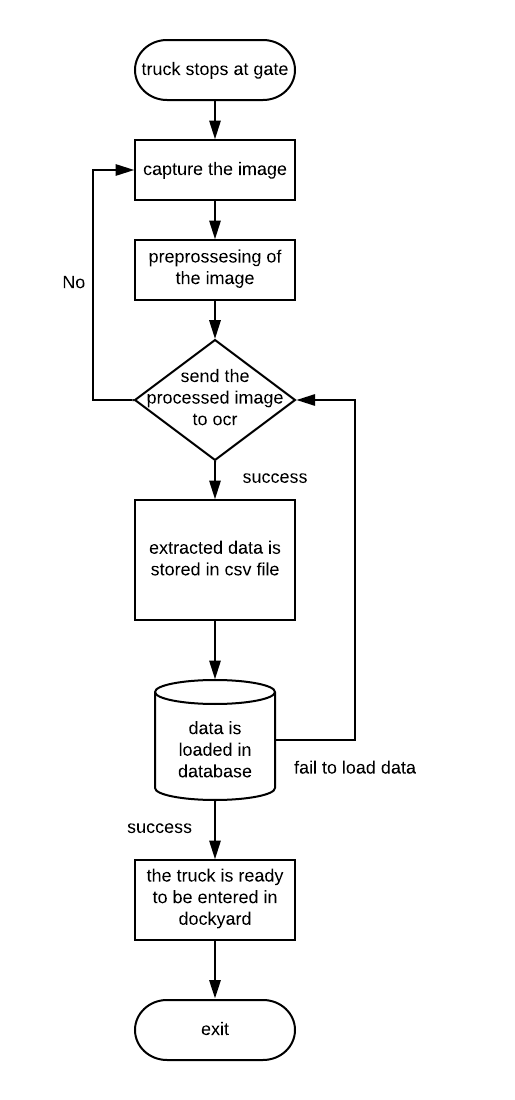
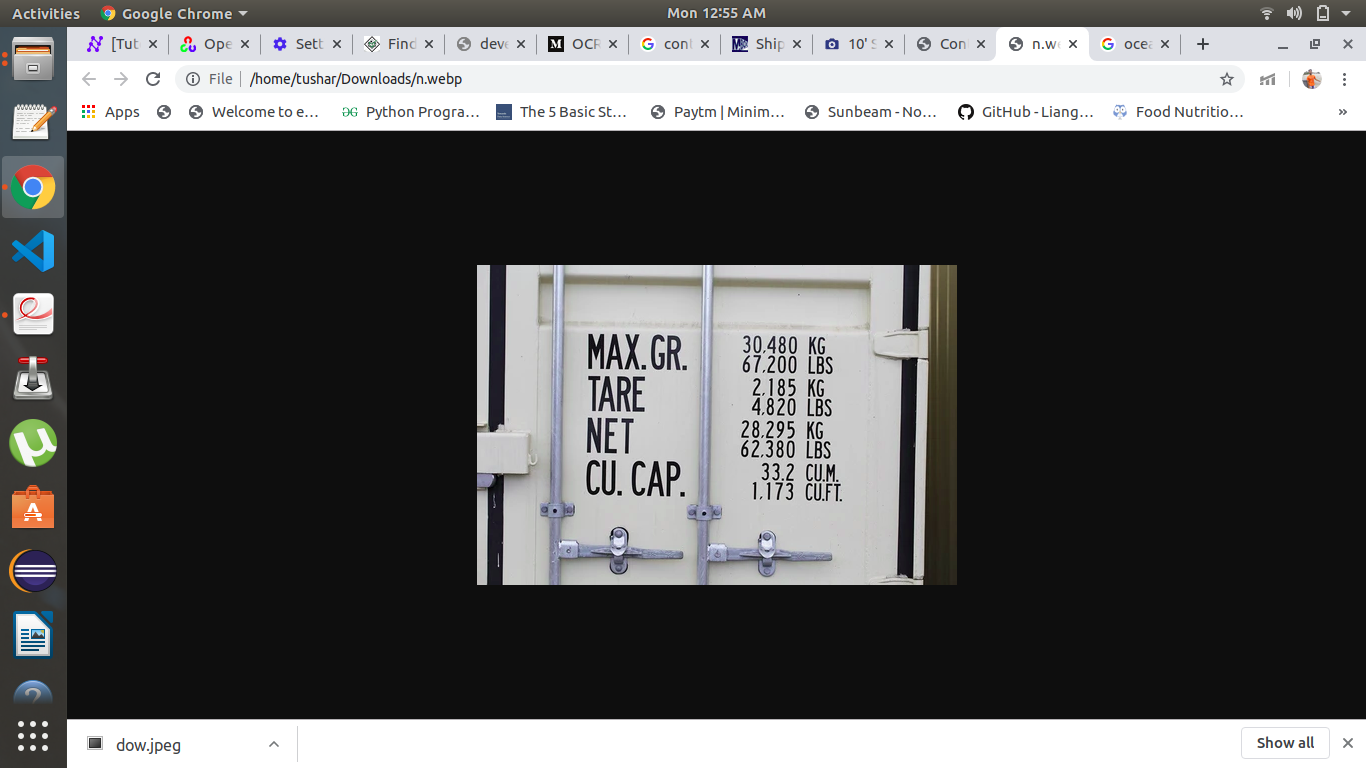
**1)steps to retrive text:**

