Optimization Software:
www.balesio.com

EVALUASI KINERJA MINI PC DENGAN ALGORITMA ALPR STUDI KASUS :DETEKSI PLAT MOBIL

TUGAS AKHIR

Disusun dalam rangka memenuhi salah satu persyaratan

Untuk menyelesaikan program Strata-1 Departemen Teknik Informatika

Fakultas Teknik Universitas Hasanuddin

Makassar

Disusun Oleh:

NUR AZIZAH NOVITAMI WINARKO

D421 15 009

DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
MAKASSAR
2019

LEMBAR PENGESAHAN

"EVALUASI KINERJA MINI PC DENGAN ALGORITMA ALPR STUDI KASUS: DETEKSI PLAT MOBIL"

Disusun Oleh:

NUR AZIZAH NOVITAMI W. **D421 15 009**

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 30 Desember 2019. Diterima dan disahkan sebagai salah satu syarat memperoleh gelar Sarjana Teknik (S.T) pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Gowa, 30 Desember 2019

Disetujui Oleh:

Pembimbing I,

Adnan S.T., M.T., Ph.D.
NIP. 19740426 200501 1 002

Pembimbing II,

Dr. Eng. Intan Sari Areni, S.T., M.T.
NIP. 19750203 200012 2 002

Diterima dan disahkan oleh:

Ketua Departemen Teknik Informatika

Dr. Amil Ahmad Ilham, S.T., M.IT
NIP. 19731010 199802 1 001

Optimization Software:
www.balesio.com

Scanned by CamScanner

KATA PENGANTAR

Bismillahirrahmanirrahim.

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya sehingga tugas akhir yang berjudul " EVALUASI KINERJA MINI PC DENGAN ALGORITMA ALPR, STUDI KASUS : DETEKSI PLAT MOBIL" ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Penulis menyadari bahwa dalam penyusunan dan penulisan laporan tugas akhir ini tidak lepas dari bantuan, bimbingan serta dukungan dari berbagai pihak, dari masa perkuliahan sampai dengan masa penyusunan tugas akhir. Oleh karena itu, penulis dengan senang hati menyampaikan terima kasih kepada:

1. Kedua Orang tua penulis, Bapak Budi Winarko dan Ibu Helmi H Lateng, S.H yang selalu memberikan dukungan, doa, dan semangat serta selalu sabar dalam mendidik penulis sejak kecil;
2. Bapak Adnan S.T., M.T.,Ph.D., selaku pembimbing I dan Ibu Dr. Eng. Intan Sari Areni, S.T., M.T., selaku pembimbing II yang selalu menyediakan waktu, tenaga, pikiran dan perhatian yang luar biasa untuk mengarahkan penulis dalam penyusunan tugas akhir;
3. Bapak Dr. Amil Ahmad Ilham, ST., M.IT., selaku Ketua Departemen Teknik Informatika dan juga selaku Dosen Pembimbing Akademik penulis yang selalu membimbing dan menyediakan waktu, tenaga, dan perhatiannya

Optimization Software:
www.balesio.com

selama masa perkuliahan penulis di Fakultas Teknik Universitas Hasanuddin;

4. Ibu Elly Warni,S.T.,M.T. , dan Bapak Dr. Amil Ahmad Ilham, ST., M.IT., selaku Dosen Penguji yang telah memberikan saran dan masukan-masukan yang luar biasa dalam rangka perbaikan kualitas tugas akhir penulis;
5. Para sahabat, teman-teman dan kakak-kakak di Lab Riset IOTPC FT UH yang telah memberikan begitu banyak bantuan selama penelitian, pengambilan data dan diskusi *progress* penyusunan tugas akhir;
6. Nani, Dewi, Billa, Arief, William dan Isma yang senantiasa mendukung, memberikan masukan, dan nasehat dalam kehidupan perkuliahan selama ini;
7. Teman-teman Hypervisor FT UH atas dukungan dan semangat yang diberikan selama ini;
8. Segenap Staf dan Dosen Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu penulis.
9. Orang-orang berpengaruh lainnya yang tanpa sadar telah menjadi inspirasi penulis.

Akhir kata, penulis berharap semoga Allah SWT. berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga Tugas Akhir ini dapat memberikan manfaat bagi pengembangan ilmu. Aamiin.

Wassalam

Makassar, Desember 2019

Optimization Software:
www.balesio.com

ABSTRAK

Pajak Parkir adalah salah satu sumber Pendapatan Asli Daerah yang dibayarkan ke DISPENDA melalui pengelola parkir. Namun salah satu kendala yang dihadapi DISPENDA dalam pemungutan pajak parkir ini adalah DISPENDA tidak dapat memastikan apakah jumlah pajak yang disetor oleh pengelola sesuai dengan jumlah penerimaan yang sebenarnya. Hal ini dikarenakan data pembanding yang dimiliki tidak aktual dan masih menggunakan pengumpulan dengan sistem berbasis *counter* manual. Untuk itu dibutuhkan sebuah sistem otomatisasi *realtime* terhadap perhitungan tersebut. Dalam rangka membantu DISPENDA menghasilkan sistem yang efisien untuk diimplementasikan, penelitian ini mengevaluasi kinerja Mini PC terhadap algoritma ALPR (*Automatic License Plat Recognition)* yang telah ada. Kinerja yang diukur adalah: (1) rata-rata waktu eksekusi satu gambar hingga karakter plat dikenali, dengan skenario pengujian yaitu perubahan nilai kernel *GaussianBlur* 3x3, 5x5, 7x7 dan 11x11, (2) Kinerja CPU Mini PC dengan skenario pengujian yaitu diamati dalam keadaan startup dan dalam keadaan mengeksekusi program. Hasil yang diperoleh: nilai kernel dengan akurasi terbaik 83% yaitu nilai kernel 7x7, dan nilai kernel dengan rata-rata waktu eksekusi tercepatadalah nilai kernel 11x11 yaitu 23.44806444 detik. Hasil kinerja CPUpada keadaan *startup*yaitu sebesar 0.21% dan dalam keadaan mengeksekusi program sebesar 23.40% , artinya penggunaan CPU naik sebesar 23.19% dari keadaan startup.

Kata kunci: DISPENDA, ALPR, evaluasi kinerja, *mini* PC.

Optimization Software:
www.balesio.com

DAFTAR ISI

LEMBAR PENGESAHAN ii

KATA PENGANTAR iii

ABSTRAK v

DAFTAR ISI vi

DAFTAR GAMBAR viii

DAFTAR TABEL x

BAB I PENDAHULUAN 1

1.1. Latar Belakang 1
1.2 Rumusan Masalah 4
1.3 Tujuan Penelitian 4
1.4 Manfaat Penelitian 4
1.5 Batasan Masalah Penelitian 5
1.6 Sistematika Penulisan 5

BAB II TINJAUAN PUSTAKA 7

2.1 Plat Nomor Kendaraan 7
2.2 Raspberry Pi 8
2.3 OpenCV 10
2.4 Automatic License Plate Recognition (ALPR) 12
2.5 K-Nearest Neighbour 15
2.6 Pengolahan Citra 17
2.6.1 Citra Digital 17
2.6.2 Citra Grayscale 19
2.6.3 *Thresholding* 20
2.7 Konvolusi Citra 21
2.7.1 Kernel 22
2.8 *Gaussian Blur* 27
2.9 *Profilling* 30
2.9.1 CProfile 31
Mpstat 33
Penelitian Terkait 36
ETODOLOGI PENELITIAN 39

Optimization Software:
www.balesio.com

3.1 Tahapan Penelitian 39
3.2 Waktu dan Lokasi Penelitian 40
3.3 Instrumen Penelitian 40
3.4 Pengambilan Data 41
3.5 Perancangan Sistem 42
3.6 Evaluasi Kinerja 64
BAB IVHASIL DAN PEMBAHASAN 69
4.1 Hasil Pengujian Perubahan Nilai Kernel *Gaussian Blur* terhadap Akurasi Pengenalan Plat 69
4.2 Hasil Evaluasi Kinerja MiniPC dengan Parameter System Time, User Time dan Idle Time 73
BAB VPENUTUP 77
5.1 Kesimpulan 77
5.2 Saran 78
DAFTAR PUSTAKA 79
LAMPIRAN 1 : Hasil Deteksi dengan Nilai Kernel *Gaussian Blur* 11x11.....Lamp.1-1
LAMPIRAN 2 : Hasil Deteksi dengan Nilai Kernel *Gaussian Blur* 9x9.....Lamp.2-1
LAMPIRAN 3 : Hasil Deteksi dengan Nilai Kernel *Gaussian Blur* 7x7.....Lamp.3-1
LAMPIRAN 4 : Hasil Deteksi dengan Nilai Kernel *Gaussian Blur* 5x5.....Lamp.4-1
LAMPIRAN 5 : Hasil Deteksi dengan Nilai Kernel *Gaussian Blur* 3x3.....Lamp.5-1

Optimization Software:
www.balesio.com

DAFTAR GAMBAR

Gambar 2. 1 Struktur Raspberry Pi 10
Gambar 2. 2 Tahapan ALPR 13
Gambar 2. 3 Contoh Kernel 2x2 dan 3x3 23
Gambar 2. 4 Ilustrasi Konvolusi 23
Gambar 2. 5 Konvolusi Piksel Pinggir 27
Gambar 2. 6 Contoh Penerapan Konvolusi Matriks Gambar Asli dan Kernel 29
Gambar 2. 7Contoh Pemanggilan cProfile dan Output yang Dihasilkan 31
Gambar 2. 8 Contoh Pemanggilan Mpstat dan Output yang dihasilkan 33
Gambar 2. 9 Contoh Pemanggilan Mpstat dengan Interval 34
Gambar 3. 1Diagram Tahapan Penelitian 38
Gambar 3. 2Ilustrasi Pengambilan Data Objek Penelitian 41
Gambar 3. 3 Flowchart Sistem 42
Gambar 3. 4Citra Latih 43
Gambar 3. 5Contoh Citra Uji 44
Gambar 3. 6Hasil Tahap *Grayscale* 46
Gambar 3. 7Hasil Tahap *Gaussian Blur* 48
Gambar 3. 8Perbedaan Hasil Metode *Threshold* 50
Gambar 3. 9Perbedaan Hasil 2 Jenis*Threshold* 51
Gambar 3. 10Hasil Segmentasi Karakter 52
Gambar 3. 11Flowchart Tahap Lokalisasi Plat 53
Gambar 3. 12Hasil Kontur Deteksi Plat 54
Gambar 3. 13Flowchart Tahap Pengenalan Karakter 56
Gambar 3. 14Contoh Hasil Pencarian Semua Kontur 56
Gambar 3. 15Contoh Kontur yang Terlihat Seperti Karakter 57
Gambar 3. 16Contoh Hasil Penghapusan Karakter yang *Overlap* 58
Gambar 3. 17Contoh Kandidat Karakter dengan *Bounding Box* 58
Gambar 3. 18 Pengurutan list Karakter untuk Setiap Kandidat Plat 59
Gambar 3. 19 Contoh Hasil Deteksi yang Salah 60
Gambar 3. 20 Hasil Penerapan Filter Tambahan 61
. 21 Struktur Tabel deteksi_plat 61
. 22 Tampilan Database Sebelum dan Setelah Diolah 63
. 1 Grafik kernel terhadap jumlah citra yang berhasil dikenali 69

Optimization Software:
www.balesio.com

Gambar 4. 2 Grafik kernel terhadap rata-rata waktu eksekusi 69
Gambar 4. 3 Hasil *Profiling* Eksekusi Program dengan Kernel 3x3 71
Gambar 4. 4 Hasil *Profiling* Eksekusi Program dengan Kernel 11x11 71
Gambar 4. 5 Hasil Rata-rata Kinerja Mini PC dalam Keadaan Menjalankan *Startup* Program 73
Gambar 4. 6 Hasil Rata-rata Kinerja Mini PC dalam Keadaan Mengeksekusi Program ALPR 73

Optimization Software:
www.balesio.com

DAFTAR TABEL

Tabel 2. 1 Spesifikasi Raspberry Pi 3 Model B+ 9
Tabel 4. 1 Hasil Perbandingan Akurasi dan Waktu Eksekusi……..….68

Optimization Software:
www.balesio.com

BAB I

PENDAHULUAN

1.1. Latar Belakang

Parkir merupakan salah satu hal krusial dalam tata perkotaan. Keberadaan sistem parkir akan mempermudah pengaturan kendaraan-kendaraan yang masuk ke dalam ruang lingkup area tertentu seperti mall, perkantoran dan area kampus. Dengan adanya sistem parkir juga, bisa diperoleh data-data penting seperti jumlah total kendaraan yang berkunjung, lama waktu parkir, dan *manifest* data lainnya yang nantinya bisa dimanfaatkan untuk hal analisis data.

Salah satu hal menarik jika berbicara mengenai parkir adalah pajak parkir itu sendiri. Berdasarkan arsip data dari PD Parkir Makassar Raya, penerimaan pajak parkir tiap tahunnya menunjukkan penetapan target yang hampir selalu lebih tinggi dari realisasi penerimaan tahun sebelumnya bisa dikatakan berhasil. Fenomena tersebut mengindikasikan bahwa pajak sektor parkir menunjukkan progress yang baik dalam tampilan statistiknya. Namun apabila dibenturkan dengan fakta lapangan mengenai kondisi kepadatan penduduk Kota Makassar, kondisi jalan raya protokol yang hampir selalu terlihat *high traffic* di jam berangkat dan pulang kerja, dan psikologi budaya konsumerisme masyarakatnya yang menganjurkan untuk beraktualisasi diri dengan mengunjungi pusat-pusat keramaian. Maka menjadi lumrah untuk mempertanyakan nominal angka penerimaan pajak parkir yang diperoleh. Apakah benar jumlah-jumlah penerimaan yang diperoleh telah menggambarkan potensi sebenarnya dari pajak parkir,

Optimization Software:
www.balesio.com

mengingat jumlah titik-titik parkir yang cukup banyak tersebar di Kota Makassar.[1]

Salah satu sumber pendapatan pajak parkir yang cukup besar adalah pajak parkir Mall [1] .Sistem parkir Mall di kota Makassar di kelola oleh pihak swasta dimana pajak parkirnya wajib dibayarkan dan dilaporkan kepada DISPENDA (Dinas Pendapatan Daerah).Dalam wawancara penulis dengan Kepala Bidang DISPENDA kota Makassar pada Februari 2019, Bapak Dahyar menjelaskan, sebagai data pembanding, DISPENDA harus menghitung sendiri potensi parkir Mall tersebut dengan menerjunkan orang-orang dari Laskar Peduli Pajak (LPP) secara bergantian untuk menghitung manual jumlah kendaraan yang masuk dan keluar dalam sehari. Hal inilah yang menjadi kendala DISPENDA, karena perhitungan potensi parkir seperti itu membutuhkan banyak energi dan sumber daya manusia, serta data yang dihasilkan tidaklah aktual. Untuk itu dibutuhkan sebuah model sistem otomatisasi yang aktual (*realtime)* untuk membantu perhitungan potensi parkir tersebut.

