

Sistem Pendeteksi Plat Nomor Dengan Optical Character Recognition Metode HOG dan Linear SVM Menggunakan Raspberry Pi

Bab 1 Pendahuluan

Tanda nomor kendaraan bermotor atau TNKB merupakan identitas atau kode unik yang menjadi pengenal sebuah kendaraan bermotor. TNKB terdiri dari baris pertama yang menunjukkan dimana kendaraan tersebut berasal yang biasanya adalah kabupaten, lalu bagian selanjutnya adalah kode unik kepolisian yang berbeda tiap kendaraan, lalu bagian berikutnya menunjukkan tempat tinggal domisili pemilik kendaraan, dan yang terakhir terdapat masa berlaku TNKB yang harus diperbaharui 5 tahun sekali. Beberapa tujuannya adalah untuk parkir otomatis atau di mall, mendeteksi dan pengenalan identitas pemilik kendaraan, dan pelacaran pelanggaran lalu lintas.

Meningkatnya jumlah kendaraan bermotor dan semakin padatnya arus lalu lintas membuat teknologi pengolahan citra ini semakin penting. Yang dimana awalnya dengan menggunakan tenaga manual menjadi menggunakan gambar sebagai input untuk dikenali plat nomornya dengan menggunakan teknik pengenalan pola atau tepi. Salah satu cara agar dapat meraih tujuan tersebut dengan menggunakan *computer vision* untuk mengenali karakter plat nomor kendaraan. Tujuan teknologi ini adalah meniru kemampuan manusia agar dapat mengenali dan memahami informasi dari sebuah gambar selayaknya manusia.

Untuk mengatasi masalah dari keterbatasan biaya, tempat dan keakuratan program dapat menggunakan raspberry pi. Dengan menggunakan raspberry pi maka sistem pendeteksi plat nomor ini dapat diterapkan dimana-mana dan dikarenakan raspberry pi merupakan komputer kecil yang bekerja selayaknya komputer pada umumnya. Pada dasarnya sistem pendeteksi plat nomor ini bekerja dengan beberapa tahap yaitu deteksi dan memotong bagian plat nomor lalu dijadikan sebagai input frame, ekstrak karakter yang ada di plat nomor, lalu menerapkan suatu bentuk

Optical Character Recognition (OCR) untuk mengenali karakter yang diekstrak. Dengan menggunakan library Tesseract dan OpenCV dari python sebagai mesin OCR.

Bab 2 Tinjauan Pustaka

2.1 Raspberry Pi

Raspberry pi merupakan modul micro komputer yang mempunyai input dan output (GPIO). Jika dibandingkan dengan mikrokontroller lain seperti arduino, raspberry pi memiliki *connector* usb, keyboard dan mouse. Raspberry pi bisa dimodifikasi sesuai dengan kebutuhan penggunaannya. Sistem operasi utama bagi raspberry menggunakan debian yang dimodifikasi yang bernama raspbian OS.

Di dalam penelitian ini raspberry pi memiliki tugas sebagai image processing yang akan dilakukan dan data yang dihasilkan akan disimpan di SD card. Walaupun raspberry pi merupakan pengganti komputer yang artinya dengan menggunakan komputer kecil raspberry pi ini memiliki kemampuan serba minim kita membutuhkan berbagai cara agar kemampuannya tetap maksimal, yaitu dengan menggunakan metode, algoritma, dan berbagai cara agar tidak terjadi kendala. (Humonggio et al., 2019)

2.2 Python

Python adalah bahasa pemrograman yang berorientasi obyek. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi. Beberapa fitur yang dimiliki python adalah : (Wakhidah, 2012)

Beberapa fitur unggul yang dimiliki Python adalah:

1. Memiliki library yang luas; dalam distribusi Python telah disediakan modul-modul yang bermacam macam.
2. Memiliki tata bahasa yang jelas dan mudah dipelajari.
3. Memiliki aturan layout kode sumber yang memudahkan pengecekan, pembacaan kembali dan penulisan ulang kode sumber.

2.3 Dasar Pengolahan Citra

Pengolahan citra *atau image processing* adalah ilmu yang mempelajari dalam mengolah citra, yang dimaksud citra disini adalah gambar diam atau foto maupun video. Lalu diproses secara matematis di komputer. Agar sebuah citra dapat diolah dibutuhkannya nilai-nilai distrik yang dipisah-pisahkan dari citra sehingga dapat dilakukan segmentasi citra agar dapat dipisahkan gambar yang akan diolah dan yang tidak. (Gurav, 2019)

Segmentasi Citra adalah proses pemisahan objek satu dengan lainnya di sebuah citra yang akhirnya memiliki karakteristik tertentu. Prosesnya akan berhenti jika objeknya dapat dikenali dan ditemukan. Pada dasarnya segmentasi citra ini bertujuan untuk menemukan karakterisik yang unik di gambar atau objek di sebuah citra. Jadi semakin akurat segmentasi citra maka hasil output pengolahan citra dari computer vision semakin jelas dan bagus. (Darmawan, 2020)

Suatu citra dapat memiliki gangguan (noise) yang merupakan bagian gambar atau pixel yang mengurangi kualitas citra. Noise biasanya terjadi karena komposisi cahaya yang tidak teratur dan gangguan seperti titik-titik yang tidak jelas di sebuah gambar. Maka dibutuhkannya filtering agar noise ini dihilangkan sehingga informasi yang didapatkan dari gambar bisa lebih akurat. (Dodi, 2020)

Walaupun di dalam segmentasi citra sebuah gambar sudah dibagi-bagi namun objek yang ingin diproses belum dapat ditentukan karena segmentasi hanya membagi antara tiap objek yang terdeteksi dengan backgroundnya. Maka dari itu kita perlu memberikan label tiap komponen citra sehingga berbeda satu sama lain, yaitu dengan memberikan nilai berbeda dari tiap objek yang terdeteksi di segmentasi citra. Lalu dilakukan cropping untuk mendapatkan objek atau area yang diinginkan dengan cara menarik dua koordinat $(x1,y1)$ dan $(x2,y2)$ sehingga terbentuk persegi berisi objek yang berfokus pada citra yang ingin diamati. (Sugeng et al., 2020)

2.4 Optical Character Recognition

Setelah proses pengolahan citra maka proses selanjutnya menggunakan metode Optical Character Recognition (OCR) untuk mengenali gambar dari hasil segmentasi citra metode ini juga disebut dengan metode *Image Matching*. OCR bekerja dengan cara menerjemahkan gambar ke dalam bentuk teks yang dikenali agar dapat diedit dan disimpan. Dalam mengenali teksnya OCR mencoba mengenali apakah citra masukan yang diterima cocok dengan citra yang telah dilatih sebelumnya yang ada di database. Lalu selanjutnya terdapat post processing yang pada umumnya proses yang dilakukan di tahap ini proses koreksi ejaan sesuai dengan bahasa yang digunakan. (Suharyanto, 2020)

Bab 3 Desain Sistem

Dalam pembuatan sistem pendeteksi plat nomor ini ada beberapa langkah yang akan dilakukan

:

- Mendeteksi dan melokalisasi plat nomor dalam sebuah bingkai gambar.
- Mengekstrak karakter yang ada dari plat nomor.
- Menerapkan sistem OCR untuk mengenali karakter yang diekstrak.

Dalam pembuatan sistem pendeteksi plat nomor ini terdapat tantangan sendiri yang harus dilewati dan memperumit pengenalan objek, seperti :

- Kondisi pencahayaan gambar seperti gangguan pantulan, dan bayangan.
- Gambar kendaraan yang tidak jelas sehingga menyebabkan *blurring*.
- hambatan lainnya seperti dari kemampuan sistem itu sendiri.

Poin-poin dalam pembuatan sistem pendeteksi plat nomor dijabarkan dalam berbagai bagian dengan menggunakan pengetahuan tentang OpenCV, python, dan Tesseract-OCR.

3.1 Mendeteksi Letak Plat Nomor

Langkah pertama dalam membuat aplikasi pendeteksi plat nomor di raspberry pi ini adalah mendeteksi dimanakah letak plat nomornya. Dengan menggunakan opsi kontour dari OpenCV untuk mendeteksi objek persegi agar dapat mendapatkan bagian plat nomor. Akurasi dapat ditingkatkan jika kita mengetahui besar ukuran plat nomor, warna, dan perkiraan lokasi dimanakah letak plat nomor. Biasanya algoritma deteksi ini dilatih berdasarkan posisi dari kamera dan tipe plat nomor yang digunakan. Hal ini semakin rumit jika gambar tidak memiliki objek mobil di dalamnya. Maka dari itu akan dilakukan segmentasi karakter untuk proses selanjutnya.

3.2 Segmentasi karakter

Merupakan bagian untuk segmentasi atau memisah plat nomor dari gambar keseluruhan dengan cara memotong bagian tersebut lalu disimpan dalam bentuk file gambar yang baru dengan menggunakan openCV. Hal ini disebut dengan ROI (region of interest)

3.3 Pengenalan Karakter Menggunakan Tesseract-OCR

Sekarang, gambar baru telah didapatkan dengan angka dan huruf di dalamnya proses selanjutnya adalah menerapkan OCR (optical character recognition) untuk mendeteksi angka dan huruf tersebut. Dengan menggunakan *package* pytesseract untuk membaca karakter yang ada di dalam gambar. Karakter yang telah dideteksi lalu akan ditampilkan dalam sebuah konsol ketika di *compile*.

Namun sebelum memulai proyeknya kita rencanakan terlebih dahulu susunan direktori proyek yang ada.

1. license_plates
 - a. group1 (berisi gambar yang bagus untuk di proses)
 - b. group2 (berisi gambar yang kurang bagus untuk di proses)
2. ocrmobil
 - a. anpr
 - ❖ __init__.py
 - ❖ anpr.py

__init__.py
3. ocr_license_plate.py

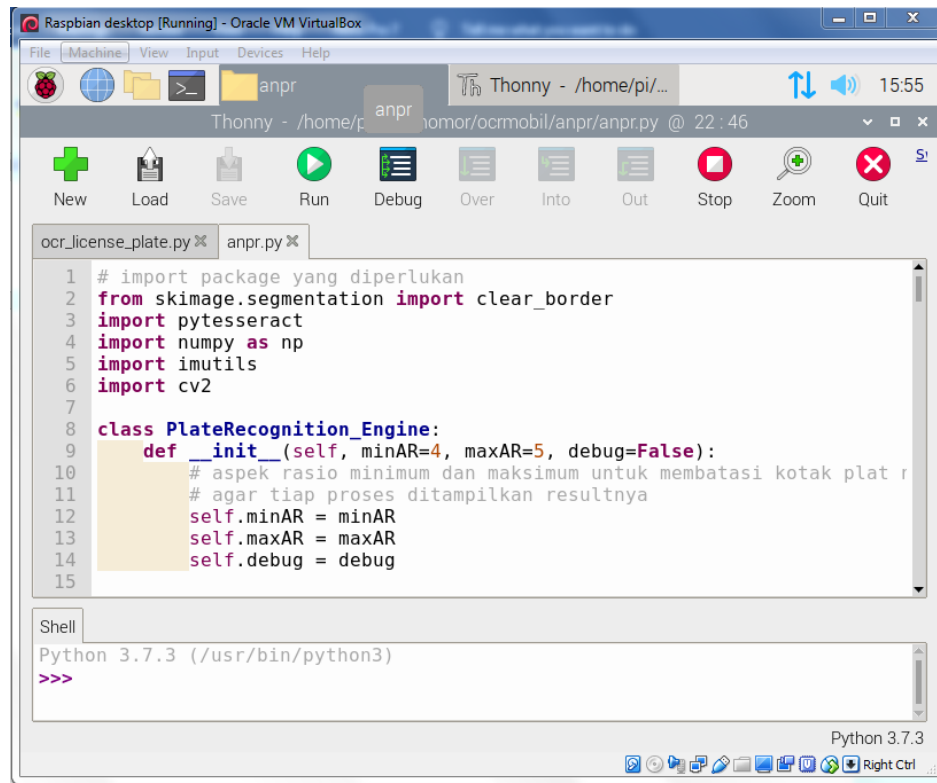
Tiap komponen di dalam folder ini memiliki beberapa bagian fungsi, yaitu :

- license_plates : Bagian folder yang berisi 2 sub folder yang berisi gambar JPG
- anpr.py : berisi tentang class PlateRecognition_Engine yang bertanggung jawab untuk melokalisasi plat nomor dan melakukan OCR (Optical Character Recognition).
- ocr_license_plate.py : script driver python utama yang menggunakan kelas PlateRecognition_Engine untuk melakukan OCR ke seluruh grup atau folder gambar.

