

BAB I

PENDAHULUAN

A. Latar Belakang

Seiring meningkatnya jumlah kendaraan di Indonesia, semakin meningkat pula kejahatan yang terjadi. Salah satu kejahatan yang sering terjadi adalah pencurian kendaraan bermotor. Saat ini kejahatan tersebut sudah tertata rapi dan juga mempunyai jaringan yang terorganisir. Dengan semakin terorganisirnya kejahatan pencurian kendaraan bermotor, maka tidak hanya terbatas sampai pencurian kendaraan bermotor saja melainkan sampai pemalsuan surat kendaraan bermotor seperti Surat Tanda Nomor Kendaraan (STNK). Masyarakat semakin sulit mengetahui keaslian STNK dengan cepat dan mudah. Masyarakat juga perlu berhati-hati ketika akan membeli sepeda motor bekas. Di sisi lain, pihak Satuan Administrasi Manunggal Satu Atap (SAMSAT) Jakarta sebagai pihak yang bertanggung jawab terhadap pencatatan nomor kendaraan, telah menyediakan website yang menyediakan informasi nomor kendaraan yang telah terdaftar. Alamat website tersebut adalah http://www.samsat-pkb.jakarta.go.id/INFO_PKB. Pemilik kendaraan atau calon pembeli kendaraan dapat mengetahui keaslian STNK dan informasi nomor kendaraan dengan memeriksa nomor kendaraan pada website SAMSAT tersebut. Pengguna harus membuka website dan memasukkan secara manual nomor kendaraannya. Pada penelitian ini, dirancang sebuah aplikasi berbasis mobile yang dapat mempermudah penggunaanya dalam mengakses

informasi kendaraan berdasarkan plat nomor kendaraan tersebut. Aplikasi dirancang sedemikian rupa agar mudah diakses dimanapun dan kapanpun. Dengan aplikasi ini diharapkan dapat meningkatkan efisiensi dan efektifitas dalam mendapatkan informasi nomor kendaraan. Pengguna juga akan lebih mudah mengetahui keaslian nomor kendaraan atau surat nomor kendaraan. Proses identifikasi nomor kendaraan dilakukan dengan memanfaatkan kamera yang tersedia di perangkat mobile dan menggunakan metode pengolahan citra digital. Nomor kendaraan diidentifikasi dengan metode Optical Character Recognition (OCR) (Patel dkk., 2012) dan Automatic Number Plat Recognition (ANPR). Proses identifikasi diawali dengan pengambilan citra melalui kamera yang terdapat pada perangkat Python. Selanjutnya dilakukan proses segmentasi, ekstraksi ciri dan pengenalan karakter. Untuk mengenali karakter pada nomor kendaraan, dilakukan proses klasifikasi menggunakan metode Learning Vector Quantization (LVQ). Pengujian terhadap aplikasi dilakukan untuk mendapatkan tingkat akurasi proses identifikasi nomor kendaraan.

B. Rumusan Masalah

Dari latar belakang masalah yang ditemukan diatas maka pokok permasalahan yang terdapat dalam penelitian ini adalah Sebagai Berikut:

1. Bagaimana membuat suatu Aplikasi Pengenalan Plat kendaraan berbasis Python untuk identitas kepemilikan kendaraan?
2. Bagaimana cara mengetahui data Plat kendaraan berbasis Python untuk identitas kepemilikan kendaraan?

C. Batasan Masalah

Batasan masalah dalam penelitian tugas akhir ini adalah sebagai berikut :

1. Aplikasi yang dirancang pada aplikasi Python.
2. Data data Plat kendaraan berbasis Python untuk identitas kepemilikan kendaraan

D. Tujuan Penelitian

Adapun tujuan yang ingin dicapai pada penelitian ini yaitu mengetahui data Plat kendaraan berbasis Python untuk identitas kepemilikan kendaraan.

E. Manfaat Penelitian

1. Manfaat bagi mahasiswa

Sebagai penerapan teori yang telah di dapatkan dari bangku kuliah dan diharapkan dapat menambah pengetahuan dan wawasan tentang bagaimana cara membuat aplikasi pada Python .

2. Manfaat bagi penulis

Menambah pengetahuan, pengalaman, serta wawasan tentang metode perkembangan dan melatih diri dalam menyelesaikan masalah untuk maju dan berkembang.

BAB II

TINJAUAN PUSTAKA

A. Penelitian Terdahulu

Pada penelitan sebelumnya sudah ada sistem pengenalan plat dengan wajah manusia menggunakan metode *Hidden Markov Models* (HMM) oleh Sepritahara mencapai tingkat akurasi pengenalan sebesar 84,28%, dengan *database* 70 gambar yang terdiri dari 10 individu dengan masing-masing individu memiliki 7 variasi ekspresi yang berbeda.

Pada penelitian selanjutnya juga sudah dilakukan deteksi plat dengan penulis Esty dkk membuat aplikasi yang mampu pendeteksian beberapa wajah dengan berbagai posisi pada foto menggunakan metode *eigenface*. Posisi pengujian yang dilakukan adalah tampak depan, atas, bawah, kanan dan kiri, Plat, pencahayan dan latar belakang. Akurasi PCA yang diteliti oleh Dian Esti Pratiwi dan Agus Harjoko pada tahun 2013 dengan judul Implementasi Pengenalan Wajah Menggunakan PCA mempunyai akurasi sebesar 82,81%. Eko dkk melakukan pengujian terhadap metode chi square dan Local binary Pattern (LBP) dengan menggunakan satu wajah secara bergantian.

Hasil dari penelitian mengatakan bahwa metode *chi square* lebih baik jika dibandingkan dengan metode PCA untuk fitur *Local binary Pattern* (LBP) yaitu tingkat pengenalan mencapai 92,4 %. Penelitian dilakukan oleh *Timo Ahonen*,

Abdenour Hadid, and Matti Pietikäinen dengan Face Recognition with Local Binary Patterns University of Oulu, Finland dan penelitian menghasilkan akurasi 93%.

B. Plat Nomor Kendaraan

Plat nomor kendaraan berfungsi sebagai tanda identifikasi atau registrasi kendaraan. Plat nomor kendaraan biasanya terdiri atas susunan kombinasi huruf dan angka yang sifatnya unik dan khusus untuk kendaraan tersebut. Huruf dan angka yang terdapat pada plat nomor kendaraan atau nomor polisi tersebut menunjukkan informasi mengenai kendaraan seperti model, tahun pembuatan, wilayah kendaraan dan lainnya. Plat nomor atau nomor polisi berjumlah dua buah atau sepasang dan wajib dipasang di bagian depan dan belakang kendaraan.

Karena fungsinya sebagai tanda identifikasi kendaraan, pemasangan nomor polisi di bagian depan dan belakang kendaraan harus sesuai aturan dan tidak boleh ditutupi atau terhalang benda lain. Angka dan huruf pada plat nomor kendaraan yang dipasang harus jelas terlihat.

1. Jenis Plat Nomor Kendaraan

Di jalanan umum, mungkin banyak dari kita hanya pernah melihat kendaraan dengan plat warna hitam, kuning atau merah melintas. Umumnya plat hitam digunakan untuk kendaraan pribadi, plat kuning untuk kendaraan umum, serta plat merah untuk kendaraan pemerintah.



Adapun jenis plat nomor kendaraan yang ada antara lain:

1. Dasar hitam dan tulisan putih, digunakan untuk kendaraan bermotor perseorangan dan sewa. Akan tetapi jenis plat ini akan digantikan dengan warna plat putih dan tulisan hitam sehingga ke depannya semua kendaraan perseorangan dan sewa akan menggunakan plat dasar putih dan tulisan hitam.
2. Dasar kuning dan tulisan hitam digunakan untuk kendaraan bermotor umum
3. Dasar merah dan tulisan putih digunakan untuk kendaraan operasional milik instansi pemerintah
4. Dasar putih dan tulisan hitam CD / CC digunakan untuk kendaraan dinas kedutaan besar negara asing
5. Plat dengan garis biru menandakan kendaraan tersebut adalah kendaraan listrik yang ramah lingkungan
6. Dasar hijau dengan tulisan hitam secara khusus digunakan untuk kendaraan di kawasan *FTZ / Free Trade Zone*

2. Arti Huruf & Angka Pada Plat Kendaraan

Apabila warna dasar plat kendaraan memiliki arti tersendiri, maka baik huruf maupun angka yang terdapat pada plat nomor kendaraan juga memiliki arti dan informasi tertentu loh.



Nomor polisi yang ada pada kendaraan di Indonesia umumnya dimulai dengan kode huruf yang menandakan domisili atau wilayah kendaraan tersebut terdaftar. Kode huruf tersebut diikuti dengan nomor urut dan setelah nomor diikuti dengan huruf abjad. Contoh: B 2234 PAI, Huruf pertama atau B menunjukkan bahwa kendaraan tersebut terdaftar di wilayah DKI Jakarta, Depok, Tangerang dan Bekasi. Sedangkan huruf di belakang setelah nomor urut memiliki fungsi dan arti masing – masing, antara lain:

- a) Huruf P merupakan wilayah Jakarta Pusat
- b) Huruf A merupakan penanda kendaraan tersebut termasuk jenis sedan / pickup

- c) Huruf I atau huruf abjad terakhir berfungsi sebagai pembeda saja dan tidak ada artinya

Secara lebih lengkap, kode abjad pertama di akhir di nomor polisi memiliki arti berikut:

- 1) B – wilayah Jakarta Barat
- 2) C – wilayah Kota Tangerang
- 3) E – wilayah Depok
- 4) F – wilayah Kabupaten Bekasi
- 5) G – wilayah Kabupaten Tangerang SAMSAT Tigaraksa
- 6) K – wilayah Kota Bekasi
- 7) N – wilayah Kabupaten Tangerang SAMSAT BSD
- 8) P – wilayah Jakarta Pusat
- 9) S – wilayah Jakarta Selatan
- 10) T – wilayah Jakarta Timur
- 11) U – wilayah Jakarta Utara
- 12) V – wilayah Kota Tangerang SAMSAT Ciledug
- 13) W – wilayah Kota Tangerang Selatan
- 14) Z – wilayah Kota Depok

Masing-masing provinsi memiliki ketentuan kode huruf yang berbeda-beda untuk mengidentifikasi daerah kendaraan terdaftar. Sedangkan kode abjad kedua memiliki arti berikut:

- 1) A – plat nomor berjenis Sedan/Pick Up
- 2) D – plat nomor berjenis Truk

- 3) F – plat nomor berjenis Minibus, Hatchback, City Car
- 4) J – plat nomor berjenis Jip dan SUV
- 5) Q – plat nomor staf pemerintahan
- 6) T – plat nomor berjenis Taksi
- 7) U – plat nomor staf pemerintahan
- 8) V – plat nomor berjenis Minibus

C. Tanda Nomor Kendaraan Bermotor

Berdasarkan Peraturan Kepala Kepolisian RI Nomor 5 Tahun 2012 Tentang Registrasi dan Identifikasi Kendaraan Bermotor, Tanda Nomor Kendaraan Bermotor (TNKB) atau sering disebut plat nomor merupakan tanda *regident* Ranmor yang berfungsi sebagai bukti legitimasi pengoperasian Ranmor berupa pelat atau berbahan lain dengan spesifikasi tertentu yang diterbitkan Polri dan berisikan kode wilayah, nomor registrasi, serta masa berlaku dan dipasang pada Ranmo. Secara teknis, spesifikasi TNKB berupa plat aluminium dengan cetakan tulisan dua baris. Baris pertama menunjukkan kode wilayah (huruf), nomor polisi (angka), dan kode/seri akhir wilayah (huruf). Baris kedua menunjukkan bulan dan tahun masa berlaku. Bahan baku TNKB adalah aluminium dengan ketebalan 1 mm. Plat TNKB untuk kendaraan bermotor roda 2 dan roda 3 adalah 250×105 mm, sedangkan untuk kendaraan bermotor roda 4 atau lebih adalah 395×135 mm. Terdapat cetakan garis lurus pembatas lebar 5 mm di antara ruang nomor polisi dengan ruang angka masa berlaku (yang lama), sedangkan yang baru terdapat garis putih di sekitar TNKB dan tidak ada batas pemisah antara nomor polisi dan masa

berlaku (dari tahun 2011). TNKB memiliki 2 macam warna yang umum dipakai, yang pertama memiliki warna latar hitam dan warna depan putih untuk perseorangan. Warna latar kuning dengan warna depan hitam untuk umum.

D. Optical Character Recognition dan Automatic Number Plate Recognition

Emanuel Goldberg 2018, *Optical Character Recognition* (OCR) adalah teknik mengubah sebuah gambar berisi teks, tulisan tangan menjadi teks yang dapat diubah untuk proses selanjutnya, pertama kali ditemukan oleh *Emanuel Goldberg* pada akhir dekade 1920, yang dipakai untuk mencari arsip film mikro. Teknologi ini memungkinkan mesin untuk mengenali teks secara otomatis. Teknik ini seperti kombinasi dari mata dan otak manusia. Sebuah mata dapat melihat sebuah teks dari gambar tetapi sebenarnya otak yang memproses dan mengolah teks tersebut sehingga dapat dibaca oleh mata. Dalam pengembangan OCR terdapat beberapa masalah, diantaranya yang pertama adalah terdapat sedikit perbedaan antara beberapa huruf dengan angka yang sulit dibedakan. Contohnya antara angka 0 dengan huruf o. Faktor kedua yaitu faktor cahaya dalam citra yang mempersulit sistem dalam mengenali huruf. Sementara itu, *Automatic Number Plate Recognition* (ANPR) atau *Plate Number Recognition* adalah bentuk khusus dari *Optical Character Recognition*. ANPR adalah jenis teknologi, yaitu perangkat lunak yang memungkinkan sistem komputer dapat membaca nomor kendaraan secara otomatis dari citra digital. Yang dimaksud dengan dengan membaca nomor kendaraan secara otomatis adalah mengubah pixel dari citra digital ke dalam karakter ASCII dari plat nomor.

E. Pengertian Aplikasi

Aplikasi menurut Jogiyanto (2018:12) adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output. Menurut Kamus Kamus Besar Bahasa Indonesia (2018 : 52) , “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu”. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer. Program merupakan kumpulan *instruction set* yang akan dijalankan oleh pemroses, yaitu berupa *software*. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini. Program inilah yang mengendalikan semua aktifitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada *instruction set* . Program aplikasi merupakan program siap pakai. Program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh-contoh aplikasi ialah program pemroses kata dan *Web Browser*. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung. Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak tahun 1993, yang biasanya juga disingkat dengan app. Secara *historis*, aplikasi adalah *software* yang dikembangkan oleh sebuah perusahaan. *App* adalah *software* yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya

menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang paralel dengan yang terjadi antar sistem operasi yang dimunculkan.

F. Metode *Canny Edge Detection*

Deteksi tepi adalah proses mengidentifikasi dan menemukan diskontinuitas tajam dalam gambar. Deteksi tepi telah digunakan oleh pengenalan obyek, pelacakan sasaran, deteksi obyek dan lain - lain. Pada penelitian ini, metode deteksi tepi yang digunakan adalah metode *Canny*. Metode *canny* adalah salah satu metode deteksi tepi untuk menemukan tepi dari citra masukan tanpa mempengaruhi fitur dari tepi (Nisha, 2015). Metode *Canny* terdiri dari beberapa langkah, yaitu menghilangkan *noise* dengan gaussian blur, mencari tepi dengan menggunakan sobel operator dan menentukan arah garis tepi, melakukan penipisan tepi dengan *Non Maximum Supression* dan *hysteresis*. Contoh hasil dari deteksi tepi Canny dapat dilihat dari gambar berikut:



Gambar 2.1 Hasil dari Proses *Canny Edge Detection*

1. Kalibrasi

Menurut ISO/IEC Guide 17025 (2007) dan *vocabulary of International Metrology* (VIM), kalibrasi adalah serangkaian kegiatan yang membentuk hubungan antara nilai yang ditunjukkan oleh instrument ukur atau *system*

pengPlat, atau nilai yang diwakili oleh bahan ukur, dengan nilai – nilai yang sudah diketahui yang berkaitan dari besaran yang diukur dalam kondisi tertentu. Dengan kata lain, kegiatan ini untuk menentukan kebenaran konvensional nilai penunjukan alat ukur dan bahan ukur dengan cara membandingkan terhadap standar ukur yang mampu telusur (*traceable*) ke standar nasional maupun internasional. Tujuan kalibrasi ini adalah :

- 1) Untuk menentukan deviasi (penyimpangan) kebenaran nilai konvensional penunjukan suatu instrument ukur
- 2) Menjamin hasil – hasil pengPlat

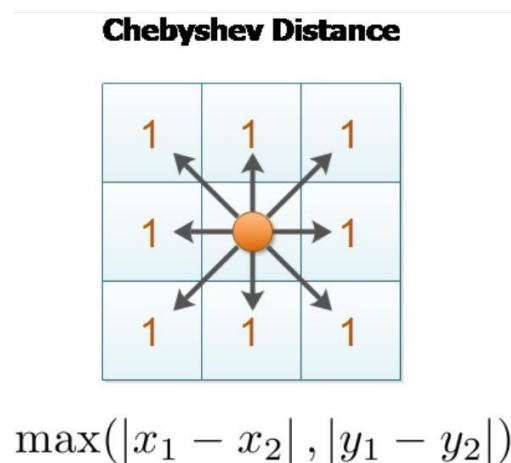
Hasil kalibrasi antara lain :

