DETEKSI DAN PENGENALAN PLAT NOMOR MOBIL MENGGUNAKAN *VERTICAL EDGE DETECTION* DAN *BACKPROPAGATION NEURAL NETWORK*

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun oleh :

DWY SAPUTRO NUGROHO

NIM. 105090600111025

PROGRAM STUDI INFORMATIKA / ILMU KOMPUTER

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2014

LEMBAR PERSETUJUAN

DETEKSI DAN PENGENALAN PLAT NOMOR MOBIL MENGGUNAKAN *VERTICAL EDGE DETECTION* DAN *BACKPROPAGATION NEURAL NETWORK*

SKRIPSI
KONSENTRASI KOMPUTASI CERDAS DAN VISUALISASI

Untuk memenuhi sebagian persyaratan untuk
Mencapai gelar Sarjana Komputer

Disusun Oleh:
DWY SAPUTRO NUGROHO
105090600111025

Telah diperiksa dan disetujui oleh
Dosen Pembimbing

Pembimbing I	Pembimbing II
Ahmad Afif Supianto, S.Si., M.Kom.	**Imam Cholissodin, S.Si., M.Kom.**
NIP. 820623 16 1 1 0425	**NIP. 850719 16 1 1 0422**

LEMBAR PENGESAHAN

DETEKSI DAN PENGENALAN PLAT NOMOR MOBIL MENGGUNAKAN *VERTICAL EDGE DETECTION* DAN *BACKPROPAGATION NEURAL NETWORK*

SKRIPSI

LABORATORIUM KOMPUTASI CERDAS DAN VISUALISASI
Untuk memenuhi sebagian persyaratan untuk mencapai gelar Sarjana Komputer

Disusun oleh :

Dwy Saputro Nugroho
105090600111025

Setelah dipertahankan di depan Majelis Penguji
pada tanggal 2 Oktober 2014
dan dinyatakan memenuhi syarat untuk memperoleh
gelar sarjana dalam bidang Ilmu Komputer

Penguji I,	**Penguji II,**
Indriati, ST.,M.Kom	Rekyan Regasari MP, ST.,MT.
NIK. 831013 06 1 2 0035	**NIK. 770414 06 1 2 0253**

Penguji III,

Adharul Muttaqin, ST.,MT.
NIP. 197601212005011001

Mengetahui,
Ketua Program Studi Informatika / Ilmu Komputer

Drs. Marji, MT.
NIP. 196708011992031001

PERNYATAAN
ORISINALITAS SKRIPSI

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata di dalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 danPasal 70.

Malang, 22 Agustus 2014

Mahasiswa,

Dwy Saputro Nugroho

NIM. 105090600111025

KATA PENGANTAR

Syukur Alhamdulillah penulis panjatkan ke hadirat Allah SWT karena atas limpahan Rahmat dan Hidayah-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Deteksi dan Pengenalan Plat Nomor Mobil Menggunakan *Vertical Edge Detection* dan *Backpropagation Neural Network*". Sholawat serta salam tak lupa juga penulis panjatkan kepada Nabi Besar Muhammad SAW.

Skripsi ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer di Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, penulis mengucapkan terima kasih yang tulus kepada Ayahanda Riyanto dan Ibunda Qodriyah atas segala kasih sayang, kesabaran, nasehat dan Do'a dalam membesarkan dan mendidik penulis. Serta kakak dan kedua adik saya Nuvy, Vany dan Sandy, terima kasih atas segala perhatian motivasi dan Do'anya.

Terima kasih kepada Bapak Ahmad Afif Supianto, SSi., M.Kom selaku Dosen pembimbing skripsi pertama atas ilmu, motivasi dan waktu yang telah diberikan kepada penulis sehingga penulis dapat menyelesaikan skripsi ini.

Terima kasih kepada Bapak Imam Cholissodin, S.Si.,M.Kom selaku Dosen pembimbing skripsi kedua yang telah meluangkan waktu dan juga memberikan pengarahan kepada penulis.

Terima kasih kepada Bapak Drs. Marji, MT. selaku Ketua Program Studi Informatika/Ilmu Komputer Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya. Terima kasih kepada Bapak Ir. Sutrisno, MT., selaku Ketua Program Teknologi Informasi & Ilmu Komputer Universitas Brawijaya.

Terima kasih kepada teman-teman seperjuangan yang telah menjadi tempat cerita, keluh kesah dan canda, khususnya untuk sahabat – sahabat Ilmu Komputer angkatan 2010 dan sahabat – sahabat NOS 2010. Terima kasih kepada segenap Bapak/Ibu Dosen Informatika/Ilmu Komputer atas ilmu dan motivasi yang telah diberikan dan Seluruh pihak yang tidak dapat penulis sebutkan baik secara langsung atau tidak lansung telah membantu pada penyusunan skripsi ini. Semoga mendapatkan balasan yang berlipat.

Penulis menyadari bahwa skripsi ini masih jauh dari kesempurnaan. Untuk itu penulis mengharapkan segala saran dan kritik yang membangun untuk penyempurnaan skripsi ini. Penulis juga berharap semoga skripsi ini dapat menjadi manfaat bagi penyusun maupun pihak lain yang berkepentingan.

Malang, 22 agustus 2014

Penulis,

Dwy Saputro N
10509060011102

ABSTRAK

Plat nomor kendaran merupakan nomor identitas kendaraan yang mewakili identitas kendaraan dan asal kendaraan yang biasa digunakan untuk catatan masuk parkir, catatan surat tilang atau bahkan catatan masuk jalan tol. Hingga tahun ajaran 2013-2014 Universitas Brawijaya telah memiliki 52.376 mahasiswa, 1.852 dosen tetap dan 972 karyawan. Berdasarkan surat edaran Rektor Nomor : 5088/UN10/LL/2013 untuk seluruh Dosen, Tenaga Administrasi, Mahasiswa, dan Tamu yang memiliki kendaraan dan memasuki kawasan Universitas Brawijaya wajib meminta karcis masuk dengan alasan keamanan dan tanggung jawab apabila terjadi kehilangan atau kerusakan. Sebuah sistem deteksi plat nomor diperlukan untuk menjamin hak karcis bagi pengendara terpenuhi dan memberikan rasa nyaman dan praktis bagi pengendara. Otomatisasi, *running time*, dan akurasi merupakan beberapa tujuan dalam suatu sistem dan penelitian untuk terus dikembangkan dan menjadi lebih baik.

Deteksi plat nomor dengan *edge-based method* adalah salah satu metode yang sering digunakan karena dinilai sederhana dan cepat. Penelitian ini mengadaptasi *edge-based method* dengan menggunakan *vertical edge detection* untuk deteksi plat nomor dan *backpropagation neural network* untuk pengenalan karakter. Hasil dari pengujian, didapatkan akurasi terbaik deteksi plat nomor pada variabel uji perbesaran kamera 5.6x dengan akurasi 89%. Sedangkan akurasi pengenalan karakter terbaik yang didapatkan sebesar 80% dengan variabel uji pada 7 data latih, MSE 10^{-4} dan *learning rate* 0.25.

Kata Kunci : Deteksi Plat Nomor, *Vertical Edge Detection*, *Histogram Projection*, Pengenalan Karakter, *Backpropagation*, *Neural Network*

ABSTRACT

Vehicle license number represent the identity of the vehicle and the origin of the vehicle that is used to record the parking entrance, entry highway, or speeding ticket. As for as 2013 – 2014 academic year, Brawijaya university has had 52.376 students, 1.852 lecturers dan 972 employees. Based on rector's letter number : 5088/UN10/LL/2013, for all lecturers, employees, students and guests who have vehicle and wanted to enter on campus area, they have to ask a entrance ticket for a waranty of security and responsibility in Brawijaya University, from loss and damage of Vehicle. System of vehicle license number detection is needed to ensure Driver get entrance ticket and provide convenient and efficiency to the Driver. Automation, running time and accuration are some of purpose of the system to develop for better result.