Bab 4 Implementasi

1. Mengimplementasikan mesin OCR

A. Implementasi pytesseract dan openCV dengan import ke dalam script python



```

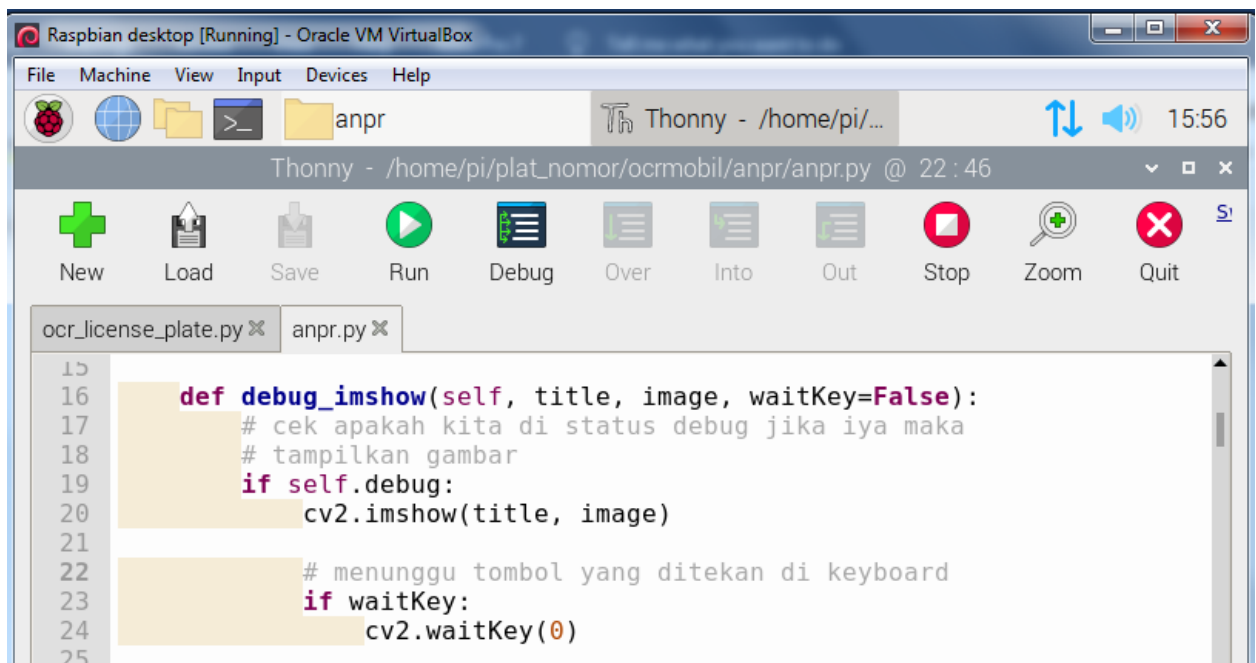
1 # import package yang diperlukan
2 from skimage.segmentation import clear_border
3 import pytesseract
4 import numpy as np
5 import imutils
6 import cv2
7
8 class PlateRecognition_Engine:
9     def __init__(self, minAR=4, maxAR=5, debug=False):
10         # aspek rasio minimum dan maksimum untuk membatasi kotak plat r
11         # agar tiap proses ditampilkan hasilnya
12         self.minAR = minAR
13         self.maxAR = maxAR
14         self.debug = debug
15
16
17 Shell
18 Python 3.7.3 (/usr/bin/python3)
19 >>>

```

Selain dilakukannya import package yang dibutuhkan ada parameter-parameter yang ada di bagian ini.

- minAR : aspek minimum rasio yang digunakan untuk mendeteksi dan memfilter kotak plat nomor, yang memiliki nilai default 4.
- maxAR : aspek maksimum rasio kotak plat nomor yang memiliki nilai default 5.
- debug : sebuah tanda untuk menunjukkan apakah kita harus menampilkan hasil secara langsung dari pemrosesan gambar

B. Debug Pipeline Citra Komputer

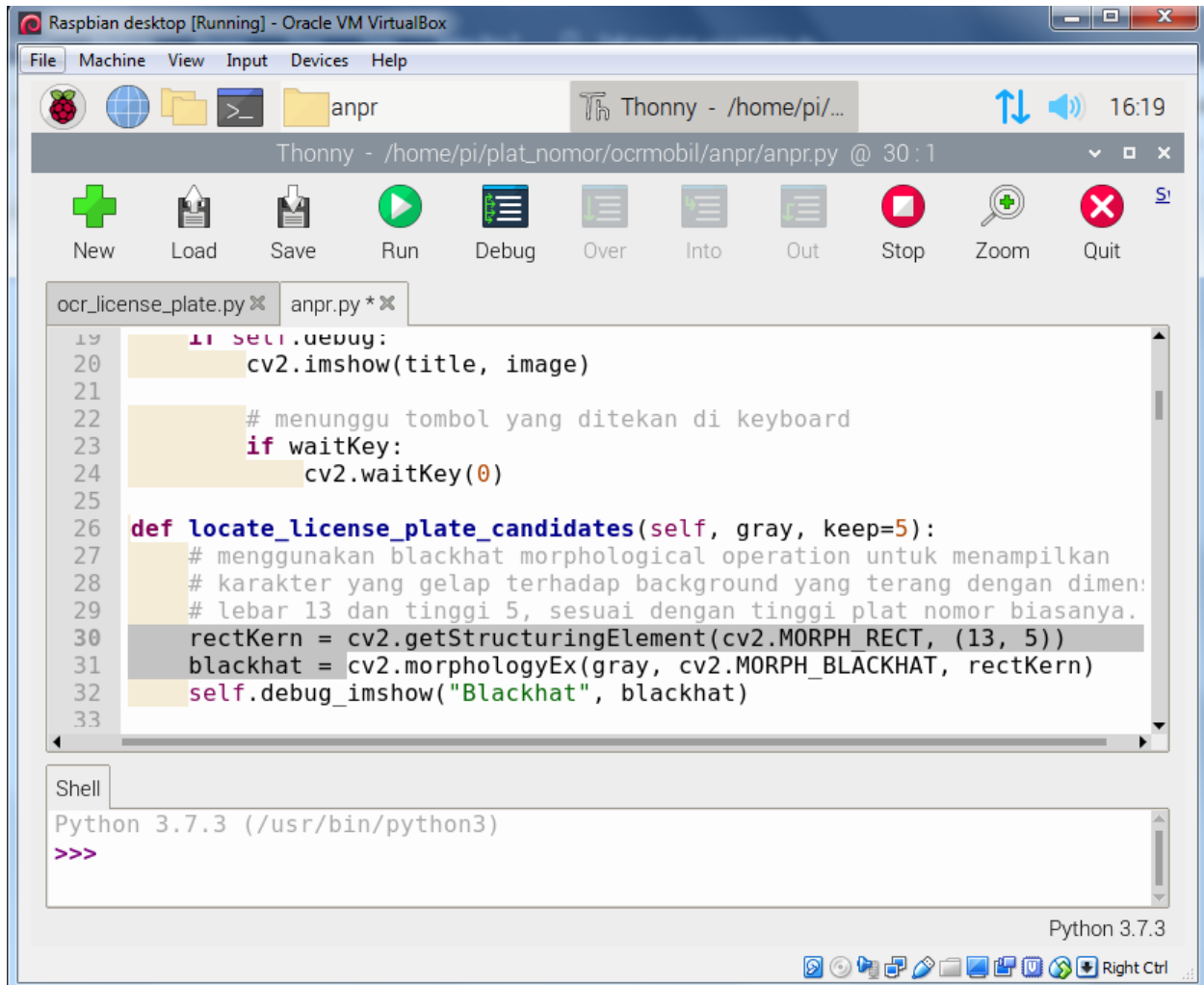


Dengan dibuatnya konstruktor, maka akan didefinisikan fungsi helper untuk menampilkan hasil gambar pada titik tertentu dengan menggunakan debug mode. Ada 3 parameter dalam fungsi helper ini

- title : Judul yang diinginkan, sehingga gambar yang dihasilkan tidak akan menumpuk dan mengganti satu sama lain melainkan akan dibuatnya jendela baru
- image : Menampilkan gambar di dalam window program openCV ini.

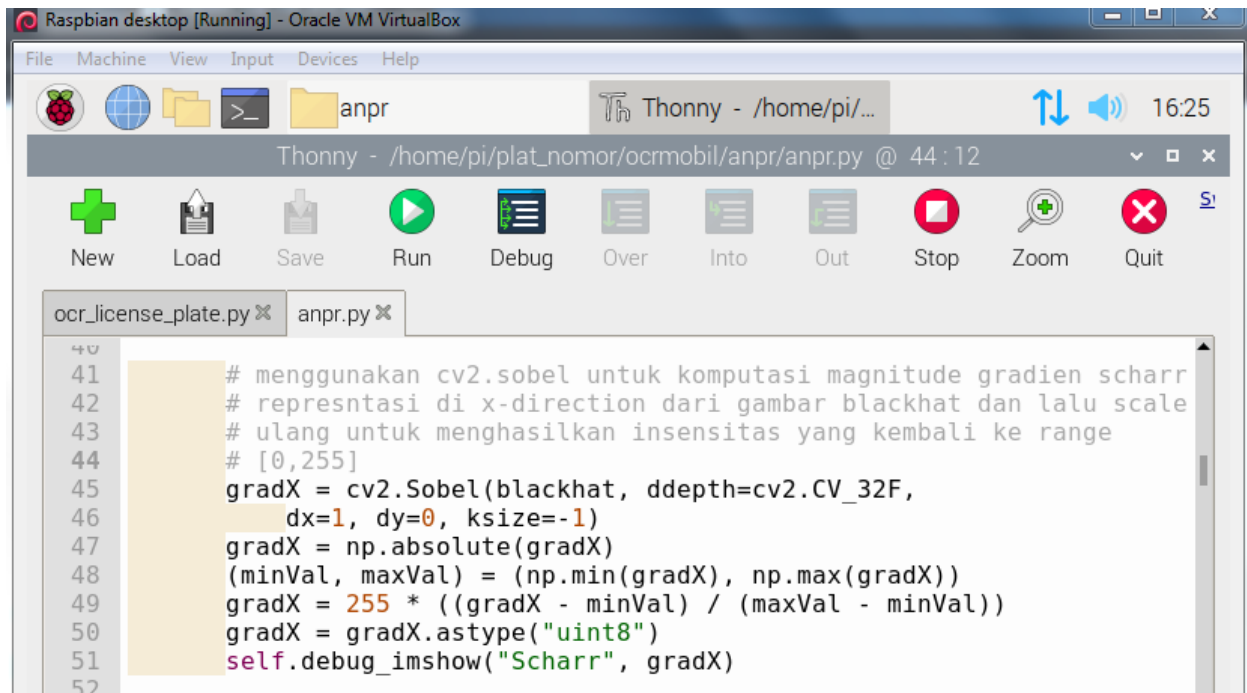
- waitKey : Pertanda apakah tampilan gambar selanjutnya dilakukan dengan mendeteksi tombol yang akan ditekan.