The below one is the process which we have done while working on the given assignment



step 1: load the image



step 2: preprocessing :

the image is converted in grayscale so that while passing it to thresholdig or binarizing the image will give better result

step 3: binarize the image

after converting the image to grayscal the image is processed to binarize .it convert the grayscale image to the image which will contain on black and white color no other color will exist in the image because of the threshold function

we can increase the contrast in the image to make it easier for the OCR to deted the text and recognize it

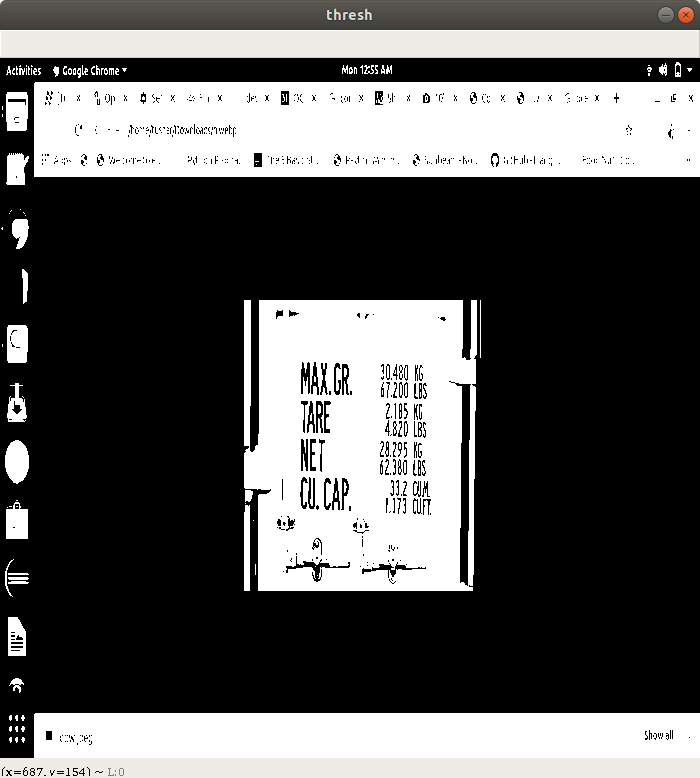


image after converting gray to thresh

step 4: erosion process

as we can see the image ,the character in the image will be converted in the black color,it is used to remove small white nosie in the image. so here it comes the process to make the digits more thicker .erosion replace the neighbouring pixel at the boundary black pixel to black and it results in making the text or digits bold which will help in detecting contours making text recognizable

