 Invoice #1051
11 Casey Young
12 Issued
13 Nol sent yel
14 289 Norlnwesi 198ih Streel
15 Jan 28, 2022
16 Shoreline, Washinglon 98177
17 PO#
18 1.0 M
19 Total
20 32,058.00
21 Ceccot
22 12,L50 C
23 For Services Rendered
24 product
25 service
26 deScriPtION
27 QTY:
28 UnI
29 TOTAL
30 price
31 Labcur
32 Cne Iull day $ labor Ior =
33 lecnnicians &nd
34 575.00
35 Sed0.00
36 equipmenl 575 per nour:
37 Startiped Corcrete
38 Slamped concrele palh running lenglh l yard
39 8510.00
40 8510.00
41 Slonpwctk
42 Sione paln rurning lerigth ol yard_
43 $475.00
44 3475.00
```

```
text = [result[res][1] for res in range(len(result))]
text
```

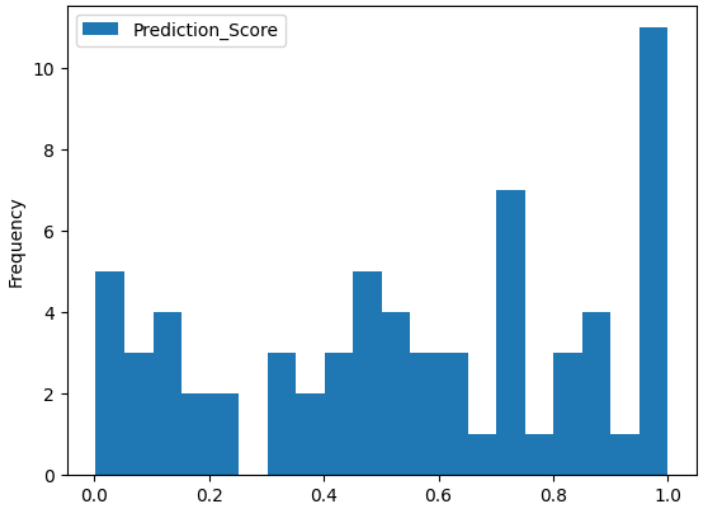
```
['LANDSCAPE',
 'SERVICES',
 'Landscape Services',
 '#300 10130',
 '03 Sireet Norhiwest',
 'Edrionton; Alberta T5J 3N9',
 '888-721-1115',
 'nellb@serviceprovider_Diz',
 'WW/A.serviceprovider biz',
 'AECIPIENT:',
 'Invoice #1051',
 'Casey Young',
 'Issued',
 'Nol sent yel',
 '289 Norlnwesi 198ih Streel',
 'Jan 28, 2022',
 'Shoreline, Washinglon 98177',
 'PO#',
 '1.0 M',
 'Total',
 '32,058.00',
 'Ceccot',
 '12,L50 C',
 'For Services Rendered',
 'product',
 'service',
 'deScriPtION',
 'QTY:',
 'UnI',
 'TOTAL',
 'price',
 'Labcur',
 'Cne Iull day $ labor Ior =',
 'lecnnicians &nd',
 '575.00',
 'Sed0.00',
 'equipmenl 575 per nour:',
 'Startiped Corcrete',
 'Slamped concrele palh running lenglh l yard',
 '8510.00',
 '8510.00',
 'Slonpwctk',
 'Sione paln rurning lerigth ol yard_',
 '$475.00',
 '3475.00',
```

```
'Brick',
'Brick walkway running length ol yard.',
'8375.00',
'5375.00',
'Sublolal',
'51,960.00',
'Landscane Senices Falds Oir cstomers Make thelr dream:',
'cmarnue',
'Corgeou3 /303',
'Ol Misgipn I5 10 provice exkglte [andscapina :E ices',
'GST (5.0%)',
'598.00',
```

4. Prediction Score Histogram