Dalam mengimplementasikan sistem sesuai kebutuhan DISPENDA tersebut, dapat menggunakan sistem yang sama yang digunakan pada parkir Mall pada umumnya, yaitu dengan mengambil data plat kendaraan yang masuk dan keluar, kemudian membandingkan waktunya untuk mengetahui lama parkir kendaraan tersebut. Akumulasi jumlah kendaraan yang parkir, lama waktu parkir dan biaya parkirnya akan dikirimkan kepada DISPENDA secara *realtime.* Hanya keseluruhan sistem ini (mulai dari deteksi hingga pengiriman data) haruslah secara otomatis tanpa bantuan manusia.

Optimization Software:
www.balesio.com

Dari segi metode dan algoritma, implementasi deteksi otomatis plat kendaraan sudah banyak dikerjakan di Indonesia, salah satunya adalah denganalgoritma Automatic License Plate Recognition (ALPR) . Algoritma ALPR terdiri atas tiga buah tahapan, yaitu *plate localization, character segmentation* serta *character recognition*. Algoritma ALPR akan bekerja untuk mengenali plat kendaraan tersebut dan menggunakannya untuk menghitung lama parkir kendaraan tersebut.

Namun dalam mengimplementasikan sistem sesuai kebutuhan DISPENDA, yang harus dipikirkan bukan hanya keefektifan *software* atau akurasi dari program deteksi plat, tetapi juga keefektifan *hardware* yang akan digunakan (biaya, waktu maupun proses pengiriman data). Karena tantangan yang sering dihadapi adalah bagaimana alat yang dipasang tidak mencolok, hemat daya tetapi proses pengiriman data dapat berjalan lancar dan *realtime*, sistem menggunakan Mini PC bisa menjadi solusinya. Dengan memanfaatkan Mini PC sebagai pengontrol jarak jauh melalui bahasa pemrograman tertentu menjadikan sistem lebih efisien dalam segi ukuran dan daya yang digunakan [2]. Selain itu dengan bantuan jaringan internet, proses *monitoring* bisa berjalan secara *realtime*.

Karena itu, penelitian ini bertujuan untuk mengevaluasi kinerja Mini PC terhadap algoritma deteksi plat agar sistem yang dihasilkan untuk DISPENDA memang efisien sebelum diimplementasikan nantinya.

Optimization Software:
www.balesio.com

1.2 Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah pada tugas akhir ini adalah:

1. Berapa rata-rata waktu yang dibutuhkan Mini PC dalam memproses algoritma ALPR untuk menghasilkan data plat yang diinginkan?
2. Bagaimana kinerja CPU Mini PC dalam memproses algoritma ALPR?

1.3 Tujuan Penelitian

Tujuan dari tugas akhir ini adalah :

1. Untuk mengetahui rata-rata waktu yang dibutuhkan *Mini PC* dalam memproses algoritma ALPR untuk menghasilkan data plat yang diinginkan.
2. Untuk mengetahui kinerja CPU Mini PC dalam memproses algoritma ALPR.

1.4 Manfaat Penelitian

Manfaat dari tugas akhir ini adalah :

1. Bagi DISPENDA, dapat digunakan sebagai data pembanding yang aktual dan efisien.
2. Bagi Peneliti, dapat digunakan untuk menambah pengetahuan dan kemampuan di bidang *Internet of Things* dan Pengolahan Citra.
3. agi Institusi pendidikan, dapat digunakan sebagai referensi dalam engembangan penelitian topik terkait.

Optimization Software:
www.balesio.com

1.5 Batasan Masalah Penelitian

Yang menjadi batasan masalah dalam tugas akhir ini adalah :

1. Mini PC yang digunakan adalah Raspberry pi 3 Model B+.
2. Objek penelitian hanya berupa mobil.
3. Plat mobil yang dideteksi hanya plat berwarna hitam.
4. Simulasi dan pengambilan data dilakukan skala kampus Teknik Unhas, Gowa.
5. Tidak berfokus pada peningkatan akurasi, tapi untuk mengecek evaluasi kinerja Mini PC.

1.6 Sistematika Penulisan

Untuk memberikan gambaran singkat mengenai isi tulisan secara keseluruhan, maka akan diuraikan beberapa tahapan dari penulisan secara sistematis, yaitu :

BAB I PENDAHULUAN

Bab ini menguraikan secara umum mengenai hal yang menyangkut latar belakang, perumusan masalah dan batasan masalah, tujuan, manfaat, dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini berisi teori-teori terkait hal-hal yang mendasari dan berhubungan engan penelitian, termasuk di dalamnya Raspberry pi, Visi Komputer, penCV, *Profiliing,* dan metode-metode yang digunakan dalam penelitian.

Optimization Software:
www.balesio.com

BAB III METODOLOGI PENELITIAN

Bab ini berisi tentang perencanaan dan proses penerapan algoritma dan metode-metode dalam pengolahan data, mulai dari preprocessing hingga evaluasi kinerja Mini PC.

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang hasil pengolahan data serta pembahasan yang disertai tabel hasil penelitian.

BAB V PENUTUP

Bab ini berisi tentang kesimpulan yang didapatkan berdasarkan hasil penelitian yang telah dilakukan serta saran-saran untuk pengembangan lebih lanjut.

Optimization Software:
www.balesio.com

BAB II

TINJAUAN PUSTAKA

2.1 Plat Nomor Kendaraan

Plat nomor kendaran atau biasa disebut juga plat nomor polisi, atau yang dalam bahasa Inggris disebut *vehicle registration plate*, merupakan plat logam yang terdapat di kendaraan dan digunakan untuk tujuan identifikasi. Plat nomor ini berisi karakteristik unik yang berbeda antara satu kendaraan dengan kendaraan yang lain.[3]

Karakteristik plat kendaraan berbeda antara negara satu dan lainnya. Indonesia memiliki karakteristik plat dengan format HH AAA HH atau HH AAAA HHH, dimana H merepresentasikan huruf dan A merepresentasikan angka 0-9. Awalan huruf pertama merepresentasikan area dimana kendaraan tersebut didaftarkan. Huruf ini diikuti dengan satu sampai empat angka kemudian diakhiri oleh satu hingga tiga huruf[3]. Misalnya, DD 2173 AK merupakan kendaraan dari daerah Makassar, karena dua huruf pertamanya, DD, merupakan kode untuk Makassar.

Terdapat beberapa kelompok plat nomor yang dibedakan melalui warnanya, yaitu :

1. Warna putih dengan latar hitam, artinya kendaraan pribadi.
2. Warna merah pada latar putih, artinya kendaraan yang belum terdaftaratau kendaraan baru yang belum ada pemiliknya.
3. Warna hitam pada latar kuning, artinya kendaraan untuk transportasi umum seperti bus, taksi, dan angkutan kota.

Optimization Software:
www.balesio.com

4. Warna putih pada latar merah, artinya kendaraan milik institusi pemerintahaan.
5. Sedangkan Militer, Polisi dan Pemadam Kebakaran memiliki warna tersendiri dan biasanya terdapat lambing pangkat pemilik kendaraan tersebut.

2.2 Raspberry Pi

Raspberry Pi adalah sebuah komputer papan tunggal (*single-board computer*) atau SBC berukuran kecil. Raspberry Pi telah dilengkapi dengan semua fungsi layaknya sebuah komputer lengkap, menggunakan sytem on a chip (SoC) dari Broadcom BCM2835 hingga BCM2837 (Raspberry Pi 3), juga sudah termasuk CPU ARM1176JZF-S MHz bahkan 1.2GHz 64-bit quad-core ARMv8 CPU untuk Raspberry Pi 3, GPU VideoCore IV dan kapasitas RAM hingga 1 GB serta perangkat ini menggunakan kartu SD untuk booting dan penyimpanan jangka panjang.[4].

2.2.1 Raspberry Pi 3 Model B+

Raspberry Pi 3 adalah generasi ketiga dari Raspberry Pi, menggantikan Raspberry Pi 2 Model B pada Februari 2016. Raspberry Pi3 memiliki bentukyang identik dengan Raspberry Pi 2 sebelumnya (dan Pi 1 Model B +) dan memiliki kompatibilitas lengkap dengan Raspberry Pi 1 dan 2. Pade aspberry pi 3 Model B+ pengontrol USB Ethernetnya menawarkan nektivitas gigabit dengan throughput maksimum secara teoritis yaitu 00Mb / s, karena penggunaannya pada channel USB tunggal. Pada

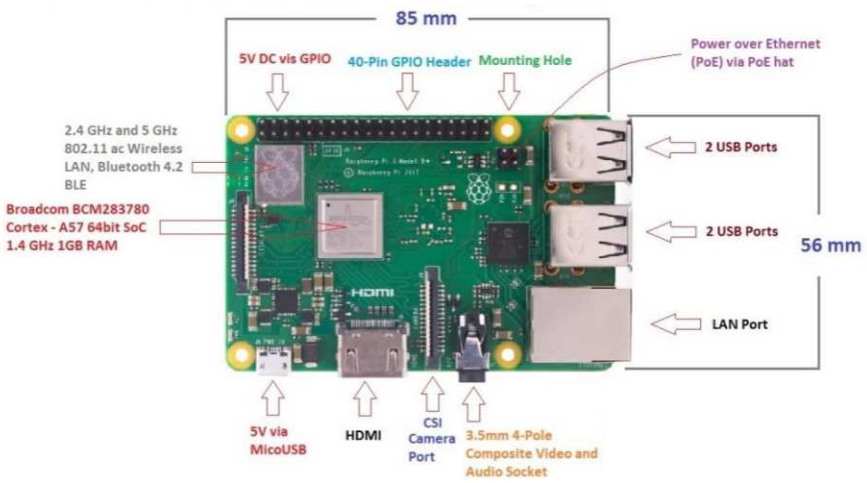
Optimization Software:
www.balesio.com

perangkat terbarunya ini Raspberry menambahkan fitur built- in wireless dan processor yang lebih bertenaga yang belum pernah dimiliki pada versi sebelumnya[4]. Spesifikasi dapat dilihat pada **tabel 2.1.**

Tabel 2.1 Spesifikasi Rasbperry Pi 3B+

Spesifikasi	Keterangan
SoC	Broadcom BCM2837B0 quad-core A53 (ARMv8) 64-bit @ 1.4GHz
GPU	Broadcom Videocore-IV
RAM	1GB LPDDR2 SDRAM
Networking	Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
Bluetooth	Bluetooth 4.2, Bluetooth Low Energy (BLE)
Storage	Micro-SD
GPIO	40-pin GPIO header, populated
Ports	HDMI, 3.5mm analogue audio-video jack, 4x USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
Dimensions	82mm x 56mm x 19.5mm, 50g
Power Input	5V/2.5A DC

Optimization Software:
www.balesio.com



Gambar 2.1 Struktur Raspberry Pi
https://www.theengineeringprojects.com

2.3 OpenCV

OpenCV (Open Computer Vision) adalah *library* dari fungsi pemrograman untuk *realtime* visi komputer [5]. OpenCV sendiri dirilis dalam lisensi BSD dan bebas digunakan untuk keperluan akademik maupun komersial. Produk ini mendukung interface C/C++, phyton dan java serta bisa berjalan diberbagai platform seperti Windows, Linux, Mac OS, iOS dan Android [6].

Computer Vision sendiri adalah salah satu cabang dari Bidang Ilmu Pengolahan Citra atau dikenal sebagai *Image Processing* yang memungkinkan komputer dapat melihat seperti manusia. Dengan *vision* tersebut komputer dapat mengambil keputusan, melakukan aksi, dan mengenali terhadap suatu objek. Beberapa pengembangan dari produk ini adalah *Face Recognition, Face* *, Face/Object Tracking, Road Tracking*, dan sebagainya.

Optimization Software:
www.balesio.com

OpenCV memiliki banyak fitur yang bisa dimanfaatkan dalam melakukan riset atau pekerjaanyang berhubungan dengan computer vision (*image processing,video processing* dan lain-lain) diantaranya :

1. Manipulation data citra (alokasi, *copying, setting*, konversi).
2. Citra dan video I/O (file dan *camera based input, image/video file output*).
3. Manipulasi Matriks dan Vektor beserta aljabar linear (*products, solvers, eigenvalues*, SVD).
4. Data struktur dinamis (*lists, queues, sets, trees, graphs*).
5. Pemroses Citra fundamental (*filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids*).
6. Analisis struktur(*connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation*).
7. Kalibrasi kamera (*calibration patterns, estimasi fundamental matrix, estimasi homography, stereo correspondence*).
8. Analisis gerakan (*optical flow, segmentation, tracking*).
9. Pengenalan obyek (*eigen-methods*, HMM).
10. Pelabelan citra (*line, conic, polygon, text drawing*).

dalam openCV terdapat 3 library utama yang bisa dipakai sesuai
n yakni :

Optimization Software:
www.balesio.com

1. CV : untuk algoritma Image processing dan Vision.
2. High gui :untuk GUI, Image dan Video I/O.
3. CXCORE : Untuk struktur data, support XML dan fungsi-fungsu grafis.

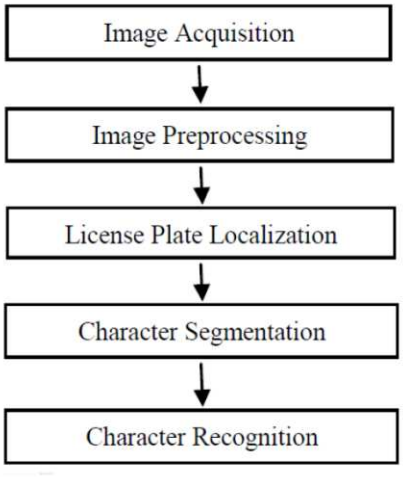
Selain itu OpenCV juga dilengkapi dengan *Machine Learning library* yang memiliki algoritma berikut :

1. *Naive Bayes classifier*.
2. KNN (*K-Nearest Neighbor Algorithm*).
3. *Support Vector Machine*.
4. *Decision Trees.*
5. *Random forest*.
6. *Neural Networks*.

2.4 Automatic License Plate Recognition (ALPR)

Automatic License Plate Recognition (ALPR) adalah teknik yang digunakan untuk mengekstraksi Plat Lisensi kendaraan dari suatu gambar. Secara khusus, algoritma ini digunakan bersama dengan berbagai sistem transportasi di area yang diaplikasikan, seperti penegakan hukum (misalnya penegakan batas kecepatan) dan penggunaan komersial seperti penegakan parkir dan pembayaran tol otomatis, pintu masuk pribadi dan publik, kontrol perbatasan, maupun pencurian dan kontrol vandalisme. ALPR telah dipelajari secara intensif di banyak negara. Karena perbedaan berbagai jenis plat yang digunakan, implementasi ini dapat berbeda untuk setiap negara. Namun umumnya,Sistem ALPR ri lima tahapan seperti pada **gambar 2.2** [7] :

Optimization Software:
www.balesio.com



Gambar 2.2 Tahapan ALPR [7]

- Image Acquisition

  Langkah pertama yang dilakukan adalah mengakuisisi gambar untuk mendapatkan citra kendaraan khususnya area plat kendaraan. Tahap ini dimulai dari objek yang akan diambil gambarnya dan difasilitasi oleh beberapa perangkat digitizer yang nantinya akan terjadi pencitraan, dimana pencitraan adalah kegiatan transformasi dari citra tampak (foto, gambar, lukisan, pemandangan dsb) menjadi citra digital, ini dapat dilakukan dengan sensor kamera atau dari penyimpanan basis data.