C. Menemukan Potensi Letak Plat Nomor



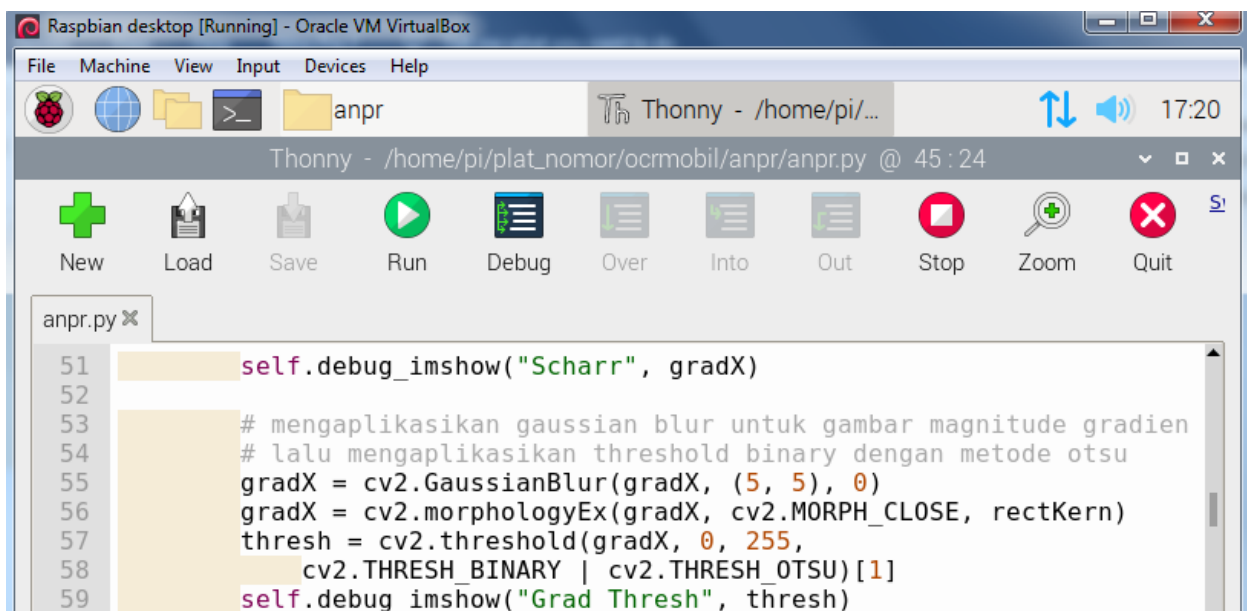
Disini kita akan melakukan *blackhat morphological operation* untuk menampilkan karakter yang gelap seperti huruf, digit, dan simbol yang dibandingkan background yang cerah.

menggunakan kernel kotak kecil (line 35), akan diaplikasikan operasi penutup (line 36) untuk mengisi lobang kecil yang akhirnya akan membantu dalam mengidentifikasi struktur yang lebih besar. Di line 37 dan 38 akan dilakukan *binary threshold* kepada gambar menggunakan metode otsu untuk menampilkan bagian yang tertepa cahaya yang mungkin merupakan tempat plat nomor tersebut.



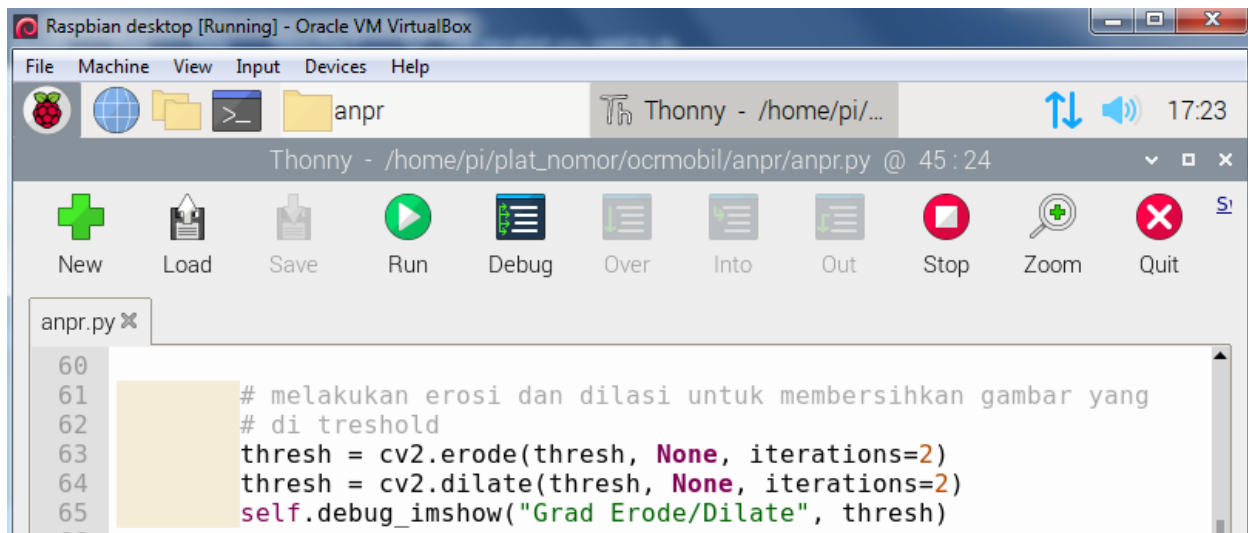
```
41 # menggunakan cv2.sobel untuk komputasi magnitudo gradien scharr
42 # represntasi di x-direction dari gambar blackhat dan lalu scale
43 # ulang untuk menghasilkan insensitas yang kembali ke range
44 # [0,255]
45 gradX = cv2.Sobel(blackhat, ddepth=cv2.CV_32F,
46                  dx=1, dy=0, ksize=-1)
47 gradX = np.absolute(gradX)
48 (minVal, maxVal) = (np.min(gradX), np.max(gradX))
49 gradX = 255 * ((gradX - minVal) / (maxVal - minVal))
50 gradX = gradX.astype("uint8")
51 self.debug_imshow("Scharr", gradX)
52
```

Scharr gradient akan mendeteksi bagian pinggir dalam gambar dan menekankan perbatasan tiap karakter yang ada di plat nomor. menggunakan cv2.Sobel, kita mendeskripsikan magnitude scharr gradient yang direpresentasikan dengan x-drection dari gambar proses blackhat tadi (line 45-46), lalu di scaling ulang dengan hasil yang kembali sesuai range [0,2550] (line 47-51).



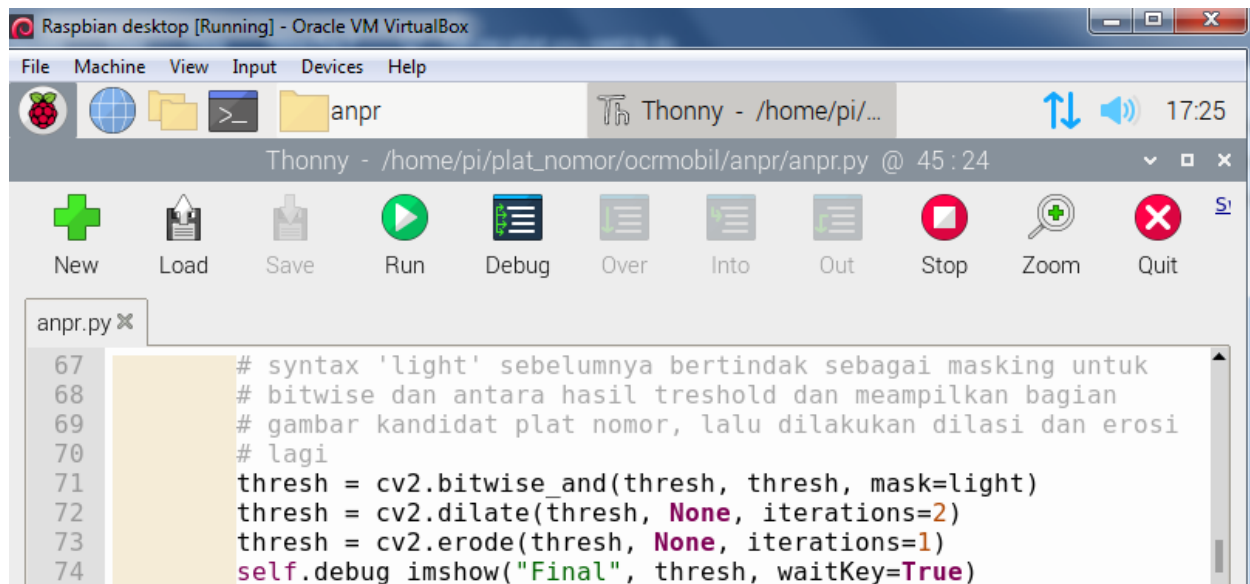
```
51 self.debug_imshow("Scharr", gradX)
52
53 # mengaplikasikan gaussian blur untuk gambar magnitudo gradien
54 # lalu mengaplikasikan threshold binary dengan metode otsu
55 gradX = cv2.GaussianBlur(gradX, (5, 5), 0)
56 gradX = cv2.morphologyEx(gradX, cv2.MORPH_CLOSE, rectKern)
57 thresh = cv2.threshold(gradX, 0, 255,
58                       cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]
59 self.debug_imshow("Grad Thresh", thresh)
```

Lalu akan dihaluskan bagian grup yang mungkin mengandung lisensi plat nomor. Disini akan diterapkan gaussian blur kepada gradient magnitude gambar (gradX) (line 55). Lalu akan dilakukan operasi penutupan (line 56) dan dilakukan binary threshold menggunakan metode otsu (line 57-58).



```
60
61     # melakukan erosi dan dilasi untuk membersihkan gambar yang
62     # di treshold
63     thresh = cv2.erode(thresh, None, iterations=2)
64     thresh = cv2.dilate(thresh, None, iterations=2)
65     self.debug_imshow("Grad Erode/Dilate", thresh)
```