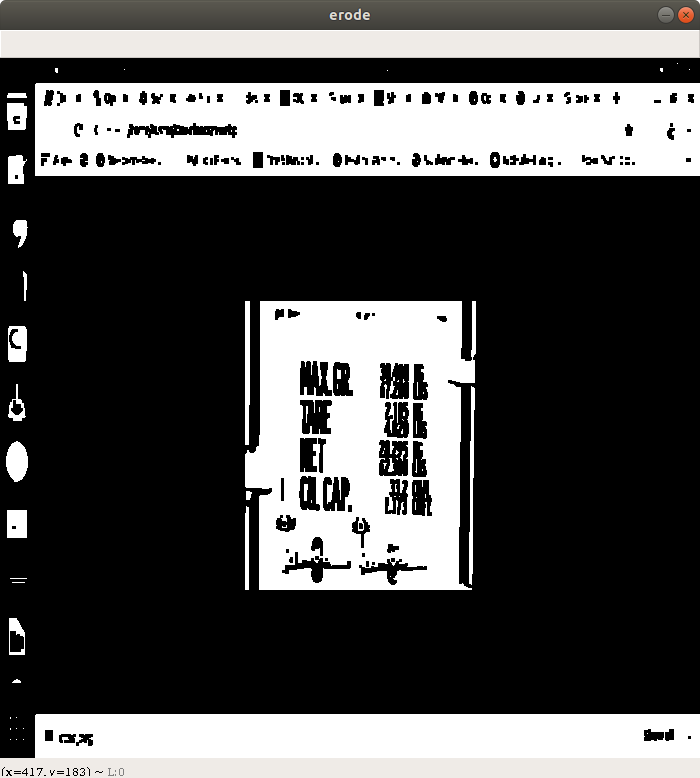


image after erosion process

step 5: dilation process

dilation is used to remove the smal black noise present in the image,it opposite to erosion it replaces the neighbouring pixels of white to white in our case dilation process is not required the characters color is black so after this process the text is not visible in the image

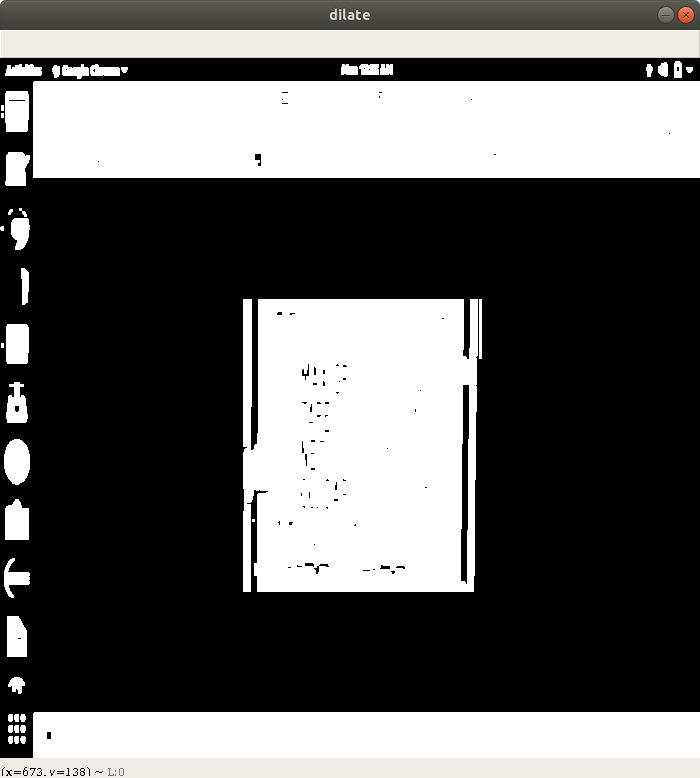
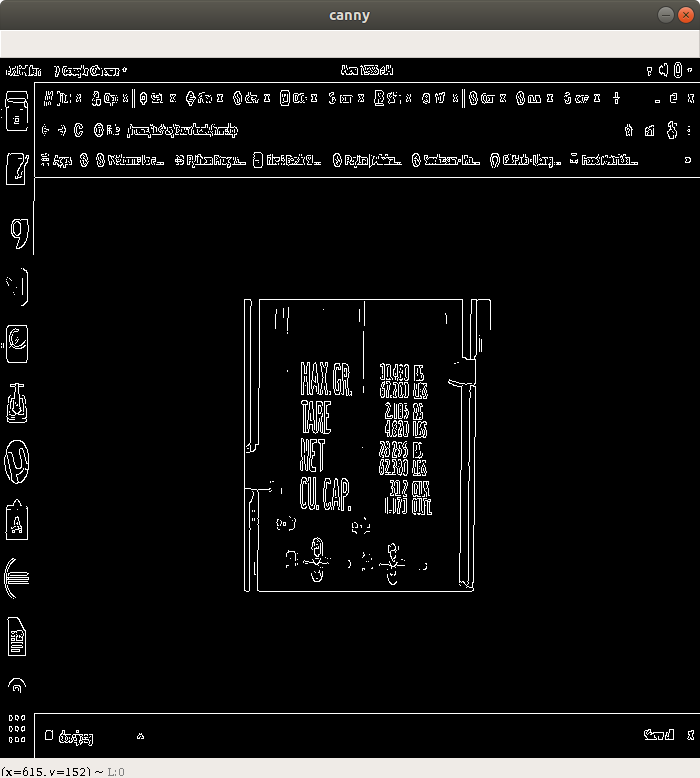