- Image Preprocessing

  Langkah kedua adalah meningkatkan gambar untuk ekstraksi plat nomor License Sebelum gambar diproses, gambar harus diproses terlebih dahulu ntuk menghilangkan *noise* tanpa mengurangi data yang diperlukan. erikut adalah beberapa langkah *preprocessing* yang sering digunakan, aitu:

Optimization Software:
www.balesio.com

1. Konversi menjadi *grayscale*

   Konversi *grayscale* mengubah nilai setiap piksel RGB (Merah Hijau Biru), menjadi satu nilai saluran pada setiap piksel. Proses-proses ini disarankan oleh banyak peneliti, untuk mengurangi ukuran gambar sehingga dapat mempercepat proses perhitungan.

2. Konversi skala abu-abu ke biner

   Beberapa peneliti menyarankan untuk mengubah gambar grayscale menjadi gambar biner menggunakan metode Thresholding. Tujuannya adalah untuk memisahkan Objek dengan latar belakangnya.

3. Mengubah ukuran gambar

   Tujuan konversi ini adalah untuk meminimalkan ukuran gambar untuk mempercepat proses perhitungan. Proses ini dilakukan oleh sebagian besar peneliti.

- Plate Localization

  Langkah ketiga adalah menemukan plat nomor dalam gambar. Sistem memindai area di mana ia akan menjadi kandidat plat nomor. Setiap kandidat telah diuji apakah itu plat nomor kendaraan atau tidak. Keberhasilan proses deteksi akan mempengaruhi proses selanjutnya, karena kesalahan pendeteksian lokasi plat maka sistem akan engekstraksi area yang bukan plat kendaraan.

Optimization Software:
www.balesio.com

- Character Segmentation

  Langkah keempat adalah untuk membagi setiap karakter yang ada di plat. Proses segmentasi ini bertujuan untuk memisahkan wilayah (region) karakter dengan wilayahlatarbelakangagarobjekdalam citramudahdianalisisdalamrangkamengenali karakter. Dengan demikian citra yang besar terdiri dari objek karakterdapat disegmentasi menjadimasing-masingkarakter.

- Character Recognition

  Langkah terakhir adalah mengenali setiap karakter yang tersegmentasi. Karakter yang telah diperoleh pada tahap ekstraksi akan dikenali menggunakan *machine learning*. Sistem akan dilatih menggunakan dataset yang berisi gambar pelatihan. Jumlah dan variasi pencitraan pelatihan yang digunakan akan menentukan keberhasilan dan akurasi sistem. Kemudian sistem diuji dengan data nyata.

2.5 K-Nearest Neighbour

K-Nearest Neighbor adalah suatu metode yang menggunakan algoritma supervised machine learning dimana hasil dari instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori k-tetangga terdekat. Algoritma *k-Nearest* menggunakan *Neighborhood Classification* sebagai nilai prediksi dari *nce* yang baru.[8]

Optimization Software:
www.balesio.com

Prinsip kerja *K-Nearest Neighbor* (KNN) adalah mencari jarak terdekat antara data yang akan dievaluasi dengan k tetangga (*neighbor*) terdekatnya dalam data pelatihan(training) . Dengan k merupakan banyaknya tetangga terdekat.

Data training diproyeksikan ke ruang berdimensi banyak, yang mana masing-masing dimensi menjelaskan fitur dari data. Ruang ini dibagi menjadi bagian-bagian berdasarkan klasifikasi data *training*[8].

Misalnya ada sebuah rumah yang berada tepat di tengah perbatasan antara Kota Makassar dan Kabupaten Maros, sehingga pemerintah kesulitan untuk menentukan apakah rumah tersebut termasuk kedalam wilayah Kota Makassar atau Kabupaten Maros.

Kasus ini bisa diselesaikan dengan menggunakan Algoritma K-NN, yaitu dengan melibatkan jarak antara rumah tersebut dengan rumah-rumah yang ada disekitarnya (tetangganya).

Pertama, ditentukan jumlah tetangga yg akan kita perhitungkan (k), misalnya ditentukan ada 3 tetangga terdekat (k = 3).

Kedua, dihitung jarak setiap tetangga terhadap rumah tersebut, lalu hasilnya diurutkan berdasarkan jarak, mulai dari yang terkecil ke yang terbesar.

Ketiga, diambil nilai 3 (k) tetangga yg paling dekat, lalu dilihat masing-masing dari tetangga tersebut apakah termasuk kedalam wilayah Kota atau Kabupaten. Ada 2 kemungkinan:

- Bila dari 3 tetangga tersebut terdapat ada 2 rumah yg termasuk kedalam wilayah Kota Bandung, maka rumah tersebut termasuk kedalam wilayah Kota Makassae.

Optimization Software:
www.balesio.com

- Sebaliknya, bila dari 3 tetangga tersebut terdapat 2 rumah yg termasuk kedalam wilayah Kabupaten Bandung, maka rumah tersebut termasuk kedalam wilayah Kabupaten Maros.

Dalam menentukan nilai k, bila jumlah klasifikasi adalah genap maka sebaiknya menggunakan nilai k ganjil, dan begitu pula sebaliknya bila jumlah klasifikasi adalah ganjil maka sebaiknya menggunakan nilai k genap, karena jika tidak seperti itu, ada kemungkinan jawaban tidak akan didapatkan.

2.6 Pengolahan Citra

Pengolahan citra adalah metode untuk melakukan beberapa operasi terhadap citra, dengan tujuanadalah untuk meningkatkan kualitas citra maupun untuk mengekstrak informasi yang penting dari citra tersebut. Di dalam mengolah sebuah citra, terdapat berbagai algoritma yang dapat diterapkan untuk menghasilkan keluaran yang lebih baik. Keluaran yang baik akan mempengaruhi hasil dari proses yang akan dilakukan selanjutnya.

2.6.1 Citra Digital

Citra merupakan kombinasi antara titik, garis, bidang, dan warna yang mewakili suatu objek atau benda. Sedangkan citra digital merupakan keluaran yang dihasilkan melalui perangkat pencitraan digital seperti kamera dan dapat disimpan oleh komputer digital.

Piksel merupakan suatu nilai dari irisan antara baris dan kolom ,y). Istilah piksel sering digunakan pada citra digital dimana bagian otak terkecil dari setiap piksel disebut sel. Nilai dari sebuah piksel

Optimization Software:
www.balesio.com

merupakan suatu nilai rata-rata yang sama untuk seluruh bagian dari sel tersebut.

Komponen lain disebut dengan resolusi citra yaitu tingkat detail suatu citra. Semakin tinggi resolusi citra maka akan semakin tinggi tingkat detail dari citra tersebut. Resolusi piksel merupakan perhitungan jumlah piksel dalam sebuah citra digital. Sebuah citra dengan tinggi N piksel dan lebar M piksel berarti memiliki resolusi sebesar M x N. Resolusi piksel akan memberikan dua buah angka integer yang secara berurutan akan memiliki jumlah piksel lebar dan jumlah piksel tinggi dari citra.

Citra digital mengandung elemen-elemen dasar. Elemen-elemen dasar yang paling penting diuraikan sebagai berikut[9]:

a. Kecerahan (*Brightness*) merupakan intensitas cahaya rata-rata dari suatu area yang melingkupinya.
b. Kontras (*Contrast*) merupakan sebaran terang (*lightness*) dan gelap (*darkness*) di dalam sebuah citra. Citra dengan kontras rendah komposisi citranya sebagian besar terang atau sebagian besar gelap. Citra dengan kontras yang baik, komposisi gelap dan terangnya, tersebar merata.
c. Kontur (*Contour*) merupakan keadaan yang ditimbulkan oleh perubahan intensitas pada *piksel-piksel* tetangga, sehingga dapat dideteksi tepi objek di dalam citra.
d. Warna (*Color*) merupakan persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya (λ) yang

Optimization Software:
www.balesio.com

dipantulkan oleh objek. Warna-warna yang dapat ditangkap oleh mata manusia merupakan kombinasi cahaya dengan panjang berbeda. Kombinasi yang memberikan rentang warna paling lebar adalah *red* (R), *green* (G) dan *blue* (B).

e. Tekstur (*Texture*) merupakan distribusi spasial dari derajat keabuan di dalam sekumpulan *piksel-piksel* yang bertetangga.

2.6.2 Citra Grayscale

Grayscale (skala keabuan) merupakan suatu citra yang memiliki warna putih, abu-abu, dan hitam. Jenis citra ini memilikiintensitas berkisar antara 0 sampai dengan 255. Nilai 0 dinyatakan sebagai hitam dan 255 dinyatakan sebagai putih. Nilai minimum atau maksimum dari suatu citra bergantung pada jumlah bitnya. Misalnya pada skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah 2^4 = 16 dan nilai maksimumnya adalah 2^4-1 = 15. Sedangkan untuk skala keabuan 8 bit, maka jumlah kemungkinan nilainya adalah 2^8=256 dimana nilai maksimumnya adalah 2^8-1 = 255[10].

Citra RGB dapat dikonversi menjadi citra *grayscale*dengan menggunakan tiga metode yaitu:

a. *Lightness Method*

Metode ini mengambil rerata pada intensitas warna yang paling mencolok dan paling tidak mencolok dengan persamaan sebagai berikut:

Optimization Software:
www.balesio.com

$$Grayscale = \frac{Max(R,G,B) + Min(R,G,B)}{2} \quad (2.1)$$

b. *Average Method*

Metode rerata (*average method*) merupakan metode yang paling sederhana. Nilai *piksel* dari tiap-tiap *channel* diambil dan dirata-ratakan. Sehingga persamaan dari metode ini yaitu:

$$Grayscale = \frac{R + G + B}{3} \quad (2.2)$$

c. *Luminosity Method*

Metode ini memberikan nilai beban-beban tertentu pada *channel-channel* tertentu sesuai dengan kebutuhan pengguna. Manusia lebih sensitif terhadap warna hijau dibandingkan warna-warna lainnya, jadi warna biru biasanya memiliki bobot yang paling besar. Adapun salah satu contoh persamaan dari metode ini adalah:

$$Grayscale = 0.21R + 0.72G + 0.07B \quad (2.3)$$

2.6.3 *Thresholding*

Thresholding merupakan salah satu teknik dasar dalam melakukan segmentasi citra. *Thresholding* digunakan untuk mengonversi citra *grayscale* menjadi citra biner dalam rangka untuk memisahkan beberapa objek target dari latar belakangnya. Citra biner hanya terdiri dari dua intensitas warna yaitu hitam yang memiliki representasi nilai 0 dan putih

Optimization Software:
www.balesio.com

yang memiliki representasi nilai 1. Sehingga jenis citra in hanya membutuhkan 1 bit memori untuk menyimpan kedua warna ini.

2.7 Konvolusi Citra

Konvolusi citra adalah teknik untuk menghaluskan suatu citra atau memperjelas citradengan menggantikan nilai piksel dengan sejumlah nilai piksel yang sesuai atau berdekatan dengan piksel aslinya. Tetapi dengan adanya konvolusi, ukuran dari citra tetap sama, tidak berubah[11].

Pada umumnya analisa suatu citra dalam domain frekuensi didasarkan pada teknik konvolusi. Operasi konvolusi akan dilakukan dengan menggeser kernel konvolusi piksel per piksel, menghitung piksel keluaran *f(i,j)*, lalu menyimpannya dalam matriks baru [12].Konvolusi menyediakan cara untuk menggabungkan dua *array*, biasanya untuk ukuran*array* yang berbeda, tetapi untuk dimensi *array* yang sama, akan menghasilkan *array* ketiga yangmempunyai dimensi yang sama.

Konvolusi dapat digunakan dalam *image processing* untukmenerapkan operator yang mempunyai nilai *output* dari piksel yang berasal dari kombinasi linear nilai*input* piksel tertentu.Proses ini sangat berguna untuk melakukan operasi penapisan (*filtering*) pada citra. Pada pengolahan citra digital, konvolusi dilakukan secara dua dimensi pada sebuah citra.

Konvolusi memiliki dua buah fungsi *f(x)* dan *g(x)* yang didefinisikan erikut:

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(a) \cdot g(x-a)da \qquad (2.5)$$

Optimization Software:
www.balesio.com

Dalam hal ini, tanda (*) menyatakan operator konvolusi dan peubah (*variable*) adalah peubahbantu.

Untuk pengolahan citra, operasi yang dilakukan adalah diskrit karena nilai koordinat piksel merupakan nilai yang diskrit. Selanjutnya *filter* atau *mask* yang digunakan pada pengolahan citra biasanya berukuran terbatas, dalam artian bobot atau pengaruh dari titik-titik yang cukup jauh sudah tidak signifikan, sehingga dapat diabaikan (dianggap nol) [11].

Untuk fungsi dengan dua dimensi, operasi konvolusi untuk fungsi diskrit didefinisikan sebagai berikut:

$$h(x,y) = f(x,y) * g(x,y) = \sum_{a=-\infty}^{\infty} \sum_{a=-\infty}^{\infty} f(a,b) \cdot g(x-a, y-b) \qquad (2.6)$$

Fungsi *g(x,y)* disebut juga konvolusi *filter*, konvolusi *mask*, konvolusi kernel / kernel, atau *template*. Dalam bentuk diskrit kernel dinyatakan dalam bentuk matriks (umumnya matriks3x3). Ukuran matriks ini biasanya lebih kecil dari ukuran citra. Setiap elemen matriks disebutkoefisien konvolusi.

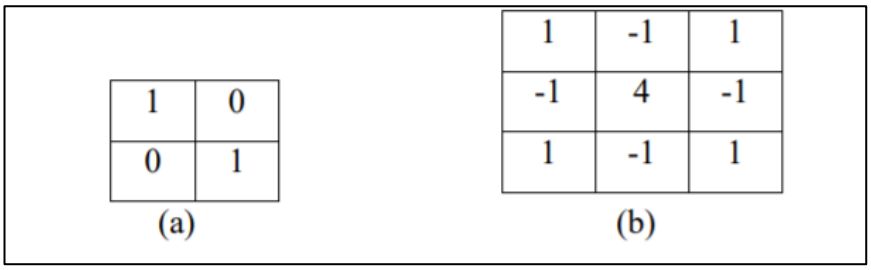
2.7.1 Kernel

Kernel adalah suatu matriks angka yang umumnya berukuran kecil yang elemen-elemennya adalah berupa bilangan. Kerneldigunakan dalam proses konvolusi. Oleh karena itu, kernel juga disebut sebagai *convolution window* (jendela konvolusi).