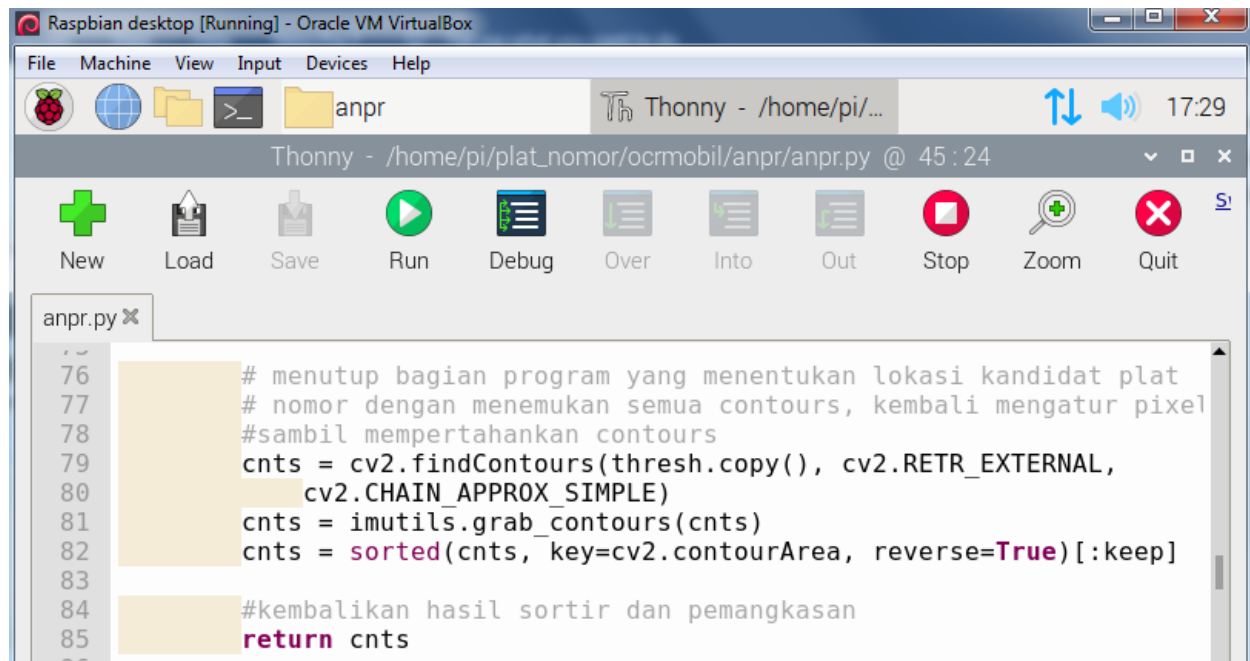
Pada awalnya hasilnya akan terlihat pecah tetapi bagian plat nomornya sedikit terlihat dan dapat didefinisikan. Proses selanjutnya akan dilakukan pembersihan noise. Di line 63 dan 64 akan dilakukan erosions dan dilations yang melakukan usaha denoise image yang sudah di treshold.



The screenshot shows the Thonny IDE interface within a Raspberry Pi desktop environment. The file explorer on the left shows the project structure: `anpr` and `anpr.py`. The main editor window displays the following Python code:

```
67 # syntax 'light' sebelumnya bertindak sebagai masking untuk
68 # bitwise dan antara hasil threshold dan menampilkan bagian
69 # gambar kandidat plat nomor, lalu dilakukan dilasi dan erosi
70 # lagi
71 thresh = cv2.bitwise_and(thresh, thresh, mask=light)
72 thresh = cv2.dilate(thresh, None, iterations=2)
73 thresh = cv2.erode(thresh, None, iterations=1)
74 self.debug_imshow("Final", thresh, waitKey=True)
```

Di line 71 sampai 74 light disini berperan sebagai masking hasil threshold dan bagian light itu sendiri akan dapat menampilkan kandidat plat nomor (line 71). Lalu akan dilakukan proses erosions dan dilations seperti proses sebelumnya.

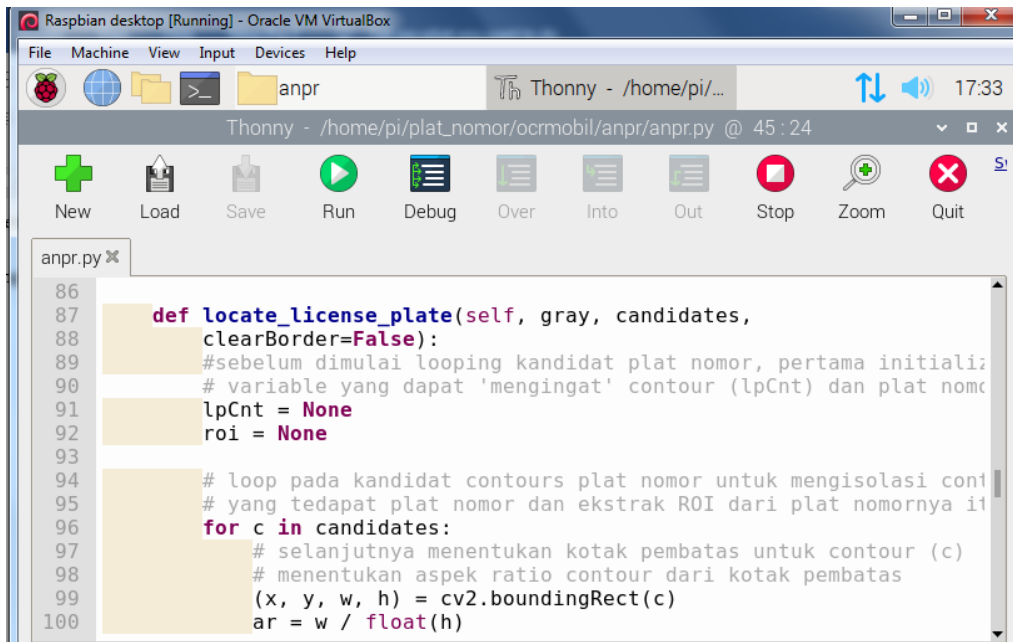


The screenshot shows the Thonny IDE interface with the same project structure. The main editor window displays the following Python code:

```
76 # menutup bagian program yang menentukan lokasi kandidat plat
77 # nomor dengan menemukan semua contours, kembali mengatur pixel
78 # sambil mempertahankan contours
79 cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
80                        cv2.CHAIN_APPROX_SIMPLE)
81 cnts = imutils.grab_contours(cnts)
82 cnts = sorted(cnts, key=cv2.contourArea, reverse=True)[:keep]
83
84 #kembalikan hasil sortir dan pemangkasan
85 return cnts
```

Untuk menutup metode bagian `locate_license_plate_candidates` akan dilakukan penemuan semua contours (line 79-81), reverse-sorting tergantung dimana area pixel terletak tergantung dengan jumlah contours, dan return hasil sorting ke dalam `cnts` (line 85).

D. Memangkas kandidat plat nomor



```
86
87     def locate_license_plate(self, gray, candidates,
88                             clearBorder=False):
89         #sebelum dimulai looping kandidat plat nomor, pertama initializ
90         # variable yang dapat 'mengingat' contour (lpCnt) dan plat nomo
91         lpCnt = None
92         roi = None
93
94         # loop pada kandidat contours plat nomor untuk mengisolasi conf
95         # yang terdapat plat nomor dan ekstrak ROI dari plat nomornya it
96         for c in candidates:
97             # selanjutnya menentukan kotak pembatas untuk contour (c)
98             # menentukan aspek ratio contour dari kotak pembatas
99             (x, y, w, h) = cv2.boundingRect(c)
100            ar = w / float(h)
```

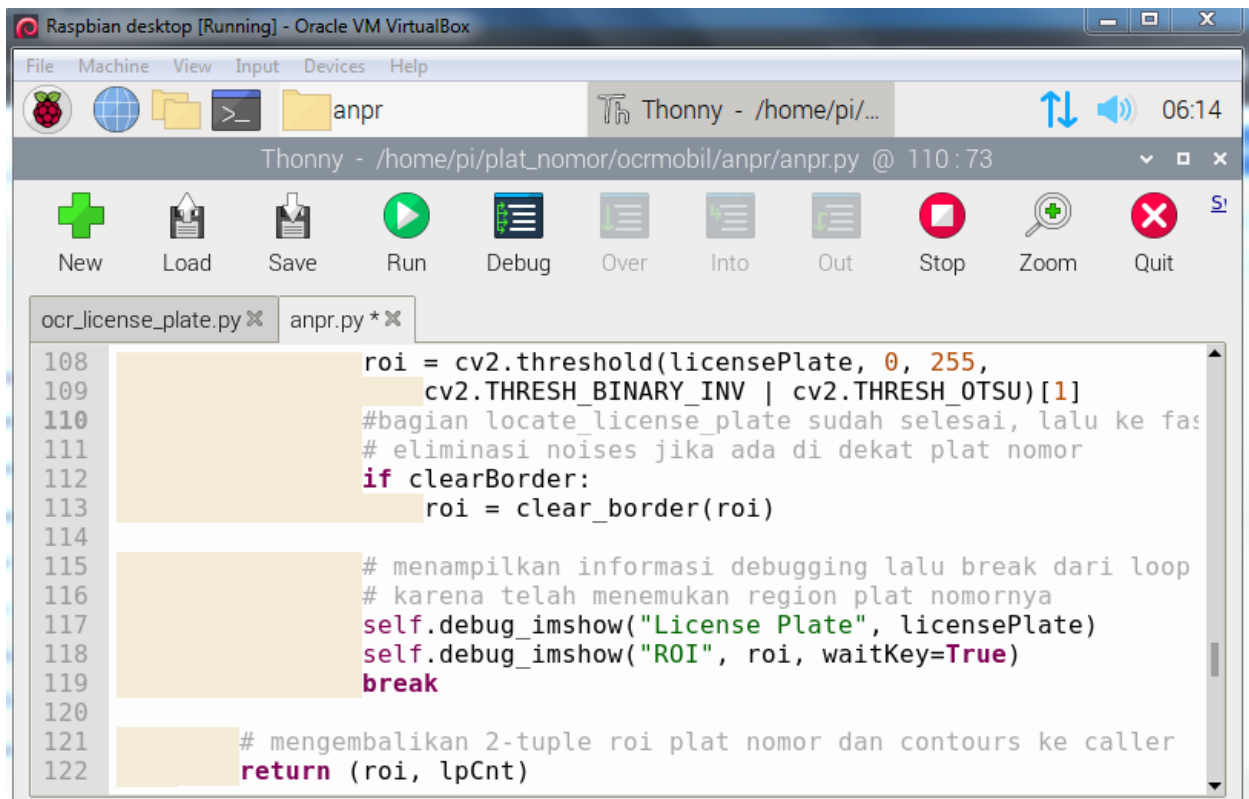
Terdapat 3 parameter yaitu :

- `gray` : input gambar grayscale
- `candidates` : contours kandidat plat nomor yang dikembalikan ke dalam method sebelumnya di class ini.
- `clearBorder` : sebuah boolean yang menandakan tiap proses pipeline untuk mengeliminasi contours yang menyentuh bagian gambar.

jika countours di kotak bounding ar tidak memenuhi ekspetasi plat nomor, sehingga tidak dilakukan pengerjaan lanjut. roi dan lpCnt akan tetap tidak memiliki nilai none, dan bagian ini terserah pada driver script mengatasi masalah ini. lalu aspek rasio besarnya diantara minAR dan maxAR. Dalam kasus ini telah dianggap memiliki bentuk garis (contours) plat nomor yang benar. Akhirnya akan dipopulasi lpCnt dan roi :

- lpCnt di set dalam contours tersebut, sebagai c (line 106)
- roi diekstrak lewat NumPy (line 107) dan dilakukan threshold lagi menggunakan metode otsu (line 108-109)

Disini bagian class locate_license_plate sudah selesai lalu dilanjut ke fase selanjutnya.



```
108         roi = cv2.threshold(licensePlate, 0, 255,  
109                             cv2.THRESH_BINARY_INV | cv2.THRESH_OTSU)[1]  
110         #bagian locate_license_plate sudah selesai, lalu ke fas  
111         # eliminasi noises jika ada di dekat plat nomor  
112         if clearBorder:  
113             roi = clear_border(roi)  
114  
115         # menampilkan informasi debugging lalu break dari loop  
116         # karena telah menemukan region plat nomornya  
117         self.debug_imshow("License Plate", licensePlate)  
118         self.debug_imshow("ROI", roi, waitKey=True)  
119         break  
120  
121         # mengembalikan 2-tuple roi plat nomor dan contours ke caller  
122         return (roi, lpCnt)
```