- 1) Nilai objek ukur
- 2) Nilai koreksi / penyimpangan
- 3) Nilai ketidakpastian pengPlat (besarnya kesalahan yang mungkin terjadi dalam pengPlat, dievaluasi setelah ada hasil pekerjaan yang diukur & analisis ketidakpastian yang benar dengan memperhitungkan semua sumber ketidakpastian yang ada dalam metode yang digunakan serta besarnya kesalahan yang mungkin terjadi dalam pengPlat)
- 4) Sifat metrology lain seperti factor kalibrasi, krva kalibrasi

2. Chess board

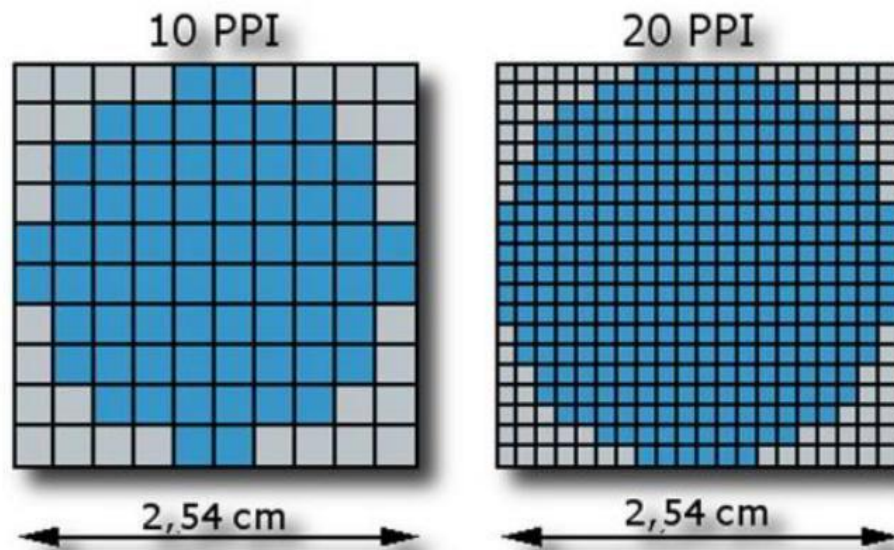
Dinamai demikian *Pafnuty Chebyshev*, seorang ahli matematika Rusia yang merupakan penerima hadiah Demidov dan dikenal karena karyanya tentang *probabilitas, statistic, mekanika, geometri analitik* dan teori bilangan.. konsep jarak *chebyshev* ditangkap oleh bilangan ini :

Jarak digunakan untuk menentukan tingkat kesamaan (*similarity degree*) dua *vector* fitur. Tingkat kesamaan berupa suatu nilai (*score*) dan berdasarkan skor tersebut dua *vector* fitur akan dikatakan mirip atau tidak. Salah satu cara yang digunakan untuk mengukur jarak dua buah titik pada citra, yaitu metode *chess board*. Metode *chebyshev distance* yang disebut dengan *chessboard distance* adalah sama dengan pergerakan king pada permainan catur yang bergerak dari satu titik ke titik yang lainnya.



Gambar 2.2 Konsep jarak *Chebyshev / Chess board*

F. Pixel Per Inch (PPI)



Gambar 2.3. PPI

Kepadatan *piksel* atau dalam bahasa Inggris *pixel density* yang biasanya dinyatakan dalam satuan *pixel per inch* (ppi) adalah indikasi penting yang memberitahu Anda tentang seberapa jelas layar dari sebuah perangkat elektronik.

Semakin besar nilai ppi maka bisa diartikan kepadatan atau kerapatan *piksel* dalam satuan area inci semakin tinggi. Hasilnya, layar yang akan Anda lihat akan semakin halus dan jelas.

Sedikit keliru jika ppi didefinisikan sebagai kepadatan *piksel*, ini sama dengan Anda mengartikan “cm” sebagai panjang, padahal panjang sendiri tidak hanya bisa dinyatakan dalam “cm” tetapi bisa dengan meter, km dan lain sebagainya.

Lebih tepatnya, ppi adalah satuan yang menyatakan kepadatan *piksel* sebuah layar. Bayangkan layar ponsel adalah sebuah grid, di mana *piksel* akan menempati setiap baris dan kolom di dalamnya.

Layar HD diukur dari segi jumlah *piksel* yang melintas sepanjang layarnya, sehingga dalam contoh Samsung Galaxy Mega 6.3 ini disebut sebagai layar 720p. Sebuah layar Full HD terdiri dari 1080 *piksel* sepanjang layarnya, sehingga Sony Xperia Z Ultra digambarkan memiliki tampilan Full HD 1080p.

1. Cara Menghitung PPI Sebuah Layar

Dalam istilah awam, ppi mengacu pada jumlah *piksel* yang tersebar di seluruh area permukaan layar, tetapi karena layar diukur secara diagonal maka tidak bisa sesederhana mengalikan jumlah pixel mendatar dengan jumlah *piksel* yang menurun dan kemudian membaginya dengan pengPlat diagonal.

Agar mudah dan cepat, Anda dapat menggunakan [kalkulator ppi](#) siap pakai untuk mengetahui nilai ppi dari layar apapun tanpa terlibat dalam *Teorema Pythagoras*. Uniknya ppi adalah, bahwa semakin besar layar maka semakin rendah ppi tersebut. Hal ini disebabkan karena layar yang lebih besar membuat tiap individu *piksel* menjadi lebih besar (meregang) agar dapat mengisi seluruh ruang. meskipun memiliki jumlah *piksel* yang sama, tetapi ketika **Plat layarnya** berbeda maka akan berbeda pula nilai kepadatan *piksel*. Dengan resolusi yang sama, semakin kecil layar maka *piksel* akan terkonsentrasi pada area yang lebih kecil sehingga akan semakin tinggi nilai kepadatan *piksel*.

2. Cara Manual Menghitung ppi

Secara teoritis, ppi dapat dihitung dengan mengetahui Plat diagonal dari sebuah layar dalam satuan inci dan resolusi layar dalam satuan *piksel* (lebar dan tinggi). Menghitung ppi dapat dilakukan dengan dua langkah.

- a. Menghitung resolusi diagonal dalam satuan *piksel* menggunakan teorema Pythagoras

$$d_p = \sqrt{w_p^2 + h_p^2}$$

- b. Menghitung ppi dengan rumus berikut

$$PPI = \frac{d_p}{d_i}$$

dimana:

- 1) **dp** adalah resolusi diagonal dalam satuan *piksel*
- 2) **wp** adalah resolusi lebar (horizontal) layar dalam satuan *piksel*
- 3) **hp** adalah resolusi tinggi (vertikal) layar dalam satuan *piksel*
- 4) **di** adalah Plat diagonal dalam satuan inci

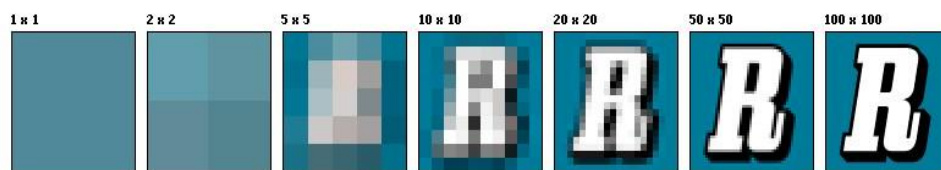
Contoh:

1. Sebuah perangkat dengan Plat layar 21,5 inci dan *resolusi* Full HD (1920×1080) akan memiliki kepadatan *piksel* 102,46 ppi (di mana **wp** = 1920, **hp** = 1080 dan **di** = 21,5) → $\text{akar}(1920^2 + 1080^2) / 21,5 = \mathbf{102,46 \text{ ppi}}$.
2. Perangkat dengan Plat layar 10,1 inci dan resolusi 1024×600 *piksel* akan memiliki kepadatan *piksel* 117,5 ppi (di mana **wp** = 1024, **hp** = 600 dan **di** = 10,1) → $\text{akar}(1024^2 + 600^2) / 10,1 = \mathbf{117,5 \text{ ppi}}$.
3. Dengan cara yang sama, perangkat dengan layar 27 inci dan resolusi 2560×1440 *piksel* akan menghasilkan kepadatan *piksel* **109 ppi**.

G. Pixel dan Resolusi Gambar

Pixel atau *picture elements* merupakan unsur gambar atau representasi sebuah titik terkecil dalam sebuah gambar grafis yang dihitung per inci. Semakin tinggi jumlah *pixel* dalam suatu citra, maka semakin tajam citra tersebut.

Selain *pixel*, terdapat pula resolusi gambar, yang menentukan kualitas suatu citra. Resolusi gambar mendeskripsikan tentang banyaknya detil gambar yang tersimpan, atau sering didefinisikan sebagai jumlah *pixel* dalam pencitraan gambar digital. Semakin besar nilai resolusi, maka semakin tajam gambar yang diperlihatkan seperti Gambar 2.5.



Gambar 2.4. Contoh Perbedaan Resolusi Gambar

1. Grayscale

Suatu citra *grayscale* adalah suatu citra yang hanya memiliki warna tingkat keabuan. Penggunaan citra *grayscale* dikarenakan membutuhkan sedikit informasi yang diberikan pada tiap *pixel* dibandingkan dengan citra berwarna. Warna abu-abu pada citra keabuan adalah warna R (*Red*), G (*Green*), B (*Blue*) yang memiliki intensitas yang sama. Sehingga dalam *grayscale image* hanya membutuhkan nilai intensitas tunggal dibandingkan dengan citra berwarna membutuhkan tiga intensitas untuk tiap *pixel*nya.

Pada umumnya citra *grayscale* memiliki penyimpanan 8 bit sehingga dapat menampung 256 tingkatan skala abu-abu dimana tiap *pixel* memiliki intensitas 0 hingga 255 dengan 0 menjadi hitam dan 255 menjadi putih. Contoh dari *grayscale image*

Grayscale dilakukan dengan cara mengambil semua *pixel* pada gambar kemudian warna tiap *pixel* akan diambil informasi mengenai 3 warna dasar yaitu merah, hijau dan biru, ketiga warna dasar ini akan dijumlahkan kemudian dibagi tiga sehingga didapat nilai rata-rata. Nilai rata-rata inilah yang akan dipakai untuk memberikan warna pada *pixel* gambar sehingga warna menjadi *grayscale* seperti pada persamaan dibawah ini.

$$S = \frac{r+g+b}{3}$$

Dimana : S = Citra *grayscale*
 R = nilai *red* dari sebuah *piksel*
 G = nilai *green* dari sebuah *piksel*
 B = nilai *blue* dari sebuah *piksel*

2. *Thresholding*

Thresholding merupakan teknik binerisasi yang digunakan untuk mengubah citra keabuan menjadi citra biner. *Thresholding* dapat digunakan dalam proses segmentasi citra untuk mengidentifikasi objek yang diinginkan dari *background* berdasarkan distribusi tingkat keabuan atau tekstur citra. Proses *thresholding*

menggunakan nilai batas (*threshold*) untuk mengubah nilai *piksel* pada *grayscale image* menjadi hitam atau putih.

Citra hasil berupa citra hitam dan putih, citra hitam (0) diperoleh apabila nilai *piksel* citra keabuan lebih kecil dari nilai *threshold* sebaliknya citra putih (1) diperoleh jika nilai *piksel* citra keabuan lebih besar dari nilai *threshold*. Pada proses *thresholding* citra dibagi kedalam beberapa bagian, masing-masing dari bagian tersebut memiliki Plat yang sama besar dengan sebuah citra dibagi dengan derajat keabuan (Aruan, 2017). Persamaan yang digunakan terdapat pada persamaan dibawah ini.

$$g(x,y) \begin{cases} 1 \text{ if } (x,y) > T \\ 0 \text{ if } F(x,y) \leq \end{cases}$$

Dimana : $g(x,y)$ = *piksel* citra hasil binerisasi

$f(x,y)$ = *piksel* citra asal

T = nilai *threshold*

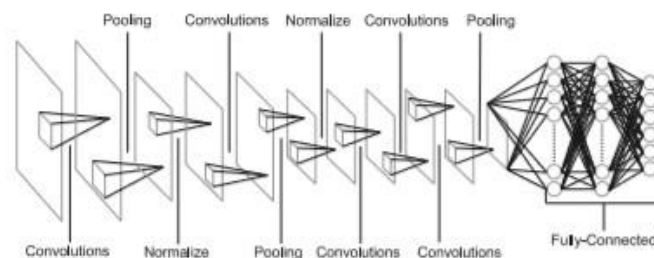
3. *Feature Extraction*

Fitur merupakan karakteristik unik dari suatu objek. Fitur dibedakan menjadi dua yaitu fitur “alami” merupakan bagian dari gambar, misalnya kecerahan dan tepi objek. Sedangkan fitur “buatan” merupakan fitur yang diperoleh dengan operasi tertentu pada gambar, misalnya histogram tingkat keabuan. Sehingga ekstraksi fitur adalah proses untuk mendapatkan ciri-ciri pembeda yang membedakan suatu objek dari objek yang lain (Putra,2010).

4. *Convolutional Neural Network*

Convolutional Neural Network (CNN) adalah pengembangan dari *Multilayer Perceptron* (MLP) yang didesain untuk mengolah data dua dimensi yang terdiri dari lapisan input dan output, serta beberapa lapisan tersembunyi (Yao, 2019). CNN menjadi sangat populer setelah memenangkan *ImageNet Large Scale Recognition Challenge* ILSVRC 2012. Dalam makalah tersebut, mereka menggunakan lebih dari 600.000 neuron dan 7 lapisan tersembunyi untuk memberikan model data yang baik untuk menghindari overfitting.

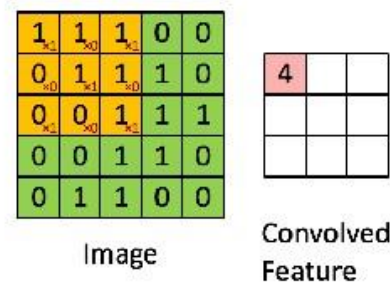
Pada CNN, setiap neuron direpresentasikan dalam bentuk dua dimensi, tidak seperti MLP yang setiap neuron hanya berPlat satu dimensi. CNN termasuk dalam *Deep Neural Network* karena kedalaman jaringan yang tinggi. *Convolutional Neural Network* (CNN) merupakan salah satu jenis neural network yang berisi kombinasi beberapa layer yaitu *convolutional layer*, *pooling layer* dan *fully connected layer* (Hu, Huang, Wei, Zhang, & Li, 2015) seperti pada Gambar 2.5.



Gambar 2.5 *Convolutional neural network*

5. *Convolutional Layer*

Konvolusi merupakan suatu istilah matematis yang berarti mengaplikasikan sebuah fungsi pada output fungsi lain secara berulang (Vista Lab, 2013) seperti Gambar 2.6.



Gambar 2.6 Operasi Konvolusi

Convolutional Layer adalah sebuah inti utama dari CNN, dimana layer ini memiliki sebuah kumpulan filter yang dapat digunakan untuk mempelajari citra masukan. Melalui layer ini, fitur akan di ekstraksi dan kemudian di lanjutkan ke layer berikutnya dengan tujuan untuk mengekstraksi fitur yang lebih kompleks (Bui & Chang, 2017).

Convolutional Layer melakukan operasi konvolusi terhadap input ataupun output dari layer sebelumnya. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra input. Konvolusi akan menghasilkan transformasi linear dari data input sesuai informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan input pada CNN seperti pada persamaan rumus.

$$C_{ij} = A \times P_1 + B \times P_2 + C \times P_3 + D \times P_4 + E \times P_5 \\ + F \times P_6 + G \times P_7 + H \times P_8 + I \times P_9$$

Keterangan :

$C_{i,j}$ = Nilai *feature map* yang dihasilkan proses konvolusi

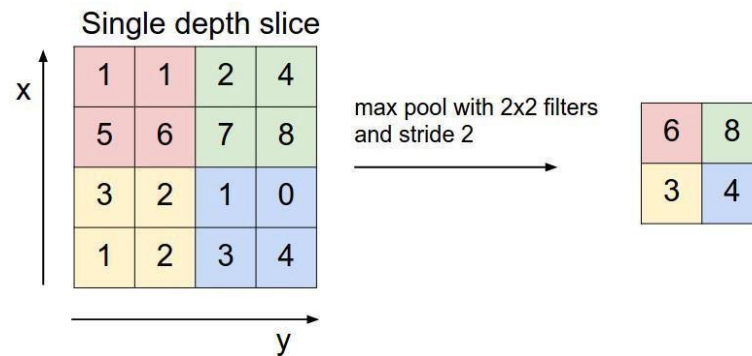
$A-I$ = Nilai matriks gambar *input*

P_{1-9} = Nilai matriks *feature detector*

6. *Pooling Layer*

Pooling Layer merupakan proses resizing yaitu proses untuk mengubah Plat citra input yang berbeda, salah satunya dengan menggunakan operasi *MAX*. Hal ini bertujuan untuk membantu mengurangi jumlah parameter dan waktu perhitungan yang dibutuhkan saat melatih network.

Max pooling membagi output dari *convolutional layer* menjadi beberapa grid kecil lalu mengambil nilai maksimal dari setiap grid untuk menyusun matriks citra yang telah direduksi seperti yang ditunjukkan pada Gambar. Grid yang berwarna merah, hijau, kuning dan biru merupakan kelompok grid yang akan dipilih nilai maksimumnya. Sehingga hasil dari proses tersebut dapat dilihat pada kumpulan grid disebelah kanannya. Proses tersebut memastikan fitur yang didapatkan akan sama meskipun objek citra mengalami translasi (pergeseran).