Optimization Software:
www.balesio.com

Ukuran kernel dapat berbeda-beda, seperti 2x2, 3x3, 5x5, dan sebagainya. Elemen-elemen kernel juga disebut sebagai bobot (*weight*) merupakan bilangan-bilangan yang membentuk pola tertentu. Dalam proses konvolusi, kernel berukuran berbeda yang mengandung pola angka yang berbeda menimbulkan hasil yang berbeda pula.

Gambar 2.3 adalah contoh dari kernel 2x2 dan 3x3 :



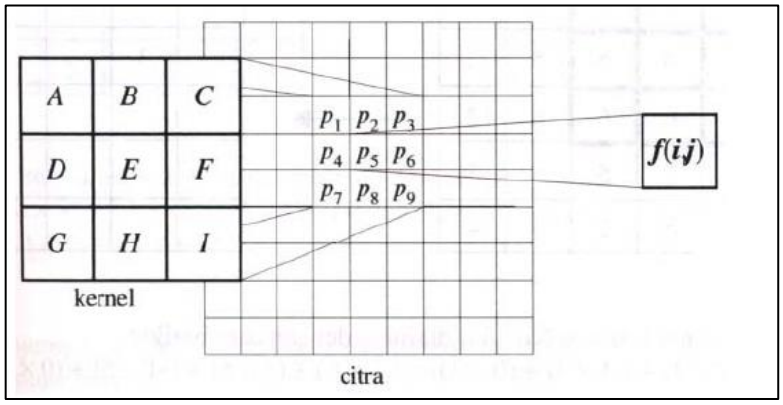
Gambar 2.3 (a) contoh kernel yang berukuran 2x2 dan (b) adalah contoh kernel yang berukuran 3x3[13].

Kernel menggunakan konsep piksel tetangga (*neighbouring pixels*), di mana matriks kernel dibuat dengan asumsi bahwa nilai sebuah piksel bisa dipengaruhi oleh piksel-piksel tetangganya.

Adapun Ilustrasi konvolusi ditunjukkan pada **gambar 2.4**, dimana :

$$f(i,j) = Ap1 + Bp2 + Cp3 + Dp4 + Ep5 + Fp6 + Gp7 + Hp8 + Ip9$$

Optimization Software:
www.balesio.com



Gambar 2.4 Ilustrasi Konvolusi [11].

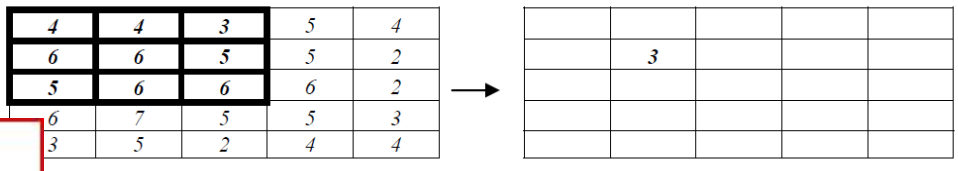
Contoh Misalkan citra *f(x,y)* yang berukuran 5x5 dan sebuah kernelatau *mask* yangberukuran 3x3 masing-masing adalah sebagai berikut

$$f(x,y)=\begin{bmatrix}4&4&3&5&4\\6&6&5&5&2\\5&6&6&6&2\\6&7&5&5&3\\2&5&2&4&4\end{bmatrix} \quad \text{dan} \quad g(x,y)=\begin{bmatrix}0&-1&0\\-1&0.4&-1\\0&-1&0\end{bmatrix}$$

Keterangan : tanda – menyatakan posisi (0,0) dari kernel.

Operasi konvolusi antara citra *f(x,y)* dengan kernel *g(x,y),* yaitu *f(x,y)* * *g(x,y)* dapatdiilustrasikan sebagai berikut:

1. Tempatkan kernelpada sudut kiri atas, kemudian hitung nilai piksel pada posisi (0,0) dari kernel.

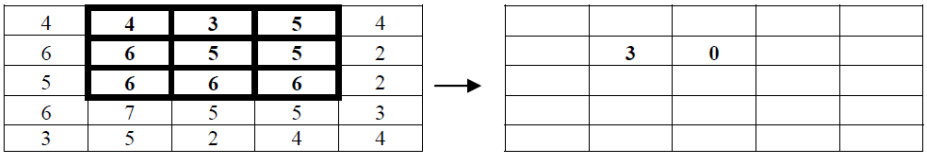


l konvolusi = 3. Nilai ini dihitung dengan cara berikut:

) + (-1x4) + (0x3) + (-1x6) + (4x6) + (-1x5) + (0x5) + (-1x6) + (0x6) = 3

Optimization Software:
www.balesio.com

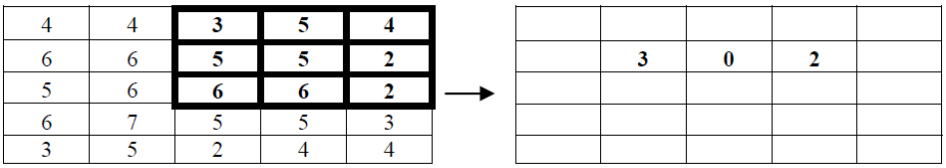
2. Geser kernelsatu piksel ke kanan, kemudian hitung nilai piksel pada posisi (0,0) dari kernel:



Hasil konvolusi = 0. Nilai ini dihitung dengan cara berikut:

(0x4) + (-1x3) + (0x5) + (-1x6) + (4x5) + (-1x5) + (0x6) + (-1x6) + (0x6) = 0

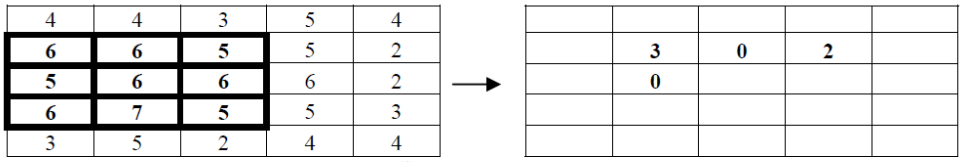
3. Geser kernelsatu piksel ke kanan, kemudian hitung nilai piksel pada posisi (0,0) dari kernel:



Hasil konvolusi = 2. Nilai ini dihitung dengan cara berikut:

(0x3) + (-1x5) + (0x4) + (-1x5) + (4x5) + (-1x2) + (0x6) + (-1x6) + (0x2) = 2

4. Geser kernelsatu piksel ke bawah, lalu mulai lagi melakukan konvolusi dari sisi kiri citra. Setiap kali konvolusi, geser kernel atau piksel ke kanan:

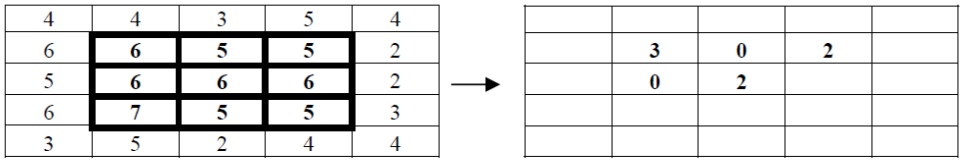


(i)

Hasil konvolusi = 0. Nilai ini dihitung dengan cara berikut:

) + (-1x6) + (0x5) + (-1x5) + (4x6) + (-1x6) + (0x6) + (-1x7) + (0x5) = 0

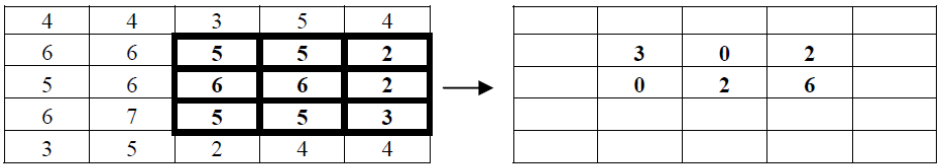
Optimization Software:
www.balesio.com



(ii)

Hasil konvolusi = 2. Nilai ini dihitung dengan cara berikut:

(0x6) + (-1x5) + (0x5) + (-1x6) + (4x6) + (-1x6) + (0x7) + (-1x5) + (0x5) = 2



(iii)

Hasil konvolusi = 6. Nilai ini dihitung dengan cara berikut:

(0x5) + (-1x5) + (0x2) + (-1x6) + (4x6) + (-1x2) + (0x5) + (-1x5) + (0x3) = 6

Dengan cara yang sama, piksel-piksel pada baris ketiga di konvolusi sehingga akan menghasilkan:

	3	**0**	**2**	
	0	**2**	**6**	
	6	**0**	**2**	

Jika hasil konvolusi menghasilkan nilai piksel *negative*, nilai tersebut dijadikan 0. Sebaliknya jika hasil konvolusi menghasilkan nilai piksel lebih besar dari nilai keabuan maksimum (255), nilai tersebut dijadikan ke nilai keabuan m[11].

Optimization Software:
www.balesio.com

Masalah timbul bila piksel yang dikonvolusi adalah piksel pinggir, karena beberapa koefisien konvolusi tidak dapat diposisikan pada piksel-piksel citra, seperti pada **gambar 2.5**:

4	4	3	**5**	**4**	**?**
6	6	5	**5**	**2**	**?**
5	6	6	**6**	**2**	**?**
6	7	5	5	3	
3	5	2	4	4	

Gambar 2.5 Konvolusi piksel pinggir

Solusi untuk masalah ini adalah: (1) piksel-piksel pinggir diabaikan, tidak dikonvolusi, jadi nilai piksel pinggir sama dengan nilai pada citra semula; (2) duplikasi elemen citra, misalnya elemen kolom pertama disalin ke kolom M+1 dst.; (3) elemen bertanda "?" diasumsikan bernilai 0 atau konstanta lain, sehingga konvolusi pinggir-pinggir dapat dilakukan [11].

Solusi dengan ketiga pendekatan di atas mengasumsikan bagian pinggir citra lebarnya sangat kecil (hanya satu piksel) *relative* dibandingkan dengan ukuran citra, sehingga piksel-piksel pinggir tidak memperlihatkan efek yang kasat mata.

2.8 *Gaussian Blur*

engolahan citra dengan menggunakan konvolusi *gaussian blur* bkan suatu citra menjadi kabur sehingga sudut-sudut tajam pada citra

Optimization Software:
www.balesio.com

akan menjadi lebih halus. Pengolahan citra ini dapat berdampak suatu citra menjadi semakin baik, tapi bisa juga menjadi semakin buruk.

Gaussian Blur atau yang sering disebut *Gaussian Filter* merupakan sebuah metode untuk *image smoothing* dimana nilai pembobotan untuk setiap piksel berdasarkan pada fungsi *Gaussian*. Filter *Gaussian* dapat digunakan untuk mengurangi derau yang ada pada citra digital. Secara teori derau yang memiliki sebaran *Gaussian* akan dinetralkan dengan fungsi lain yang memiliki sifat fungsi *Gaussian.*

*Gaussian Filter*yang diperoleh dari operasi konvolusi dilakukan dengan melakukan perkalian antara matriks kernel dengan matriks citra asli. Nilai kernel *Gaussian* memiliki ukuran yang beragam dan pasti adalah bilangan ganjil, hal ini dikarenakan kernel *Gaussian* memiliki nilai di posisi koordinat tengah. Semakin besar ukuran kernel maka *noise* pada gambar akan semakin hilang. Akan tetapi, ukuran kernel yang terlalu besar juga bisa menyebabkan semakin hilangnya *feature* pada gambar sehingga hasil *pre-processing* menjadi kurang baik [10].

Perkalian antara bobot matriks gambar asli dengan bobot matriks kernel

$$\frac{1}{K}\sum_{p=0}^{N-1}\left(\sum_{q=0}^{M-1} G(p,q)\, Pixel\ A\left(i+p-\frac{(N-1)}{2}, i+q-\frac{(M-1)}{2}\right)\right) \qquad (2.4)$$

dapat dirumuskan seperti di bawah ini

Keterangan:

Pixel A = Gambar A (Gambar asli).

N = Jumlah kolom matriks kernel.

Optimization Software:
www.balesio.com

M = Jumlah baris matriks kernel.

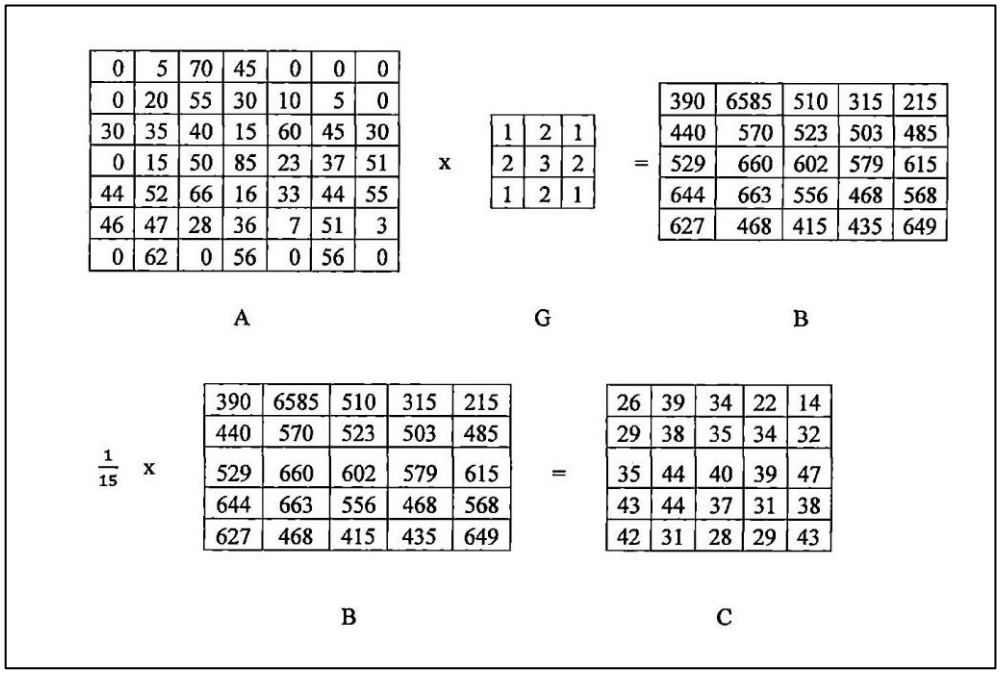
K = Penjumlahan semua bobot di G

G(p,q) = Elemen matriks kernel pada posisi (p,q).

Gambar yang akan diproses dibagi menjadi dua jenis piksel, yaitu piksel batas dan piksel dalam. Piksel batas yaitu piksel yang posisinya berada pakling luar pada gambar. Piksel dalam yaitu piksel yang posisinya berada di dalam piksel batas. Untuk piksel yang berada di dalam, perkalian dilakukan dengan mencari nilai piksel baru sebagai piksel tengah dan bobotnya dikalikan dengan bobot pada piksel tengan matriks kernel. Hasil dari perkalian tersebut dijumlahkan dengan hasil perkalian antara bobot piksel yang ada pada tetangganya dengan bobot piksel dari matriks kernel. Untuk piksel yang berada pada sudut perbatasan, sebelum melakukan perkalian, maka harus mencari bobot pada piksel-piksel luar (*dummy*). Bobot piksel-piksel ini dicari dengan menggunakan metode interpolasi yaitu melihat kedua piksel yang berdekatan juga searah (*horizontal* atau *vertical*). Jika bobot piksel kurang dari 0 maka bobot tersebut akan dijadikan 0. Apabila ada piksel yang memiliki bobot lebug besar dari 255 maka bobotnya dijadikan 255 [10].