Vehicle license number detection using edge-based method is one of the method which is often used on research because it is simple and fast of the process. This research was adapting edge based method by using vertical edge detection for vehicle license plat detection and backpropagation neural network for recogniting character. The result of testing, best accuration was obtained at testing variable on magnification 5.6x with 89% of accuration. While the best accuration of recognition character was 80% by testing variabel on 7 training data, MSE 10^{-4} and *learning rate* 0.25.

Keyword : Vehicle license plat detection, Vertical Edge Detection, Histogram Projection, Recognition character, Backpropagation, Neural Network

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR SOURCECODE

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi terjadi begitu cepat sejalan dengan kebutuhan manusia yang semakin banyak dan luas. Dengan tujuan mempermudah pekerjaan manusia, teknologi mulai berkembang kesegala aspek untuk memenuhi kebutuhan manusia misal dibidang ekonomi dan bisnis, penelitian, hukum, transportasi dan lain sebagainya. Salah satu pekerjaan manusia yang berhubungan dengan transportasi adalah pencatatan plat nomor kendaraan. Plat nomor kendaran merupakan nomor identitas kendaraan yang mewakili identitas kendaraan dan asal kendaraan yang biasa digunakan untuk catatan masuk parkir, catatan surat tilang atau bahkan catatan masuk jalan tol. Selama ini di Indonesia untuk beberapa kasus pencatatan plat nomor masih dilakukan secara manual tetapi beberapa telah melakukan otomatisasi dalam pencatatannya untuk memberikan rasa nyaman dan praktis bagi pengendara. Otomatisasi, *running time*, dan akurasi merupakan beberapa tujuan dalam suatu sistem dan penelitian untuk terus dikembangkan dan menjadi lebih baik.

Masalah yang diangkat pada penelitian ini adalah mengenai hak karcis di Universitas Brawijaya. Hingga tahun ajaran 2013-2014 Universitas Brawijaya telah memiliki 52.376 mahasiswa, 1.852 dosen tetap dan 972 karyawan [ANO-13]. Jumlah mahasiswa yang sangat banyak untuk suatu Universitas Negeri dan tentu akan memberikan dampak untuk lingkungan disekitarnya. Salah satu dampak buruk dari kasus ini adalah bertambahnya jumlah kendaraan yang keluar masuk di wilayah kampus Universitas Brawijaya. Hal ini mengakibatkan banyaknya antrian pada pencatatan karcis di pintu masuk Universitas Brawijaya. Ditambah lagi berdasarkan surat edaran Rektor Nomor : 5088/UN10/LL/2013 untuk seluruh Dosen, Tenaga Administrasi, Mahasiswa, dan Tamu yang memiliki kendaraan dan memasuki kawasan Universitas Brawijaya wajib meminta karcis masuk dengan alasan keamanan dan tanggung jawab apabila terjadi kehilangan atau kerusakan. Universitas Brawijaya memiliki 3 pintu masuk untuk kendaraan

roda 4 dan 2 pintu masuk untuk kendaraan roda 2. Jika dibandingkan dengan jumlah kendaraan yang akan memasuki kampus, hal ini tidak seimbang dan sering terjadi antrian. Antrian yang bertambah panjang akan menyulitkan pihak pencatat plat nomor sehingga sering kali pencatat plat nomor tidak memberikan hak karcis pada mereka yang memasuki wilayah kampus. Padahal karcis sendiri merupakan jaminan ganti rugi atas kerusakan atau kehilangan dari pihak universitas. Pada kasus ini penulis mengusulkan bahwa perlu adanya suatu sistem yang dapat mendeteksi plat nomor pada kendaraan yang akan masuk dengan harapan lebih cepat dan nyaman untuk memenuhi hak dari pengendara yang memasuki wilayah Universitas Brawijaya. Sistem yang diusulkan ini hanya akan membantu pihak terkait tanpa menggeser fungsinya dilapangan karena nantinya pihak pencatatan plat nomor dapat difokuskan untuk mengatur antrian dan melakukan perawatan sistem.

Penelitian mengenai deteksi plat nomor pada citra telah banyak dilakukan baik di luar maupun dalam negeri. Setiap penelitian tentu memiliki acuan untuk mendapatkan hasil yang lebih baik dari penelitian sebelumnya. Beberapa diantaranya adalah penelitian yang pernah dilakukan *Tran Duc Duan* pada tahun 2004 untuk kasus citra dengan pengambilan gambar secara *close up* pada bagian depan mobil dengan memanfaatkan *Contour Algorithm* untuk deteksi garis. Hasil dari deteksi garis ini dikombinasikan dengan *Hough Transform* untuk mendapatkan garis yang terhubung dan membuat bagian didalamnya tertutup. Kombinasi *Contour Algorithm* dan *Hough Transform* akan menghasilkan kandidat plat nomor yang dicari. Kandidat ini selanjutnya diseleksi dengan menghitung rasio yang mendekati rasio plat nomor acuannya [TRA-04]. Penelitan berikutnya dilakukan oleh *Danian Zheng* dkk pada tahun 2005, dengan memanfaatkan operator vertikal Sobel untuk mendapatkan tepi vertikal dari citra yang telah ditingkatkan kualitas gambarnya menggunakan *intensity variance*. Hasil yang didapatkan kemudian dihapus tepi yang terlalu panjang karena dianggap bagian dari *background* dan menghapus bagian yang terlalu kecil karena dianggap sebagai *noise* [DAN-05]. Penelitian Berikutnya dikembangkan oleh *Vahid Abolghasemi* pada tahun 2009, dengan melakukan perbaikan pada metode *preprosesing*-nya. *Intensity Variance* dinilai tidak efisien pada citra dengan

pencahayaan yang tinggi. Hasilnya diberikan metode preprosesing yaitu *Edge Density* [VAH-09]. Penelitian yang dilakukan oleh *Abolghasemi* ini menerapkan analisis warna dalam melakukan seleksi kandidat plat nomor. Penelitian berikutnya juga dikembangkan oleh *Mahmood Ashori* dengan menggabungkan *preprosesing* dari *Abolghasemi* yaitu *Edge* Density dan deteksi plat yang dilakukan oleh *Zheng* yaitu menghapus tepian background dan *noise* [MAH-11].

Di Indonesia sendiri telah ada beberapa penelitian mengenai deteksi plat nomor. Salah satunya adalah Fajar Astuti Hermawati pada tahun 2010, dengan melakukan proyeksi vertikal untuk nilai periodogram pada setiap baris dari citra. Periodogram dihitung dari total energi hasil *Fast Fourier Transform* (FFT) nilai *luminance* (Y) pada setiap baris citra YUV. FFT bertujuan menemukan kandidat sub-image yang mengandung karakter karena nilai FFT akan berbeda untuk gambar yang mengandung karakter dengan gambar lain [FAJ-10]. Hasil kandidat vertikal ini diberikan perlakuan yang sama untuk menemukan posisi plat dengan horizontal periodogram. Penelitian lainya dilakukan oleh Ikhwan Ruslianto pada tahun 2010, dengan melakukan peningkatan kualitas gambar dengan *contrast stretching* dan melakukan penajaman citra untuk mendapatkan tepi yang lebih jelas. Hasil *preprosesing* ini akan dicari tepi dengan Sobel Operator. Langkah berikutnya melakukan anlisis pada setiap *connected component* yang dianggap sebagai kandidat plat dengan perbandingan panjang dan lebar dari *connected component* tersebut [IKH-12].