image after dilation process

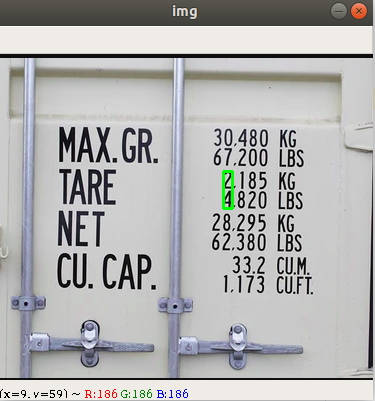
step 6: edge detection

In edge detection all the closing boundaries are highlighted this lead the image to provide more accuracy to detect the objects available in the image as the image quality is low it is not able to detect the contours in the image so we have skiped this stage

 image after canny edge detection

step 7: finding segments in image

trying to findout the segements in the image with pytesseract.image\_to\_data and then drawing

boxes around them but it is not detecting the text in image and noit drawing the segments

step 8:

we have used pytesseract which is a tool is retrive text from the image. We can retrive the text from the image by using the method from pytesseract and passing the image to it

pytesseract.image\_to\_string (“image path”)

then retrived is stored in file beacuse if somehow while loading the data some exception is occured and data is lost we can reload the data in database

**2)ans**

tesseract does it internally it organized text lines into blob,and the lines and regions are analysed for fixed pitch or proportional text .then text lines are are broken into words differently according to the kind of character spacing .recognition then process as a two pass process.in the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to more accurately recognize text lower down the page.

When we send image to OCR engine it first draw segments like rectangles on image and then these segments are send for text detection whether it contains any text if that segment contains text then the detected part is send for the character recognition

**3)ans**

the factors which play important role for text extraction **quality of image** the OCR engine scans the image and finds the edges, font, and paragraphs and individual characters.the OCR engine determines whether each character and/or word, matches characters it is ‘trained’ to recognize.

we can imporve the accuracy by converting image to different color space or **contrasting** the image so that the text can be detected properly.