Gambar 2.7 Operasi Max Polling

7. Fully Connected Layer

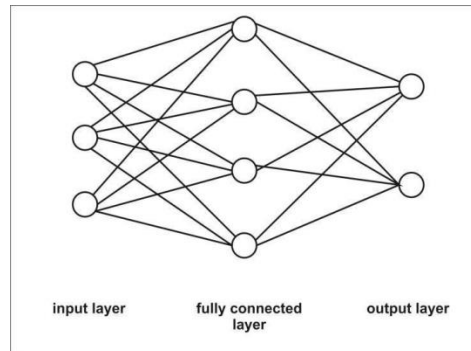
Dan layer ketiga pada CNN adalah *Fully Connected Layer*, dimana layer ini mengambil seluruh neuron pada layer sebelumnya (*Convolutional Layer* dan *MAX Pooling Layer*) dan menghubungkannya ke setiap *single neuron* yang ada (Devikar, 2016).

Fully connected layer adalah layer yang biasanya digunakan dalam penerapan MLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear (Suartika et al., 2016).

Setiap *neuron* pada *convolution layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan kedalam sebuah *fully connected layer*. Karena hal tersebut menyebabkan data kehilangan informasi spesialnya dan tidak reversibel, *fully connected layer* hanya dapat diimplementasikan diakhir jaringan seperti pada Gambar.

Dalam sebuah jurnal oleh Lin et al. (2014), dijelaskan bahwa *convolutional layer* dengan Plat kernel 1 x 1 melakukan fungsi yang sama dengan sebuah *fully connected layer* namun dengan tetap mempertahankan karakter spasial dari data.

Hal tersebut membuat penggunaan *fully connected layer* pada CNN sekarang tidak banyak dipakai.



Gambar 2.8 Proses Fully Connected Layer

H. Object Tracking

Dalam visi komputer, *objecttracking* (pelacakan objek) adalah suatu proses untuk melacak satu objek atau lebih dari suatu citra. *Objecttracking* termasuk dalam salah satu fungsi yang sangat penting di bidang visi komputer. Ada tiga langkah penting dalam analisa video: deteksi objek yang bergerak, mendeteksi beberapa objek di setiap frame, dan analisa objek yang dilacak untuk mengenali pergerakan objek pada citra. Dalam bentuk yang paling sederhana, tracking dapat didefinisikan sebagai suatu masalah untuk memperkirakan lintasan dari sebuah objek yang bergerak dalam sebuah gambar. Secara konsisten, pelacak memberikan label pada objek yang dilacak pada frame-frame yang berbeda dalam sebuah video. Berdasarkan dari pelacakan domain, sebuah pelacak juga dapat memberikan informasi suatu objek, seperti sebuah orientasi gerak, area, atau bentuk dari objek. Satu hal yang dapat memudahkan pelacakan, yaitu dengan memberikan batasan secara paksa pada pergerakan dan/atau bentuk dari objek. Sebagai contoh, hampir

semua algoritma tracking berasumsi bahwa objek bergerak secara tetap tanpa ada perubahan yang mendadak serta memaksakan kecepatan atau percepatan objek menjadi konstan berdasarkan informasi sebelumnya. Pembelajaran sebelumnya mengenai Plat, tampilan, dan bentuk objek, juga dapat digunakan untuk memudahkan masalah

I. *Computer Vision*

Computer Vision adalah ilmu dan teknologi mesin yang melihat, di mana mesin mampu mengekstrak informasi dari gambar yang diperlukan untuk menyelesaikan tugas tertentu. Sebagai suatu disiplin ilmu, visi komputer berkaitan dengan teori di balik sistem buatan bahwa ekstrak informasi dari gambar. Data gambar dapat mengambil banyak bentuk, seperti urutan video, pandangan dari beberapa kamera, atau data multi-dimensi dari scanner medis. Sedangkan sebagai disiplin teknologi, *computer vision* berusaha untuk menerapkan teori dan model untuk pembangunan sistem *computer vision*.

Computer Vision didefinisikan sebagai salah satu cabang ilmu pengetahuan yang mempelajari bagaimana komputer dapat mengenali obyek yang diamati. Cabang ilmu ini bersama *Artificial Intelligence* akan mampu menghasilkan *Visual Intelligence System*. Perbedaannya adalah *Computer Vision* lebih mempelajari bagaimana komputer dapat mengenali obyek yang diamati. Namun komputer grafik lebih ke arah pemanipulasian gambar (visual) secara digital. Bentuk sederhana dari grafik komputer adalah grafik komputer 2D yang kemudian

berkembang menjadi grafik komputer 3D, pemrosesan citra, dan pengenalan pola.

Grafik komputer sering dikenal dengan istilah visualisasi data.

Computer Vision adalah kombinasi antara :

- a. Pengolahan Citra (*Image Processing*), bidang yang berhubungan dengan proses transformasi citra/gambar (*image*). Proses ini bertujuan untuk mendapatkan kualitas citra yang lebih baik.
- b. Pengenalan Pola (*Pattern Recognition*), bidang ini berhubungan dengan proses identifikasi obyek pada citra atau interpretasi citra. Proses ini bertujuan untuk mengekstrak informasi/pesan yang disampaikan oleh gambar/citra.

a. Fungsi / Proses pada *Computer Vision*

Untuk menunjang tugas *Computer Vision*, terdapat beberapa fungsi pendukung ke dalam sistem ini, yaitu :

1. Proses penangkapan citra (*Image Acquisition*)
 - a. *Image Acquisition* pada manusia dimulai dengan mata, kemudian informasi visual diterjemahkan ke dalam suatu format yang kemudian dapat dimanipulasi oleh otak.
 - b. Senada dengan proses di atas, *computer vision* membutuhkan sebuah mata untuk menangkap sebuah sinyal visual.
 - c. Umumnya mata pada *computer vision* adalah sebuah kamera video.
 - d. Kamera menerjemahkan sebuah *scene* atau *image*.
 - e. Keluaran dari kamera adalah berupa sinyal analog, dimana frekuensi dan amplitudonya (frekuensi berhubungan dengan jumlah sinyal

dalam satu detik, sedangkan amplitudo berkaitan dengan tingginya sinyal listrik yang dihasilkan) merepresentasikan detail ketajaman (*brightness*) pada *scene*.

- f. Kamera mengamati sebuah kejadian pada satu jalur dalam satu waktu, memindainya dan membaginya menjadi ratusan garis horizontal yang sama.
- g. Tiap-tiap garis membuat sebuah sinyal analog yang amplitudonya menjelaskan perubahan *brightness* sepanjang garis sinyal tersebut.
- h. Kemudian sinyal listrik ini diubah menjadi bilangan biner yang akan digunakan oleh komputer untuk pemrosesan.
- i. Karena komputer tidak bekerja dengan sinyal analog, maka sebuah *analog-to-digital converter* (ADC), dibutuhkan untuk memproses semua sinyal tersebut oleh komputer.
- j. ADC ini akan mengubah sinyal analog yang direpresentasikan dalam bentuk informasi sinyal tunggal ke dalam sebuah aliran (*stream*) sejumlah bilangan biner.
- k. Bilangan biner ini kemudian disimpan di dalam memori dan akan menjadi data raw yang akan diproses.

2. Proses pengolahan citra (*Image Processing*)

- a. Tahapan berikutnya *computer vision* akan melibatkan sejumlah manipulasi utama (*initial manipulation*) dari data binary tersebut.

- b. *Image processing* membantu peningkatan dan perbaikan kualitas image, sehingga dapat dianalisa dan di olah lebih jauh secara lebih efisien.
- c. *Image processing* akan meningkatkan perbandingan sinyal terhadap noise (signal-to-noise ratio = s/n).
- d. Sinyal-sinyal tersebut adalah informasi yang akan merepresentasikan objek yang ada dalam image.
- e. Sedangkan noise adalah segala bentuk interferensi, kurangpengaburan, yang terjadi pada sebuah objek.

3. Analisa data citra (*Image Analysis*)

- a. *Image analysis* akan mengeksplorasi *scene* ke dalam bentuk karakteristik utama dari objek melalui suatu proses investigasi.
- b. Sebuah program komputer akan mulai melihat melalui bilangan biner yang merepresentasikan informasi visual untuk mengidentifikasi fitur-fitur spesifik dan karekteristiknya.
- c. Lebih khusus lagi program image *analysis* digunakan untuk mencari tepi dan batas-batasan objek dalam *image*.
- d. Sebuah tepian (*edge*) terbentuk antara objek dan latar belakangnya atau antara dua objek yang spesifik.
- e. Tepi ini akan terdeteksi sebagai akibat dari perbedaan level brightness pada sisi yang berbeda dengan salah satu batasnya.

4. Proses pemahaman data citra (*Image Understanding*)

- a. Ini adalah langkah terakhir dalam proses *computer vision*, yang mana sprsifik objek dan hubungannya diidentifikasi.
- b. Pada bagian ini akan melibatkan kajian tentang teknik-teknik artificial intelligent.
- c. Understanding berkaitan dengan template matching yang ada dalam sebuah scene.
- d. Metoda ini menggunakan program pencarian (*search program*) dan teknik penyesuaian pola (*pattern matching techniques*).

J. Pengolahan Citra

Citra merupakan suatu fungsi kontinu dari intensitas cahaya dalam bidang 2D, dengan (x,y) menyatakan tingkat kecerahan atau derajat keabuan. Citra digital merupakan array 2D dengan nilai $f(x,y)$ nya telah dikonversi ke dalam bentuk diskrit baik pada koordinat citra maupun kecerahannya. Pengolahan citra secara umum dapat didefinisikan sebagai pemrosesan sebuah gambar 2D secara digital. Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual (Anil K. Jain, 1989). Proses ini mempunyai data masukan dan informasi keluaran yang berbentuk citra.

Operasi pengolahan citra digital umumnya dilakukan dengan tujuan memperbaiki kualitas suatu gambar sehingga dapat dengan mudah diinterpretasikan oleh mata manusia dan untuk mengolah informasi yang terdapat pada suatu gambar untuk keperluan pengenalan objek secara otomatis.

1. Pengolahan Citra *Grayscale*

Citra *grayscale* yaitu citra yang nilai *pixel*-nya merepresentasikan derajat keabuan atau intensitas warna putih. Nilai intensitas paling rendah merepresentasikan warna hitam dan nilai intensitas paling tinggi merepresentasikan warna putih.

Pada umumnya citra *grayscale* memiliki kedalaman *pixel* 8 bit. Maka jumlah kemungkinan nilainya adalah $2^8 = 256$, dan nilai maksimumnya adalah $2^8 - 1 = 255$. Sehingga semakin besar angka *grayscale*, citra yang berbentuk makin mendekati kenyataan



Pada gambar di atas dimana Citra *grayscale* ini merupakan citra kanal, yang dimana $f(x,y)$ merupakan fungsi dari tingkat keabuan dari hitam keputih, x merupakan variabel baris atau posisi *pixel* di garis jelajah dan y merupakan variabel kolom atau posisi *pixel* di garis jelajah. Intensitas f dari gambar hitam putih pada titik (x,y) disebut derajat keabuan (*gray level*), yang dalam hal ini derajat keabuannya bergerak dari hitam keputih. Derajat keabuan memiliki rentang

nilai dari I_{min} sampai I_{max} , atau $I_{min} < f < I_{max}$, selang (I_{min}, I_{max}) disebut skala keabuan.

Biasanya selang (I_{min}, I_{max}) sering digeser untuk alasan-alasan yang praktis menjadi selang $[0, L]$, yang dalam hal ini nilai intensitas 0 yaitu menyatakan hitam, nilai intensitas L yaitu menyatakan putih, sedangkan nilai intensitas antara 0 sampai L bergeser dari hitam keputih. Sebagai contoh citra *grayscale* dengan 256 level artinya mempunyai skala abu-abu dari 0 sampai dengan 255 atau $[0, 255]$, yang dalam hal ini intensitas 0 yaitu menyatakan hitam, intensitas 255 yaitu menyatakan putih, dan nilai antara 0 sampai dengan 255 menyatakan warna keabuan yang terletak antara hitam dan putih.

Untuk mengubah citra *grayscale* ke citra biner bisa dilakukan dengan cara *threshold*, dan biasanya penulis menentukan nilai citra *threshold* tertentu terlebih dahulu dan kemudian yang ada di bawah nilai *threshold* nilainya dijadikan 0 yang ada di atas *threshold* nilainya 1. Inilah gambaran bagaimana proses konversi *grayscale* melalui perhitungan manual ilmu matematika. Dimisalkan ada sebuah citra yang memiliki Plat matriks 7×7 *pixel* dengan nilai-nilai sebagai berikut :

Tabel 2.1 Nilai RGB suatu citra

(i,j)	0	1	2	3	4	5	6	7
0	31	42	46	41	121	119	84	111
	32	44	48	43	116	121	86	113
1	43	44	64	82	92	76	6	61
	44	46	66	84	94	78	8	63
2	188	186	193	157	177	156	191	231
	189	187	194	158	178	158	193	233
3	137	134	34	177	156	173	152	150
	138	135	35	178	157	174	153	152

4	255	255	73	178	97	211	186	159
	255	255	74	179	98	212	187	160
5	141	134	125	153	137	139	152	150
	142	135	126	154	138	140	153	151
6	116	118	151	120	132	137	145	118
	116	118	151	120	132	138	145	119
7	123	129	139	135	130	120	140	129
	123	129	139	135	134	120	140	129

Untuk konversi warna menjadi *grayscale*, misalnya *koordinat* (5,3), perhitungannya adalah : $Gray [7,5] = \frac{150 + 151 + 145}{3} = 148,6$

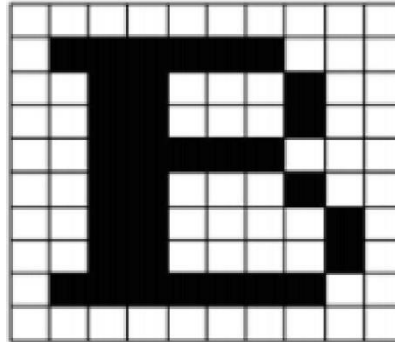
(2) Perhitungan tersebut dilakukan untuk semua *koordinat*, sehingga *matriks* citra hasil konversi menjadi berikut :

Tabel 2.2 hasil konversi nilai RGB menjadi *grayscale*

(i,j)	0	1	2	3	4	5	6	7
0	29	40	44	39	111	116	81	108
1	40	42	62	80	90	73	5	58
2	186	184	191	155	175	154	189	229
3	136	132	33	175	154	171	150	148
4	254	253	72	177	996	210	185	157
5	140	133	124	152	136	138	151	149
6	115	117	150	119	131	136	144	117
7	122	128	138	134	133	119	139	128

2. Pengolahan Citra biner

Citra *biner*, yaitu citra yang hanya terdiri atas dua warna, yaitu hitam dan putih. Oleh karena itu, setiap *pixel* pada citra biner cukup direpresentasikan dengan 1 *bit* (Balza dan Kartika, 2005:9). Gambar 2.2 merupakan contoh citra dan Gambar 2.3 merupakan representasi citra *biner*.



Gambar 2.9 Contoh citra biner (Biben Nurbani Hasan, 2012)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0

Meskipun saat ini citra berwarna lebih disukai karena memberi kesan uang lebih kaya dari citra *biner*, namun tidak membuat citra *biner* mati. Pada beberapa aplikasi citra *biner* masih tetap dibutuhkan, misalnya citra logo *instansi* (yang hanya terdiri dari warna hitam dan putih), citra kode barang (*bar code*) yang tertera pada label barang, citra hasil pemindai dokumen *teks*, dan sebagainya.

Seperti yang sudah disebutkan diatas, citra *biner* hanya mempunyai dua nilai derajat keabuan yaitu hitam dan putih. *Pixel-pixel* objek bernilai 1 dan *pixel-pixel* latar belakang bernilai 0. Pada waktu menampilkan gambar, 0 adalah putih dan 1 adalah hitam. Jadi pada citra *biner*, latar belakang berwarna putih sedangkan objek berwarna hitam seperti tampak pada gambar 4 di atas. Meskipun komputer saat ini dapat memproses citra hitam-putih (*grayscale*) maupun citra berwarna, namun citra *biner* masih tetap dipertahankan keberadaannya.

Alasan penggunaan citra *biner* adalah karean citra biner memiliki sejumlah keuntungan sebagai berikut :

- a. Kebutuhan memori kecil karena nilai derajat keabuan hanya membutuhkan representasi 1 *bit*.
- b. Waktu pemrosesan lebih cepat dibandingkan dengan citra hitam-putih ataupun warna.

Citra *biner* adalah citra yang terdiri dari warna hitam dan putih. Pada langkah citra biner ini adalah nilai konversi dari *grayscale*. Sekarang saya mencoba menentukan nilai *threshold* terlebih dahulu secara manual dari nilai *grayscale* yang sudah ada.