Metode dan algoritma yang telah dilakukan untuk mencari posisi plat nomor telah banyak dikembangkan dan tentunya terdapat kelebihan dan kekurangan untuk masing-masing algoritma. *Du*, telah melakukan ulasan mengenai metode - metode dasar deteksi plat nomor hingga tahun 2013. Salah satunya yaitu *edge-based algorithm* yang dinilai sederhana, cepat dan mudah tetapi akan sangat sulit dilakukan pada citra yang terlalu komplek [SHA-13]. Metode pengenalan karakter yang juga menjadi hasil review oleh *Shan du* adalah jaringan saraf tiruan metode backpropagation dengan tingkat akurasi 85.5%. Sehingga nantinya untuk sistem deteksi dan pengenalan plat nomor parkir yang diusulkan akan menggunakan metode dasar yaitu deteksi tepi dan *Backpropagation*.

Penelitian ini nantinya akan dihadapkan pada permasalahan eksekusi waktu sistem dan masalah - masalah lain dilapangan yang bisa saja terjadi dan belum didefinisikan. Nantinya hasil dari penelitian ini akan sangat bermanfaat khususnya bagi pihak pencatat plat nomor kendaraan Universitas Brawijaya dan rasa nyaman dan aman bagi pengendara.

1.2 Rumusan Masalah

Dari latar belakang yang telah yang dikemukakan diatas, maka perumusan permasalahan dalam skripsi ini adalah :

1. Bagaimana mengimplementasikan *Edge-based algorithm* untuk mendapatkan posisi plat nomor pada sistem parkir.
2. Bagaimana implementasi segmentasi karakter dari plat nomor yang didapatkan.
3. Bagaimana mengimplementasikan *Backpropagation Neural Network* dan pengaruh parameter *Backpropagation Neural Network* (*learning rate* dan MSE) untuk mengenali karakter dari karakter plat yang didapatkan.

1.3 Batasan Masalah

Dari permasalahan yang dirumuskan diatas, batasan permasalahan yang digunakan pada skripsi ini adalah :

1. Pengambilan data hanya dilakukan pada siang hari dan tidak pada kondisi hujan.
2. Objek yang digunakan dalam penelitian adalah kendaraan roda empat (mobil).
3. Gambar yang diambil adalah pada saat mobil berhenti untuk mengambil karcis masuk di pintu gerbang.
4. Sistem hanya akan mengambil gambar plat nomor dan melakukan pengenalan karakter.
5. Data diambil pada pintu masuk roda 4 (mobil) Universitas Brawijaya, gerbang Veteran.
6. Pengambilan gambar dilakukan dengan jarak dan dengan kamera sesuai dengan skenario pengambilan gambar di bab perancangan.

1.4 Tujuan

Tujuan dari skripsi ini adalah :

1. Mengimplementasikan *Edge-based algorithm* untuk deteksi plat nomor dan dapat mensegmentasi karakter dari plat.
2. Mengimplementasikan *Backpropagation Neural Network* dan parameter jaringan (*learning rate* dan MSE) untuk mengenali karakter dari plat.
3. Mengevaluasi metode yang digunakan pada sistem deteksi dan pengenalan plat nomor.

1.5 Manfaat

Manfaat yang diperoleh dari pembuatan skripsi ini dapat memberikan kemudahan bagi pihak pencatatan plat nomor di pintu masuk Universitas Brawijaya tanpa menggantikan fungsinya. Pihak pencatatan plat nomor nantinya dapat difokuskan untuk mengatur antrian dan melakukan perawatan sistem. Selain hal tersebut, bagi pengendara yang memasuki kawasan Universitas Brawijaya bisa mendapatkan hak karcis sebagai jaminan keamanan dan memberikan rasa nyaman.

1.6 Sistematika Penulisan

Sistematika penyusunan laporan tugas akhir ini disusun menjadi 6 bab dengan masing – masing bab diuraikan sebagai berikut :

1. BAB I PENDAHULUAN

   Menguraikan mengenai latar belakang masalah yang diangkat, perumusan masalah, batasan masalah, tujuan dari penelitian yang dilakukan dan manfaat yang dapat diambil dari penelitian ini.
2. BAB II KAJIAN PUSTAKA DAN DASAR TEORI

   Menguraikan kajian pustaka dan dasar teori tentang plat nomor dan metode untuk deteksi plat nomor pada citra dan pengenalan karakter plat nomor.
3. BAB III METODOLOGI DAN PERANCANGAN

   Menguraikan mengenai langkah kerja dan metode yang digunakan untuk proses perancangan dalam mendeteksi dan mengenali plat nomor kendaraan.

4. BAB IV IMPLEMENTASI

   Berisi tentang penjelasan implementasi dari sistem, bagaimana user interface sistem dan *source code* untuk mengembangkan sistem.

5. BAB V PENGUJIAN DAN ANALISIS

   Menguraikan mengenai hasil implementasi dan pengujian dari perancangan sistem deteksi dan pengenalan plat nomor kendaraan dan melakukan analisis terhadap hasil implementasi untuk evaluasi.

6. BAB VI PENUTUP

   Menguraikan mengenai kesimpulan yang didapatkan dari hasil perancangan, implementasi dan pengujian sistem deteksi dan pengenalan plat nomor dan membuat saran untuk dasar pengembangan atau penelitian berikutnya.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 Kajian Pustaka

Berdasarkan objek skripsi yang akan dibahas yaitu deteksi dan pengenalan plat nomor kendaraan maka perlu dilakukan pengkajian terhadap pustaka yang ada mengenai penelitian terkait untuk mendukung penelitian pada skripsi ini. Dasar metode yang digunakan pada skripsi ini adalah *edge detection.* Metode yang diambil ini berdasarkan penelitan yang dilakukan oleh *Shan Du* tahun 2013 dari beberapa metode dasar yang telah di-*review* bahwa *edge-based algorithm* merupakan metode sederhana dan cepat untuk deteksi plat nomor [SHA-13]. Kekurangan dari metode ini yaitu sulit untuk melakukan analisis pada citra yang terlalu banyak tepi (komplek). Sedangkan tepi yang tidak diinginkan dapat dihapus atau dikurangi dengan operasi morfologi [SHA-13]. *Shan Du* juga telah me-*review* mengenai metode untuk segmentasi dan pengenalan karakter salah satunya adalah *pixel connectivity* dan *Backpropagation*. *Pixel connectivity* memiliki kelebihan mudah, cepat dan dapat mengatasi efek rotasi pada plat, sedangkan *backpropagation* dari hasil *review* salah satu jurnal oleh *Shan Du* menghasilkan akurasi sebesar 85.5%.

Penelitian *AMR Badr, Mohamed M, Thabet dan Abdelsadek* pada tahun 2011 pada jurnal berjudul "*Automatic Number Plate Recognition System*" memaparkan metode yang digunakan untuk deteksi plat nomor dengan memanfaatkan *vertical edge detection.* Hasil dari *edge detection* kemudian dibinerisasikan dan dilakukan operasi *average filter* untuk memperkuat tepi vertikal dari karakter yang merupakan element dari plat nomor. Hasilnya dilakukan *vertical projection* dan mengambil puncak tertinggi yang dianggap memuat plat nomor [BAD-11].

Penelitian *Md.Mostafa Kamal Sarker, Sook Yoon, Jaehwan Lee* dan *Dong Sun Park* pada tahun 2013, dalam jurnal internasional yang berjudul *Novel Lisense Plate Detection Method Based on Heuristic Energy Map* menjelaskan mengenai metode yang dikerjakan untuk mendeteksi plat nomor dari citra dengan

edge based algorithm dan *Morphological Operation*. *Md.Mostafa Kamal Sarker*, menentukan area pada citra yang dianggap sebagai plat dengan melakukan analisis *histogram projection* pada citra hasil deteksi tepi untuk memperkecil area pencarian plat. Area hasil kemudian dicari tepi vertikal dan dilakukan operasi *box filtering* untuk mendapatkan area plat dan titik pusat dari plat [MOS-13].