```
data = []
for i in range(len(result)):
  # Append the prediction score in the list
  data.append([result[i][2]])
  # Create the dataframe of the prediction score
  df = pd.DataFrame(data,columns=["Prediction_Score"])

display(df.head())
df.plot.hist(bins=20)
plt.savefig("Prediction_Score.png", dpi=100)
```

	Prediction_Score
0	0.998997
1	0.995329
2	0.812625
3	0.739384
4	0.705763




```
# Statistical Summary of score
df.describe()
```

	Prediction_Score
count	67.000000
mean	0.552530
std	0.322105
min	0.002256
25%	0.314942
50%	0.579132
75%	0.822412
max	0.999909

5. Write Output Text in the File

```
textfile = open("Output.txt", "w")
# Write output text in the file
for i in range(len(result)):
  textfile.write(str(i) +': ' + str([result[i][1]])+ "\n")

textfile.close()
```

6.Extracting and storing Detected Bounding Boxes

```
# Create Directory to store patches
os.mkdir ('/content/Patches')
%cd '/content/Patches'
```

```
/content/OCR
```

```
for i in range(len(result)):
  X= int(result[i][0][0][0])            # //Column
  Y= int(result[i][0][0][1])            # //Row
  W= int(result[i][0][1][0])            # //Width
  H= int(result[i][0][2][1])            # //Height
# Slicing of particular boxes
  cropped_image = img[Y:Y+H, X:X+W]
# Save an image
  cv2.imwrite(str(i)+'.png', cropped_image)
```

```
# Zip the patches Folder
# /Destination /Source
!zip -r /content/Patches.zip /content/Patches
```

```
updating: content/Patches/ (stored 0%)
updating: content/Patches/64.png (deflated 37%)
updating: content/Patches/32.png (deflated 17%)
updating: content/Patches/30.png (deflated 33%)
updating: content/Patches/49.png (deflated 33%)
updating: content/Patches/25.png (deflated 16%)
updating: content/Patches/12.png (deflated 19%)
updating: content/Patches/13.png (deflated 21%)
updating: content/Patches/4.png (deflated 11%)
updating: content/Patches/28.png (deflated 36%)
updating: content/Patches/15.png (deflated 22%)
updating: content/Patches/58.png (deflated 29%)
updating: content/Patches/40.png (deflated 38%)
updating: content/Patches/23.png (deflated 12%)
updating: content/Patches/16.png (deflated 12%)
updating: content/Patches/59.png (deflated 35%)
updating: content/Patches/21.png (deflated 27%)
updating: content/Patches/54.png (deflated 25%)
updating: content/Patches/6.png (deflated 12%)
updating: content/Patches/8.png (deflated 14%)
updating: content/Patches/9.png (deflated 7%)
updating: content/Patches/46.png (deflated 23%)
updating: content/Patches/55.png (deflated 32%)
updating: content/Patches/56.png (deflated 29%)
updating: content/Patches/34.png (deflated 36%)
updating: content/Patches/29.png (deflated 36%)
updating: content/Patches/61.png (deflated 26%)
updating: content/Patches/14.png (deflated 11%)
updating: content/Patches/35.png (deflated 36%)
updating: content/Patches/52.png (deflated 26%)
updating: content/Patches/19.png (deflated 21%)
updating: content/Patches/5.png (deflated 12%)
updating: content/Patches/27.png (deflated 33%)
updating: content/Patches/39.png (deflated 31%)
updating: content/Patches/48.png (deflated 49%)
updating: content/Patches/10.png (deflated 18%)
updating: content/Patches/42.png (deflated 17%)
updating: content/Patches/51.png (deflated 23%)
updating: content/Patches/37.png (deflated 18%)
updating: content/Patches/47.png (deflated 39%)
updating: content/Patches/62.png (deflated 32%)
updating: content/Patches/17.png (deflated 22%)
updating: content/Patches/38.png (deflated 18%)
updating: content/Patches/43.png (deflated 30%)
updating: content/Patches/44.png (deflated 42%)
updating: content/Patches/26.png (deflated 14%)
updating: content/Patches/60.png (deflated 33%)
updating: content/Patches/3.png (deflated 10%)
updating: content/Patches/24.png (deflated 14%)
updating: content/Patches/36.png (deflated 17%)
updating: content/Patches/22.png (deflated 28%)
updating: content/Patches/41.png (deflated 19%)
updating: content/Patches/7.png (deflated 16%)
updating: content/Patches/65.png (deflated 34%)
updating: content/Patches/18.png (deflated 23%)
updating: content/Patches/2.png (deflated 10%)
updating: content/Patches/53.png (deflated 23%)
updating: content/Patches/50.png (deflated 29%)
```