Table 2.3 Nilai *Grayscale*

(i,j)	0	1	2	3	4	5	6	7
0	29	40	44	39	111	116	81	108
1	40	42	62	80	90	73	5	58
2	186	184	191	155	175	154	189	229
3	136	132	33	175	154	171	150	148

4	254	253	72	177	996	210	185	157
5	140	133	124	152	136	138	151	149
6	115	117	150	119	131	136	144	117
7	122	128	138	134	133	119	139	128

Nilai *High* = 255 dan Nilai *low* = 0 Nilai *threshold* kita tentukan yaitu median dari 0-255, yaitu 127. Untuk nilai dibawah 127 kita jadikan warna putih yang nilainya 0, untuk nilai diatas 127 kita jadikan warna hitam yang nilainya 1

K. Pengertian Aplikasi

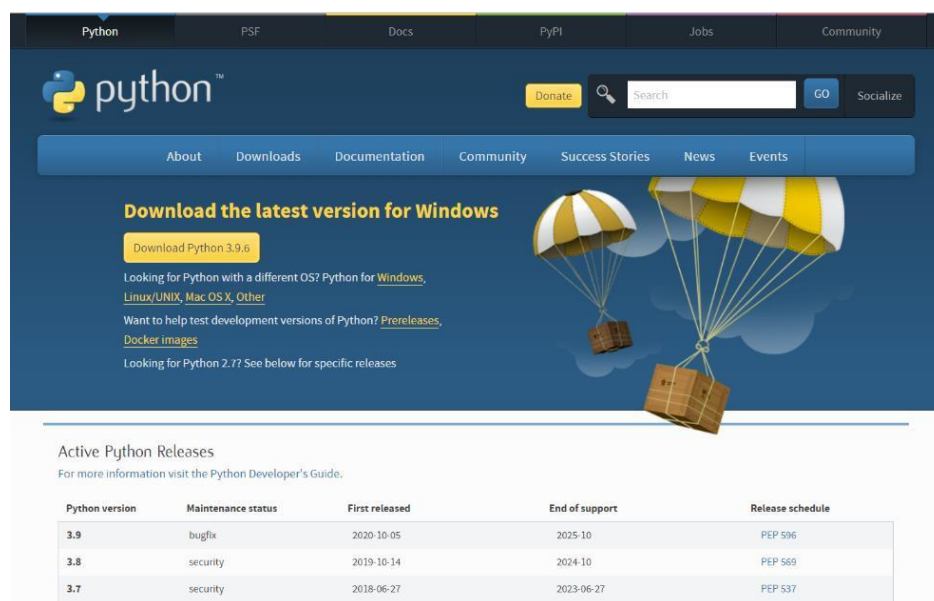
Aplikasi menurut Jogiyanto (2018:12) adalah penggunaan dalam suatu komputer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output. Menurut Kamus Kamus Besar Bahasa Indonesia (2018 : 52) , “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu”. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer. Program merupakan kumpulan *instruction* set yang akan dijalankan oleh pemroses, yaitu berupa *software*. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini. Program inilah yang mengendalikan semua aktifitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada

instructionset .Program aplikasi merupakan program siap pakai. Program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh-contoh aplikasi ialah program pemproses kata dan *Web Browser*. Aplikasi akan menggunakan sistem operasi (OS) komputer dan aplikasi yang lainnya yang mendukung. Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak tahun 1993, yang biasanya juga disingkat dengan app. Secara *historis*, aplikasi adalah *software* yang dikembangkan oleh sebuah perusahaan. *App* adalah *software* yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang paralel dengan yang terjadi antar sistem operasi yang dimunculkan.

L. Python

1. Instalasi Python

Pada penelitian ini, penulis menggunakan bahasa Python untuk melakukan pendeteksian.



Download the latest version for Windows

[Download Python 3.9.6](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases

Active Python Releases

For more information visit the [Python Developer's Guide](#).

Python version	Maintenance status	First released	End of support	Release schedule
3.9	bugfix	2020-10-05	2025-10	PEP 596
3.8	security	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537

Gambar 2.10 Tampilan *website python*

2. Instalasi OpenCV

Selanjutnya melakukan instalasi OpenCV. Disini penulis menggunakan OpenCV 3 dengan code seperti pada Gambar 4.7.

```
>pip install opencv-python
```

Gambar 4.11 Kode program instalasi OpenCV

M. Deteksi Plat

Pada proses deteksi, setelah semua *software* yang dibutuhkan telah terinstalasi dengan baik. Langkah selanjutnya adalah menggabungkan file *weight*, *configuration*, dan dataset pada satu folder.

1. Import library

Menggunakan library *cv2*, *numpy*, dan *time*. Library *cv2* berfungsi untuk melakukan *computer vision task*. Library *numpy* berfungsi untuk pengolahan data numerical.

```
import cv2
import numpy as np
import time

print(cv2.__version__)
```

Gambar 2.12 Kode program *import library*

```
C:\Users\HP\anaconda3\envs\OpenCV\python.exe C:/Users/HP/PycharmProjects/pythonProject/detect.py
4.5.1
```

Gambar 2.13 Hasil dari keluaran versi OpenCV

2. Membaca Masukan plat

Pada bagian ini dilakukan membaca masukan benda dengan mendefinisikan `cv2.VideoCapture()` untuk mendapatkan objek dari benda capture pada kamera sesuai dengan lokasi penyimpanan. Lalu menggunakan variabel '*writer*' yang digunakan untuk menulis *frame* yang diproses. Kemudian menggunakan variabel '*h*' dan '*w*' untuk dimensi *frame*.

```
writer = None
h, w = None, Non
```

Gambar 2.14 Kode program membaca masukan plat

```
with open('Plat.names') as f:
    labels = [line.strip() for line in f]

print('List with labels names:')
print(labels)

layers_names_all = network.getLayerNames()
layers_names_output = \
    [layers_names_all[i[0] - 1] for i in
network.getUnconnectedOutLayers()]
probability_minimum = 0.5
threshold = 0.5
colours = np.random.randint(0, 255, size=(len(labels), 3),
dtype='uint8')
```

Gambar 2.15 Kode program memuat *network*

```
List with labels names:
['mobil', 'sepeda motor', 'becak', 'truk', 'bus']
```

Gambar 2.16 Hasil keluaran *labels*

3. Membaca *frame* untuk perulangan

Kemudian melakukan perulangan pada *frame*. Disini akan dilakukan fungsi perulangan untuk menangkap *frame-by-frame* dengan fungsi *read*, yaitu membaca *frame* dari Plat masukan. Jika *frame* tidak diambil, misalnya pada akhir aplikasi memunculkan sebuah Plat sebuah benda, maka

```
f = 0
t = 0

while True:
    ret, frame = benda.read()
    if not ret:
        break
    if w is None or h is None:
        h, w = frame.shape[:2]
```

Gambar 2.17 Kode program membaca *frame* untuk perulangan

4. Mengambil fungsi *blob* dari *frame*

Kemudian, data akan diolah menggunakan *library* OpenCV untuk diubah menjadi bentuk blob (Binary Large Object) dari frame. Menggunakan fungsi 'cv2.dnn.blobFromImage' akan mengembalikan 4-dimensional blob dari frame saat ini setelah pengurangan rata-rata, normalisasi, dan pertukaran saluran RB (Irwan, Putrada, & Prabowo, 2019). Pada fungsi

'cv2.dnn.blobFromImage' berisi parameter frame yang merupakan masukan gambar yang akan diproses melalui jaringan saraf dalam bentuk klasifikasi. 1/255.0 sebagai scale factor untuk melakukan pengurangan rata-rata. Plat (416,416) adalah Plat benda. swapRb memberikan asumsi pada gambar berada dalam saluran BGR, namun nilai mean mengasumsikan kita menggunakan urutan RGB. Untuk mengatasi perbedaan ini, kita dapat menetapkan nilai menjadi 'True' untuk menukarkan saluran R dan B. Bentuk yang dihasilkan memiliki jumlah bingkai, jumlah

saluran, lebar dan tinggi. Blob digunakan untuk mengekstrak fitur gambar dan mengubah Platnya, yaitu 320x320,

416x416, dan 609x609. Disini penulis menggunakan Plat frame 416x416 agar proses deteksi memiliki kecepatan dan akurasi yang seimbang.

```
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
                              swapRB=True, crop=False)
```

Gambar 2.18 Kode program mengambil fungsi *blob* dari *frame*

5. Menerapkan *Forward Pass*

Mengimplementasi *forward pass* dengan '*blob*' melalui lapisan keluaran.

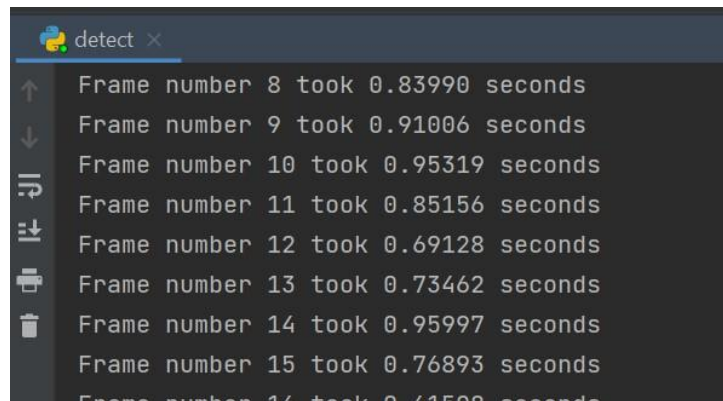
Lalu, menghitung waktu yang dibutuhkan untuk *forward pass*.

```
network.setInput(blob)
start = time.time()
output_from_network = network.forward(layers_names_output)
end = time.time()

f += 1
t += end - start

print('Frame number {0} took {1:.5f} seconds'.format(f, end - start))
```

Gambar 2.19 Kode program menerapkan *Forward Pass*



Gambar 2.20 Contoh hasil dari proses deteksi per *frame*

6. Mendapatkan *bounding boxes*

Pada langkah ini dilakukan ekstraksi seluruh informasi objek yang dideteksi dan menampilkannya pada layar. Melakukan inisialisasi list *bounding_boxes* untuk kotak pembatas yang mengelilingi objek. *Confidences* memberikan nilai keyakinan untuk suatu objek, nilai keyakinan (*confidence*) dari objek yang terdeteksi bernilai 0 sampai 1. *ClassIDs* untuk memberikan label kelas objek yang terdeteksi.

Melakukan perulangan untuk setiap *output_from_network* dan perulangan untuk setiap *detected_object* pada *result* dengan mengekstrak *class_current* dan *confidence_current*. Menggunakan perintah *confidence_current* untuk menyaring prediksi yang lemah dengan memastikan objek yang terdeteksi dengan probabilitas lebih besar dari probabilitas minimum. Kemudian melakukan skala koordinat kotak pembatas untuk menampilkan dengan benar pada gambar asli. Lalu, mengekstraksi koordinat dan dimensi dari kotak pembatas dengan mengembalikan koordinat kotak pembatas dalam bentuk: *x_center*, *y_center*, *box_width*, *box_height*. Menggunakan informasi untuk mendapatkan *top-left (x, y)-coordinates* dari kotak pembatas. Lalu memperbarui list *bounding_boxes*, *confidences*, dan *classIDs*.

```

bounding_boxes = []
confidences = []
classIDs = []

for result in output_from_network:

    for detected_objects in result:
        scores = detected_objects[5:]
        class_current = np.argmax(scores)
        confidence_current = scores[class_current]

        if confidence_current > probability_minimum:
            box_current = detected_objects[0:4] *
np.array([w, h, w, h])
            x_center, y_center, box_width, box_height =
box_current
            x_min = int(x_center - (box_width / 2))
            y_min = int(y_center - (box_height / 2))
            bounding_boxes.append([x_min, y_min,
int(box_width),
int(box_height)])

```

7. Non-maximum Suppresion

Menerapkan *non-maximum suppression* atau yang disebut juga *non-maxima suppression* (NMS). Objek yang terdapat pada video atau gambar dapat memiliki Plat dan bentuk yang berbeda, dan untuk menangkap setiap objek maka algoritma akan memuat kotak pembatas (*bounding boxes*). Jadi, untuk setiap objek yang terdapat pada video atau gambar harus memiliki satu kotak pembatas (*bounding boxes*). Untuk mendapatkan kotak pembatas (*bounding boxes*) terbaik dari beberapa kotak pembatas (*bounding boxes*) yang diprediksi, algoritma akan menggunakan fungsi *non-max*. teknik ini digunakan untuk “menekan” kotak pembatas yang kemungkinannya kecil dan hanya menyimpan yang terbaik. Tujuan dari *non- maximum suppression* adalah untuk memilih kotak pembatas (*bounding boxes*) terbaik pada suatu objek. NMS akan mempertimbangkan dua hal, yang pertama skor objektifitas yang diberikan oleh model, dan yang kedua IoU dari kotak pembatas (*bounding boxes*) (Hosang & May, n.d.).

Pada Gambar 4.18 memanfaatkan implementasi modul DNN pada OpenCV, kemudian melakukan *non-maximum suppression* dengan parameter *bounding_boxes*, *confidences*, *probability_minimum*, dan *threshold*.

```
results = cv2.dnn.NMSBoxes(bounding_boxes, confidences,
                           probability_minimum, threshold)
```

Gambar 2.22 Kode program *non-maximum suppression*

8. Mendapatkan *bounding boxes* dengan *labels*

Menggunakan perintah *if* untuk memeriksa jika setidaknya ada satu objek yang terdeteksi setelah NMS. Dengan mendapatkan *bounding box* saat ini, lebar, dan tingginya. Kemudian mempersiapkan warna untuk *bounding box* saat ini dan mengkonversi dari *array* *numpy*. Setelah mendapatkan warna, selanjutnya menggambar *bounding box* pada *frame*. Setelah itu, mempersiapkan teks dengan label dan nilai *confidence* untuk *bounding box* saat ini dan menampilkan hasilnya.

```
if len(results) > 0:
    for i in results.flatten():
        x_min, y_min = bounding_boxes[i][0], bounding_boxes[i][1]
        box_width, box_height = bounding_boxes[i][2], bounding_boxes[i][3]
        colour_box_current = colours[classIDs[i]].tolist()

        cv2.rectangle(frame, (x_min, y_min), (x_min + box_width, y_min
+ box_height), colour_box_current, 2)
        text_box_current = '{}: {:.4f}'.format(labels[int(classIDs[i])],
        confidences[i])
        cv2.putText(frame, text_box_current, (x_min, y_min - 5),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, colour_box_current,
        2)
```

9. Menyimpan video proses deteksi

Pada bagian ini, akan menyimpan hasil prediksi masukan video dengan mengecek '*writer*', kemudian '*writer*' akan diinisialisasi pada iterasi pertama dari

perulangan. Mengeluarkan perkiraan waktu yang dibutuhkan untuk proses video. Dan mengeluarkan hasil akhir dari proses deteksi pada masukan video.

```

if writer is None:
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    writer = cv2.VideoWriter('resultvideo8juli.mp4', fourcc, 30,
                            (frame.shape[1], frame.shape[0]),
                            True)

writer.write(frame)

# Printing final results print()
print('Total number of frames', f)

```

G. UML (*Unified Modeling Language*)

Menurut Adi nugroho dalam buku “Rekayasa Perangkat Lunak Berorientasi Objek” (2010:6) UML (*Unified Modeling Language*) adalah ‘bahasa’ pemodelan untuk *system* atau perangkat lunak yang berparadigma ‘berorientasi objek’. Pemodelan (*Modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami.

a. *Use Case Diagram*

Use case diagram bersifat statis, memperlihatkan himpunan *use-case* dan aktor-aktor. Diagram ini sangat penting terutama untuk memodelkan ataupun mengorganisasikan perilaku dari *system* yang dibutuhkan pengguna.

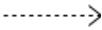
b. *Activity Diagram*

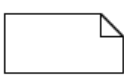
Activity Diagram bersifat dinamis, merupakan tipe khusus dari diagram state yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lain dalam suatu system.

c. *Class Diagram*

Class Diagram bersifat statis tetapi sering pula memuat kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi.

Tabel 2.4 Simbol Diagram *Use Case*

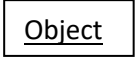
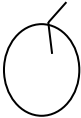
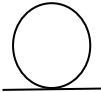
NO	GAMBAR	NAMA	KETERANGAN
1		<i>Actor</i>	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan <i>use case</i> .
2		<i>Dependency</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempengaruhi elemen yang bergantung padanya sehingga elemen yang tidak mandiri (<i>independent</i>).
3		<i>Generalization</i>	Hubungan dimana objek anak (<i>descendent</i>) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (<i>ancestor</i>).
4		<i>Include</i>	Menspesifikasikan bahwa <i>use case</i> sumber secara <i>eksplisit</i> .
5		<i>Extend</i>	Menspesifikasikan bahwa <i>use case</i> target memperluas perilaku dari <i>use case</i> sumber pada suatu titik yang diberikan.
6		<i>Association</i>	Apa yang menghubungkan antara objek satu dengan objek lainnya.

NO	GAMBAR	NAMA	KETERANGAN
7		<i>System</i>	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		<i>Use Case</i>	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor
9		<i>Collaboration</i>	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan perilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		<i>Note</i>	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

Tabel 2.4 Simbol Diagram Activity

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Activity</i>	Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
2		<i>Action</i>	State dari sistem yang mencerminkan eksekusi dari suatu aksi
3		<i>Initial Node</i>	Bagaimana objek dibentuk atau diawali.
4		<i>Activity Final Node</i>	Bagaimana objek dibentuk dan dihancurkan
5		<i>Fork Node</i>	Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran

Tabel 2.5 Simbol Sequence Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		<i>Object</i> (Partisipan)	Object atau biasa juga disebut partisipan merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama objek didalamnya yang diawali dengan sebuah titik koma.
2		<i>Actor</i>	Actor jug dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom.
3		<i>Life line</i>	Life line mengidentifikasi sebuah keberadaan object dalam basis waktu. Notasi untuk lifeline adalah garis putus-putus vertical yang ditarik oleh sebuah object.
4		<i>Collaboration</i>	Activication dinotasikan sebagai kotak segi empat yang digambar pada sebuah lifeline action mengidentifikasi sebuah object yang akan melakukan sebuah aksi.
5		<i>Boundary</i>	Boundary terletak diantara system dengan dunia sekelilingnya. Semua form, laporan-laporan, antar muka ke perangkat keras seperti printer atau scanner dan antar muka ke system lainnya adalah termasuk dalam kategor.
6		<i>Control</i>	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (<i>independent</i>) akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		<i>Entity</i>	Entity digunakan menangani informasi yang mungkin akan disimpan secara permanen. Entity bias juga merupakan sebuah table.