Penelitian lain tentang deteksi plat nomor dengan *edge based algorithm* dilakukan oleh *Ronak P Patel, Narendra M Pateli* dan *Keyur Brahmbhatt* (2013). Metode yang diberikan berbeda pada tahap *preprosesing* yaitu melakukan *morphological operation* pada citra *grayscale* dan *substraction operation* hasil *morphological operation* dan citra *grayscale*-nya untuk memperjelas gambar plat dan memperkecil area yang tidak diinginkan dari citra. *Morphological Operation* yang digunakan adalah opening. Tahap prosessing yang diberikan hampir sama secara umum seperti yang dijelaskan oleh *Shan Du* yaitu *Vertical Edge Detection* sebagai *edge based methods* dan *morphological operation* [RON-13]. Ronak Dkk, juga melanjutkan penelitiannya hingga pengenalan karakter dan metode yang digunakan adalah *Backpropagation Neural Network*. Akurasi uji coba pengenalan karakter dengan metode *Backpropagation* ini adalah 84%.

Penelitian lain mengenai deteksi plat nomor berdasarkan pada dateksi tepi dilakukan oleh *Reza Azad* dan *Hamid Reza Shayegh*. Metode yang diberikan untuk tahap *preprosessing* yaitu *denoising* untuk mengurangi *noise* yang mungkin akan mengganggu pada saat deteksi tepi. Metode lain yang ditambahkan pada penelitian ini adalah *filling* kandidat plat nomor. Penelitian yang dilakukan oleh *Reza Azad* dan *Hamid Reza Shayegh* juga melakukan uji terhadap beberapa metode deteksi tepi. Hasilnya Deteksi tepi dengan *sobel* memberikan hasil yang paling besar diantara deteksi tepi lain yaitu 98.66% dari 150 data. Sedangkan untuk penentuan *Threshold* yang paling besar adalah *Adaptive Threshold* [REZ-13].

Berdasarkan kajian pada jurnal di atas, maka penelitian dalam skripsi ini akan mengadaptasi *edge based algorithm* yang dirasa merupakan metode yang masih populer untuk dikembangkan menjadi metode yang diajukan. Secara khusus metode yang diajukan adalah *vertical edge detection*, *Otsu method* sebagai penentu nilai *threshold*, *morphological operation* dan *histogram projection*.

Metode yang masih mengadaptasi pada *edge based detection* ini diharapkan dapat memberikan tingkat akurasi yang baik untuk dapat diterima dengan waktu eksekusi yang cepat.

2.2 Plat Nomor

Plat nomor adalah salah satu syarat kelengkapan kendaraan bermotor di Indonesia. Hal ini telah diatur pada pasal 280 UU 22/2009 mengenai Undang Undang Lalu Lintas dan Angkutan Jalan (UULAJ) yang berbunyi "Setiap orang yang mengemudikan Kendaraan Bermotor di Jalan yang tidak dipasangi Tanda Nomor Kendaraan Bermotor (TNKB) yang ditetapkan oleh Kepolisian Negara Republik Indonesia sebagaimana dimaksud dalam Pasal 68 ayat (1) dipidana dengan pidana kurungan paling lama 2 (dua) bulan atau denda paling banyak Rp500.000,00 (lima ratus ribu rupiah)." Syarat dan ketentuan mengenai plat nomor juga diatur pada pasal 68 ayat (3) yang menyebutkan bahwa plat nomor berisi kode wilayah, nomor registrasi dan masa berlaku. Nomor registrasi terdiri dari angka dan huruf, Angka mewakili no urut dan huruf yang paling depan pada bagian nomor registrasi menunjukan subwilayah selebihnya adalah huruf yang digunakan sebagai pembeda. Gambar 2.1 adalah contoh plat nomor kendaraan.

Gambar 2.1 Contoh Plat Nomor Kendaraan

Ukuran, warna, bentuk, bahan dan cara pemasangan plat nomor kendaraan Indonesia juga telah diatur dalam PP 44/1993 tentang kendaraan dan pengemudi pada pasal 178. Bentuk plat nomor yaitu lempengan persegiempat tipis dengan ukuran 250mm x 105mm untuk sepeda motor dan 395mm x 135mm untuk kendaraan lain. Gambar 2.2 adalah contoh ukuran plat nomor kendaraan.

Gambar 2.2 Contoh Ukuran Plat Nomor Indonesia

Plat nomor dipasang pada dibagian depan dan belakang ditempat yang telah disediakan. Warna plat nomor kendaraan dibagi berdasarkan fungsinya. Kendaraan pribadi dan kendaraan lain bukan kendaraan umum dan sewa warna plat adalah hitam dengan huruf berwarna putih. Kendaraan Umum warna plat adalah kuning dengan tulisan hitam. Kendaraan dinas pemerintahan warna plat adalah merah dengan warna huruf adalah putih. Kendaraan Korps diplomatik Negara asing warna plat adalah putih dengan warna karakter adalah hitam.

2.3 Citra

Citra adalah suatu media untuk mempresentasikan informasi secara visual. Manusia mengambil informasi visual melalui mata dan melakukan pengolahan data visual didalam otak. Pengambilan informasi oleh mata ini membutuhkan cahaya. Informasi mengenai objek didalam citra divisualkan oleh variasi cahaya dan warna cahaya yang tertangkap. Objek secara nyata adalah 3 dimensi tetapi gambaran objek akan terekam dalam bentuk 2 dimensi dengan nilai intensitas cahayanya [NIC-00].

Penjabaran diatas menunjukan bahwa citra merupakan representasi 2 dimensi dengan fungsi intensitas cahaya f(x,y). Secara umum citra digolongkan menjadi 2 macam yaitu citra continyu dan citra diskrit. Citra continyu dihasilkan dari sinyal analog sedangkan citra diskrit adalah hasil diskritisasi dari sinyal analog itu sendiri [MAR-10]. Citra diskrit inilah yang dimaksud sabagai citra digital. Citra digital diilustraikan melalui array 2 dimensi yang menyimpan suatu

nilai yaitu derajat keabuan. Setiap element dalam array tersebut disebut dengan *pixel* atau pel yang merepresentasikan derajat keabuan. Pemetakan koordinat ke dalam citra digital ini biasa disebut sampling dan diskritisasi derajat keabuan biasa disebut kuantisasi. Representasi citra digital dapat diformulasikan pada persamaan (2.1). Diskritisasi nilai keabuan berada pada interval $0 \leq f(x,y) \leq 2^n-1$ dengan n adalah jumlah bit yang digunakan untuk kuantisasi.

$$f(x,y) = \begin{pmatrix} f(1,1) & \cdots & f(1,n) \\ \vdots & \ddots & \vdots \\ f(m,1) & \cdots & f(m,n) \end{pmatrix} \quad \text{.....(2.1)}$$

Cahaya merupakan bagian tampak dari gelombang Elektro Magnetik yang dianggap sebagai warna. Cahaya ditangkap oleh alat optik dan dipecah menjadi 3 warna berdasarkan panjang gelombangnya yaitu merah, hijau dan biru. Citra digital dari warna penyusunnya dibebedakan menjadi 3 yaitu citra berwarna, citra *grayscale* dan citra biner. Citra berwarna memiliki 3 nilai derajat keabuan yang masing - masing mewakili warna dasar diatas. Citra *grayscale* memiliki 1 warna yaitu keabuan sedangkan citra biner hanya memiliki warna hitam atau putih.