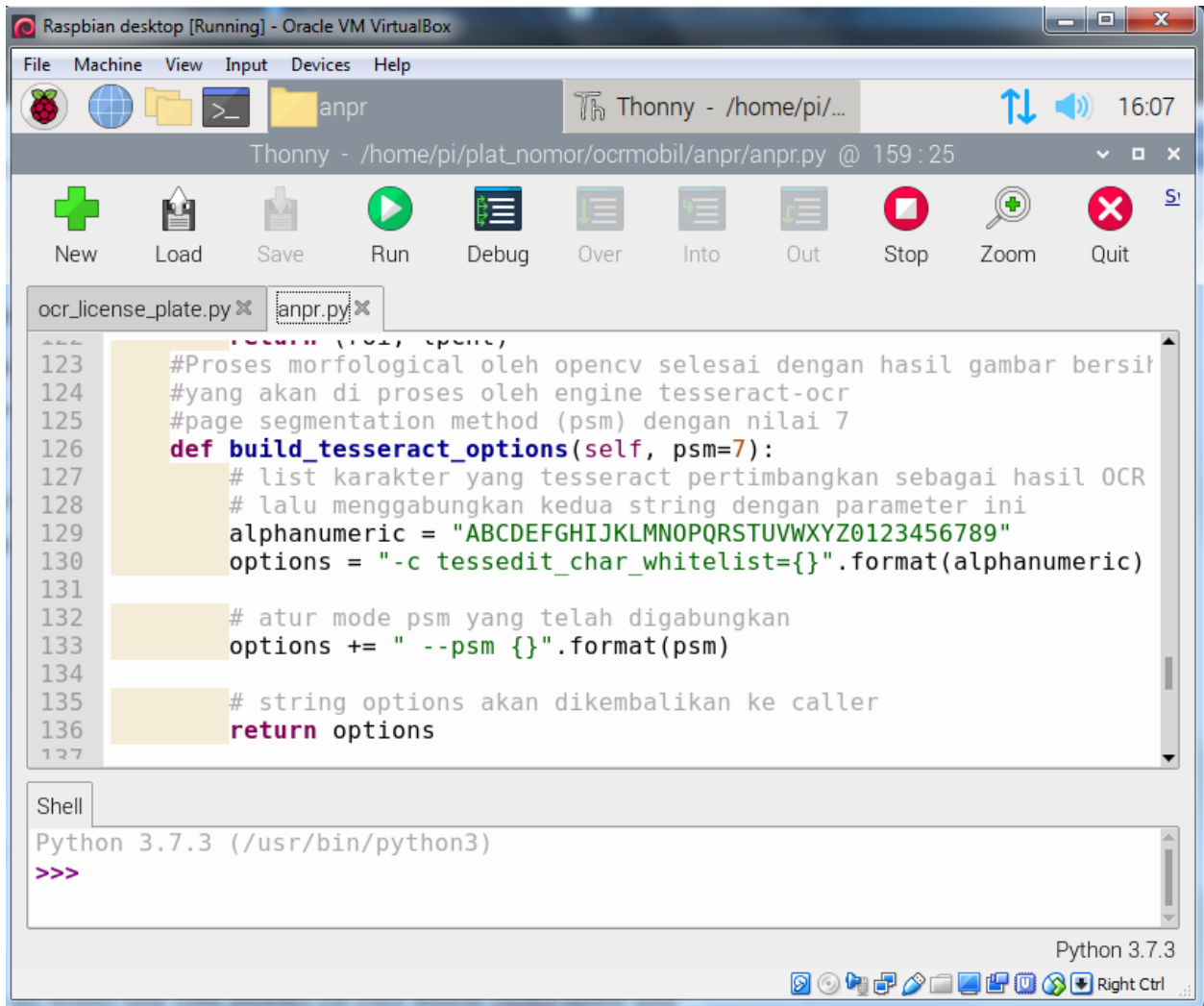
Jika `clearBorder` sudah ditentukan, lalu bisa dibersihkan pixel yang melewati ROI plat nomor (line 112-113). Hal ini akan membantu mengeliminasi noise yang bisa berpengaruh pada hasil Tesseract-OCR.

- license Plate : ROI pre-thresholding dan pembersihan border (line 116)
- roi : proses ROI final dari proses plat nomor ini (line 117)

fungsi `debug_imshow` disini menimpa fungsi `waitKey` sehingga memiliki nilai `True`, diberikan kesempatan pada user sehingga dapat menginspeksi proses sebelum menekan tombol untuk lanjut ke gambar selanjutnya. Setelah tombol itu ditekan, maka `break` dari loop akan dilakukan sehingga tidak lompat ke kandidat gambar lainnya. akhirnya akan dilakukan `return` yang memiliki ROI dan contours plat nomor yang dipanggil caller.

- E. Definisikan opsi tesseract untuk sistem pendeteksi plat nomor termasuk whistlist karakter untuk proses OCR dan page segmentation mode (PSM) untuk menambah akurasi

Di bagian ini kita berfokus pada konfigurasi mesin tesseract OCR dengan mendefinisikan `build_tesseract_options` method :



```
123     #Proses morfological oleh opencv selesai dengan hasil gambar bersih
124     #yang akan di proses oleh engine tesseract-ocr
125     #page segmentation method (psm) dengan nilai 7
126     def build_tesseract_options(self, psm=7):
127         # list karakter yang tesseract pertimbangkan sebagai hasil OCR
128         # lalu menggabungkan kedua string dengan parameter ini
129         alphanumeric = "ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
130         options = "-c tessedit_char_whitelist={}".format(alphanumeric)
131
132         # atur mode psm yang telah digabungkan
133         options += " --psm {}".format(psm)
134
135         # string options akan dikembalikan ke caller
136         return options
137
```

Shell

Python 3.7.3 (/usr/bin/python3)

>>>

Python 3.7.3

Tesseract dan python saling berkaitan, ada beberapa opsi konfigurasi di tesseract ini, yaitu

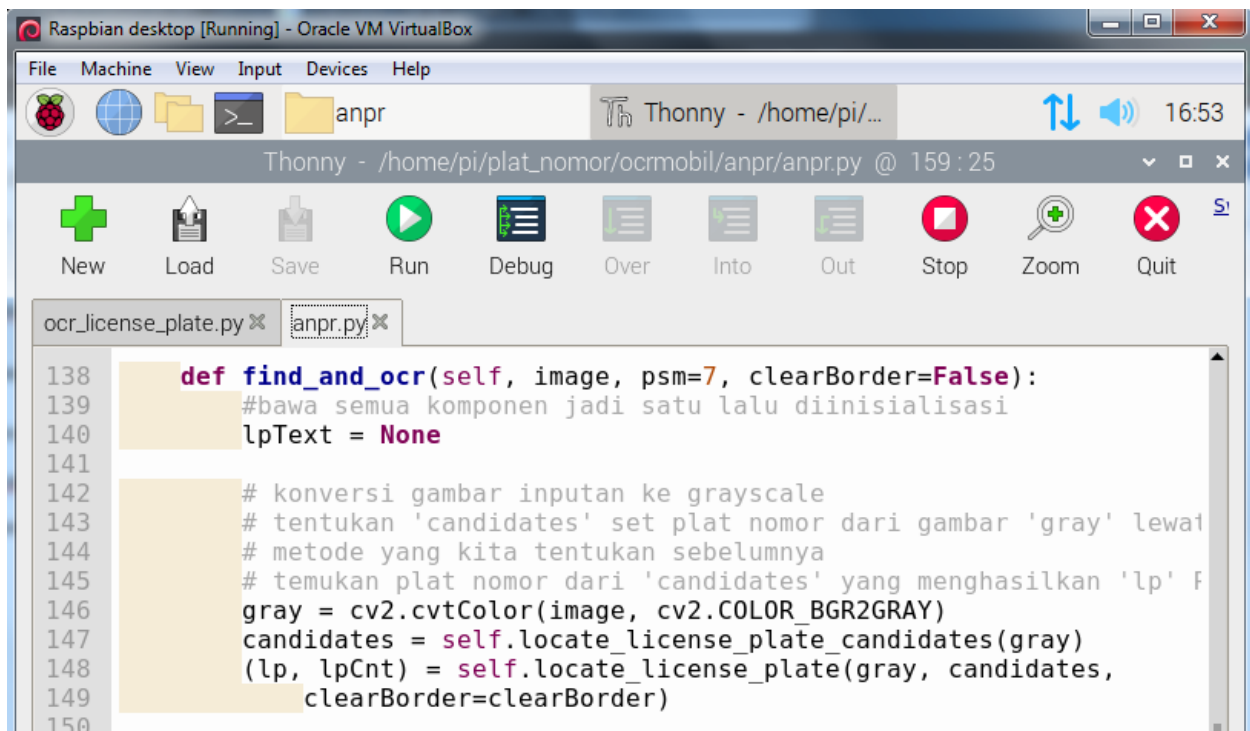
:

- Page segmentation method (PSM) : opsi ini memberikan nilai 7 yang artinya analisis tampilan dan pengolahan teksnya memperlakukan gambar sebagai satu baris teks.
- Whitelist : daftar karakter dari huruf, angka maupun simbol yang dipertimbangkan oleh tesseract yang di daftar di line 129.

Line 130-133 menggabungkan string yang sudah di format dengan parameter option yang ada di command line argumen tesseract, yang akhirnya akan di return di line 133.

F. Method utama dari class PlateRecognition_Engine

Merupakan metode usaha agar semua komponen menyatu dalam tempat satu kesatuan sehingga script drivenya bisa menginisialisasi object PlateRecognition_Engine, dan memakai hanya satu fungsi caller. Ini akan didefinisikan sebagai find_and_ocr :



```

138     def find_and_ocr(self, image, psm=7, clearBorder=False):
139         #bawa semua komponen jadi satu lalu diinisialisasi
140         lpText = None
141
142         # konversi gambar inputan ke grayscale
143         # tentukan 'candidates' set plat nomor dari gambar 'gray' lewat
144         # metode yang kita tentukan sebelumnya
145         # temukan plat nomor dari 'candidates' yang menghasilkan 'lp' f
146         gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
147         candidates = self.locate_license_plate_candidates(gray)
148         (lp, lpCnt) = self.locate_license_plate(gray, candidates,
149         clearBorder=clearBorder)
150
  
```

```

150
151     # asumsi bahwa plat nomor yang dipilih sudah cocok
152     if lp is not None:
153         # lakukan ocr pada plat nomor lewat 'image_to_string'
154         options = self.build_tesseract_options(psm=psm)
155         lpText = pytesseract.image_to_string(lp, config=options)
156         self.debug_imshow("License Plate", lp)
157
158     # kembalikan 2 tuple yang terdiri dari OCR 'lpText' dan 'lpCnt'
159     return (lpText, lpCnt)

```

Di method ini ada 3 parameter :

- image : gambar yang merupakan gambar mobil bagian depan atau belakang dengan bagian plat nomor.
- PSM : tesseract page segmentation mode.
- clearBorder : bagian yang menunjukkan yang membersihkan contours yang menyentuh batas ROI plat nomor