Adapun contoh penerapan persamaan 2.4 bisa dilihat pada **gambar 2.6**:

Optimization Software:
www.balesio.com



Gambar 2.6 Contoh penerapan Konvolusi Matriks Gambar Asli dan Matriks Kernel [10]

Keterangan**gambar 2.6**:

A = Matriks gambar asal.

B = Matriks hasil perkalian.

G = Matriks kernel *Gaussian.*

C = Matriks gambar hasil.

2.9 *Profilling*

Profiling pada suatu program adalah suatu bentuk analisis untuk mengukur hal-hal seperti penggunaan memori, penggunaan waktu, penggunaan instruksi atau frekuensi dan durasi pemanggilan fungsi[14]. Ini adalah cara untuk ni di mana saja jumlah sumber daya terbesar yang digunakan untuk

Optimization Software:
www.balesio.com

menargetkan optimasi pada bagian tertentu dari program tersebut.Ada dua tipe *profiling*, yaitu[15]:

1. *Profil*ing Deterministik: Semua kejadian/proses akan dimonitor. *Profiling* ini memberikan informasi yang akurat tetapi memiliki dampak besar pada kinerja (*overhead*), dimana ini berarti kode akan berjalan lebih lambat saat melakukan *profiling*. Jenis *profiling* ini cocok digunakan untuk fungsi-fungsi kecil.
2. *Profiling* statistik: *Profiling* jenis ini akan mengambil sampel kondisi eksekusi secara berkala untuk memperoleh indikatornya. Metode ini kurang akurat, tetapi juga dapat mengurangi overhead.

2.9.1 CProfile

Melakukan *profiling* pada program Python dapat dilakukan dengan *library* standar yang dimiliki Python, maupun modul dan program pihak ketiga. Python dilengkapi dengan dua modul untuk *profil*ing deterministic yaitu cProfile dan Profile. Keduanya adalah implementasi yang berbeda dari antarmuka yang sama.

Modul Profilelebih sesuai jika ingin memperluas *profiling* dengan cara tertentu. Sedangkan cProfile lebih banyak digunakan untuk program-program yang akan berjalan lama / terus menerus [15].

CProfile dapat dijalankan dari terminal, maupun diimpor sebagai ul dengan Python. CProfile akan memberikan informasi hasil *profiling* suatu fungsi, total waktu yang digunakan untuk setiap pemanggilan

Optimization Software:
www.balesio.com

(tidak termasuk pemanggilan ke fungsi lain), waktu kumulatif fungsi dan jumlah panggilan ke fungsi tersebut.Contoh pemanggilan cProfile dan output yang dihasilkan terlihat pada **gambar 2.7**

```
➜ python3 -m cProfile script.py

58 function calls in 9.419 seconds

Ordered by: standard name

ncalls  tottime  percall  cumtime  percall filename:lineno(function)
     1    0.000    0.000    9.419    9.419 part1.py:1(<module>)
    51    9.419    0.185    9.419    0.185 part1.py:1(computation)
     1    0.000    0.000    9.419    9.419 part1.py:10(function1)
     1    0.000    0.000    9.243    9.243 part1.py:15(function2)
     1    0.000    0.000    0.176    0.176 part1.py:20(function3)
     1    0.000    0.000    9.419    9.419 part1.py:24(main)
```

Gambar 2.7 Contoh pemanggilan cProfile dan output yang dihasilkan

Hasil **gambar 2.7** bisa diuraikansebagai berikut :

- *ncalls*: Seperti namanya, variabel ini memuat jumlah panggilan terhadap fungsi. Dengan data ini, fungsi yang memiliki banyak panggilan atau menghabiskan terlalu banyak waktu per panggilan dapat dioptimalkan.
- *tottime*: Total waktu yang dihabiskan dalam fungsi itu sendiri, tetapi tidak termasuk sub-panggilan. Pada contoh diatas, dapat dilihat bahwa fungsi ’computation’ dipanggil sebanyak 51 kali, dan setiap kali pemanggilan mengkonsumsi 0,185 detik.
- *cumtime*: Waktu kumulatif. Waktu ini termasuk panggilan sub, itulah sebabnya pada contoh di atas, function1 dan function2 memiliki waktu kumulatif yang mirip dengan total waktu perhitungan.

Optimization Software:
www.balesio.com

- *percall*: Terdapat 2 jenis variabel ’percall’. Yang pertama berarti total waktu per panggilan, dan yang kedua berarti waktu kumulatif per panggilan.

2.9.2 Mpstat

Perintah mpstat digunakan untuk mengumpulkan dan menampilkan statistik kinerja untuk semua *logical processor* dalam sistem. Pengguna dapat menentukan berapa kali statistik ditampilkan, dan interval di mana data diperbarui.

Parameter interval akan menentukan jumlah waktu dalam satuan detik antara setiap laporan yang ditampilkan. Jika parameter interval tidak digunakan, maka perintah mpstat akan menghasilkan laporan tunggal yang berisi statistik untuk waktu sejak sistem dalam keadaan *startup* hingga akhirnya keluar. Kemudian, jika parameter hitungan digunakan, nilainya akan menentukan berapa jumlah laporan yang dihasilkan dan jumlah detik secara terpisah. Jika hanya parameter interval yang digunakan, tanpa parameter hitungan, maka laporan akan terus menerus dihasilkan.

Perintah mpstat tanpa opsi menghasilkan laporan tunggal yang berisi statistik kinerja untuk semua prosesor logis sejak waktu boot.Ketika perintah mpstat dipanggil, ini akan menampilkan dua bagian statistik. Bagian pertama ampilkan Konfigurasi Sistem, yang ditampilkan ketika perintah dimulai pun setiap kali ada perubahan dalam konfigurasi sistem. Bagian kedua

Optimization Software:
www.balesio.com

menampilkan Statistik Utilisasi yang ditampilkan dalam interval dan kapan saja nilai-nilai metrik ini adalah jumlah rata-rata dari interval sebelumnya.

Contoh pemanggilan mpstat dan output yang dihasilkan ditunjukkan pada**gambar 2.8**, dimana diperoleh laporan setiap penggunaan prosesor CPU dalam satu kali pemanggilan.

```
$ mpstat -P ALL

Linux 3.2.0-57-generic (USERNB01) 12/12/2013 _x86_64_ (2 CPU)

04:07:36 PM CPU %usr %nice %sys %iowait %irq %soft %steal %guest %idle
04:07:36 PM all 6.02 0.04 1.72 2.99 0.00 0.05 0.00 0.00 89.17
04:07:36 PM 0 3.84 0.01 1.15 3.72 0.00 0.06 0.00 0.00 91.21
04:07:36 PM 1 13.55 0.15 3.66 0.46 0.00 0.03 0.00 0.00 82.15
```

Gambar 2.8 Contoh pemanggilan mpstat dan output yang dihasilkan[16].

Untuk memperoleh laporan pergerakan dalam pemanfaatan keseluruhan CPU dalam interval waktu tertentu, maka ditambahkan parameter interval. Perintah pada **gambar 2.9** akan menghasilkan laporan pemanfaatan keseluruhan CPU dalam interval 3 detik.

```
$ mpstat 3 4

Linux 3.2.0-57-generic (USERNB01) 12/12/2013 _x86_64_ (2 CPU)

04:27:11 PM CPU %usr %nice %sys %iowait %irq %soft %steal %guest %idle
04:27:14 PM all 0.67 0.00 0.34 0.00 0.00 0.00 0.00 0.00 98.99
04:27:17 PM all 1.17 0.00 0.33 1.33 0.00 0.00 0.00 0.00 97.17
04:27:20 PM all 0.84 0.00 0.17 0.00 0.00 0.00 0.00 0.00 98.99
04:27:23 PM all 1.00 0.00 0.17 1.51 0.00 0.00 0.00 0.00 97.32
Average: all 0.92 0.00 0.25 0.71 0.00 0.00 0.00 0.00 98.12
```

Gambar 2.9 Contoh pemanggilan mpstatdengan interval [16].

Optimization Software:
www.balesio.com

Adapun informasi-informasi yang ditampilkan di bagian konfigurasi sistem pada **gambar 2.8**dan **gambar 2.9**adalah [16] :

- *%usr* : waktu yang digunakan CPU menjalankan *un-niced user process. Un-niced process* adalah proses standard, yang tidak diberi attribute nice. Sedangkan *niced process* adalah proses yang diberi prioritas tertentu, yang berbeda dengan standard — bisa lebih rendah atau lebih tinggi.
- *%nice* : waktu yang digunakan CPU menjalankan niced user process. Nice, pada sistem operasi UNIX (dan keluarganya), adalah sebuah fungsi (juga tool) untuk mengubah prioritas sebuah proses. Proses dengan atribut nice lebih rendah memiliki prioritas lebih tinggi, demikian sebaliknya, nilai nice tinggi memiliki prioritas lebih rendah.
- *%sys* :waktu yang digunakan CPU untuk menjalankan kernel sistem operasi.
- *%idle*:waktu ketika tidak ada proses yang perlu dijalankan oleh CPU.
- *%iowait* :waktu ketika tidak ada proses yang perlu dijalankan oleh CPU (seperti idle di atas), dan ada proses yang statusnya sedang menunggu disk I/O.
- *%irq*: waktu yang digunakan oleh kernel sistem operasi untuk menangani Hardware Interrupt Request (IRQ).
- *%soft* :waktu yang digunakan oleh kernel sistem operasi untuk menangani Software Interrupt Request (Softirq). Untuk mendapatkan %soft tinggi, dapat dilakukan dengan membangkitkan beban tinggi pada webserver misalnya.

Optimization Software:
www.balesio.com

- *%steal* :waktu yang diambil oleh hypervisor dari virtual machine kita untuk melakukan hal lain.
- *%guest* : waktu yang dihabiskan oleh CPU untuk menjalankan prosesor virtual.

2.10 Penelitian Terkait

Beberapa penelitian terkait yang telah dilakukan dari tahun ke tahun.

1. ***E-Park: Automated-Ticketing Parking Meter System* (Kulesza, Mateusz J (2015)).**

   Peneliti membuat suatu sistem pelaporan & pembayaran parkir otomatis bagi kendaraan yang parkir secara illegal. Kendaraan dikenali dengan menangkap citra plat dengan arduCam & arduino MEGA 2560. Hasilnya 14 dari 15 tes yang dilakukan pada sistem berhasil menghasilkan ekstraksi yang benar dari plat ketika wilayah plat memenuhi resolusi minimum yang diperlukan dan tingkat kemiringan tidak lebih dari 55 °.

2. ***Automatic Parking Access Using Openalpr On Raspberry Pi3* (Elena Roxana, dkk (2016)).**

   Sistem ini adalah hasil implementasi sistem ALPR dengan library open source C/C++ bernama Open ALPR berbasis OpenCV dan TesseractOCR. Sistem ini menggunakan image processing untuk mengidentifikasi setiap kendaraan yang masuk/keluar melalui palang

Optimization Software:
www.balesio.com

parkir.ketika kendaraan mendekati palang, unit pengidentifikasi plat secara otomatis membaca nomor plat dan membandingkannya dengan daftar yang telah ditetapkan. Palang akan terbuka jika keduanya cocok. Inputan sistem ini adalah berbentuk video streaming, kemudian library openALPR akan melakukan tiga tahap yaitu *License plate detection, License plate character recognition* dan *post processing* untuk mendapatkan data plat.

Hasil pengujian resolusi maksimal adalah 1280x720 piksels, jarak minimal dari bagian depan mobil ke Kamera adalah 1m, jarak jangkauan maksimum dari kamera yang diamati selama pengujian antara 2-3m, namun ini tergantung pada kondisi penerangan. Total waktu pemrosesan adalah 800,2 ms.

3. ***Accurate Vehicle Number Plate Regognition And Real Time Identification Using Raspberry Pi*** **(N.Abirami, Dr.J.S Leena Jasmine (2018)).**

Penelitian ini menggunakan raspberry pi 3, kamera dan speaker. Di mana pengenalan dilakukan oleh raspberry pi dari gambar yang ditangkap kamera secara real time. Proses ekstraksi fitur dilakukan oleh algoritma PCA sedangkan klasifikasi dilakukan oleh algoritma CNN. Untuk segmentasi, wilayah yang terpisah dibagi dan disegmentasi, setelah itu langkah klasifikasi dilakukan. Klasifikasi CNN digunakan untuk mengenali karakter yang ada di plat kendaraan.

Optimization Software:
www.balesio.com

Tidak ada hasil kinerja pengujian yang ditampilkan pada penelitian ini.

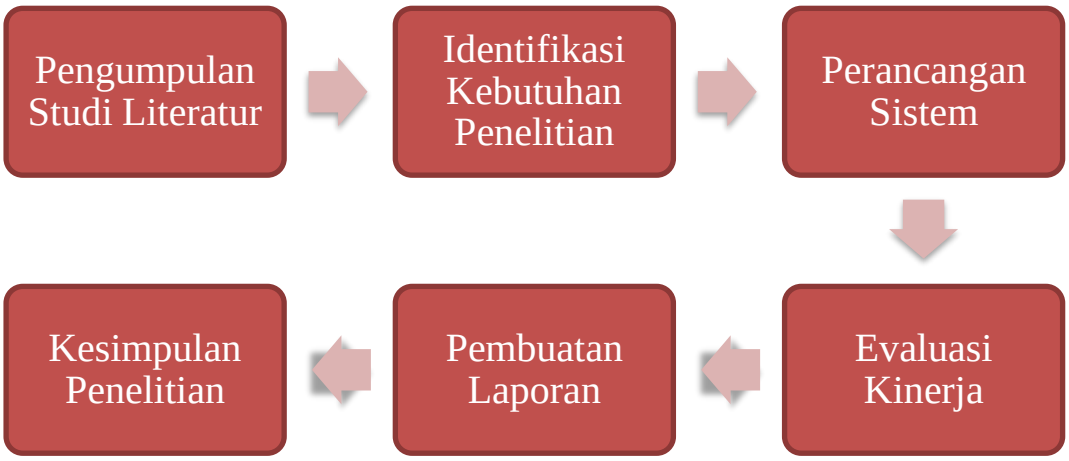
Optimization Software:
www.balesio.com

BAB III

METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Tahapan penelitian ini sebagaimana ditunjukkan pada **gambar 3.1**



Gambar 3.1. Diagram Tahapan Penelitian

Tahapan secara garis besar dijelaskan sebagai berikut:

a. Pengumpulan Studi Literatur. Pada tahap ini, dilakukan pencarian literatur dan dokumentasi penelitian dari berbagai sumber terkait sistem deteksi plat kendaraan secara *realtime* berbasis Mini PC.

b. Identifikasi kebutuhan dan desain sistem. Dalam tahap ini dilakukan identifikasi data yang dibutuhkan untuk mengevaluasi kinerja sistem, alat dan bahan yang diperlukan dalam mendukung pengambilan data tersebut, termasuk pengumpulan dataset gambar plat mobil sebagai data latih pada sistem yang akan dibuat.