2.3.1 Citra *Grayscale*

Citra berwarna terdiri dari 3 komponen warna yaitu merah (R), Hijau (G) dan biru (B) yang masing masing biasanya dikuantisasikan menjadi 8 bit sehingga totalnya adalah 24 bit atau lebih dari 16 juta kombinasi warna. Citra warna disimpan dalam memori membutuhkan alokasi sebesar 3 *byte* untuk setiap *pixel*nya. Citra warna yang berukuran besar akan lebih banyak menghabiskan memori untuk pengolahanya [RAF-02]. Representasi lain adalah citra *grayscale* hanya memiliki 1 komponen warna yaitu keabuan dan membutuhkan 1 *byte* untuk setiap *pixel* dalam memori. Citra *grayscale* sering kali digunakan dalam proses pengolahan citra digital untuk efisiensi kebutuhan memori. *Pixel* pada citra *grayscale* berwarna antara hitam hingga putih. nilai *pixel* citra *grayscale* didapatkan dari konversi citra warna yaitu rata-rata dari masing masing R, G dan B atau dapat dirumuskan pada persamaan 2.2 berikut.

$$f'(x,y) = \frac{R(x,y)+G(x,y)+B(x,y)}{3} \quad \text{..... (2.2)}$$

Di mana $f'(x,y)$ adalah intensitas *grayscale* pada *pixel* (x,y), $R(x,y)$ adalah intensitas warna merah pada *pixel* (x,y), $G(x,y)$ adalah intensitas warna hijau pada *pixel* (x,y), dan $B(x,y)$ adalah intensitas warna biru pada *pixel* (x,y).

2.3.2 Citra Biner

Citra biner adalah citra yang hanya memiliki warna hitam atau putih pada setiap *pixel*nya. Citra biner hanya membutuhkan 1 bit alokasi memori untuk setiap *pixel*. Pada citra biner biasanya warna hitam bernilai 0 untuk menunjukan background sedangkan warna putih bernilai 1 untuk menunjukan objek. Citra biner biasa dimanfaatkan pada pengolahan citra digital untuk proses segmentasi objek, operasi morfologi, *thining* dan lain sebagainya [DAR-10]. Citra biner ditentukan berdasarkan nilai ambang dari citra *grayscale* atau dapat dirumuskan pada persamaan 2.3 berikut.

$$f'(x,y) = \begin{cases} 0, f(x,y) < th \\ 1, f(x,y) \geq th \end{cases} \quad \text{..... (2.3)}$$

Di mana $f'(x,y)$ adalah nilai biner pada *pixel* (x,y) dan th adalah ambang batas untuk menentukan nilai *pixel* (x,y).

2.4 Pengolahan Citra Digital

Citra Digital sangat bermanfaat selain dapat menyimpan visualisasi dari citra sebenarnya, Citra digital juga dapat memuat informasi mengenai objek. Informasi objek dalam citra digital ada yang dapat langsung dikenali oleh mata dan ada juga yang sulit atau bahkan tidak dapat dikenali langsung oleh mata. Maka perlu dilakukan manipulasi atau meningkatkan kualitas citra agar dapat diterima baik oleh mata. Pengolahan citra digital adalah kegiatan manipulasi citra dengan berbagai cara [NIC-00]. Banyak sekali tujuan pengolahan citra digital selain mendapatkan informasi seperti dijelaskan diatas, contohnya mendapatkan penampakan global dari citra, meningkatkan kualitas citra, pemugaran citra, ekstraksi fitur dari citra dan bahkan melakukan kompresi pada citra [MAR-10]. Contoh pengenalan plat nomor dalam penelitian ini melakukan deteksi tepi dari citra dan memotong citra yang menunjukan informasi plat nomor.

Pengolahan citra digital tidak lepas dari manipulasi secara langsung pada *pixel* atau biasa disebut spatial operation [RAF-02], Spatial operation dibedakan

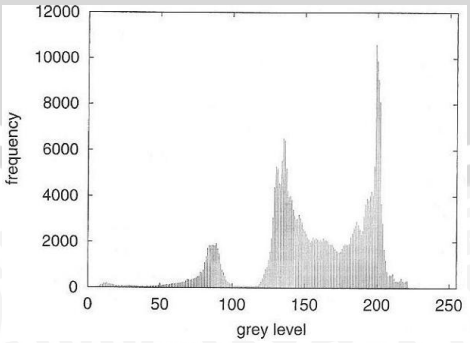
menjadi 5 kategori operasi yaitu operasi titik, operasi global, operasi ketetanggaan, operasi geometri dan operasi pada banyak citra. Operasi titik melakukan manipulasi nilai derajat keabuan pada *pixel* dengan persamaan tertentu. Operasi global melakukan manipulasi pada setiap *pixel* berdasarkan fitur global dari citra. Operasi global biasa dilakukan pada citra dengan memanfaatkan visualisasi global dari citra yaitu histogram. Operasi ketetanggaan melakukan manipulasi pada *pixel* dengan mempertimbangkan derajat keabuan tetangganya. Operasi Geomatrik melakukan manipulasi pada *pixel* dengan mengubah posisi geomatriknya dalam citra. Operasi pada banyak citra melakukan manipulasi pada lebih citra dari satu citra dengan fungsi tertentu untuk mendapatkan hasil keluaran yang diinginkan.

2.4.1 Histogram

Histogram citra adalah deskripsi global dari jumlah *pixel* dengan pada setiap titik derajat keabuan antara 0 hingga 2^n-1 [RAF-02]. Dengan n adalah jumlah bit yang digunakan dalam kuantisasi citra. Representasi histogram ditunjukan pada diagram kartesius dengan nilai x menunjukan derajat keabuan dan y adalah jumlah *pixel*. Histogram citra ditunjukan dengan pada persamaan 2.4 berikut.

$$h(r_k) = n_k \quad \text{..... (2.4)}$$

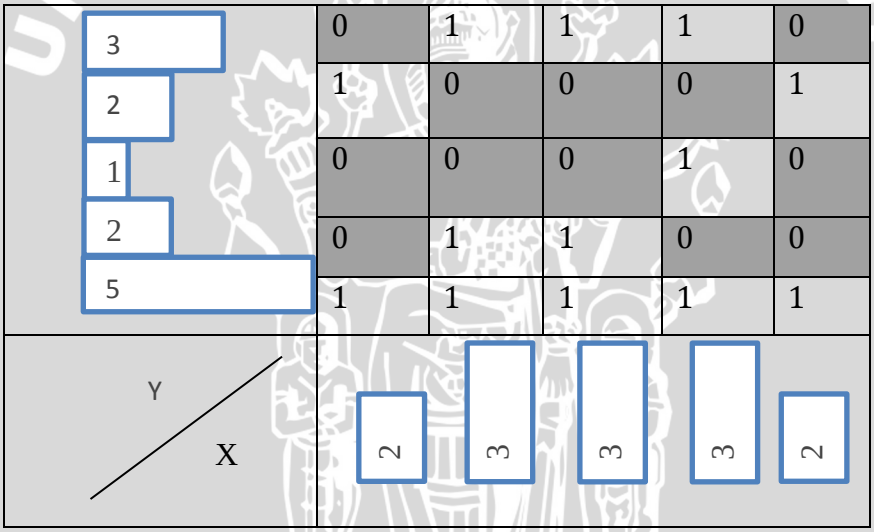
Di mana r_k adalah derajat keabuan ke-k dan n_k adalah jumlah *pixel* dengan nilai derajat keabuan k. Histogram pada suatu citra akan tetap sama meskipun telah dilakukan operasi geomatrik pada citra tersebut. Gambar 2.3 adalah contoh histogram citra.



Gambar 2.3 Histogram Citra

2.4.2 *Histogram Projection*

Pengolahan citra digital memiliki tujuan yang salah satunya adalah mengekstrak fitur dalam citra untuk dilakukan analisis dan proses selanjutnya. Penelitian pengenalan plat nomor ini berbasis pada deteksi tepi. Penentuan plat nomor dalam citra dilakukan dengan menganalisis hasil dari *histogram projection* seperti pada penelitian pengenalan *handwriting* oleh Rodolfo. *Histogram projection* adalah representasi dari jumlah nilai derajat keabuan pada citra biner pada setiap baris atau colom. Menurut Rodolfo, ide tentang histogram projection dinilai sederhana dan cepat untuk kasus pengenalan *handwriting* [ROD-09]. Gambar 2.4 adalah ilustrasi dari *histogram projection*.