BAB III

METODE PENELITIAN

A. Jenis Penelitian

Jenis penelitian yang digunakan adalah penelitian deskriptif dimana memberikan gambaran mengenai apa yang sesungguhnya terjadi. Dalam pembuatan Skripsi ini digunakan metode deskripsif yang menggambarkan fakta-fakta dan informasi secara sistematis, faktual dan akurat.

Penelitian ini dilakukan melalui internet yang dapat memberikan sumber data dan pengetahuan mengenai sistem yang diteliti, kemudian mencocokkan dengan kemungkinan yang terjadi dalam usaha penyelesaian masalah.

B. Lokasi dan Waktu Penelitian

Lokasi penelitian dilaksanakan di SAMSAT Kabupaten Pinrang. Adapun waktu penelitian dilakukan selama 2 (dua) bulan tahun 2023.

C. Metode Pengumpulan Data

Untuk memperoleh data-data yang dibutuhkan dalam rangka melakukan penelitian, maka penulis mengumpulkan data melalui beberapa cara yaitu :

1. Analisis Data

Menganalisa data-data yang sebelumnya telah dikumpulkan.

2. Perancangan Program

Sebagai pedoman dalam penulisan program atau kode-kode agar berjalan sesuai rencana.

3. Uji Coba Program

Pengujian program dilakukan untuk memastikan bahwa program yang dibuat dapat berjalan dengan baik.

4. Evaluasi

Sistem yang telah selesai dibangun perlu adanya evaluasi untuk menemukan kelemahan yang terdapat pada program yang telah dibangun tadi, yang nantinya bisa digunakan sebagai acuan untuk memperbaiki program sehingga lebih sempurna.

D. Jenis Data

Data yang digunakan dalam penelitian ini berupa data-data yang telah dikumpulkan melalui Penelitian Pustaka (*Library Research*). Adapun jenis data primer dan data sekunder yang relevan dengan masalah yang akan dibahas.

1. Data primer

Data Primer adalah data yang berasal atau data yang diperoleh langsung dari sumber data dan pengetahuan.

2. Data Sekunder

Data sekunder adalah data yang diperoleh tidak secara langsung dari objek penelitian. Peneliti mendapatkan data yang sudah jadi dari internet, *website* dan jurnal.

E. Tahapan Penelitian

Tahapan penelitian yang dimaksud dalam penelitian ini ada beberapa tahapan yaitu persiapan penelitian, pengumpulan data, analisis perancangan, pengujian dan implementasi. Adapun Uraian dari tahapan tersebut adalah sebagai berikut :

1. Persiapan Penelitian

Pada tahapan ini peneliti melakukan persiapan penelitian. Persiapan penelitian yang dimaksud adalah menyiapkan buku-buku, artikel-artikel tentang topik penelitian serta *software* yang digunakan selama penelitian.

2. Pengumpulan Data

Pada tahap ini peneliti melakukan observasi dengan peninjauan, pencatatan dan pengamatan langsung di tempat penelitian.

3. Analisis

Pada tahap analisis, peneliti melakukan analisa terhadap sistem yang di terapkan sekarang berdasarkan kemudian merumuskan masalah yang menjadi pokok penelitian sehingga dapat dibuat alternatif pemecahan masalah.

4. Perancangan

Peneliti kemudian merancang aplikasi yang ingin dibuat berdasarkan alternatif pemecahan masalah.

5. Pengujian

Setelah melakukan perancangan, peneliti kemudian menguji hasil perancangan yang telah dibuat. Jika hasil perancangan terdapat kekurangan atau kelemahan maka kembali ke tahap analisis.

6. Implementasi

Setelah pada perancangan tidak terdapat kekurangan maka aplikasi siap untuk di gunakan oleh user.

F. Metode Pengujian Aplikasi

Beberapa *test-case* harus dilaksanakan dengan beberapa perbedaan strategi, query, atau jalur navigasi yang mewakili penggunaan sistem yang *tipical*, kritis atau abnormal. Isu kunci pada pengembangan sistem adalah pemilihan sekelompok *test-case* yang cocok, sekecil dan secepat mungkin, untuk meyakinkan perilaku sistem secara detail. Pengujian harus mencakup *unit testing*, yang mengecek validasi dari prosedur dan fungsi-fungsi secara independen dari komponen sistem yang lain. Kemudian modul testing harus menyusul dilakukan untuk mengetahui apakah penggabungan beberapa unit dalam satu modul sudah berjalan dengan baik, termasuk eksekusi dari beberapa modul yang saling berelasi, apakah sudah berjalan sesuai karakteristik sistem yang diinginkan.

Jika struktur kendali antar modul sudah terbukti bagus, maka pengujian yang tak kalah pentingnya adalah pengujian unit. Pengujian unit digunakan untuk menguji setiap modul untuk menjamin setiap modul menjalankan fungsinya dengan baik. Ada 2 metode untuk melakukan unit testing, yaitu :

1. *White Box Testing*

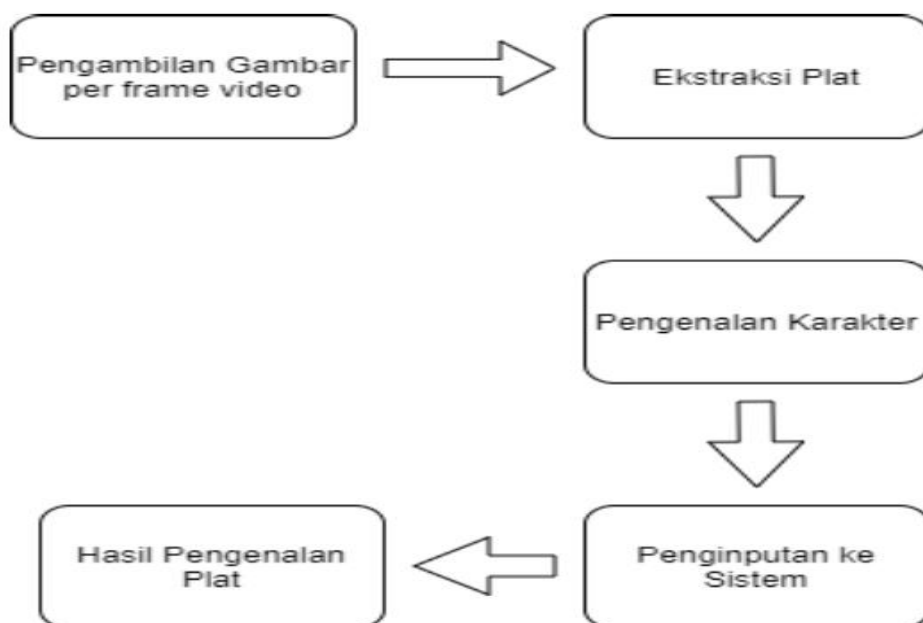
Uji coba *white box testing* merupakan metode perancangan *testcase* yang menggunakan struktural untuk mendapatkan *testcase*, test ini digunakan untuk meramal cara kerja perangkat lunak secara rinci kepada *logic path* (jalur logika), perangkat lunak di tes dengan kondisi dan perulangan secara fisik.

Contoh pengujian *white box testing* ini merupakan peringatan ketika *user* menginputkan password user yang salah, untuk kesalahan semacam ini akan memberikan suatu informasi kepada user mengenai kesalahan yang di lakukan.

2. *Black Box*

Berdasarkan hasil uji coba yang dilakukan, seluruh navigasi dan tombol fasilitas program lainnya serta proses yang di jalankan tidak terjadi kesalahan, tetapi aplikasi mempunyai aturan-aturan yang sudah di tetapkan dan harus di ikuti karena apabila di hiraukan maka sistem akan menolak perintah yang tidak sesuai seperti kesalahan ketika user belum menginput data yang harusnya di input sesuai ketentuan sistem yang di jalankan dan sistem memberikan informasi kepada user karena data yang ingin diproses belum lengkap atau tidak memenuhi ketentuan untuk proses selanjutnya.

G. Rancangan Sistem Yang Diusulkan



Gambar 3.1. Desain Sistem

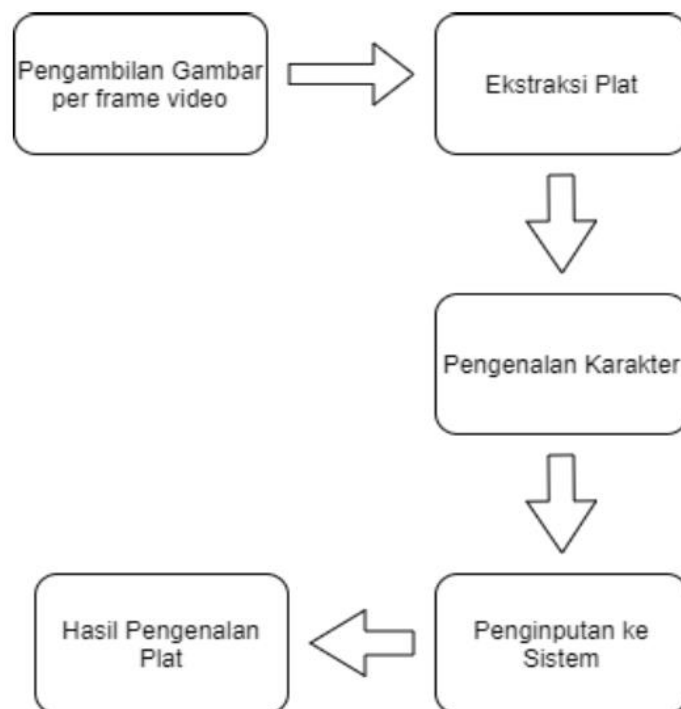
Keterangan :

1. pengambilan gambar per frame dari video adalah menentukan masukkan untuk sistem berupa citra digital.
2. ekstraksi plat adalah proses menentukan letak plat yang berada pada citra gambar dan nantinya akan diproses untuk pengenalan karakter huruf pada plat di citra gambar tersebut.
3. pengenalan karakter dilakukan dengan menggunakan metode template matching yang mana hasil dari citra gambar yang telah terekstrak akan dibandingkan dengan hasil dari segmentasi karakter yang telah dibuat di dalam dataset.
4. penginputan ke sistem yaitu menentukan langkah atau proses yang dikerjakan pada proses sebelumnya.
5. hasil pengenalan plat adalah plat nomor yang berhasil di deteksi dan bisa disimpan.

BAB IV

HASIL DAN PEMBAHASAN

A. Desain Sistem yang di Usulkan



Gambar 4.1. Gambaran umum system yang di usulkan

6. pengambilan gambar per frame dari video adalah menentukan masukkan untuk sistem berupa citra digital.
7. ekstraksi plat adalah proses menentukan letak plat yang berada pada citra gambar dan nantinya akan diproses untuk pengenalan karakter huruf pada plat di citra gambar tersebut.

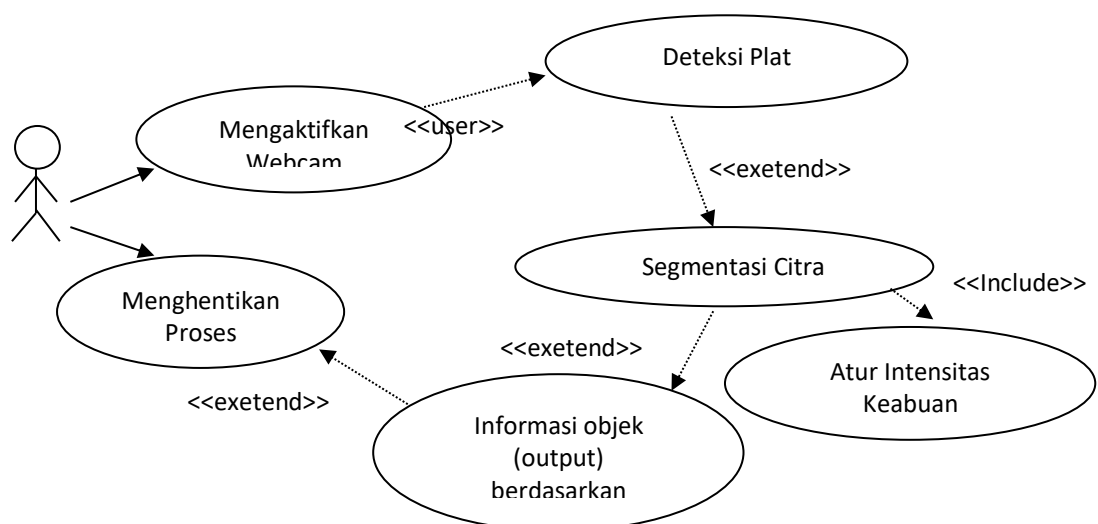
8. pengenalan karakter dilakukan dengan menggunakan metode template matching yang mana hasil dari citra gambar yang telah terekstrak akan dibandingkan dengan hasil dari segmentasi karakter yang telah dibuat di dalam dataset.
9. penginputan ke sistem yaitu menentukan langkah atau proses yang dikerjakan pada proses sebelumnya.
10. hasil pengenalan plat adalah plat nomor yang berhasil di deteksi dan bisa disimpan.

B. Analisis Aliran Data Dengan UML

Analisis aliran data adalah analisis yang dilakukan untuk mempelajari pemanfaatan data pada setiap aktifitas dan juga bertujuan mengetahui aliran proses informasi. Dalam analisis sistem ini, penulis menggunakan pengembangan orientasi objek sehingga menggunakan *Use Case Diagram*, *Activity Diagram*, dan *Sequence Diagram*.

1. Use Case Diagram

Use Case diagram berfungsi untuk menjalankan manfaat sistem jika dilihat menurut pandangan orang yang berada diluar sistem (aktor).

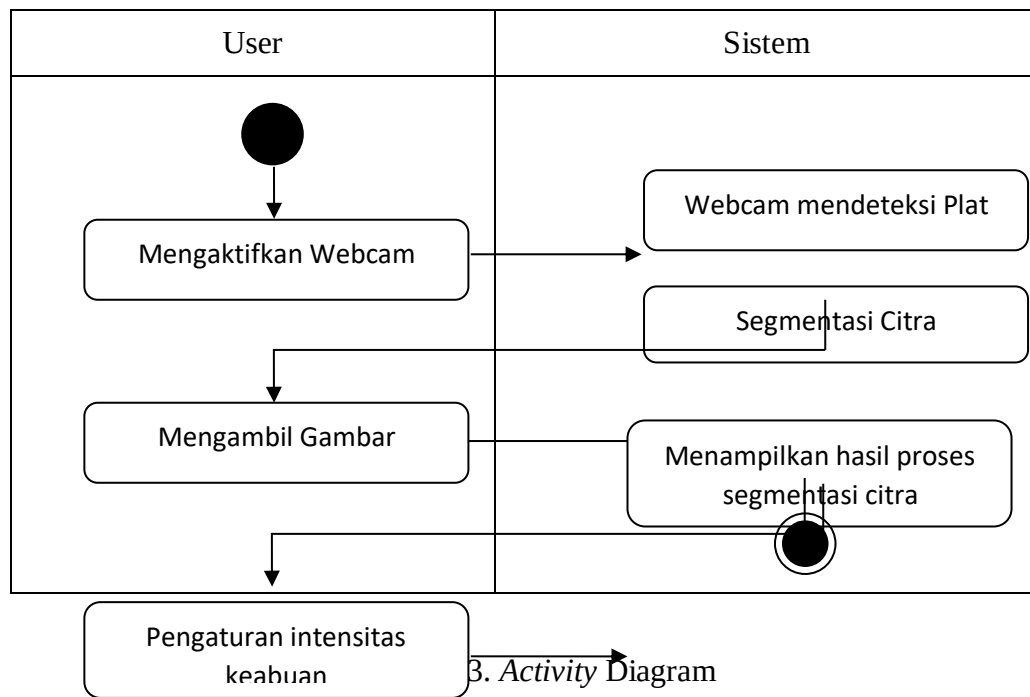


Gambar 4.2. *Use Case Diagram*Penjelasan *Use Case Diagram*a. *Actor Pengguna*Tabel 4.1. Tabel *Actor Pengguna*

Nama <i>Use Case</i>	Deskripsi <i>Use Case</i>
Mengaktifkan Webcam	<i>Use Case</i> ini menjelaskan proses untuk mengaktifkan kamera (webcam)
Deteksi Plat	<i>Use Case</i> ini menjelaskan proses pengambilan gambar benda untuk medeksi Plat
Segmentasi Citra	<i>Use Case</i> ini menjelaskan proses segmentasi dari citra asli ke citra berskala keabuan (<i>grayscale</i>) kemudian ke citra monokrom (biner).
Atur Intensitas Keabuan	<i>Use Case</i> ini menjelaskan proses pengaturan ambang intensitas keabuan agar citra biner dapat membedakan antara objek plat dengan latar belakang
(<i>Output</i>)	<i>Use Case</i> ini merupakan informasi hasil dari proses segmentasi citra
Menghentikan Proses	<i>Use Case</i> ini menghentikan proses aplikasi

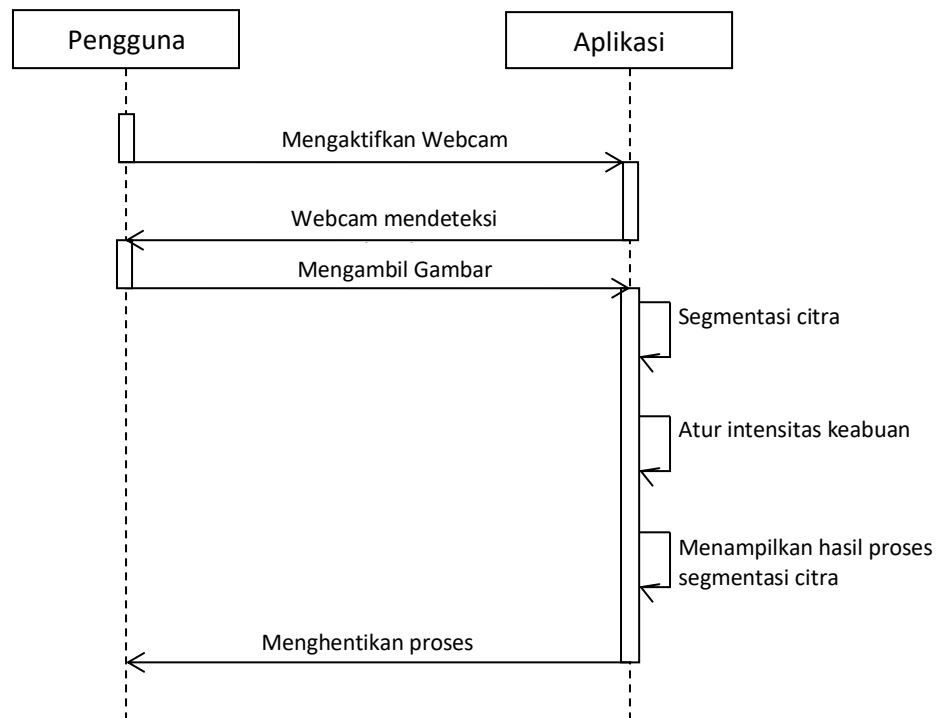
2. Activity Diagram

Activity diagram ini menjelaskan tentang aktifitas-aktifitas yang terjadi dalam sebuah aliran proses pada sebuah sistem.