```
# Download the patches folder
from google.colab import files
files.download("/content/Patches.zip")
```

7. Converting output text in Json format

```
import json

# Assuming you have extracted text data from images and stored it in a variable ca
extracted_text = """0: ['LANDSCAPE']
1: ['SERVICES']
2: ['Landscape Services']
3: ['#300 10130']
4: ['03 Sireet Norhiwest']
5: ['Edrionton; Alberta T5J 3N9']
6: ['888-721-1115']
7: ['nellb@serviceprovider_Diz']
8: ['WW/A.serviceprovider biz']
9: ['AECIPIENT:']
10: ['Invoice #1051']
11: ['Casey Young']
12: ['Issued']
13: ['Nol sent yel']
14: ['289 Norlnwesi 198ih Streel']
15: ['Jan 28, 2022']
16: ['Shoreline, Washinglon 98177']
17: ['PO#']
18: ['1.0 M']
19: ['Total']
20: ['32,058.00']
21: ['Ceccot']
22: ['12,L50 C']
23: ['For Services Rendered']
24: ['product']
25: ['service']
26: ['deScriPtION']
27: ['QTY:']
28: ['UnI']
29: ['TOTAL']
30: ['price']
31: ['Labcur']
32: ['Cne Iull day $ labor Ior =']
33: ['lecnnicians &nd']
34: ['575.00']
35: ['Sed0.00']
36: ['equipmenl 575 per nour:']
37: ['Startiped Corcrete']
38: ['Slamped concrele palh running lenglh l yard']
39: ['8510.00']
40: ['8510.00']
41: ['Slonpwctk']
42: ['Sione paln rurning lerigth ol yard_']
43: ['$475.00']
44: ['3475.00']
45: ['Brick']
46: ['Brick walkway running length ol yard.']
47: ['8375.00']
48: ['5375.00']
49: ['Sublolal']
50: ['51,960.00']
51: ['Landscane Senices Falds Oir cstomers Make thelr dream:']
52: ['cmarnue']
53: ['Corgeou3 /303']
54: ['O1 Misgipn I5 10 provice exkglte [andscapina :E ices']
55: ['GST (5.0%)']
56: ['598.00']
57: ["Meri' Edmonton homes"]
58: ['pos3ible']
59: ['Tolal']
60: ['32,058.00']
61: ['Knank"\'ouior curcubine3s']
62: ['Please contec: Us witn 24} quesbons regerding this']
63: ['MYcce']
64: ['AccounL palance']
65: ['82,058.00']
66: ['Jaakute']
"""

# Create a dictionary to store the extracted text
extracted_data = {
    "text": extracted_text
```

```
}

# Convert the dictionary to JSON format
json_data = json.dumps(extracted_data)

# Print the JSON data
print(json_data)
```

```
{"text": "0: ['LANDSCAPE']\n1: ['SERVICES']\n2: ['Landscape Services']\n3: ['
```