Gambar 2.4 *Histogram Projection*

2.4.3 *Edge Detection*

Deteksi tepi adalah menentukan batas antara 2 region dalam citra yang memiliki jarak intensitas yang relatif tinggi. Deteksi tepi biasa digunakan untuk melakukan segmentasi antara objek dengan background. Tepi dapat dibedakan menjadi 3 berdasarkan tingkat perubahanya yaitu tepi curam, tepi landai dan tepi dengan derau yaitu *noise* sehingga tepi tidak begitu menonjol. Ide utama dari deteksi tepi adalah menggunakan operator *edge detection* untuk menemukan posisi tepi lokal, dan menandai berdasarkan jarak intensitasnya [DON-14].

Deteksi tepi pada objek 1 berdimensi dapat diperoleh dari turunan fungsinya. Deteksi tepi pada citra atau objek 2 dimensi secara lokal dapat diperoleh dengan melewatkan citra pada suatu operator atau mask vertikal dan horizontal yang telah dirancang untuk deteksi tepi ini [IDH-08]. Citra dilewatkan pada suatu mask dengan persamaan *sum of products. sum of products* ditunjukan pada persamaan 2.5 berikut.

$$f'(x,y) = \sum_{u=-M}^{M} \sum_{v=-N}^{N} g(M+u, \mathrm{N}+v).f(x+u, y+v) \quad \text{..... (2.5)}$$

Dengan $f'(x,y)$ nilai intensitas pengganti pada *pixel* (x,y), M adalah setengah jumlah colomn pada kernel, N adalah setengah jumlah baris pada kernel, $g(M+u, \mathrm{N}+v)$ adalah *pixel* kernel pada posisi M+u, N+v dan $f(x+u, y+v)$ adalah *pixel* citra pada posisi x+u,y+v. Sehingga contoh mask dengan M,N dapat ditunjukkan pada Gambar 2.5 berikut.

$g(u-1, v-1)$	$g(u, v-1)$	$g(u+1, v-1)$
$g(u-1, v)$	$g(u, v)$	$g(u+1, v)$
$g(u-1, v+1)$	$g(u, v+1)$	$g(u+1, v+1)$

Gambar 2.5 Contoh Indek Kernel

Terdapat beberapa mask yang biasa digunakan sebagai operator deteksi tepi sebagai berikut [RAF-02]:

1. Roberts

1	0
0	1

0	1
1	0

2. Prewit

1	1	1
0	0	0
1	1	1

1	0	1
1	0	1
1	0	1

3. Sobel

1	2	1
0	0	0
1	2	1

1	0	1
2	0	2
1	0	1

Hasil dari kedua mask yang dilewatkan pada suatu sampel citra dapat dikombinasikan untuk mengisi pada titik f'(x,y) dengan salah satu persamaan berikut [IDH-08]:

1. Penjumlahan

$$f'(x,y) = |f'1(x,y)| + |f'2(x,y)| \quad \text{..... (2.6)}$$

2. Maksimum

$$f'(x,y) = \max(|f'1(x,y)|, |f'2(x,y)|) \quad \text{..... (2.7)}$$

3. Rata – rata

$$f'(x,y) = \frac{|f'1(x,y)| + |f'2(x,y)|}{2} \quad \text{..... (2.8)}$$

4. Rata – rata Geometrik

$$f'(x,y) = \sqrt{f'_1(x,y) * f'_1(x,y) + f'_2(x,y) * f'_2(x,y)} \quad \text{..... (2.9)}$$

Di mana $f'_1(x,y)$ adalah hasil *sum of products pixel* pada citra (x,y) dengan *pixel* (x,y) pada kernel-1, $f'_2(x,y)$ adalah hasil *sum of products pixel* pada citra (x,y) dengan *pixel* (x,y) pada kernel-2. Dan $f'(x,y)$ adalah hasil kombinasi untuk mengisi *pixel* (x,y) pada citra hasil.

Edge Detection adalah dasar pengolahan citra yang biasa digunakan untuk ekstraksi fitur. Hasil *Edge Detection* secara umum adalah garis batas yang terhubung. Penelitian deteksi plat nomor ini menggunakan deteksi tepi sebagai dasar metode untuk mengambil fitur dengan data yang diambil secara *close up* untuk mengurangi tepi *background* yang bukan termasuk fitur. *Mask edge detection* yang digunakan pada penilitian ini yaitu operator sobel dengan kombinasi operator rata-rata Geometrik.

2.4.4 *Thresholding* (metode *Otsu*)

Thresholding adalah salah satu cara untuk mengubah citra menjadi citra biner dengan nilai ambang tertentu [SRI-10]. Tresholding merupakan cara sederhana untuk mensegmentasi objek dalam citra. Hasil dari *edge detection* berupa citra *grayscale* memiliki tingkat intensitas masing masing *pixel* yang berbeda. Hal ini akan mempersulit proses pengolahan citra digital jika tidak ditentukan dalam batas berapa intensitas dianggap sebagai tepi. *Thresholding* melakukan penegasan tepi sehingga tepi dapat terlihat jelas.

Penentuan batas *threshold* biasanya diambil dari fitur fitur yang diperoleh dari citra atau bisa saja berupa konstanta. Berdasarkan fitur yang diambil thresholding dibagi menjadi 2 macam yaitu *Thresholding* global dan lokal. Thresholding global menentukan nilai threshold berdasarkan representasi global dari citra yaitu histogram. Thresholding global juga bisa ditentukan dengan mencoba-coba titik ambang dengan target adalah hasil yang terbaik. Thresholding lokal menentukan nilai ambang secara subjektif dengan membagi citra terlebih dahulu dan menentukan nilai threshold untuk citra tersebut. *Thresholding* lokal biasanya lebih baik untuk citra dengan kontras yang tidak merata.

Metode *thresholding* telah banyak dikembangkan dengan berbagai kelebihannya masing masing. Salah satu metode yang digunakan dalam penilitian ini adalah metode *Otsu*. Metode *Otsu* menentukan nilai ambang global yang dapat membedakan antar objek dengan background dalam citra berdasarkan histogram. Kelebihan dari meode *Otsu* adalah penentuan batas nilai threshold pada suatu citra secara objektif sehingga threshold untuk masing masing citra akan memungkinkan berbeda. Jadi, metode *Otsu* ini akan lebih baik dari pada nilai konstanta yang biasa dibuat untuk menentukan nilai threshold dalam citra.

Pencarian nilai threshold dengan metode *Otsu* secara konsep yaitu membagi citra menjadi 2 kelompok di mana nilai ambang tersebut memaksimalkan varians antar kelompok dan meminimalkan varians dalam kelompok. Sebelum menentukan varians maksimal antar 2 kelompok histogram dinormalisasi dengan persamaan probabilitas seperti pada persamaan 2.10 berikut.

$$P_i = \frac{n_i}{N} \quad \text{..... (2.10)}$$

Di mana, Pi adalah probabilitas yang mewakili nilai histogram ke-i (n_i) dan N adalah jumlah keseluruhan *pixel* dalam citra tersebut. Penentuan varians maksimal didapatkan pada persamaan 2.11 berikut.

$$\sigma_B^2(k^*) = \max\left(\frac{[\mu_T.\omega(k)-\mu_k]^2}{\omega(k)[1-\omega(k)]}\right) \quad \text{..... (2.11)}$$

Di mana, masing – masing persamaan diatas dideskripsikan pada persamaan statistik berikut :

$$\omega(k) = \sum_{i=1}^{k} P_i \quad \text{..... (2.12)}$$

$$\mu(k) = \sum_{i=1}^{k} i.P_i \quad \text{..... (2.13)}$$

$$\mu_T = \sum_{i=1}^{L} i.P_i \quad \text{..... (2.14)}$$

Dengan $\omega(k)$ adalah nilai momen kumulatif ke-0, $\mu(k)$ adalah momen kumulatif ke-1, μ_T adalah rata-rata. Penelitian deteksi dan pengenalan plat nomor ini menggunakan salah satu metode thresholding *Otsu* ini dengan harapan tepi dari citra yang diambil dapat dimaksimalkan.