3. Sequence Diagram

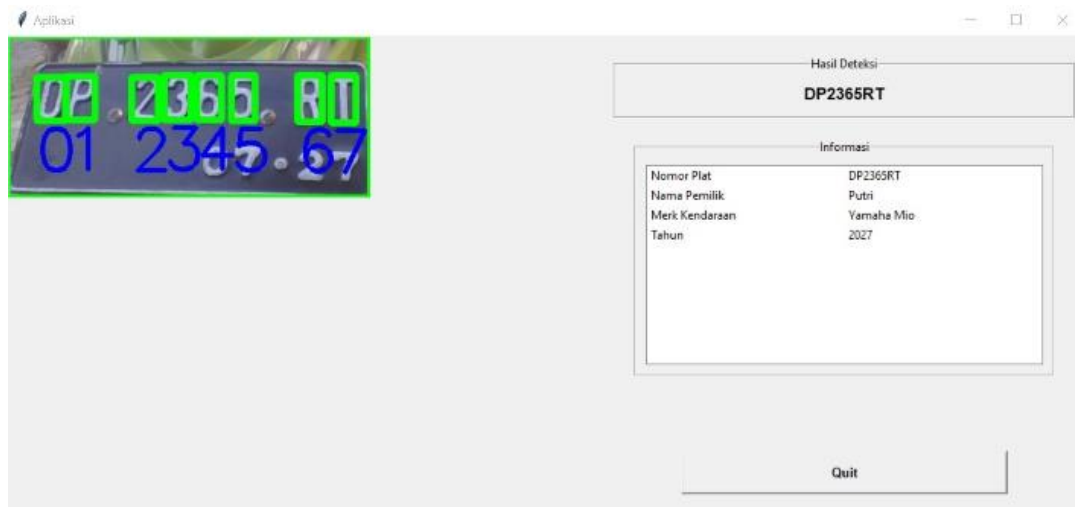
Diagram *sequence* merupakan salah satu diagram *Interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan; *message* (pesan) apa yang dikirim dan kapan pelaksanaannya.



Gambar 4.4. *Sequence Diagram*

C. Detail Aplikasi

Aplikasi deteksi Plat berbasis Python ini merupakan aplikasi yang digunakan untuk mengukur dimensi benda/objek. Aplikasi ini terkoneksi dengan kamera, jadi hanya dengan menggunakan kamera telepon genggam atau Handphone dapat dengan cepat mengukur benda, mendeteksi dimensi benda persegi dan persegi panjang secara otomatis dan juga dapat melihat riwayat deteksi Plat dan menyimpan foto hasil deteksi Plat. Jadi fungsi aplikasi tersebut sangat simpel, yakni untuk mengukur dimensi benda/objek di sekitar kita dengan akurat tanpa harus mengandalkan penggaris maupun meteran.

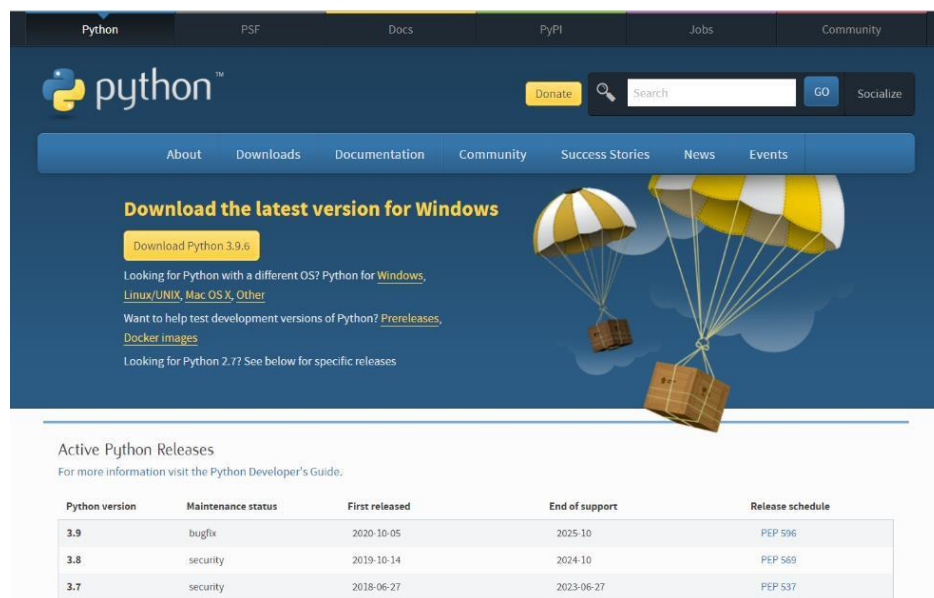


Gambar 4.5 deteksi Plat

D. Instalasi Phyton

3. Instalasi Python

Pada penelitian ini, penulis menggunakan bahasa Python untuk melakukan pendeteksian.



Gambar 4.6 Tampilan website python

4. Instalasi OpenCV

Selanjutnya melakukan instalasi OpenCV. Disini penulis menggunakan OpenCV 3 dengan code seperti pada Gambar 4.7.

```
>pip install opencv-python
```

Gambar 4.7 Kode program instalasi OpenCV

E. Deteksi Plat

Pada proses deteksi, setelah semua *software* yang dibutuhkan telah terinstalasi dengan baik. Langkah selanjutnya adalah menggabungkan file *weight*, *configuration*, dan dataset pada satu folder.

10. Import library

Menggunakan *library cv2, numpy, dan time*. Library cv2 berfungsi untuk melakukan computer vision task. Library numpy berfungsi untuk pengolahan data numerical.

```
import cv2
import numpy as np
import time

print(cv2.__version__)
```

Gambar 4.8 Kode program import library

```
C:\Users\HP\anaconda3\envs\OpenCV\python.exe C:/Users/HP/PycharmProjects/pythonProject/detect.py
4.5.1
```

Gambar 4.9 Hasil dari keluaran versi OpenCV

11. Membaca Masukan benda

Pada bagian ini dilakukan membaca masukan benda dengan mendefinisikan `cv2.VideoCapture()` untuk mendapatkan objek dari benda capture pada kamera sesuai dengan lokasi penyimpanan. Lalu menggunakan variabel `'writer'` yang digunakan untuk menulis *frame* yang diproses. Kemudian menggunakan variabel `'h'` dan `'w'` untuk dimensi *frame*.

```
writer = None
h, w = None, None
```

```
with open('Plat.names') as f:
    labels = [line.strip() for line in f]

print('List with labels names:')
print(labels)

layers_names_all = network.getLayerNames()
layers_names_output = \
    [layers_names_all[i[0] - 1] for i in
network.getUnconnectedOutLayers()]
probability_minimum = 0.5
threshold = 0.5
colours = np.random.randint(0, 255, size=(len(labels), 3),
dtype='uint8')
```

Gambar 4.10 Kode program memuat *network*

```
List with labels names:
['mobil', 'sepeda motor', 'becak', 'truk', 'bus']
```

Gambar 4.11 Hasil keluaran *labels*

12. Membaca *frame* untuk perulangan

Kemudian melakukan perulangan pada *frame*. Disini akan dilakukan fungsi perulangan untuk menangkap *frame-by-frame* dengan fungsi *read*, yaitu membaca *frame* dari Plat masukan. Jika *frame* tidak diambil, misalnya pada akhir aplikasi memunculkan sebuah Plat sebuah benda, maka

```
f = 0
t = 0

while True:
    ret, frame = benda.read()
    if not ret:
        break
    if w is None or h is None:
        h, w = frame.shape[:2]
```

Gambar 4.12 Kode program membaca *frame* untuk perulangan

13. Mengambil fungsi *blob* dari *frame*

Kemudian, data akan diolah menggunakan library OpenCV untuk diubah menjadi bentuk blob (*Binary Large Object*) dari *frame*. Menggunakan fungsi '*cv2.dnn.blobFromImage*' akan mengembalikan 4-dimensional blob dari frame saat ini setelah pengurangan rata-rata, normalisasi, dan pertukaran saluran RB (Irwan, Putrada, & Prabowo, 2019). Pada fungsi

'*cv2.dnn.blobFromImage*' berisi parameter frame yang merupakan masukan gambar yang akan diproses melalui jaringan saraf dalam bentuk klasifikasi. 1/255.0 sebagai scale factor untuk melakukan pengurangan rata-rata. Plat (416,416) adalah Plat benda. *swapRb* memberikan asumsi pada gambar berada dalam saluran BGR, namun nilai mean mengasumsikan kita menggunakan urutan RGB. Untuk mengatasi perbedaan ini, kita dapat menetapkan nilai menjadi '*True*' untuk

menukan saluran R dan B. Bentuk yang dihasilkan memiliki jumlah bingkai, jumlah saluran, lebar dan tinggi. Blob digunakan untuk mengekstrak fitur gambar dan mengubah Platnya, yaitu 320x320,

416x416, dan 609x609. Disini penulis menggunakan Plat frame 416x416 agar proses deteksi memiliki kecepatan dan akurasi yang seimbang.

```
blob = cv2.dnn.blobFromImage(frame, 1 / 255.0, (416, 416),
                             swapRB=True, crop=False)
```

Gambar 4.13 Kode program mengambil fungsi *blob* dari *frame*

14. Menerapkan *Forward Pass*

Mengimplementasi *forward pass* dengan '*blob*' melalui lapisan keluaran.

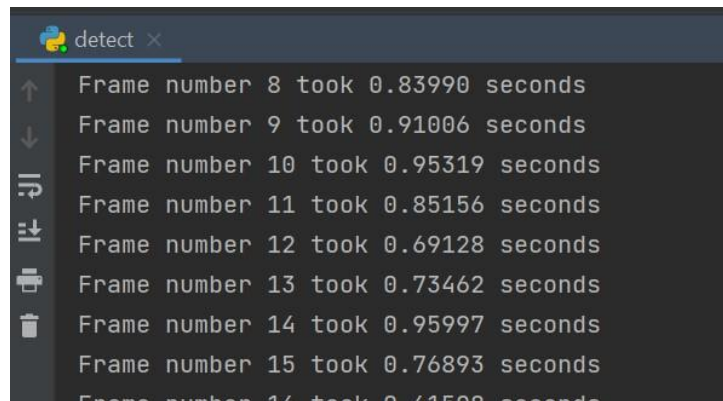
Lalu, menghitung waktu yang dibutuhkan untuk *forward pass*.

```
network.setInput(blob)
start = time.time()
output_from_network = network.forward(layers_names_output)
end = time.time()

f += 1
t += end - start

print('Frame number {0} took {1:.5f} seconds'.format(f, end - start))
```

Gambar 4.14 Kode program menerapkan *Forward Pass*



Gambar 4.15 Contoh hasil dari proses deteksi per *frame*

15. Mendapatkan *bounding boxes*

Pada langkah ini dilakukan ekstraksi seluruh informasi objek yang dideteksi dan menampilkannya pada layar. Melakukan inisialisasi list *bounding_boxes* untuk kotak pembatas yang mengelilingi objek. *Confidences* memberikan nilai keyakinan untuk suatu objek, nilai keyakinan (*confidence*) dari objek yang terdeteksi bernilai 0 sampai 1. *ClassIDs* untuk memberikan label kelas objek yang terdeteksi.

Melakukan perulangan untuk setiap *output_from_network* dan perulangan untuk setiap *detected_object* pada *result* dengan mengekstrak *class_current* dan *confidence_current*. Menggunakan perintah *confidence_current* untuk menyaring prediksi yang lemah dengan memastikan objek yang terdeteksi dengan probabilitas lebih besar dari probabilitas minimum. Kemudian melakukan skala koordinat kotak pembatas untuk menampilkan dengan benar pada gambar asli. Lalu, mengekstraksi koordinat dan dimensi dari kotak pembatas dengan mengembalikan koordinat kotak pembatas dalam bentuk: *x_center*, *y_center*, *box_width*, *box_height*. Menggunakan

informasi untuk mendapatkan *top-left (x, y)-coordinates* dari kotak pembatas. Lalu memperbarui list *bounding_boxes*, *confidences*, dan *classIDs*.

```

bounding_boxes = []
confidences = []
classIDs = []

for result in output_from_network:

    for detected_objects in result:
        scores = detected_objects[5:]
        class_current = np.argmax(scores)
        confidence_current = scores[class_current]

        if confidence_current > probability_minimum:
            box_current = detected_objects[0:4] *
np.array([w, h, w, h])
            x_center, y_center, box_width, box_height =
box_current
            x_min = int(x_center - (box_width / 2))
            y_min = int(y_center - (box_height / 2))
            bounding_boxes.append([x_min, y_min,
int(box_width),
int(box_height)])
            confidences.append(float(confidence_current))
            classIDs.append(class_current)

```

Gambar 4.16 Kode program mendapatkan *bounding boxes*

16. Non-maximum Suppresion

Menerapkan *non-maximum suppression* atau yang disebut juga *non-maxima suppression* (NMS). Objek yang terdapat pada video atau gambar dapat memiliki Plat dan bentuk yang berbeda, dan untuk menangkap setiap objek maka algoritma akan memuat kotak pembatas (*bounding boxes*). Jadi, untuk setiap objek yang terdapat pada video atau gambar harus memiliki satu kotak pembatas (*bounding boxes*). Untuk mendapatkan kotak pembatas (*bounding boxes*) terbaik dari beberapa kotak pembatas (*bounding boxes*) yang diprediksi, algoritma akan menggunakan fungsi *non-max*. teknik ini digunakan untuk “menekan” kotak

pembatas yang kemungkinannya kecil dan hanya menyimpan yang terbaik. Tujuan dari *non- maximum suppression* adalah untuk memilih kotak pembatas (*bounding boxes*) terbaik pada suatu objek. NMS akan mempertimbangkan dua hal, yang pertama skor objektifitas yang diberikan oleh model, dan yang kedua IoU dari kotak pembatas (*bounding boxes*) (Hosang & May, n.d.).