```
json_data
```

```
'{"text": "0: [\'LANDSCAPE\']\\n1: [\'SERVICES\']\\n2: [\'Landscape Services
\']\\n3: [\'#300 10130\']\\n4: [\'03 Sireet Norhiwest\']\\n5: [\'Edrionton;
Alberta T5J 3N9\']\\n6: [\'888-721-1115\']\\n7: [\'nellb@serviceprovider_Diz
\']\\n8: [\'WW/A.serviceprovider biz\']\\n9: [\'AECIPIENT:\']\\n10: [\'Invoi
ce #1051\']\\n11: [\'Casey Young\']\\n12: [\'Issued\']\\n13: [\'Nol sent yel
\']\\n14: [\'289 Norlnwesi 198ih Streel\']\\n15: [\'Jan 28, 2022\']\\n16:
[\'Shoreline, Washinglon 98177\']\\n17: [\'PO#\']\\n18: [\'1.0 M\']\\n19:
[\'Total\']\\n20: [\'32,058.00\']\\n21: [\'Ceccot\']\\n22: [\'12,L50 C\']\\n
23: [\'For Services Rendered\']\\n24: [\'product\']\\n25: [\'service\']\\n2
6: [\'deScriPtION\']\\n27: [\'QTY:\']\\n28: [\'UnI\']\\n29: [\'TOTAL\']\\n3
0: [\'price\']\\n31: [\'Labcur\']\\n32: [\'Cne Iull day $ labor Ior =\']\\n3
3: [\'lecnnicians &nd\']\\n34: [\'575.00\']\\n35: [\'Sed0.00\']\\n36: [\'equ
```

```
# Verify the required information
if "Chiu Fung Shek" in extracted_text and ("November 6th 1965" in extracted_text o
    if "M" in extracted_text or "Gentleman" in extracted_text:
        if "09-08-03" in extracted_text or "September 8th 2003" in extracted_text:
            print("He is the correct person.")
        else:
            print("The date of issue is missing.")
    else:
        print("He is not the right person.")
else:
    print("He is not the right person.")
```

```
He is not the right person.
```

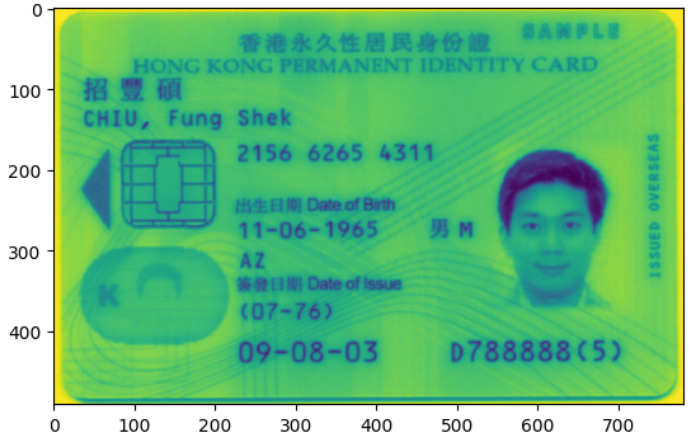
8. Image Preprocessing Using Open CV

```
im_1_path = '/gdrive/My Drive/kyc/hkidSample.png'
im_2_path = '/gdrive/My Drive/kyc/informationSample.jpg'
```

```
src = cv2.imread('/gdrive/My Drive/kyc/hkidSample.png', 1)
```

```
## Noise Removal / Black
gray = cv2.cvtColor(src, cv2.COLOR_BGR2GRAY)
gray = cv2.GaussianBlur(gray, (3, 3), 0)
plt.imshow(gray)
```