Optimization Software:
www.balesio.com

c. Perancangan sistem. Dalamtahap ini dilakukan desain dan perancangan sistem pengenalan plat.

d. Evaluasi kinerja. Setelah sistem diuji coba, maka akan dilakukan evaluasi kinerja dengan parameter rata-rata waktu yang dibutuhkan Mini PC serta kinerja CPU dalam memproses algoritma ALPR untuk menghasilkan data plat yang diinginkan.

e. Penulisan laporan. Tahap ini merupakan tahap penulisan seluruh proses penelitian yang telah dilakukan dalam bentuk skripsi. Laporan ini digunakan sebagai bahan publikasi maupun untuk acuan penelitian selanjutnya.

f. Kesimpulan penelitian. Setelah melakukan tahapan-tahapan di atas, diperolehlah kesimpulan berdasarkan penelitian yang telah dilakukan.

3.2 Waktu dan Lokasi Penelitian

Waktu penelitian dilakukan selama 9 bulan, dimulai sejak bulan Januari 2019 hingga proses pelaporan hasil tugas akhir ini pada bulan September 2019. Pengambilan data dilakukan di Parkiran Kampus Teknik Universitas Hasanuddin, Gowa.Sedangkan Evaluasi dan pengolahan data dilakukan di Laboratorium *Internet of Things* dan *Parallel Computing* (IoT-PC), Departemen Teknik Informatika, Fakultas Teknik,Universitas Hasanuddin.

Optimization Software:
www.balesio.com

trumen Penelitian

umen yang digunakan dalam penelitian ini meliputi :

1. Software
    a. Windows 7.
    b. Arduino IDE.
    c. Raspbian .
    d. Opencv 3.5.0 + Python 3.0.
2. Hardware
    a. Perangkat Personal Computer CPU intel CORE i5 (1,7GHz UP TO 2,7 GHz, 2MB L2 Cache), 14" HD Graphics, RAM 2 GB, Windows 10 Pro x64.
    b. Raspberry Pi 3 Model B+.
    c. Kamera Smartphone Oppo F1f 13MP.

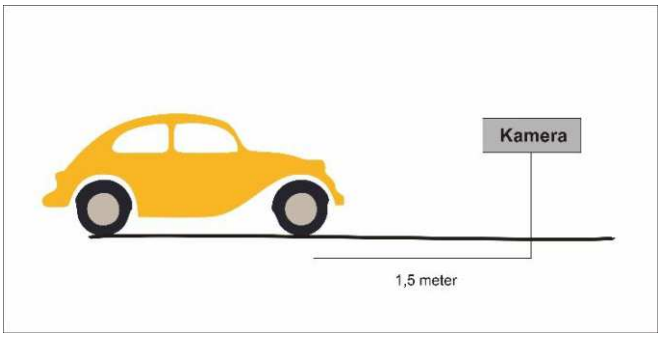
3.4 Pengambilan Data

Data yang digunakan pada penelitian ini yaitu data latih berupa citra karakter plat mobil dan data uji berupa citra mobil. Data latih diperoleh dari berbagai sumber yaitudataset penelitian terkait berupa lebih dari 1000 citra karakter. Dari data tersebut akan diolah dan dipilah kembali untuk menghasilkan satu citra latih dengan berbagai variasi karakter di dalamnya.

Sedangkan untuk citra uji diperoleh dengan melakukan pengukuran jarak antara mobil dan kamera *smartphone*. Jarak yang digunakan adalah 1,5 meter. Setelah sesuai, maka dilakukan pengambilan gambar dengan sudut 90° antara dan permukaan tanah. Tinggi kamera disesuaikan sedemikian rupa

Optimization Software:
www.balesio.com

sehingga mobil berada di tengah *frame.* Ilustrasi pengambilan data dapat dilihat pada **gambar 3.2.**



Gambar 3.2.Ilustrasi pengambilan data objek penelitian

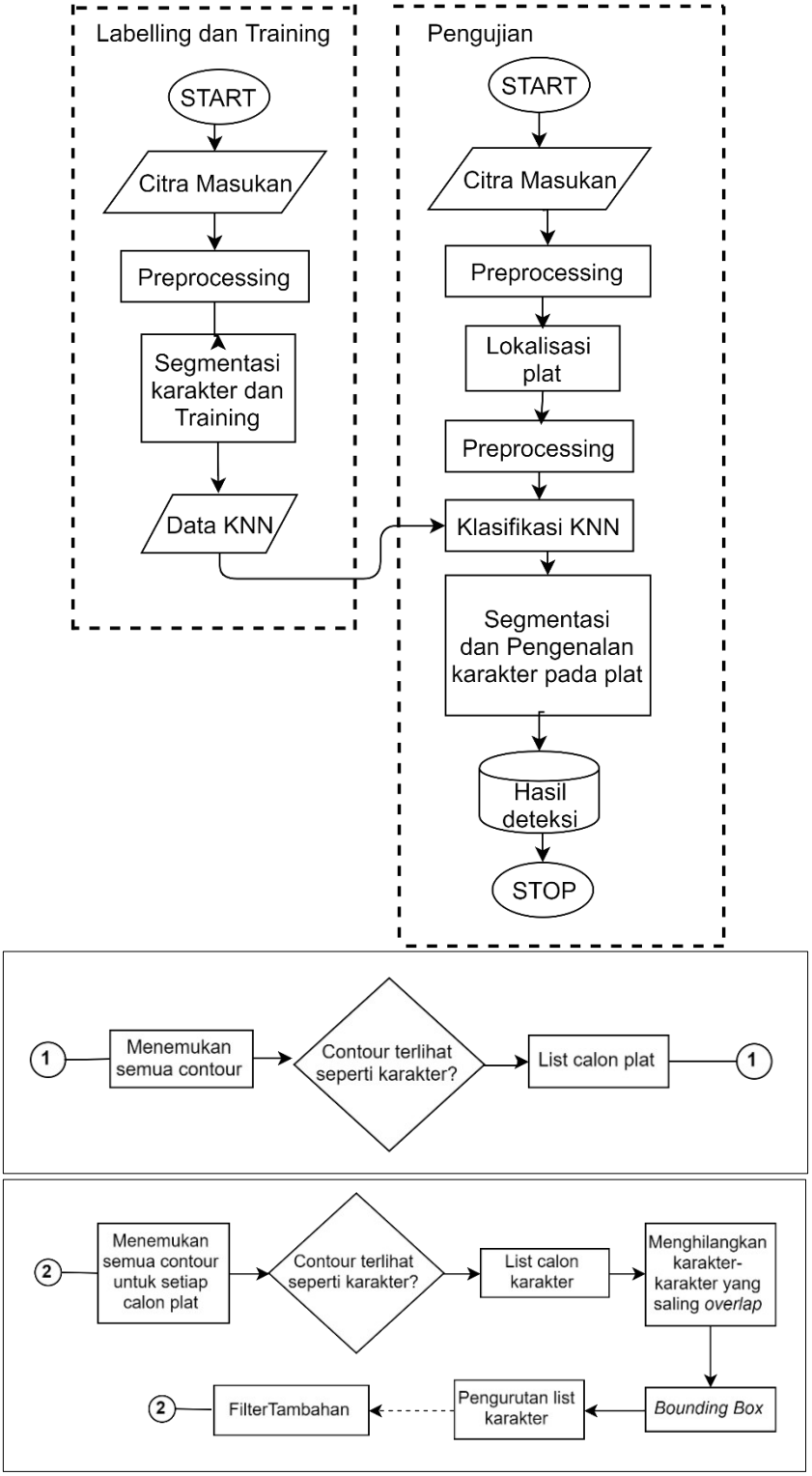
3.5 Perancangan Sistem

Sistemyang dibuat menggunakan sebuah Raspberry Pi 3 Model B+ yang digunakan sebagai *microprosessor* untuk mengolah citra masukan.Untuk bagian program, secara garis besar terbagi atas dua bagian, yaitu proses *labelling* dan *training*serta proses pengujian, seperti yang diperlihatkan pada**gambar 3.3**.

Dalam proses *training*, terdapat tiga tahap yang dilakukan, yaitu: Akuisisi Citra, *Preprocessing*, serta Segmentasi Karakter dan *Training*. Fitur yang akan diesktrak yaitu berupa bentuk dan jenis karakter 0-9 hingga A-Z. Hasil ekstraksi ini selanjutnya akan diberi label dan di*training*, kemudian hasilnya akan disimpan sebagai data untuk digunakan pada proses klasifikasi.

Untuk mengenali karakter plat pada proses pengujian digunakan algoritma ALPR dengan lima tahap, yaitu: Akuisisi Citra untuk citra masukan, *ssing*, Lokalisasi Plat, Klasifikasi KNN, serta Segmentasi dan an Karakter.

Optimization Software:
www.balesio.com



Gambar3.3 Flowchart Sistem.

Optimization Software:
www.balesio.com

Berdasarkan **gambar 3.3**, perancangan sistem dapat diuraikan sebagai berikut:

3.5.1 Citra Masukan

Langkah pertama adalah mempersiapkan data citra yang akan menjadi masukan. Data masukan yang digunakan dalam pembuatan sistem ini terdiri dari 1 citra latih (dengan lebih dari 1000 karakter di dalamnya) dan 83 citra uji. Citra latih ditunjukkan pada **gambar 3.4.**

Gambar 3.4 Citra Latih

Adapuncitra uji dalam penelitian ini diambil menggunakan kamera *nartphone* dan menghasilkan spesifikasi sebagai berikut:

Optimization Software:
www.balesio.com

1. Resolusi : 1936 x 2581*piksel*
2. Ekstensi : jpg

Citra ini merupakan hasil pengambilan yang dilakukanmenggunakan kamera belakang Oppo F1f dengan resolusi 13 *megapiksel* kemudian resolusinya diperkecil menjadi 5 *megapiksel*. Pengambilan data dilakukan pada saat kondisi cahaya yang cukup dan dilakukan pada siang hari. Adapun contoh data yang diambil sebagaimana ditunjukkan pada **gambar 3.5.**

Gambar 3.5 Contoh Citra Uji

Optimization Software:
www.balesio.com

3.5.2 Preprocessing

3.5.2.1 Grayscaling

Dalam tahap ini masukan *default* gambar yang berformat RGB diubah menjadi citra *grayscale*. Tahap ini dilakukan untuk mereduksi ukuran data dimana kombinasi warna *red, green,* dan *blue* tidak dibutuhkan,

sehingga proses komputasi menjadi lebih ringan.

Pada tahap ini digunakan fungsi cvtColor() pada OpenCV. Fungsi cvtColor ini berguna untuk mengubah gambar dari satu ruang warna ke ruang warna lainnya, yang dalam hal ini gambar dengan format RGB akan dikonversi menjadi format *grayscale* dengan menggunakan metode *luminosity*. Adapun penulisan fungsi yang digunakan adalah sebagai berikut

```
cvtColor(imgOriginal, cv2.COLOR_RGB2GRAY)
```

Keterangan :

imgOriginal =sumber gambar masukan berformat RGB.

cv2.COLOR_RGB2GRAY = kode konversi dari RGB ke *grayscale*.

Adapun fungsi cvtColor ini akan bekerja berdasarkan persamaan (3.1) berikut :

$$Gray = (0.299 * R) + (0.587 * G) + (0.144 * B) \quad (3.1)$$

Keterangan:

1. *Gray* = nilai derajat keabuan

Optimization Software:
www.balesio.com

2. R = nilai pada *channel Red*
3. G = nilai pada *channel Green*
4. B = nilai pada *channel Blue*

Sebagai contoh, misalkan pada sebuah citra warna terdapat matriks, di mana pikselnya terdiri dari tiga *channel* (*Red*, *Green*, dan *Blue*) yang bernilai [100, 55, 53]. Maka perhitungan untuk memperoleh nilai piksel keabuan dari piksel tersebut adalah sebagai berikut: 32,385

$$Gray = (0.299 * 100) + (0.587 * 55) + (0.144 * 53)$$

$$Gray = 68.327$$

Adapun contoh hasil konversi citra RGB ke *grayscale* dapat dilihat pada **gambar 3.6.**

Gambar 3.6 Hasil Tahap *Grayscale*

Optimization Software:
www.balesio.com

3.5.2.2 *Gaussian Blur*

Setelah gambar diubah menjadi format *grayscale* , dilakukan proses *blurring* dengan metode *Gaussian Blur*. Tahap ini berfungsi untuk mereduksi *noise* dan detail pada citra.

Untuk menerapkan *gaussian blur* pada citra masukan, digunakan fungsi GaussianBlur() pada OpenCV.Adapun penulisan fungsi yang digunakan adalah sebagai berikut

```
cvtColor(imgGray, nilai_kernel, 1)
```

Keterangan :

imgGray = sumber gambar masukan berformat *grayscale*.

nilai_kernel = Nilai / besar kernel yang digunakan.

1 = Nilai standar deviasi.

Area yang dipindai di sekitar setiap piksel disebut kernel, dimana semakin besar kernel yang digunakan maka semakin besar pula jumlah piksel yang dapat dipindai.

Sedangkan standar deviasi adalah filter radius yang menentukan kerapatan piksel, dimana yang digunakan adalah σ = 1.Nilai 1 dilakukan untuk mempercepat kalkulasi filter tiap piksel. Penyebab lainnya yaitu standar deviasi yang terlalu besar dapat menghilangkan banyak detail di setiap pikselnya.

Adapun contoh hasil tahap *gaussian blur*(nilai kernel 7,7 dan andar deviasi 1), hasilnya terlihat seperti **gambar 3.7.**

Optimization Software:
www.balesio.com

Gambar 3.7 Hasil Tahap *Gaussian Blur*

3.5.2.3 Adaptive Thresholding

Pada tahap inidilakukan proses *thresholding* untuk mengubah citra menjadi citra biner, dimana citra hanya punya dua nilai derajat keabuan yaitu hitam dan putih. Pada algoritma *threshold* umumnya digunakan nilai *global* sebagai nilai ambang. Tapi hasilnya tidak akan bagus di semua kondisi di mana gambar memiliki kondisi pencahayaan yang berbeda di area yang berbeda.

Dalam penelitian ini, metode yang digunakan adalah *Adaptive Thresholding*. Metode ini akan menghitung nilai ambang untuk sebagian area pada gambar, sehingga akan didapatkan ambang batas yang berbeda untuk wilayah berbeda dari gambar yang sama. Hal ini akan memberikan hasil yang lebih baik untuk gambar dengan pencahayaan beragam.