Pada Gambar 4.18 memanfaatkan implementasi modul DNN pada OpenCV, kemudian melakukan *non-maximum suppression* dengan parameter *bounding_boxes*, *confidences*, *probability_minimun*, dan *threshold*.

```
results = cv2.dnn.NMSBoxes(bounding_boxes, confidences,
                           probability_minimun, threshold)
```

Gambar 4.17 Kode program *non-maximum suppression*

17. Mendapatkan *bounding boxes* dengan *labels*

Menggunakan perintah *if* untuk memeriksa jika setidaknya ada satu objek yang terdeteksi setelah NMS. Dengan mendapatkan *bounding box* saat ini, lebar, dan tingginya. Kemudian mempersiapkan warna untuk *bounding box* saat ini dan mengkonversi dari *array* *numpy*. Setelah mendapatkan warna, selanjutnya menggambar *bounding box* pada *frame*. Setelah itu, mempersiapkan teks dengan label dan nilai *confidence* untuk *bounding box* saat ini dan menampilkan hasilnya.

```
if len(results) > 0:
    for i in results.flatten():
        x_min, y_min = bounding_boxes[i][0], bounding_boxes[i][1]
        box_width, box_height = bounding_boxes[i][2], bounding_boxes[i][3]
        colour_box_current = colours[classIDs[i]].tolist()

        cv2.rectangle(frame, (x_min, y_min), (x_min + box_width, y_min
+
        box_height), colour_box_current, 2)
        text_box_current = '{}: {:.4f}'.format(labels[int(classIDs[i])],
        confidences[i])
        cv2.putText(frame, text_box_current, (x_min, y_min - 5),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, colour_box_current,
2)
```

Gambar 4.18 Kode program mendapatkan *bounding boxes* dengan *labels*

18. Menyimpan video proses deteksi

Pada bagian ini, akan menyimpan hasil prediksi masukan video dengan mengecek 'writer', kemudian 'writer' akan diinisialisasi pada iterasi pertama dari perulangan. Mengeluarkan perkiraan waktu yang dibutuhkan untuk proses video. Dan mengeluarkan hasil akhir dari proses deteksi pada masukan video.

```
if writer is None:
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    writer = cv2.VideoWriter('resultvideo8juli.mp4', fourcc, 30,
                            (frame.shape[1], frame.shape[0]),
                            True)

writer.write(frame)

# Printing final results print()
print('Total number of frames', f)
```

Gambar 4.19 Kode program menyimpan hasil deteksi

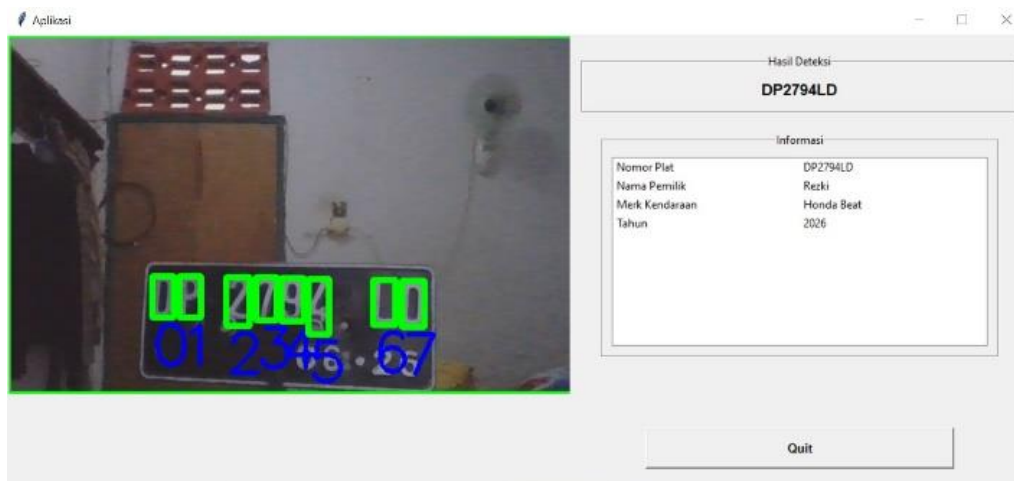
```
Total number of frames 237
Total amount of time 139.25257 seconds
```

Gambar 4.20 Contoh hasil keluaran akhir deteksi

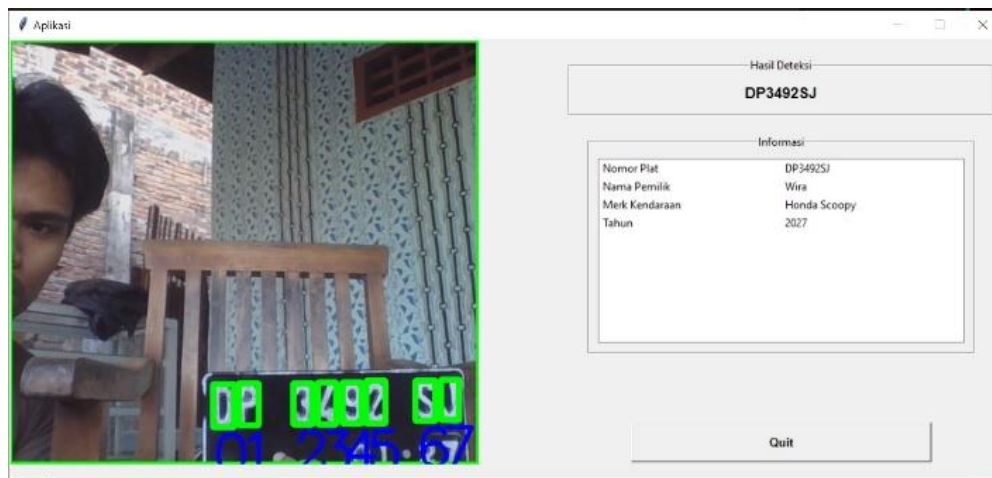
D. Pengujian

1. Pengujian pada hasil deteksi

Pada pengujian deteksi, menggunakan *pre-trained weights* dengan lima kelas dilakukan untuk mengetahui keakuratan model dalam mendeteksi kelas yang sudah dilatih. Pengujian dapat dilakukan menggunakan program yang telah dibangun dan menggunakan model *weights* yang telah dilatih. Program kemudian di-*run* dengan mengaktifkan kamera secara otomatis untuk mendapatkan seberapa akurat model *weights* dapat mendeteksi Plat. Keluaran dari hasil deteksi adalah bounding box pada objek yang terdeteksi dan nilai confidence seperti.



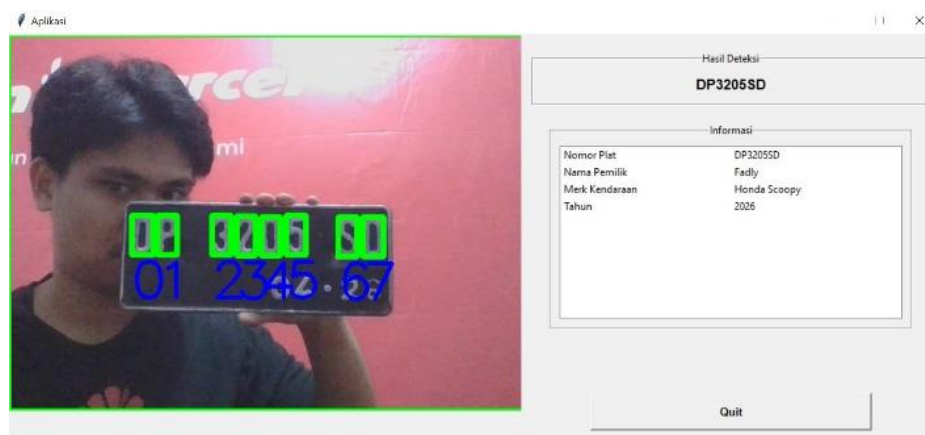
Gambar 4.21 Hasil deteksi Plat 1



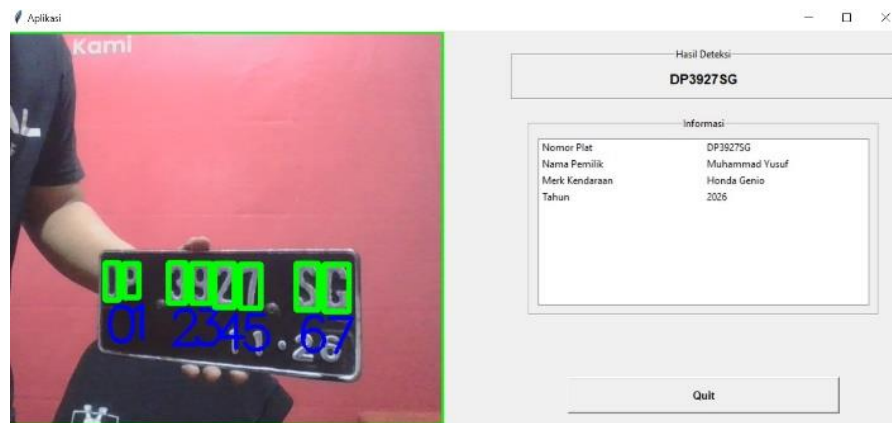
Gambar 4.22 Hasil deteksi Plat 2



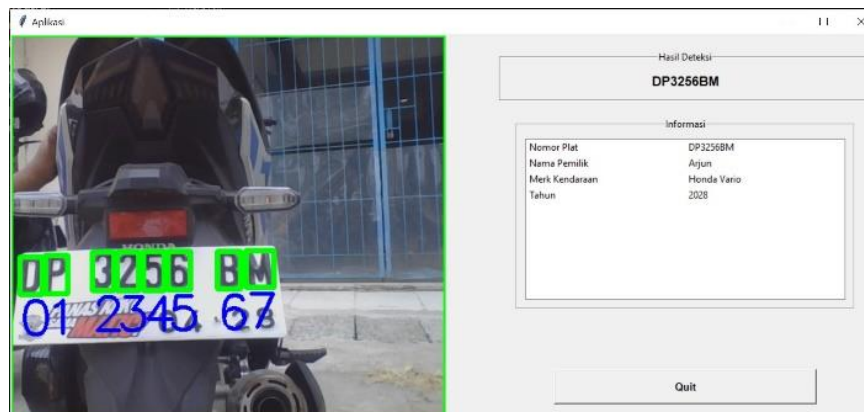
Gambar 4.23 Hasil deteksi Plat 3



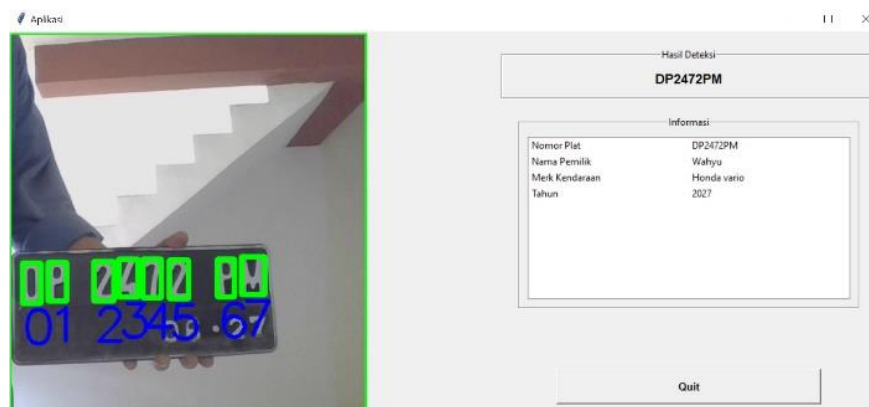
Gambar 4.24 Hasil deteksi Plat 4



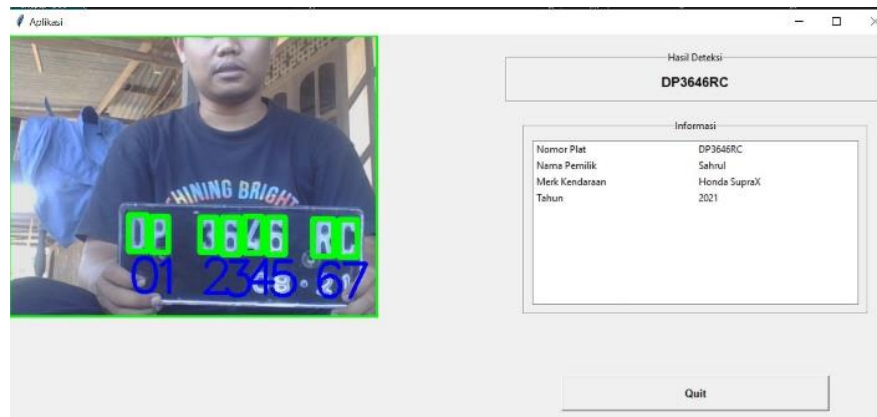
Gambar 4.25 Hasil deteksi Plat 5



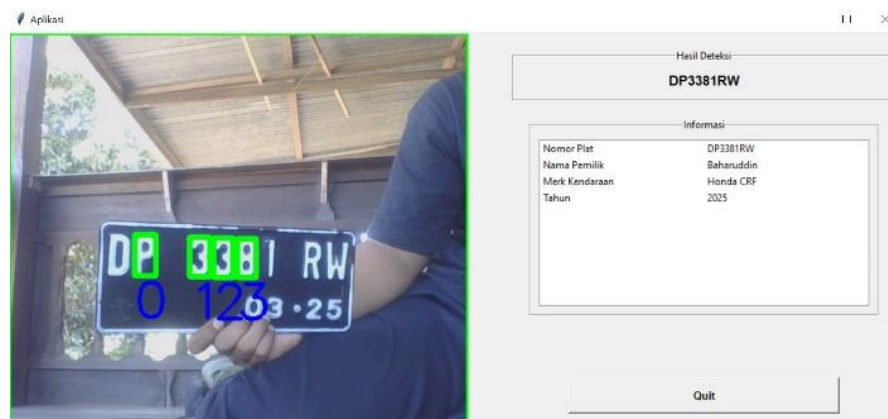
Gambar 4.26 Hasil deteksi Plat 6



Gambar 4.27 Hasil deteksi Plat 7



Gambar 4.28 Hasil deteksi Plat 8



Gambar 4.29 Hasil deteksi Plat 9

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap daftar fungsi aplikasi terhadap tujuan dibuatnya aplikasi ini, yakni melakukan deteksi plat dan menghasilkan identitas pada plat yang terdeteksi berdasarkan kepemilikannya.

2. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan Tugas Akhir ini dilakukan pada lingkungan dan alat sebagai berikut:

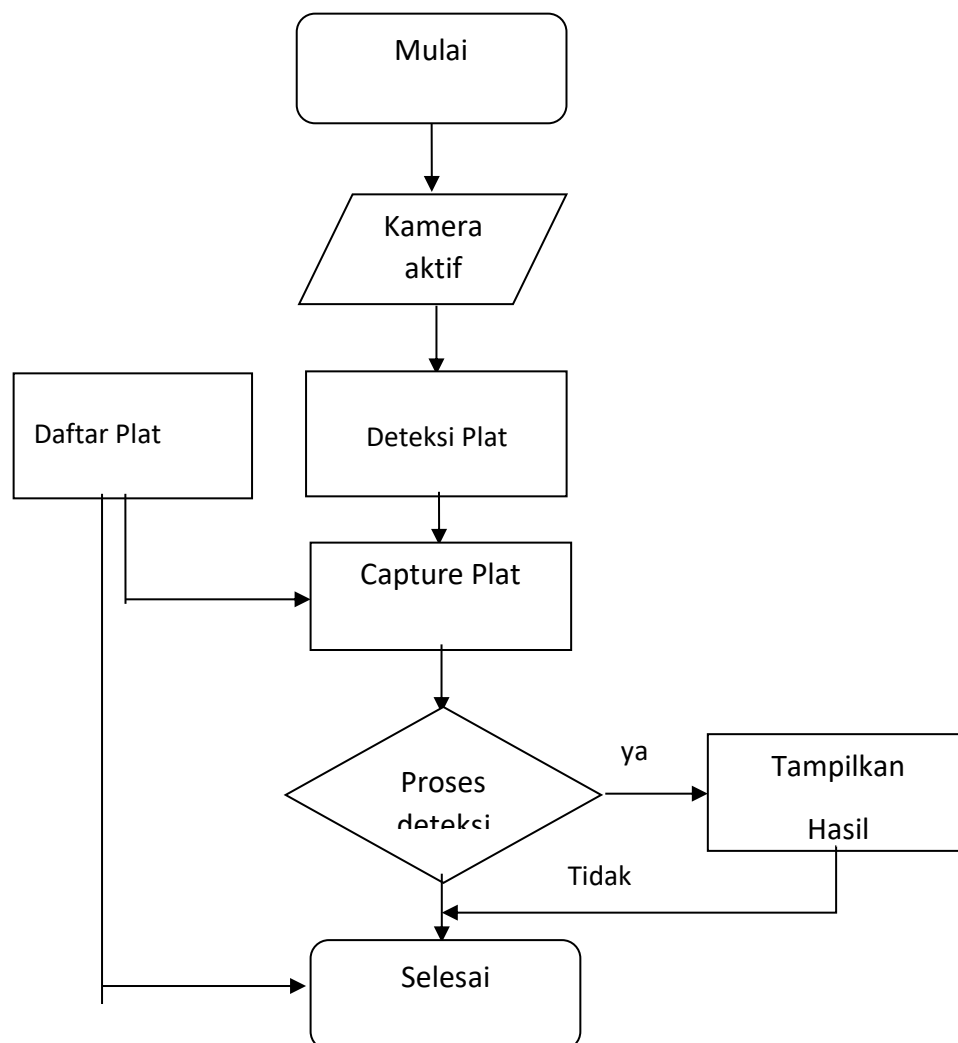
Prosesor : Prosesor Intel® Core™ i5-CPU
 RAM : 4 GB