**Quality of scans**One of the biggest factors is DPI or Dots per Inch. Setting the DPI lower than 200 will yield uninteligible results wheras setting it higher than 600dpi will just increase the size of the stored file without yielding much better results. We tend to recommend a 300dpi for in item.by keeping these factors in mind we can increase the accuracy of the text to be extracted

**program: or you can refer to ocr.py**

import re

import cv2

import numpy as np

import pytesseract

import mysql.connector

#loading image

image = cv2.imread("/home/tushar/Pictures/new.png")

# get grayscale image

def get\_grayscale(image):

return cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# noise removal:

def remove\_noise(image):

return cv2.medianBlur(image, 5)

# thresholding: converting image to binary form

def thresholding(image):

return cv2.threshold(image, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)[1]

# dilation: adding white pixel to the image at the boundary of black pixels

def dilate(image):

kernel = np.ones((3, 3), np.uint8)

return cv2.dilate(image, kernel, iterations=1)

# erosion: adding black pixel to the image at the boundary of white pixel so that the text will be sharpen

def erode(image):

kernel = np.ones((3, 3), np.uint8)

return cv2.erode(image, kernel, iterations=1)

# opening - erosion followed by dilation: not used because this is giving the blur image

def opening(image):

kernel = np.ones((5, 5), np.uint8)

return cv2.morphologyEx(image, cv2.MORPH\_OPEN, kernel)

# canny edge detection : in this case because the text size in the image is very small this will not detect the edges properly

def canny(image):

return cv2.Canny(image, 255, 255)

gray = get\_grayscale(image)

thresh = thresholding(gray)

canny = canny(gray)

noise = remove\_noise(gray)

erode = erode(thresh)

dilate = dilate(thresh)

custom\_config = r'-c tessedit\_char\_blacklist=abcdefghijklmnopqrstuvwxyz --psm 6'

text = pytesseract.image\_to\_string(thresh,config = custom\_config,lang='eng')

#print(text)

text = text.replace(",",".")

data = re.findall('\d\*\.?\d+',text)

data = data[6],data[8],data[10],data[13]

print("extracted data from the image:")

print(data)

file = open("myfile.csv","w")

for ele in data:

file.write(ele+',')

file.close()

#cv2.imshow("gray",gray)

#cv2.imshow("thresh",thresh)

#cv2.imshow("canny",canny)

#cv2.imshow("erode",erode)

#cv2.imshow("dilate",dilate)

cv2.waitKey(0)

cv2.destroyAllWindows()

myconn = mysql.connector.connect(host = "localhost", user = "dbda",passwd = "dbda", database = "classwork")

cur = myconn.cursor()

cur.execute("CREATE TABLE IF NOT EXISTS containers(max\_gross DOUBLE NOT NULL,tare\_weight DOUBLE NOT NULL,net DOUBLE NOT NULL,cu\_cap DOUBLE NOT NULL)")

print("\ntables in classwork databaes")

cur.execute("SHOW tables")

for x in cur:

print(x)

insert = "insert into containers(max\_gross,tare\_weight,net,cu\_cap) values ( ?, ?, ?, ?) "

#to load data from csv file we can execute this statement

#cur.execute("LOAD DATA INFILE '/home/tushar/Desktop/OCR/myfile.csv' INTO TABLE containers FIELDS TERMINATED BY ',' ENCLOSED BY ''' ")

#cur.execute(insert,data)

print("\nextracted data from containers table")

cur.execute("select \* from containers")

for x in cur:

print(x)

myconn.close()

#import cv2

#import pytesseract

#from pytesseract import Output

#img = cv2.imread('/home/tushar/Downloads/oc.jpeg')

#d = pytesseract.image\_to\_data(img, output\_type=Output.DICT)

#print(d.keys())

#n\_boxes = len(d['text'])

#for i in range(n\_boxes):

# if int(d['conf'][i]) > 60:

# (x, y, w, h) = (d['left'][i], d['top'][i], d['width'][i], d['height'][i])

# img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

#cv2.imshow('img', img)

#cv2.waitKey(0)