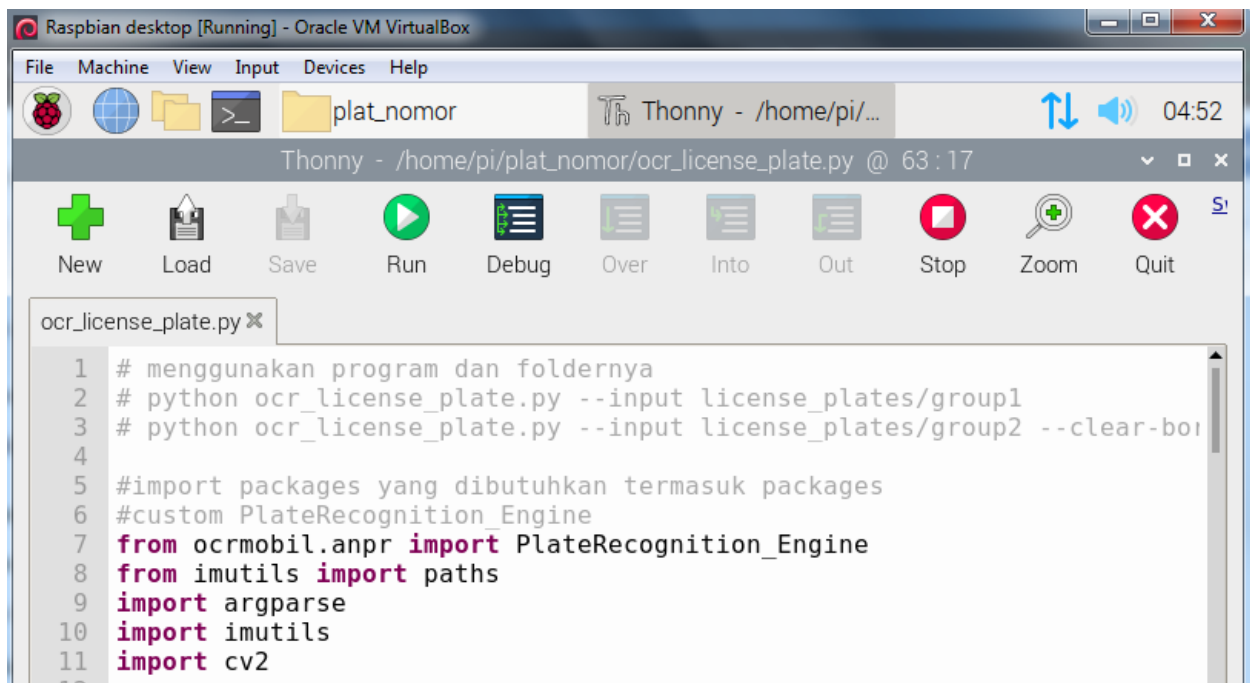
Dengan parameter-parameter ini kita telah mendapatkan :

- konversi input image ke grayscale (line 146)
- Menentukan bagian candidates plat nomor dari gambar gray dari method-method yang sebelumnya telah didefinisikan. (line 147)
- Menemukan plat nomor dari candidates yang menghasilkan lp ROI (line 148-149)
- setelah mendapatkan plat nomor yang cocok, akan dilakukan options pyTesseract dan melakukan OCR via image_to_string method. (line 152-155)
- akan dilakukan returns 2 bagian OCR yaitu lpText dan lpCnt. (line 159)

2. Membuat driver script plat nomor dengan openCV dan python

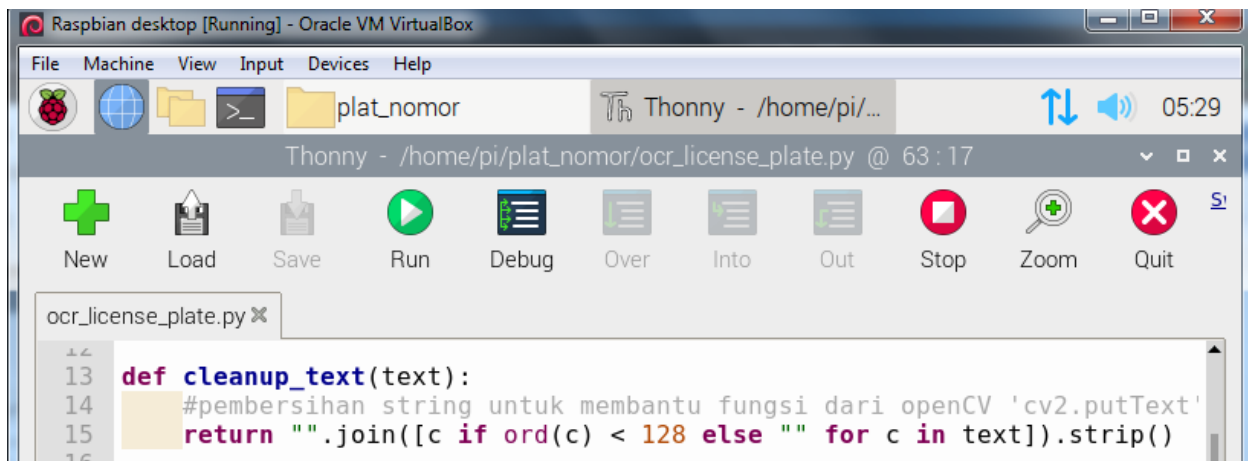
Setelah proses pembuatan mesin OCR PlateRecognition_Engine dan diimplementasikan, selanjutnya akan dibuat script driver python yang akan melakukan beberapa hal berikut :

- a. Memuat gambar dari simpanan disk
- b. menemukan plat nomor dari gambar yang di-input
- c. melakukan OCR pada plat nomor
- d. Menampilkan hasil ke dalam layar

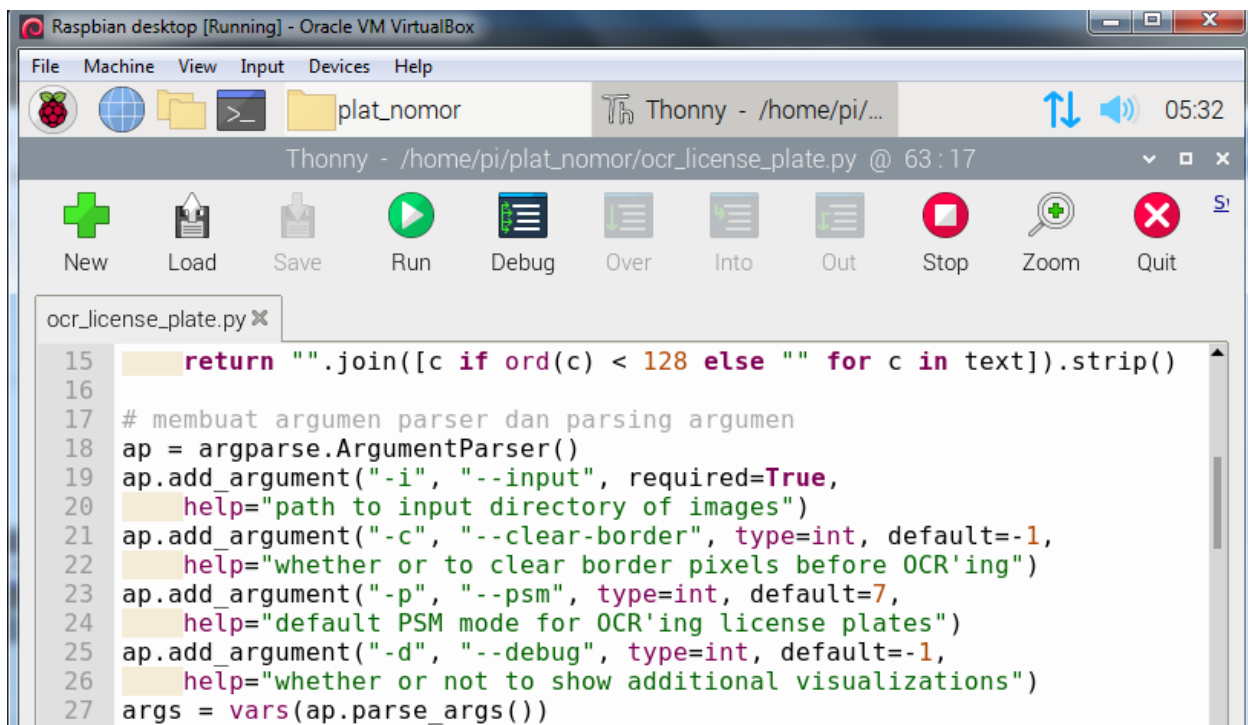


```
ocr_license_plate.py x
1 # menggunakan program dan foldernya
2 # python ocr_license_plate.py --input license_plates/group1
3 # python ocr_license_plate.py --input license_plates/group2 --clear-boi
4
5 #import packages yang dibutuhkan termasuk packages
6 #custom PlateRecognition_Engine
7 from ocrmobil.anpr import PlateRecognition_Engine
8 from imutils import paths
9 import argparse
10 import imutils
11 import cv2
12
```

Disini telah di import, class custom yang bernama PlateRecognition_Engine yang telah kita implementasi sebelumnya dan import beberapa package.



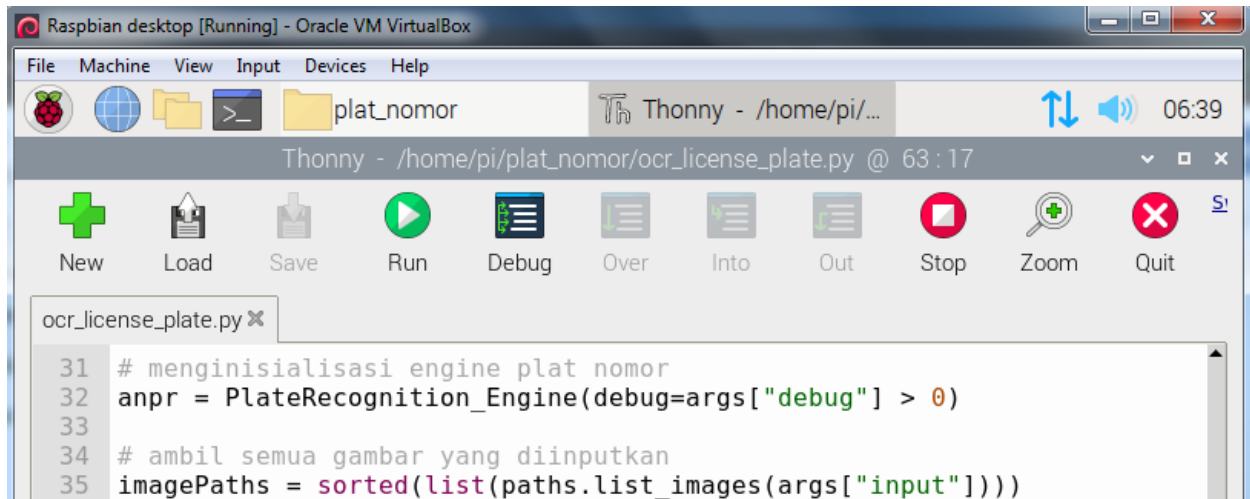
Sebelum lanjut akan dilakukan pembersihan string. `cleanup_text` hanya menerima teks dan menguraikan semua karakter non-alfanumerik.