Optimization Software:
www.balesio.com

Untuk menerapkan metode *Adaptive thresholding* digunakan fungsi adaptiveThreshold() pada OpenCV. Penulisan fungsi yang digunakan adalah sebagai berikut:

```
adaptiveThreshold(img,maxValue, adaptiveMethod,
                        thresholdTyp)
```

Keterangan :

img = Sumber gambar masukan

maxValue= Nilai yang diberikan jika nilai piksel lebih dari nilai ambang batas.

adaptiveMethod=Variabel yang menentukan metode apa yang digunakan dalam menghitung nilai ambang.

thresholdType= Variabel yang menentukan jenis ambang apa yang digunakan.

Untuk parameter adaptiveMethod, terdapat dua jenis metode yang dapat digunakan:

- cv2.ADAPTIVE_THRESH_MEAN_C : nilai ambang/*threshold* diambil dari rata-rata area di sekitarnya .
- cv2.ADAPTIVE_THRESH_*GAUSSIAN*_C : nilai ambang/ *threshold* diambil dari nilai *weighted sum area* di sekitarnya dimana *weights* adalah *Gaussian* window.

Pada penelitian ini, metode yang digunakan adalah Adaptive *aussianThreshold*ing karena hasil threshold yang diberikan tidak

Optimization Software:
www.balesio.com

berlebihandan tidak sampai merusak detail karakter pada plat. Hasil perbandingan antara kedua metode bisa dilihat pada **gambar 3.8.**

Gambar 3.8 Perbedaan hasil metode *threshold*. (a) Adaptive *GaussianThreshold*ing, (b) Adaptive Mean *Threshold*ing.

Adapun untuk parameter thresholdType, terdapat dua jenis *threshold* yang dapat digunakan yaitu *THRESH_BINARY* dan *THRESH_BINARY_INV*. Untuk *THRESH_BINARY* ,jika intensitas pikselkurang dari nilai ambang, maka intensitas piksel baru akan diatur menjadi 0, jika sebaliknya maka piksel ditetapkan menjadi *maxValue*. Sedangkan untuk *THRESH_BINARY_INV* jika intensitas piksel kurang dari nilai ambang, maka intensitas piksel yang baru ditetapkan menjadi *maxValue*, jika baliknya maka piksel ditetapkan menjadi 0.

Optimization Software:
www.balesio.com

Pada penelitian ini, tipe *thresh* yang digunakan adalah *THRESH_BINARY_INV* karena bagian yang menjadi fokus utama adalah bagian karakter plat, dimana bagian tersebut harus dijadikan terang (putih) dengan latar belakang yang gelap. Hasil perbandingan antara kedua tipe *thresh* bisa dilihat pada **gambar 3.9**.

Gambar 3.9 Perbedaan hasil 2 jenis *thresh*. (a) THRESH_BINARY_INV, (b) THRESH_BINARY.

3.5.3 Segmentasi Karakter dan *Training*

Untuk proses *labelling*dan *training*, tahap setelah *preprocessing* adalah tahap segmentasi yaitu untuk membagi-bagi fitur citra ukmemperoleh *Region of Interest* (ROI) dari setiap karakter yang akan

Optimization Software:
www.balesio.com

dipelajari. Setelah ROI diperoleh, gambar akan dicrop sesuai ukuran ROI dan ditampilkan. Hasil dari proses ini bisa dilihat pada**gambar 3.10**.

Gambar 3.10 Hasil Segmentasi Karakter (a) ROI setiap karakter, (b) ROI yang telah dicrop

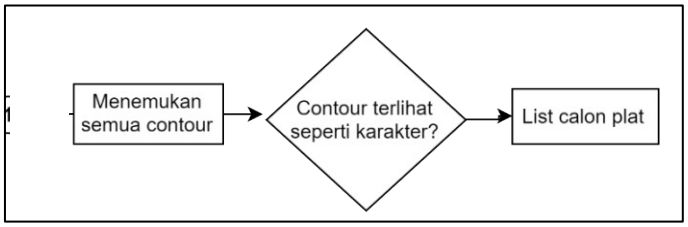
Setelah gambar berhasil dicrop, pemberian label dilakukan dengan mengetikkan karakter pada *keyboard* sesuai dengan hasil segmentasi yang muncul (contohnya pada **gambar 3.10b**). Pada saat tombol pada *keyboard* ditekan, maka data ASCII pada tombol *keyboard* akan disimpan dan gambar karakter yang ditampilkan akan diubah menjadi data array.

Setelah semua gambar karakter yang ada di citra latih selesai diproses, maka data yang dimasukkan tadi akan disimpan pada file berformat txt. Data hasil *training* inilah yang akan menjadi *training sample* pada metode KNN, dimana metode ini akan mengklasifikasikan sebuah obyek dengan cara membandingkan mayoritas atribut dengan training sample yang ada.

Optimization Software:
www.balesio.com

3.5.4 Lokalisasi Plat

Untuk proses pengujian, setelah tahap *preprocessing* selesai dilakukan, maka selanjutnya adalah melokalisasi area plat kendaraan dari citra masukan. Hal ini dilakukan karena citra masukan terdiri dari banyak objek lain selain plat seperti *background* lingkungan dan juga badan mobil itu sendiri. Karena itu, perlu dilakukan proses lokalisasi plat untuk mengenali area yang merupakan plat dan membuang area yang tidak diperlukan. Proses ini bisa dilihat pada **gambar 3.11.**



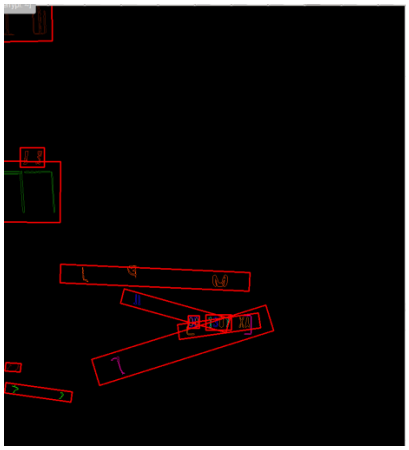
Gambar 3.11 Flowchart tahap lokalisasi plat

Berdasarkan **gambar 3.11,** langkah yang dilakukan adalah sebagai berikut :

1. Menemukan semua kontur dari citra hasil tahap *preprocessing* (*threshold*) dengan menerapkan fungsi findCountour() dari OpenCV. Contoh hasil tahap ini ditunjukkan pada gambar 3.10a.
2. Jika kontur terlihat seperti karakter, maka kontur di*resize* berdasarkan ukuran ROI masing-masing kontur, kemudian dimasukkan ke dalam list calon plat. Contoh hasil tahap ini ditunjukkan pada **gambar 3.12b** dan **3.12c**.

Optimization Software:
www.balesio.com

(a) Hasil keseluruhan kontur (b) Kontur yang diduga karakter



(c) Hasil kontur yang diduga plat

Gambar 3.12 Hasil kontur deteksi plat

3.5.5 Klasifikasi KNN

Setelah memperoleh model data dari proses *labelling* dan *training*, langkah selanjutnya adalah menggunakan model tersebut untuk proses klasifikasiyang dilakukan dengan langkah-langkah sebagai berikut

1. Membaca file hasil *labelling*dan *training*, dalam hal ini file tersebut bernama classifications.txt dan flattened_images.txt.

Optimization Software:
www.balesio.com

2. Mengubah bentuk objek *training* yang telah dibaca menjadi bentuk *numpy array* 1 dimensi. Hal ini dilakukan agar objek bisa digunakan sebagai parameter dari fungsi yangdipanggil.
3. Proses *training* dimulai dengan memanggil fungsi kNearest.train() . Berikut adalah pemanggilan fungsi yang dimaksudkan.

```
kNearest.train(npaFlattenedImages, cv2.ml.ROW_SAMPLE, npa
Classifications)
```

Keterangan parameter :

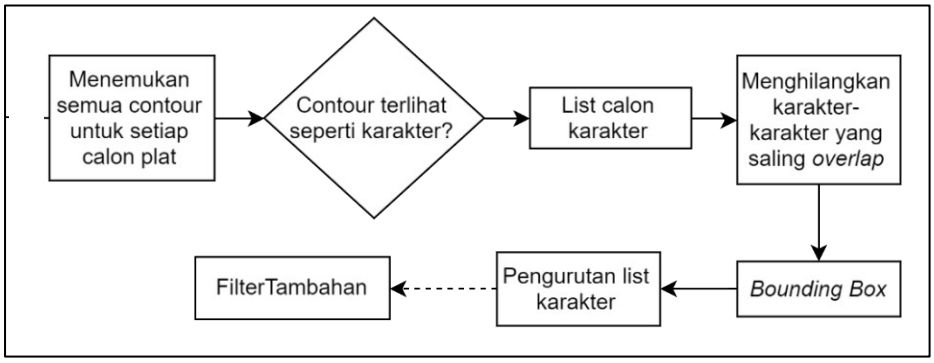
- npaFlattenedImages = Numpy array dari file *flattened_images*.txt.
- cv2.ml.ROW_SAMPLE = Variabel yang menentukan bahwa sampel data pada npaFlattenedImages akan menempati kolom.
- npaClassifications = Numpy array hasil klasifikasi.

4. *Training sample* siap digunakan untuk proses pengenalan.

3.5.6 Segmentasi dan Pengenalan Karakter

Setelah diperoleh list kontur-kontur calon plat yang telah melalui tahap *preprocessing*, langkah selanjutnya adalah seperti yang ditunjukkan pada Pada **gambar 3.13.**

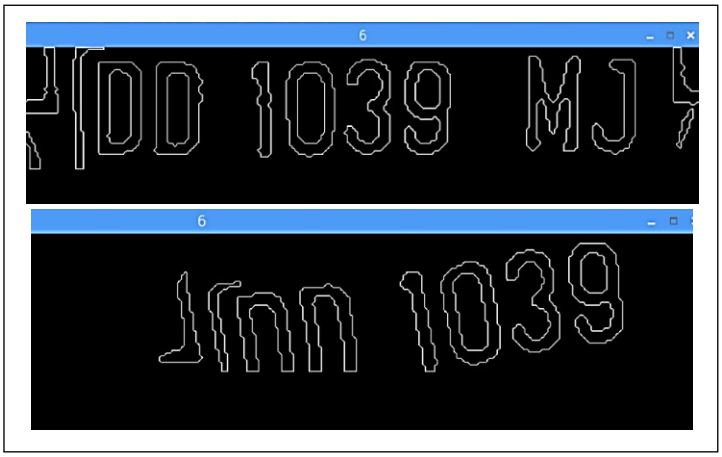
Optimization Software:
www.balesio.com



Gambar 3.13 Flowchart tahap pengenalan karakter

Berdasarkan **gambar 3.13**, langkah yang dilakukan adalah sebagai berikut :

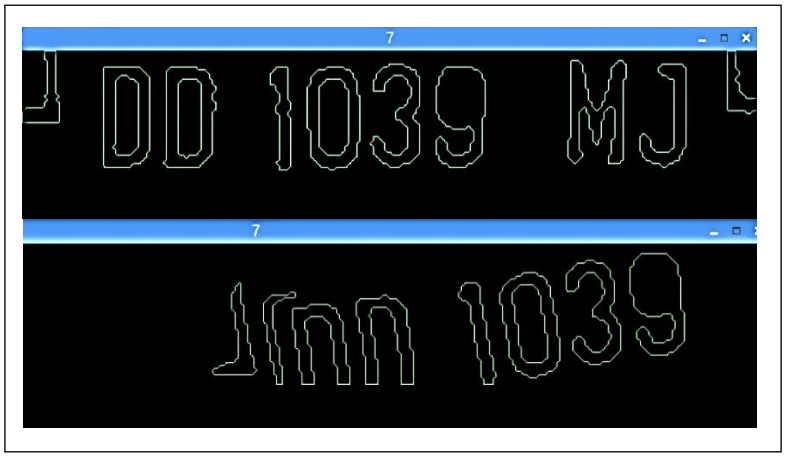
1. Menemukan semua kontur untuk masing-masing kandidatpada list calon plat dengan menggunakan fungsi findCountour() dari OpenCV. Contoh hasil tahap ini ditunjukkan pada **gambar 3.14.**



Gambar 3.14 Contoh hasil pencarian semua kontur pada setiap kandidat plat

. Jika kontur terlihat seperti karakter, maka dimasukkan ke dalam list calon karakter. Contoh hasil tahap ini ditunjukkan pada **gambar 3.15.**

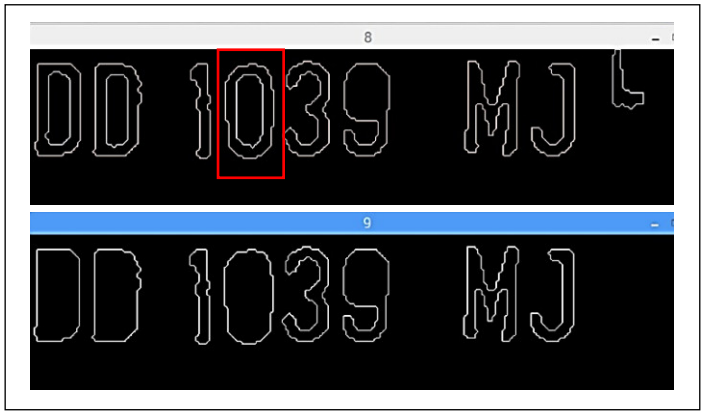
Optimization Software:
www.balesio.com



Gambar 3.15 Contoh kontur yang terlihat seperti karakter pada setiap kandidat plat

3. Menghapus karakter yang saling *overlap.* Untuk setiap kandidat karakter yang ditemukan, jika terdapat dua karakter yang saling tumpang tindih atau menutup satu sama lain, maka karakter bagian dalam (yang lebih kecil) dihapus. Hal ini dilakukan untuk mencegah memasukkan karakter yang sama dua kali jika dua kontur ditemukan untuk karakter yang sama. Contoh hasil tahap ini bisa dilihat pada **gambar 3.16.**Misalnya untuk angka '0' baik lingkaran bagian dalam dan lingkaran luar dapat ditemukan sebagai kontur, tetapi char hanya boleh dimasukkan sekali saja.