2.4.5 *Morphologi Operation*

Morphologi Operation atau operasi morfologi pada citra adalah tindakan manipulasi bentuk objek dalam citra [DAR-10]. Pengolahan citra yang mendasar pada operasi bentuk telah banyak dikembangkan, beberapa diantaranya adalah Erosi, Dilasi, *Opening*, *Closing*, *Hit and Miss*, *Thining*, *dan Thickness*. Penelitian deteksi plat nomor pada citra memerlukan proses *Opening* dengan tujuan menghapus tepi citra yang dirasa akan menggagu analisis *histogram projection* setelahnya. Penelitian ini juga akan memerlukan proses *Closing* untuk menegaskan tepi citra yang termasuk plat nomor khususnya karakter plat.

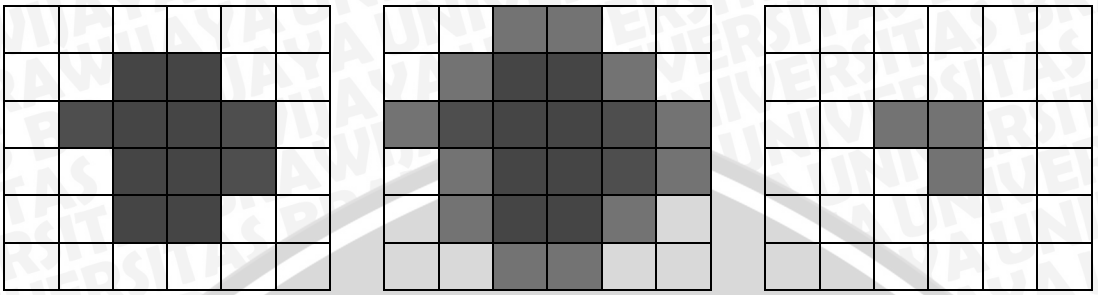
Morphologi Operation memerlukan suatu mask sebagai acuan untuk penentuan nilai *pixel* yang akan dimanipulasi atau biasa disebut *Structuring Elements* [NIC-00]. *Strukturing Elements* (SE) terdiri dari nilai 0,1 dan kosong atau tidak diperdulikan. Cara kerja SE adalah membandingkan secara berurutan pada setiap *pixel* input citra seperti template pembanding, sedemikian sehingga pusat *pixel* yang akan dirubah terletak pada pusat SE. Operasi dasar Morfologi adalah *erotion* dan *dilation* yang menjadi dasar operasi morfologi lain. Gambar 2.6 adalah beberapa contoh *Structuring Element*.

1	0	1
0	0	0
1	0	1

Gambar 2.6 Beberapa Contoh *Structuring Element*

Operasi dilasi memberikan efek perluasan area objek. Syarat operasi dilasi pada citra biner adalah jika terdapat salah satu nilai *pixel* yang sama dengan SE maka *pixel* pusatnya akan bernilai 1 atau sebagai objek. Operasi Erosi pada citra biner memberikan efek memperluas background sehingga objek akan mengecil atau bahkan hilang. Syarat dari operasi erosi ini yaitu jika semua nilai *pixel* yang

dibandingkan dengan SE sama maka *pixel* pusatnya akan diberi nilai 1 atau dianggap sebagai objek. Gambar 2.7 adalah contoh dari operasi erosi dan dilasi.



Gambar 2.7 Adalah Contoh Dilasi dan Erosi

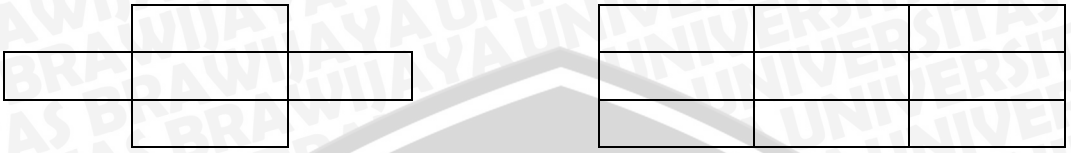
Operasi lain yang dikembangkan dari kombinasi operasi erosi dan dilasi adalah opening dan closing. Operasi *Opening* mengkombinasikan operasi erosi dan dilasi secara berurutan. Efek operasi erosi adalah memperkecil atau menghapus objek. Hasil erosi dilajutkan dengan operasi dilasi sehingga object yang kecil karena erosi diharapkan kembali kecuali objek yang hilang karena erosi. Operasi C*losing* adalah kebalikan dari operasi opening yaitu mengkombinasikan operasi dilasi dan operasi erosi secara berurutan. Operasi closing ini akan menutup lubang dalam objek dengan mengembalikan bentuk objek seperti semula.

2.4.6 *Connected Component*

Connected Component adalah dua atau lebih *pixel* pada citra biner atau *grayscale* merupakan komponen terhubung menurut aturan keterhubungan (4, 8 atau *m-connectivity*) [DAR-10]. *Pixel* terhubung dapat membentuk suatu wilayah di mana setiap *pixel* pembentuknya memiliki nilai intensitas yang sama atau nilai intensitas dideskripsikan pada suatu himpunan tertentu. *Connected Component* mewakili suatu object dalam citra, biasa digunakan untuk melakukan segmentasi object dalam citra.

Penandaan *Connected Component* untuk aturan 4-*Connectivity* adalah mengecek kesamaan intensitas pada *pixel* yang berada di atas, bawah, samping kiri dan samping kanan, jika sama maka diberikan tanda sama dengan *pixel* yang diperiksa sebelumnya. *Pixel* yang diberi tanda sama tadi kemudian di periksa kembali terhadap 4 *pixel* tetangganya hingga tidak terdapat nilai yang sama pada keempat *pixel* tetangganya. 4-*connectivity* hampir sama dengan 8-*connectivity*

hanya pemeriksaan berlanjut pada diagonal atas dan diagonal bawah *pixel* yang diperiksa tetangganya. Gambar 2.8 berikut adalah ilustrasi 4-*connectivity* dan 8-*connectivity*.



4-*Connectivity* 8-*Connectivity*

Gambar 2.0.8 Ilustrasi Keterhungungan 4-*Connectivity* dan 8-*Connectivity*

2.4.7 Ekstraksi Fitur

Ekstraksi fitur digunakan untuk mengambil informasi dari citra untuk dijadikan fitur dalam pengenalan karakter. Fitur sendiri adalah karakteristik unik dari suatu objek [DAR-10]. Fitur sebisa mungkin memiliki jumlah sedikit tetapi dapat membedakan suatu objek dengan objek yang lainnya. Fitur yang digali dari citra adalah -1 atau 1 dengan jumlah fitur telah ditentukan sebelumnya. Nilai setiap fitur ditentukan dari nilai terbanyak dari suatu region dalam citra yang mewakili nilai suatu fitur tersebut. Hal ini mirip dengan metode interpolasi untuk memperkecil skala citra. Perbedaannya, ekstraksi dilakukan pada citra biner sehingga nilainya bukan merupakan rata-rata dari region tersebut tetapi nilai modus yang dapat digunakan untuk mewakili fitur tersebut. Gambar 2.9 merupakan ilustrasi untuk ekstraksi fitur.