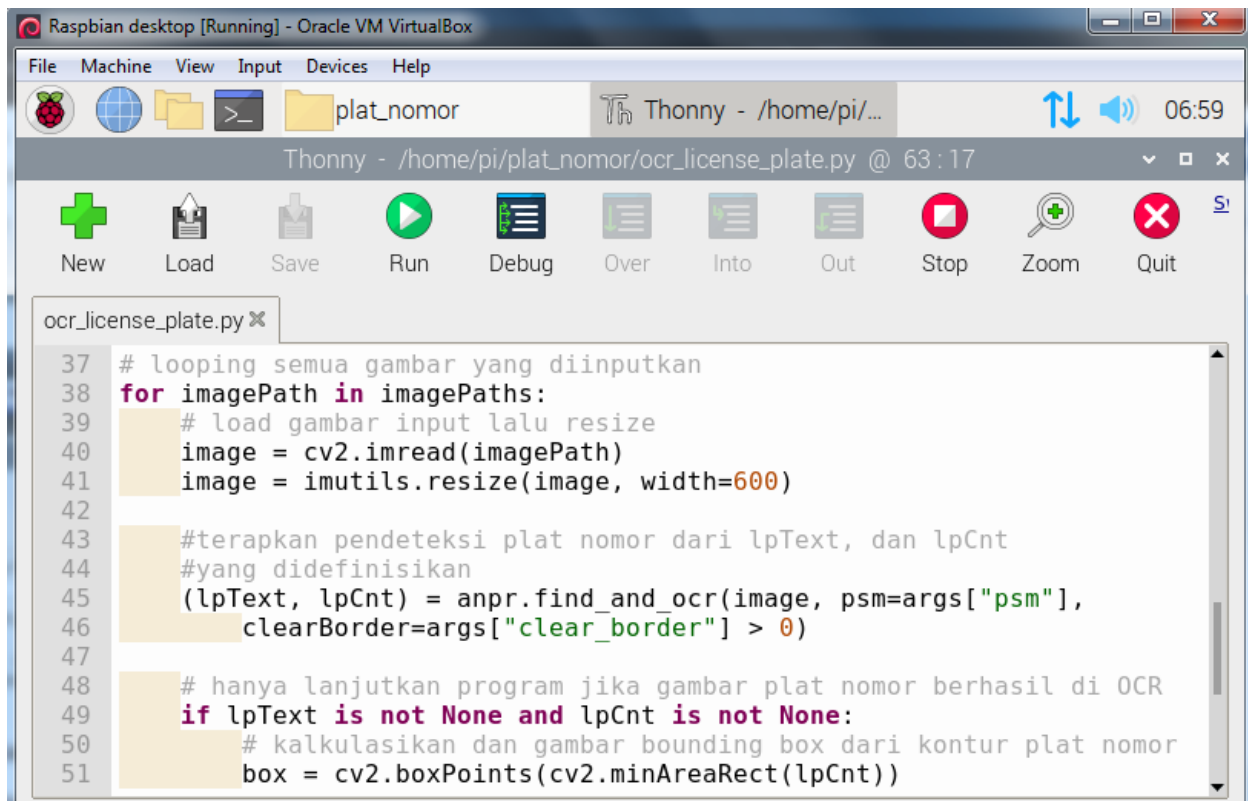
Argumen-argumen yang ada di sini, yaitu :

- `--input` : tempat direktori input yang berisi gambar-gambar kendaraan
- `--clear-border` : bagian yang menunjukkan apakah akan membersihkan tepi ROI pada plat nomor sebelum meneruskan gambar ke tesseract.
- `--psm` : Tesseract page segmentation mode dengan nilai 7, yang mengindikasikan bahwa tesseract hanya mencari satu baris teks.

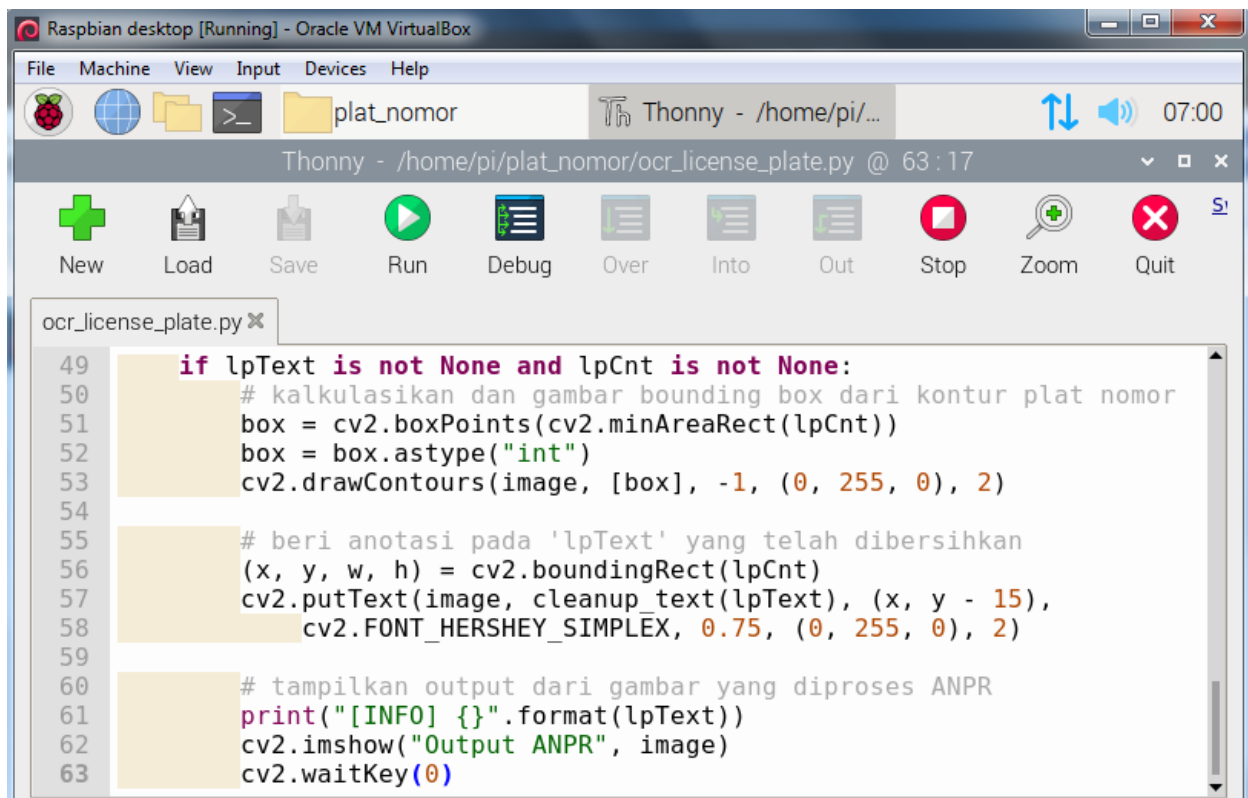
- --debug : sebuah boolean yang mengindikasikan bahwa menampilkan hasil gambar secara langsung tiap proses.



Setelah dilakukan import, pembersihan teks, dan memahami argumen dari command line. Saatnya melakukan pengenalan pada plat nomor. Pertama dilakukan inisialisasi PlateRecognition_Engine (line 32). Juga dilakukan membawa semua gambar input dengan menggunakan imutils yang bernama imagePaths.



```
ocr_license_plate.py x
37 # looping semua gambar yang diinputkan
38 for imagePath in imagePaths:
39     # load gambar input lalu resize
40     image = cv2.imread(imagePath)
41     image = imutils.resize(image, width=600)
42
43     #terapkan pendeteksi plat nomor dari lpText, dan lpCnt
44     #yang didefinisikan
45     (lpText, lpCnt) = anpr.find_and_ocr(image, psm=args["psm"],
46     clearBorder=args["clear_border"] > 0)
47
48     # hanya lanjutkan program jika gambar plat nomor berhasil di OCR
49     if lpText is not None and lpCnt is not None:
50         # kalkulasikan dan gambar bounding box dari kontur plat nomor
51         box = cv2.boxPoints(cv2.minAreaRect(lpCnt))
```



```
ocr_license_plate.py x
49     if lpText is not None and lpCnt is not None:
50         # kalkulasikan dan gambar bounding box dari kontur plat nomor
51         box = cv2.boxPoints(cv2.minAreaRect(lpCnt))
52         box = box.astype("int")
53         cv2.drawContours(image, [box], -1, (0, 255, 0), 2)
54
55         # beri anotasi pada 'lpText' yang telah dibersihkan
56         (x, y, w, h) = cv2.boundingRect(lpCnt)
57         cv2.putText(image, cleanup_text(lpText), (x, y - 15),
58         cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 255, 0), 2)
59
60         # tampilkan output dari gambar yang diproses ANPR
61         print("[INFO] {}".format(lpText))
62         cv2.imshow("Output ANPR", image)
63         cv2.waitKey(0)
```

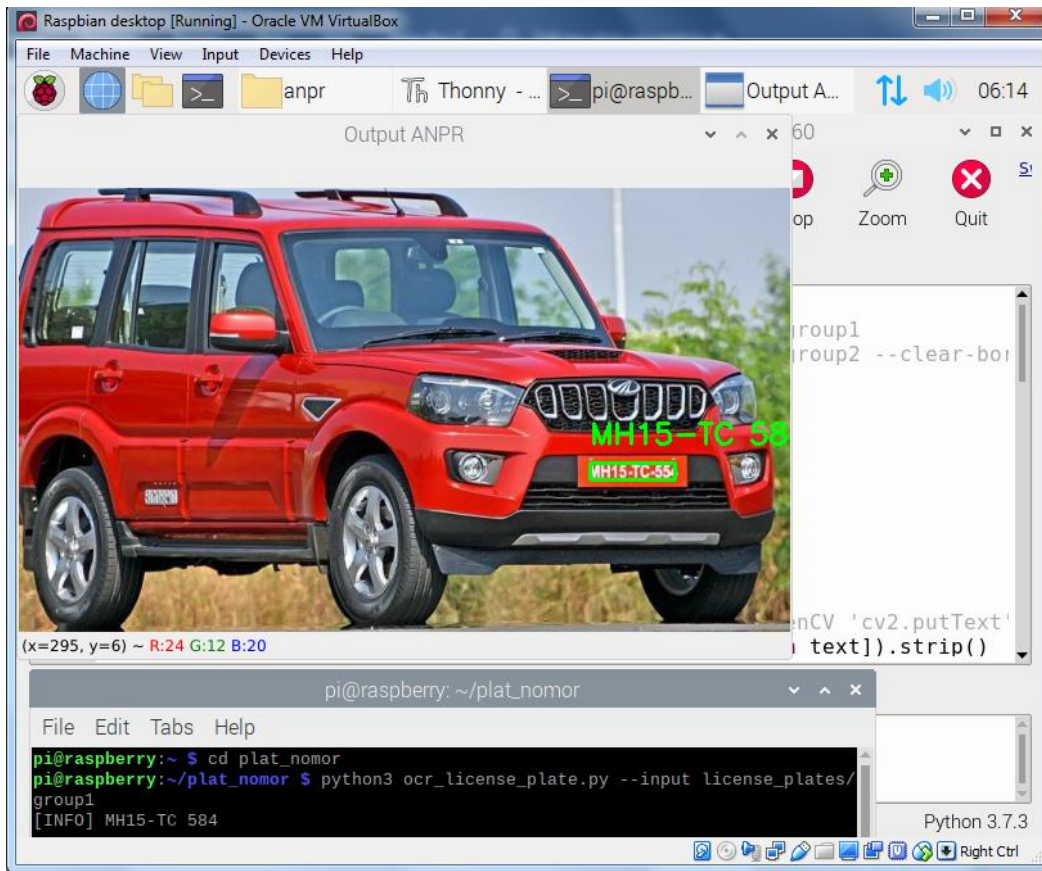
- Looping imagePaths untuk di muat dan di ubah ukuran gambar. (line 38-41)