Optimization Software:
www.balesio.com



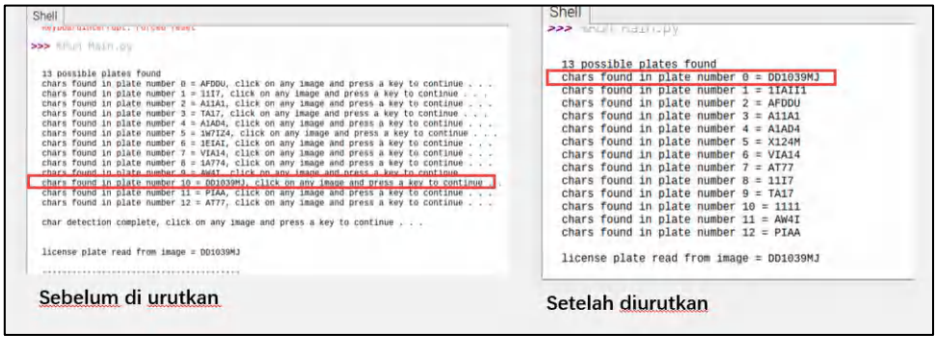
Gambar 3.16 Contoh hasil proses penghapusan karakter yang *overlap* (a) Contoh karakter yang *overlap,* (b) Hasil setelah karakter yang saling *overlap* dihapus

4. Bounding Box. Setelah membuang kontur-kontur yang saling *overlap*, maka diperoleh list yang berisi kandidat karakter untuk setiap kandidat plat. Namun, list kandidat ini harus diperkecil lagi hingga hanya tersisa satu kandidat karakter untuk setiap kandidat plat. Karena itu, kandidat karakter yang dipilih sebagai hasil akhir adalah kandidat dengan karakterterpanjang.. Setelah itu, bounding box akan digambar pada kontur yang dikenali sebagai karakter tersebut. Hasil tahap ini bisa dilihat pada **gambar 3.17.**

Gambar 3.17 Contoh kandidat karakter dengan *bounding box.*

Optimization Software:
www.balesio.com

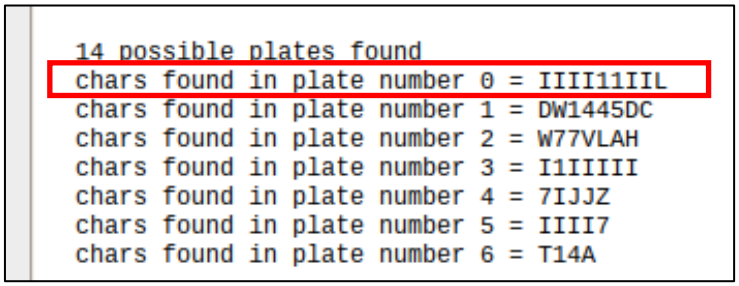
5. Pengurutan List Karakter. Setelah diperoleh satu kandidat karakter plat untuk setiap kandidat plat pada list, yang dilakukan selanjutnya adalah mengurutkan kembali list karakter tersebut dari karakter terpanjang hingga terpendek. Karakter terpanjang setelah tahap pengurutan akan berada pada index 0 dan diasumsikan sebagai hasil akhir plat yang sebenarnya (lihat **gambar 3.18).** Hal ini dilakukan agar di index berapapun karakter plat yang sebenarnya tersimpan, program tetap akan menghasilkan output dengan benar.



Gambar 3.18 Pengurutan list karakter untuk setiap kandidat plat.

Namun, proses pengurutan ini masih memiliki kekurangan. Jika ada objek selain plat yang dikenali dan memiliki karakter yang lebih panjang dari pada karakter plat sesungguhnya, maka hasil akhir akan salah.

Optimization Software:
www.balesio.com



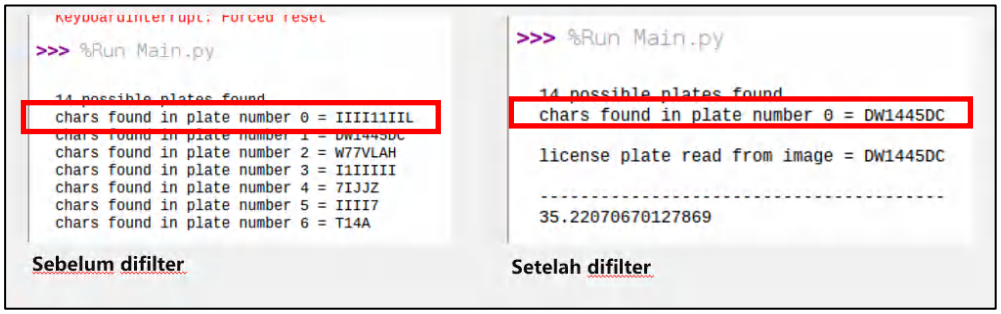
Gambar 3.19 contoh hasil deteksi yang salah.

Berdasarkan **gambar 3.19**, setelah pengurutan bisa dilihat bahwa karakter dengan index 0 memang adalah karakter terpanjang, tetapi bukan hasil yang sebenarnya. Maka program akan membaca karakter dengan indeks 0 yaitu IIII11IIL sebagai hasil akhir, padahal hasil yang sebenarnya seharusnya adalah DW1445DC.

6. Filter Tambahan.Karena masih adanya kemungkinan program salah memperoleh hasil akhir, maka filter tambahan ini diterapkan. Filter ini memastikan hasil akhir benar-benar adalah nomor plat yang sebenarnya dan jika selain itu,akandianggap bahwa plat tidak terdeteksi.

   Filter tambahan ini berjalan dengan memeriksa awalan setiap kandidat karakter. Jika karakter pada list mengandung awalan plat seperti DD, B, DP, DW, DN dan lain-lain, maka karakter tersebut dinyatakan lulus filter, kemudian dimasukkan ke dalam list baru dan dianggap sebagai hasil karakter yang benar.Hasil penerapan filter ini bisa dilihat pada **gambar 3.20**

Optimization Software:
www.balesio.com



Gambar 3.20 Hasil penerapan filter tambahan

Berdasarkan **gambar 3.20** terlihat bahwa sebelum diterapkannya filter, karakter yang berada pada index 0 adalah IIIIIIIIIL yang artinya karakter inilah yang akan dianggap sebagai hasil akhir pengenalan. Tetapi, setelah filter diterapkan, karakter-karakter yang tidak memenuhi syarat menjadi karakter plat telah hilang dari list, dimana yang tersisa hanya karakter plat yang sebenarnya.

3.5.7 Hasil Deteksi disimpan pada *Database*

Database digunakan untuk menyimpan data gambar yang dicapture oleh sistem kamera dan data hasil deteksi yang dilakukan oleh sistem *processing*. Struktur tabel yang digunakan dapat dilihat pada **gambar 3.21.**

#	Name	Type	Collation	Attributes	Null	Default
1	id	int(255)			No	*None*
2	image	longblob			No	*None*
3	date	timestamp			No	CURRENT_TIMESTAMP
4	stage	int(2)			No	0
5	result	varchar(50)	utf8mb4_general_ci		No	belum terdeteksi

Optimization Software:
www.balesio.com

Gambar 3.21 Struktur tabel deteksi_plat

Adapun penjelasan untuk masing-masing *field*pada **gambar 3.21**adalah sebagai berikut :

id: *Field* yang akan menyimpan *id* gambar yang diinput oleh sistem inputan. *Id* ini bertipe int dan akan otomatis bertambah sebanyak jumlah data yang dimasukkan *(auto increment).*

image : *Field* yang akan menyimpan informasi data gambar berupa *long blob.*

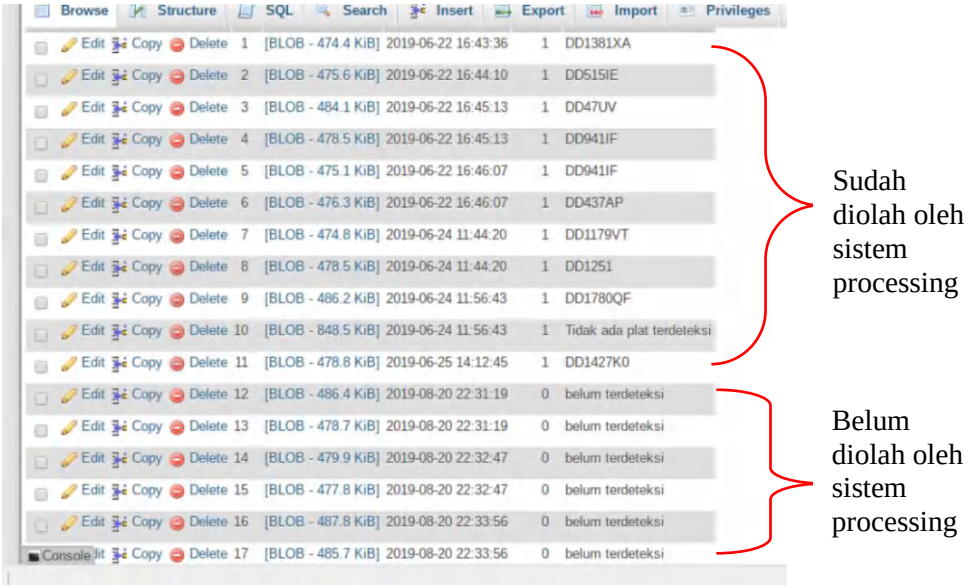
date: *Field* yang akan menyimpan informasi mengenai waktu pengambilan gambar oleh kamera.

stage : *Field* ini menyimpan informasi mengenai tahap pengolahan gambar. Ketika mengupload gambar baru di*upload* (belum diolah), *field* ini secara default akan bernilai 0 yang artinya gambar pada *id* tersebut belum diproses atau diolah oleh sistem *processing*. Ketika gambar telah diolah dan sistem *processing* berhasil mengupload hasil pengenalan karakternya, maka *stage* akan berubah menjadi nilai 1 (yang artinya gambar sudah diolah).

result :*Field* yang akan menyimpan informasi mengenai hasil pengenalan plat yang dilakukan oleh sistem *processing*. Ketika *stage* masih bernilai 0 (gambar baru diupload ke *database*), *field* ini secara *default* bernilai “belum terdeteksi”. Ketika *stage* bernilai 1, ada dua kemungkinan yaitu *field* ini akan berisi nomor plat yang

Optimization Software:
www.balesio.com

berhasil dikenali atau akan berisi “bukan plat yang terdeteksi” jika sistem prosesing gagal mendeteksi plat pada gambar (Lihat **gambar 3.22**).



Gambar 3.22 Tampilan database sebelum dan setelah diolah oleh sistem *processing*

3.6 Evaluasi Kinerja

Proses evaluasi ini dilakukan dengan menggunakan parameter pengaruh nilai kernel *Gaussian Blur* terhadap akurasi pendeteksian dan waktu processing serta kinerja Mini PC dalam mengeksekusi algoritma ALPR (dengan menghitung *system time*, *user time*, dan *idle time*).

Optimization Software:
www.balesio.com

3.6.1 Perubahan Nilai Kernel *Gaussian Blur* terhadap Akurasi Pengenalan Plat

1. Untuk mengetahui waktu yang dibutuhkan program untuk mengenali karakter plat dari suatu gambar, pengujian dilakukan dengan menggunakan parameter pengaruh nilai kernel *Gaussian Blur* terhadap akurasi pendeteksian. Nilai kernel yang digunakan adalah 3x3, 5x5, 7x7, 9x9 dan 11x11. Evaluasi dilakukan dengan mengambil data citra dari database kemudian data diuji sebanyak 5 kali dengan merubah parameter nilai kernel yang digunakan dalam tahap *Gaussian Blur* (*preprocessing*).
2. Untuk menghitung presentase akurasi dalam mengenali karakter plat digunakan dua kondisi, yaitu
    a. Pertama, kondisi karakter plat dikenali dengan benar oleh sistem, dan
    b. kedua kondisi plat gagal dikenali dengan benar oleh sistem.

Berikut persamaan yang digunakan berdasarkan dua kondisi tersebut.

$$Akurasi = \frac{JB}{JK}\ x\ 100\% \qquad (3.2)$$

Keterangan:

1. JB = Jumlah citra yang dikenali dengan benar
2. JK = Jumlah citra secara keseluruhan

Optimization Software:
www.balesio.com

3. Dalam penelitian ini, selain diperoleh jumlah citra plat yang berhasil dikenali, juga akan diperoleh lama waktu program mengeksekusi masing-masing citra.Hal ini dilakukan untuk mengetahui nilai kernel mana yang membutuhkan waktu eksekusi paling lama dan apakah waktu eksekusi tersebut berpengaruh terhadap tingkat akurasi .
    a. Untuk memperoleh nilai waktu eksekusi digunakan perintah time.time() di bagian awal program (setelah berhasil mengakuisi citra dari database) dan dibagian akhir program (setelah output pendeteksian ditampilkan).
    b. Kemudian dihitung selisih keduanya. Hal ini dilakukan untuk memperoleh waktu eksekusi yang sebenarnya tanpa dipengaruhi oleh faktor lama waktu citra diakusisi dari database. Berikut penulisan program yang dimaksudkan :

```
//Akuisisi citra dari database
Response = requests.get('http://localhost/getImage2.php').text
print(response)

blnKNNTrainingSuccessful = DetectChars.loadKNNDataAndTrainKNN()
start = time.time()  //Mulai menghitung waktu
.
.  //perintah-perintah untuk mengenali karakter
.
.
plateResult = detectPlate(imgOriginalScene)
end = time.time() //Hentikan perhitungan waktu
print(end-start) // Tampilkan waktu eksekusi
```

4. Setelah memperoleh waktu eksekusi dari program, selanjutnya dapat dilakukan profilling lebih lanjut mengenai di bagian mana saja program menghabiskan banyak waktu atau fungsi-fungsi

Optimization Software:
www.balesio.com

apa saja yang paling banyak dipanggil. *Profilling* ini dilakukan sebagai referensi tambahan jika nantinya ingin mengoptimalkan kinerja dari program tersebut. Pada proses ini digunakan *built-in* modul dari python yang bernama cProfile. Adapun perintah yang dilakukan adalah sebagai berikut:

```
pi@raspberrypi:~ $python3 -m cProfile script.py
```

3.6.2 Kinerja MiniPC dengan Parameter System Time, User Time dan Idle Time

Untuk mengevaluasi kinerja Mini PC dalam penerapan algoritma pengenalan plat pada citra mobil dilakukan dengan modul *mpstat* pada terminal raspberry. Proses pengujian dilakukan dengan dua keadaan yaitu :

1. Mengamati kinerja CPU Mini PC saat tidak melakukan ekseksusi program (hanya startup program yang berjalan). Keadaan diamati setiap 20 detik.
2. Mengamati kinerja CPU Mini PC saat mengeksekusi program dengan selang waktu 20 detik.Proses ini dilakukan dengan membuka dua terminal, dimana terminal 1 akan mengeksekusi program ALPR dan sementara terminal 1 berjalan, terminal 2 akan melakukan profiling terhadap penggunaan CPU. Keadaan diamati setiap 20 detik selama program dieksekusi.

Optimization Software:
www.balesio.com

Evaluasi dengan kedua keadaan ini akan dilakukan dengan menuliskan perintah sebagai berikut :

```
pi@raspberrypi:~ $mpstat -P ALL 20
```

Perintah ini akan menghasilkan informasi mengenai kinerja setiap CPU dan rata-ratanya dengan selang waktu 20 detik. Terdapat banyakvariabel waktu, tetapi yang akan menjadi fokus utama dalam penelitian ini yaitu *system time*, *user time* dan *idle time*. *System time*adalah waktu yang dihabiskanCPU untuk menjalankan kode di kernel sistem operasi dengan melakukan tindakan yang diminta oleh proses atas nama program yang dijalankan. Waktu ini dijumlahkan dari semua CPU yang digunakan.*User time* adalah waktu yang dihabiskan oleh CPU untuk menjalankan program atau proses, dijumlahkan dari semua CPU yang digunakan. Sedangkan *idle time* adalah waktu dimana CPU aktif tetapi tidak melakukan aktifitas.

Optimization Software:
www.balesio.com