```
<matplotlib.image.AxesImage at 0x7fc9c6d61ed0>
```

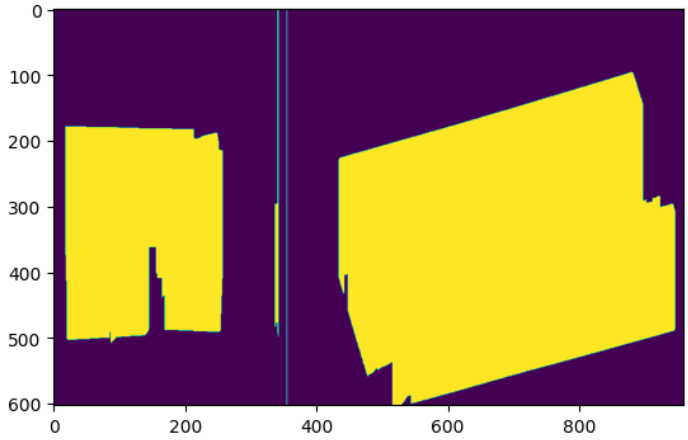



```
# Canny Edge Finding
canned = cv2.Canny(gray, 150, 300)
plt.imshow(canned)
```

```
<matplotlib.image.AxesImage at 0x7fc9c6bb48e0>
```


```
# Connection Edges
kernel = np.ones((10,1),np.uint8) # 가로 1 세로 10
mask = cv2.dilate(canned, kernel, iterations = 20)
plt.imshow(mask)
```

```
<matplotlib.image.AxesImage at 0x7fc9c7717e20>
```



```
src = cv2.imread('/gdrive/My Drive/kyc/hkidSampleTilted.png', 1)
plt.imshow(src)
```

```
<matplotlib.image.AxesImage at 0x7fc9c760eb00>
```

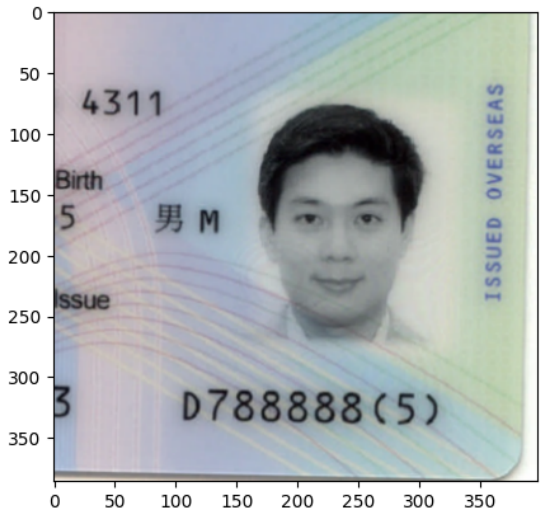

```
# Find out the contours
contours, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

# Finding Biggest contours
biggest_cntr = None
biggest_area = 0
for contour in contours:
    area = cv2.contourArea(contour)
    if area > biggest_area:
        biggest_area = area
        biggest_cntr = contour

# Outline box
rect = cv2.minAreaRect(biggest_cntr)
box = cv2.boxPoints(rect)
```

```
box = np.int0(box)


# Angle Calculation
angle = rect[-1]
if angle > 45:
    angle = -(75 - angle)

# Regulating Tilt
rotated = src.copy()
(h, w) = rotated.shape[:2]
center = (w // 2, h // 2)
M = cv2.getRotationMatrix2D(center, angle, 1.0)
rotated = cv2.warpAffine(rotated, M, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2

plt.imshow(rotated)
```

```
<matplotlib.image.AxesImage at 0x7fc9c66ad0c0>
```


```
ones = np.ones(shape=(len(box), 1))
points_ones = np.hstack([box, ones])
transformed_box = M.dot(points_ones.T).T

y = [transformed_box[0][1], transformed_box[1][1], transformed_box[2][1], transfor
x = [transformed_box[0][0], transformed_box[1][0], transformed_box[2][0], transfor

y1, y2 = int(min(y)), int(max(y))
x1, x2 = int(min(x)), int(max(x))

# crop
crop = rotated[y1:y2, x1:x2]


plt.imshow(crop)
```

```
<matplotlib.image.AxesImage at 0x7fc9c6541a80>
```



Thank you indeed for your evaluation!