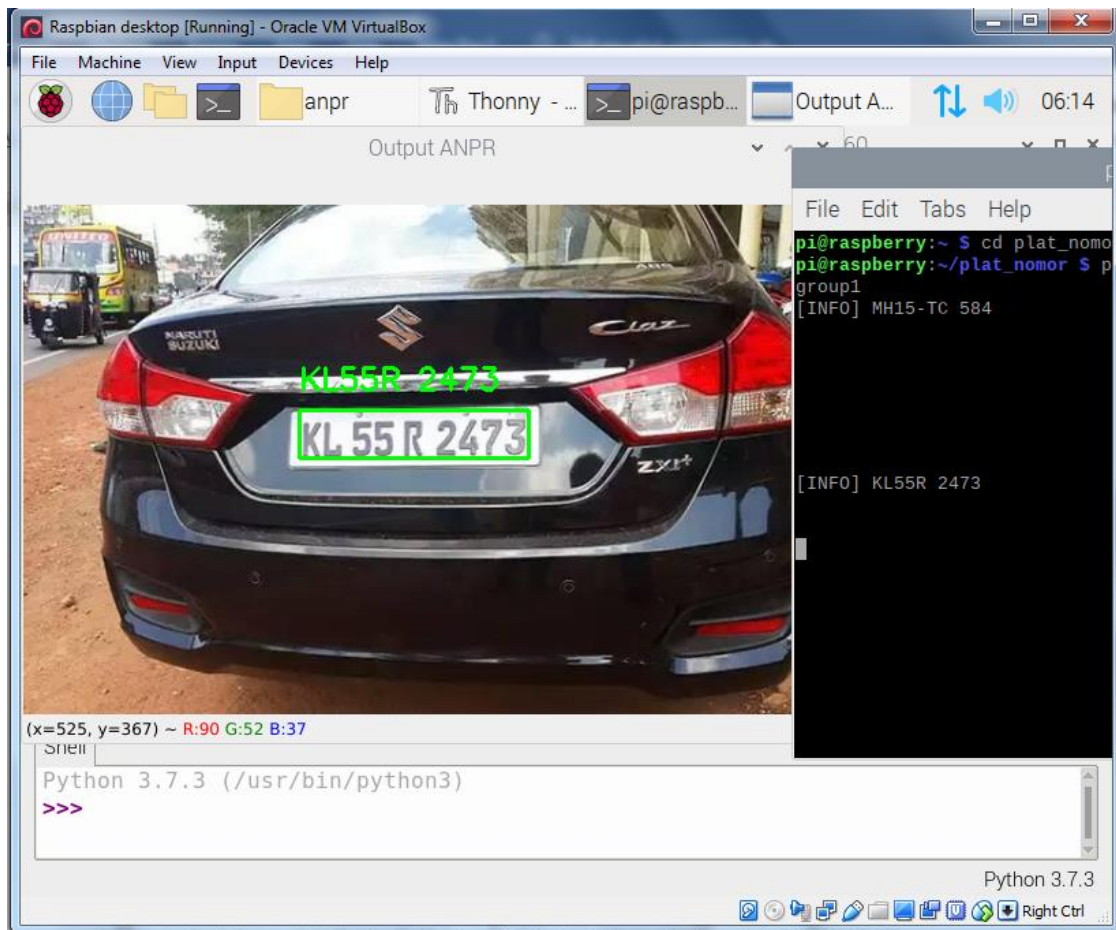
- sebuah calling kepada find_and_ocr method saat melewati image, --psm mode dan tanda -clear-border. (line 45-46)
- kalkulasi dan gambar kotak pembatas dari contours plat nomor. (line 51-53)
- lakukan cleanup pada string lpText. (line 56-58)
- Tampilkan string plat nomor pada terminal dan gambarnya sebagai program GUI window.

3. Hasil Program Pendeteksi Plat Nomor Dengan Metode Optical Character Recognition Pada Raspberry Pi

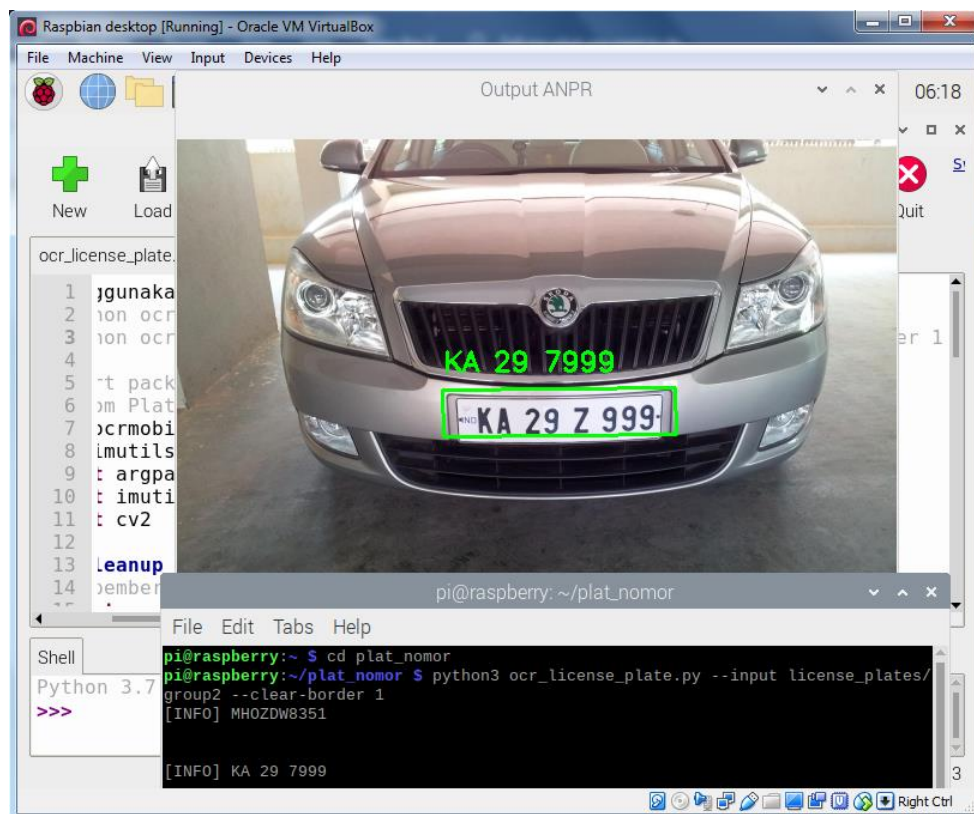
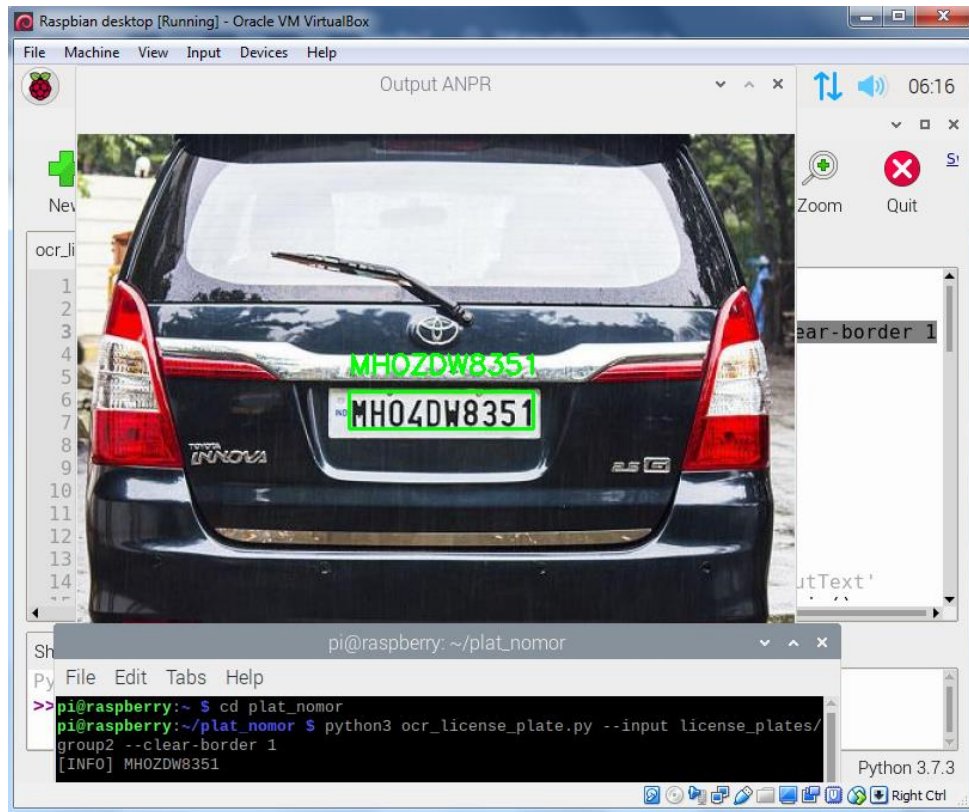
Jalankan program dengan membuka terminal dan eksekusi command ini untuk folder grup gambar tertentu :

- `python3 ocr_license_plate.py --input license_plates/group1`





- untuk folder kedua gunakan fungsi --clear-border untuk menghapus pixel yang menyentuh batas gambar yang membingungkan tesseract OCR :
python3 ocr_license_plate.py --input license_plates/group2 --clear-broder 1



REFERENCES

- Darmawan, D. (2020). *PROTOTYPE SISTEM PARKIR OTOMATIS DENGAN PROTOTYPE OF AUTOMATIC PARKING SYSTEM*.
- Humonggio, R., Abdullah, R. K., & Asri, M. (2019). Pengenalan Plat Nomor Menggunakan Image Processing Pada Perangkat Mikrokontroller. *Jurnal Teknologi Informasi Indonesia (JTII)*, 4(2), 63–70. <https://doi.org/10.30869/jtii.v4i2.400>
- Wakhidah, N. (2012). Deteksi Plat Nomor Kendaraan Bermotor Berdasarkan Area Pada Image Segmentation. *Jurnal Transformatika*, 9(2), 55. <https://doi.org/10.26623/transformatika.v9i2.58>
- Elektro, J. T., Sains, F., Teknologi, D. A. N., & Dharma, U. S. (2020). *Tugas Akhir Aplikasi Pengenalan Plat Nomor dengan Ekstraksi Ciri ICZ- ZCZ dan Similaritas Kosinus menggunakan Raspberry Pi untuk Sistem Kendali Gerbang Rumah Final Project Application for Number Plate Recognition with ICZ-ZCZ Feature Extraction and Cosine*.
- Gurav, C. (2019). *A Review Paper on Vehicle Number Plate Recognition*. 8(04), 282–286.
- Sugeng, W., Utoro, R. K., & Prabowo, M. T. (2020). Identifikasi Plat Nomor Kendaraan Dengan Metode Optical Character Recognition Menggunakan Raspberry Pi. *Jurnal Informatika*, 7(2), 116–125. <https://doi.org/10.31294/ji.v7i2.7997>
- Suharyanto, E. (2020). Pencarian Informasi Pajak Kendaraan Berdasarkan Plat Nomor Menggunakan Pustaka Tesseract dan OpenCV Python. *Jurnal Ilmu Komputer*, III(01), 14–17.