Jenis *Device* : Personal Computer

Sistem Operasi : Windows 10 Professional

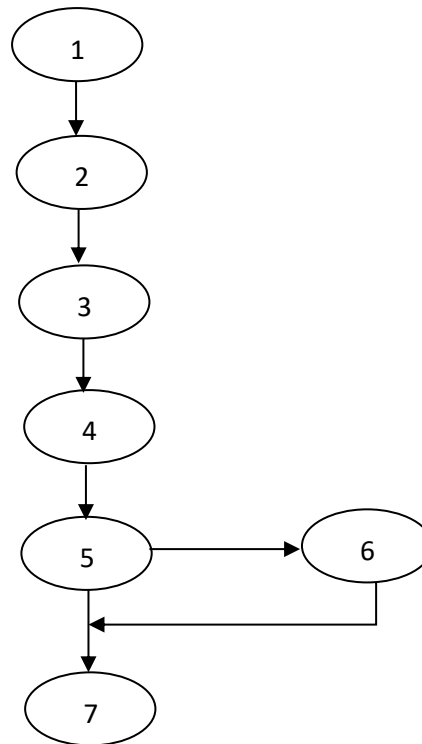
3. Pengujian *White box*

a. *Flowchart* Aplikasi



Gambar 4.30. *Flowchart* Aplikasi

b. Flowgraph Aplikasi



Gambar 4.31 . *Flowgraph* Aplikasi

1) Proses Perhitungan

Dari Gambar *FlowGraph* di atas dapat di lakukan proses perhitungan sebagai berikut :

a. Menghitung *Cyclomatic Complexity* $V(G) = E - N + 2$

$$N (\text{node}) = 7$$

$$E (\text{edge}) = 7$$

$$P (\text{predikat node}) = 1$$

$$\text{Penyelesaian : } V(G) = E - N + 2$$

$$= 7 - 7 + 2$$

$$= 2$$

$$\text{Predikat Node (N)} = P + 1$$

$$= 1 + 1$$

$$= 2$$

b. Berdasarkan perhitungan *Cyclomatic Complexity* dari Flowgraph diatas

memiliki *Region* = 2

c. *Independent Independent Path* pada *flowgraph* diatas adalah :

$$\text{Path 1} = 1 - 2 - 3 - 4 - 5 - 6 - 7$$

$$\text{Path 2} = 1 - 2 - 3 - 4 - 5 - 7$$

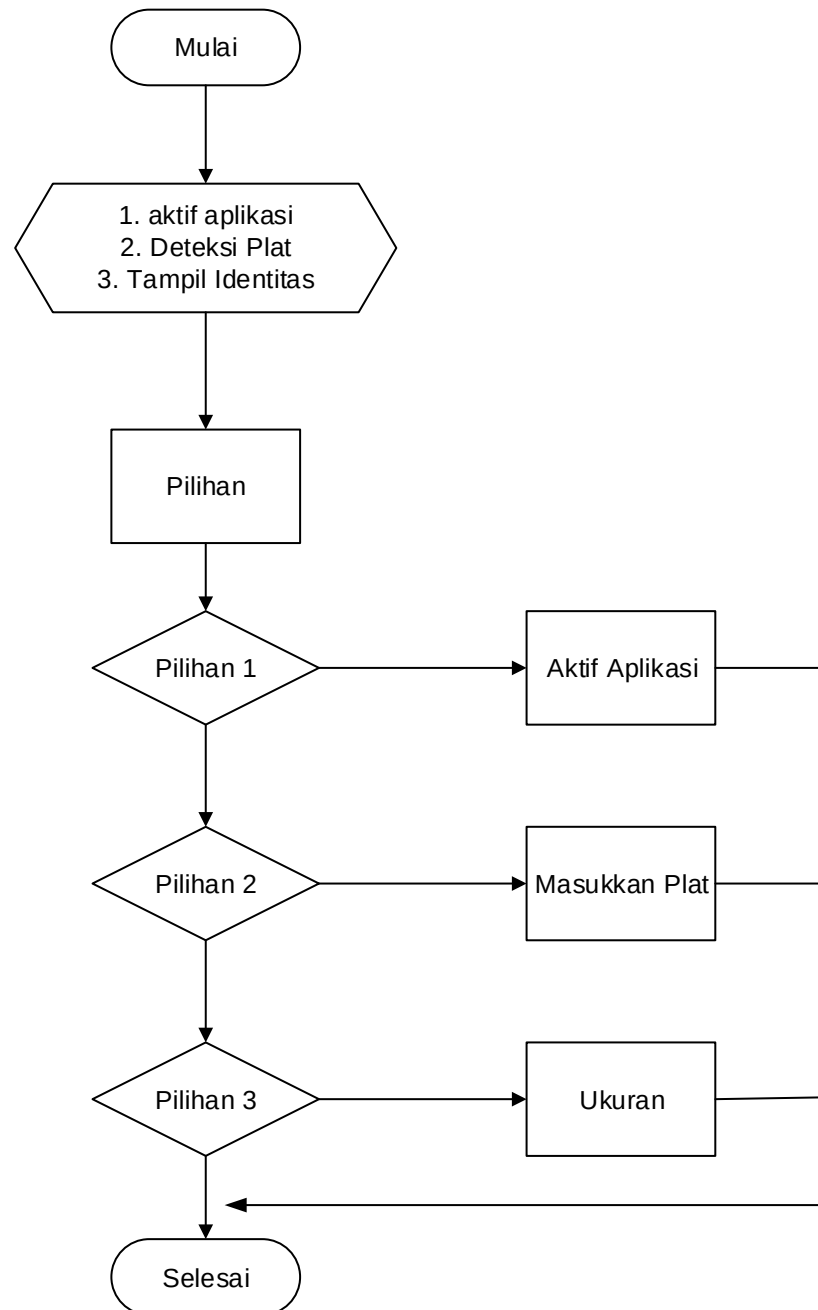
d. Grafik Matriks

Tabel 4.2. Grafik *Matriks*

	1	2	3	4	5	6	7	E - 1
1		1						1-1 = 0
2			1					1-1 = 0
3				1				1-1 = 0
4					1			1-1 = 0
5						1	1	2-1 = 1
6							1	1-1 = 1
7								
Zum (E+1)								1+1 = 2

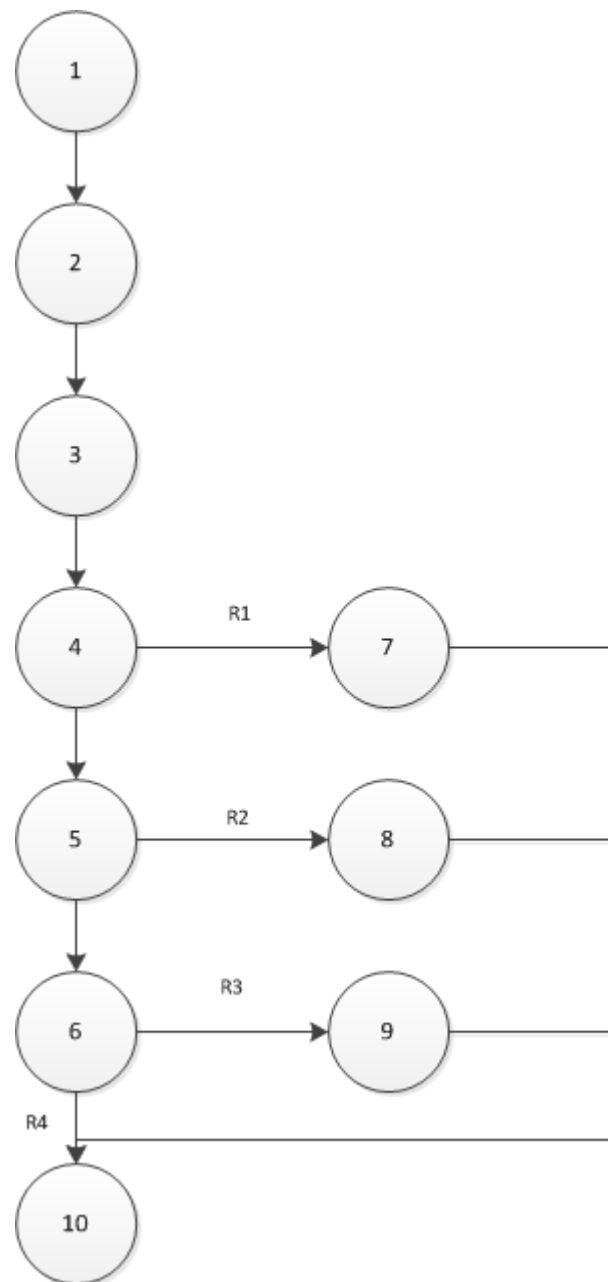
a. *White box* menjalankan aplikasi

1) *Flowchart* menjalankan aplikasi



Gambar 4.32 *Flowchart* menjalankan aplikasi

2) *Flowgraph* menjalankan aplikasi



Gambar 4.33 *Flowgraph* menjalankan aplikasi

3) Proses perhitungan menjalankan aplikasi

Dari gambar *flowgraph* diatas dapat dilakukan proses perhitungan sebagai berikut :

a) Menghitung *Cyclomatic Complexity* $V(G)$ dari *Edge* dan *Node*:

Dengan Rumus : $V(G) = E - N + 2$

$$N(\text{node}) = 10$$

$$E(\text{edge}) = 12$$

$$P(\text{predikat node}) = 3$$

$$\begin{aligned} \text{Penyelesaian : } V(G) &= E - N + 2 \\ &= 12 - 10 + 2 \\ &= 4 \end{aligned}$$

$$\begin{aligned} \text{Predikat Node (N)} &= P + 1 \\ &= 3 + 1 \\ &= 4 \end{aligned}$$

b) Berdasarkan perhitungan *Cyclomatic Complexity* dari *Flowgraph* diatas memiliki *Region* = 4

c) *Independent Path* pada *flowgraph* diatas adalah :

$$\text{Path 1} = 1 - 2 - 3 - 4 - 7 - 10$$

$$\text{Path 2} = 1 - 2 - 3 - 4 - 5 - 8 - 10$$

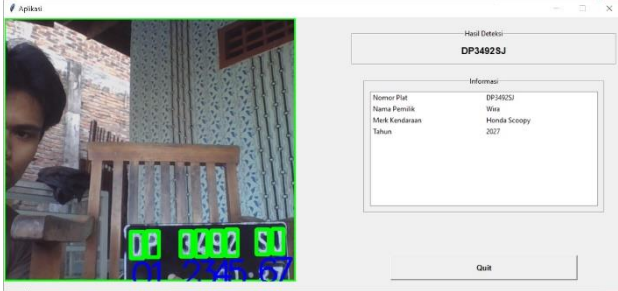
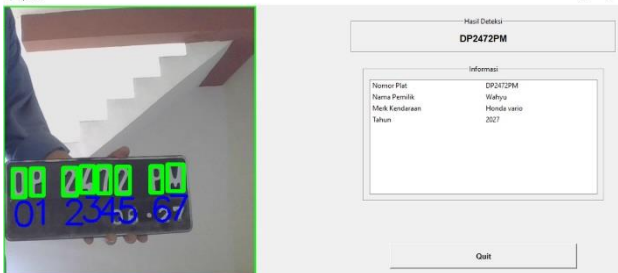
$$\text{Path 3} = 1 - 2 - 3 - 4 - 5 - 6 - 9 - 10$$

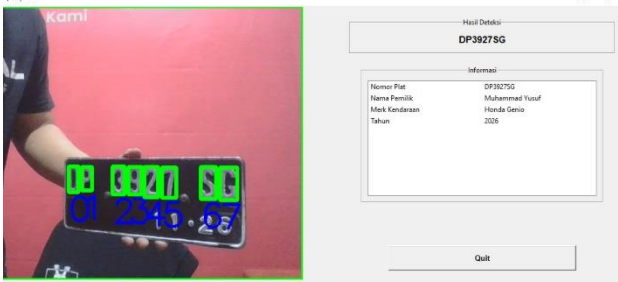
$$\text{Path 4} = 1 - 2 - 3 - 4 - 5 - 6 - 10$$

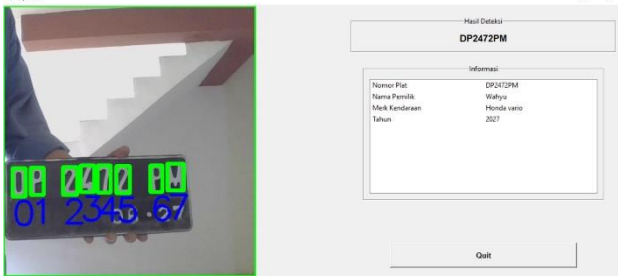
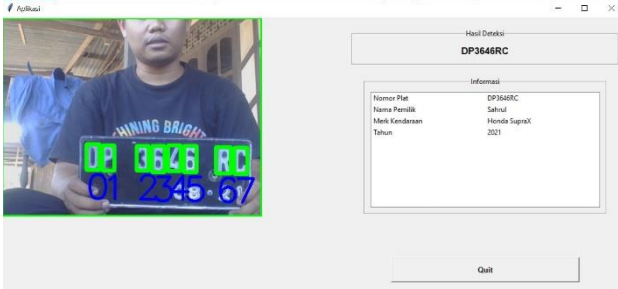
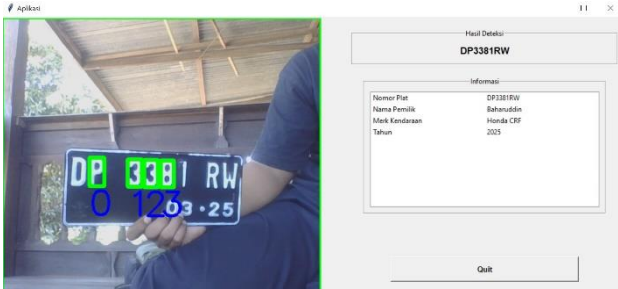
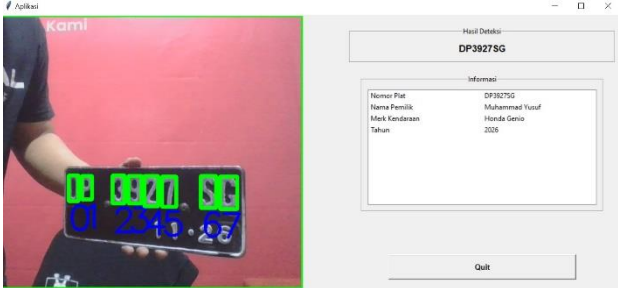
- e) Uji akurasi dari pendeteksian *Plat benda* pada *sample* yang digunakan adalah sebagai berikut :

1. Pengaruh gerak terhadap Plat Plat

Tabel 4.4 Tabel Pengujian Plat

Jarak	Gambar	Ket
10 cm		Terdeteksi
20 cm		
30 cm		Terdeteksi
35 cm		Terdeteksi
		Terdeteksi

Jarak	Gambar	Ket
10 cm		Terdeteksi
20 cm		Terdeteksi
30 cm		
35 cm	 	Terdeteksi

Jarak	Gambar	Ket
10 cm		Terdeteksi
20 cm		Terdeteksi
30 cm		Terdeteksi
35 cm		Terdeteksi

Sebagai kesimpulan bahwa pada aplikasi yang telah kami rancang bahwa tingkat akurasi untuk deteksi sebuah plat hanya 80 % kemudian plat yang hanya bisa di deteksi identitasnya hanya benda dalam bentuk segi empat saja itu disebabkan karena benda tersebut memiliki sudut masing-masing, sedangkan ketika plat tidak dalam bentuk persegi atau warna yang berbeda maka aplikasi tidak bisa mendeteksi

plat tersebut disebabkan karena aplikasi tersebut hanya membaca panjang dan lebar benda tersebut.

Jarak	Plat pertama	Plat kedua	Plat ketiga	Plat keempat	Ket
10 cm	✓	✓	✓	✓	✓ = Tampil ✕ = Tidak Tampil
30 cm	✓	✓	✓	✓	
40 cm	✓	✓	✓	✓	
50 cm	✕	✕	✕	✕	
150 cm	✕	✕	✕	✕	
160 cm	✕	✕	✕	✕	

BAB V

PENUTUP

A. Kesimpulan

Dari banyaknya hasil pembahasan di atas dapat disimpulkan beberapa diantaranya adalah :

1. Sistem aplikasi Deteksi Plat yang melintas tidak hanya menggunakan metode *object detection* melainkan juga menggunakan beberapa *libray* yang ada pada aplikasi Python Anaconda.
2. Hasil pengujian aplikasi Deteksi Plat berdasarkan *object detection* telah berhasil mendeteksi Plat sebuah benda walaupun tanpa menggunakan satu buah lampu LED, karena seluruh bagian Plat benda depan sudah cukup tersinari dan juga dibutuhkan jarak yang ideal untuk dapat mendeteksi bentuk Plat benda, tidak terlalu dekat ke kamera ataupun jauh dari kamera. Selain itu pada pengujian pergerakan Plat benda ketika jarak kamera ke benda terlalu dekat, pergerakan yang dapat dilakukan lebih sedikit, idealnya jarak yang diperlukan untuk melakukan pergerakan Plat benda pada 10 cm – 40 cm sehingga aplikasi memiliki akurasi yang baik untuk melakukan pergerakan Plat benda ke kiri, ke kanan, ke atas dan ke bawah. Jarak Plat benda ke kamera juga mempengaruhi Plat besar kecilnya.

B. Saran

Dalam perancangan dan pengujian yang telah dijalankan oleh penulis, terdapat beberapa hal yang dapat ditambahkan supaya hasil perancangan lebih baik dari penulis, diantaranya adalah:

1. Menambahkan semua jenis Plat yang terdeteksi tidak hanya mendeteksi Plat dengan benda kecil saja.
2. Dapat menambahkan nilai akurasi dari aplikasi Deteksi Plat berdasarkan metode *object detection* yang telah dibuat penulis.
3. Membandingkan nilai akurasi dari aplikasi Deteksi Plat

DAFTAR PUSTAKA

Adi nugroho 2017 dalam buku “Rekayasa Perangkat Lunak Berorientasi Objek”
Penerbit ANDI

1. Emanuel Goldberg 2018, Optical Character Recognition (OCR) adalah teknik mengubah sebuah gambar berisi teks, tulisan tangan menjadi teks
- 2.
3. Guido Van Rossum 2019, Python salah satu bahasa pemrograman yang dapat melakukan eksekusi sejumlah instruksi multi guna secara langsung (interpretatif) dengan metode orientasi objek

Irawan. 2018. *Membuat Alikasi Android untuk Orang Awam*. Palembang: Maxkom.

Jay A. Kreibich 2018, Pengenalan SQLite, Bandung, Penerbit ANDI

Jugianto, 2018. Aplikasi Pada Sebuah Komputer. Penerbit. Andi

Pratiwi dan Agus Harjoko pada tahun 2013 dengan judul Implementasi Pengenalan Wajah Menggunakan PCA

Shalahuddin 2017, *Pengenalan Pemrograman java* Penerbit Andi

Shivam Sharma, 2015. Speech Recognition with Hidden Markov Model: A Review. *international Journal of Scientific & Engineering Research*, Volume 6, Issue 11, November-2015.

Timo Ahonen, Abdenour Hadid, and Matti Pietikäinen dengan Face Recognition with Local