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

→

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

Gambar 2.9 Ilustrasi Ekstraksi Fitur Menjadi 25 Fitur

2.5 Jaringan Saraf Tiruan

Jaringan saraf tiruan adalah metode pemrosesan informasi yang mengadaptasi dari jaringan saraf biologis. Menurut Laurene, Jaringan Saraf

Tiruan dicirikan dari arsitektur atau pola hubungan antar *node* atau dalam jaringan disebut neuron, metode penentuan dan pembaharuan bobot, dan fungsi aktivasi [LAU-95]. Hal ini diadaptasi dari fungsi kerja neuron dalam jaringan saraf. Bagian neuron yang ditirukan adalah dendrit, soma dan akson.

Proses kerja jaringan dimulai dari Dendrit sebagai penerima sinyal dari neuron lain melalui *Synaptic Gap* melalui proses kimia. Proses kimia ini pada jaringan saraf tiruan diwakili sebagai bobot dan sinyal masuk sebagai nilai input. Sinyal yang diterima diteruskan ke badan sel atau soma untuk menjumlahkan semua informasi yang masuk [ANA-10]. Penjumlahan yang dimaksud pada jaringan saraf tiruan ditunjukan pada persamaan 2.15 berikut.

$$y_{in} = x_1 w_1 + x_2 w_2 + \cdots + x_n w_n \quad (2.15)$$

Di mana y_{in} adalah jumlah sinyal masuk, x_i adalah sinyal dari neuron ke-i dan w_i adalah bobot untuk sinyal ke-i. Akson melakukan aktivasi dan melanjutkan sinyal ke neuron lain. Pada jaringan saraf tiruan sinyal yang akan dikeluarkan diaktivasi dengan fungsi aktivasi seperti pada persamaan 2.16 sebagai hasil dari *node*.

$$y_{out} = f(y_{in}) \quad (2.16)$$

Secara mendasar jaringan saraf tiruan meliputi penjumlahan sinyal input dan aktivasi untuk sinyal output. Terdapat banyak fungsi aktivasi yang dapat digunakan untuk diterapkan pada suatu arsitektur sistem jaringan saraf tiruan salah satunya adalah fungsi sigmoid biner seperti ditunjukan pada persamaan 2.17 [RUD-06].

$$f(y_{in}) = \frac{1}{1+\exp(-y_{in})} \quad (2.17)$$

Di mana y_{in} adalah jumlah sinyal masuk pada *node* dan $f(y_{in})$ adalah sinyal output. Turunan dari fungsi sigmoid biner ditunjukkan pada persamaan 2.18 [RUD-06].

$$f'(x) = f(x)[1 - f(x)] \quad (2.18)$$

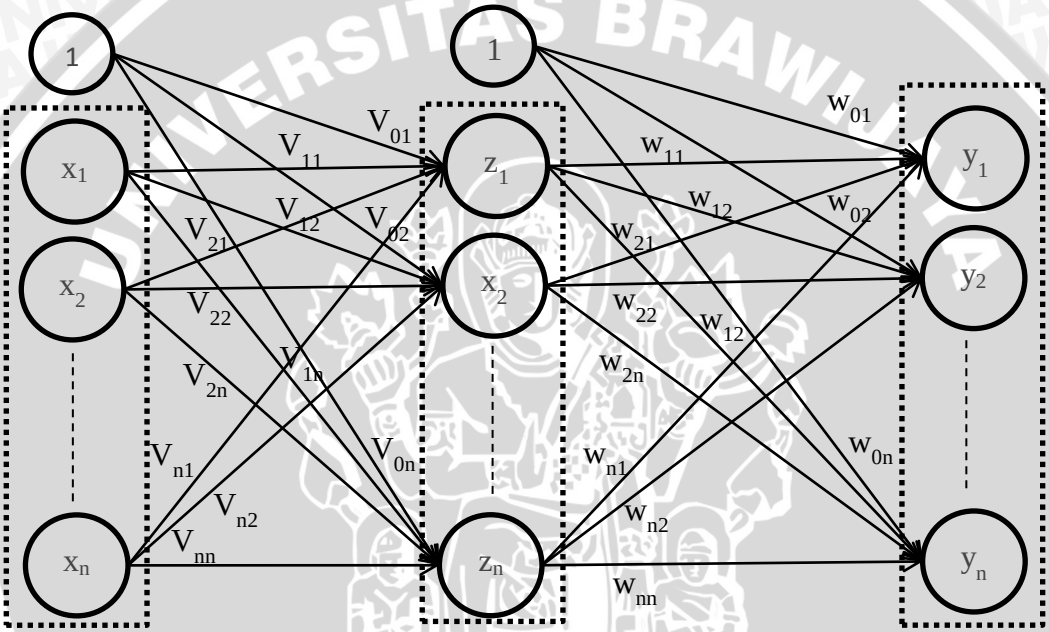
Di mana $f'(x)$ adalah turunan fungsi sigmoid biner dan $f(x)$ adalah fungsi aktivasi sigmoid biner.

Menurut Laurene jaringan saraf tiruan dicirikan dari arsitektur dan metode pembaharuan bobot. Pengenalan karakter pada penelitian ini akan diarahkan

menggunakan *Backpropagation Neural Network* (BPNN) dengan harapan meminimalkan kesalahan hasil dari pembelajaran *Neural Network.*

2.5.1 *Backpropagation Neural Network*

Secara Umum ada tiga tahap dasar metode *backpropagation* yaitu perambatan maju untuk melakukan pembelajaran dengan membawa sinyal input ke layer berikutnya, perhitungan kesalahan secara bertahap mundur, penyesuaian dan pemasangan bobot [LAU-95]. Arsitektur *Backpropagation Neural Network* secara umum dapat dilihat pada Gambar 2.10.



Gambar 2.10 Arsitektur Backpropagation Neural Network

Gambar 2.10 adalah arsitekur umum dari jaringan *Backpropagation* terdiri atas tiga layer utama. Layer pertama adalah *input*, layer tengah adalah *hidden layer* (layer tersembunyi) dan yang terakhir atau paling kanan adalah *output* [SYA-10]. Pada bagian layer input dan *hidden layer* terdapat *node* bernilai 1 sebagai bias dengan bobot masing masing. Perambatan maju seperti yang dijelaskan diatas adalah membawa pola input dari layer pertama menuju layer hasil dengan metode jaringan saraf secara umumnya. Hasil pada layer output kemudian dibandingkan dengan target jika nilai kecocokannya dapat diterima maka pembelajaran selesai tetapi jika nilai kecocokannya tidak dapat diterima maka langkah berikutnya adalah mengevaluasi bobot dengan aturan tertentu. Evaluasi bobot dilakukan dengan perambatan mundur dari layer hasil ke layer

sebelumnya. Hasil evaluasi adalah nilai bobot baru untuk diterapkan pada perambatan maju berikutnya. Proses ini dilakukan hingga mendapatkan bobot yang sesuai untuk nilai kemiripan output dengan target hingga dapat diterima [RIA-01].

Pengenalan pola dengan backpropagation membutuhkan 2 tahap yaitu training untuk mendapatkan bobot untuk hasil output yang mirip dengan target dan proses implementasi bobot untuk pengenalan input testing. Beberapa notasi yang perlu diketahui sebelum menuju ke proses training adalah sebagai berikut:

1. 'x' adalah vektor dari fitur/input data training.
2. 't' adalah vektor dari target data training.
3. 'δ_k' adalah *error correction* untuk bobot W_{jk}.
4. 'δ_j' adalah *error correction* untuk bobot V_{ij}.
5. 'α' adalah *learning rate*
6. 'X_i' adalah input unit ke-i.
7. 'Z_j' adalah Hidden unit ke-j
8. 'Y_k' adalah output unit ke-k
9. 'V_{0j}' adalah bias pada hidden unit ke-j
10. 'W_{0k}' adalah bias pada output unit ke-k

Alur prosedur training dapat dijelaskan sebagai berikut :

1) Inisialisasi awal

   Inisialisasi bobot, *learning rate*, maksimum *epoch* dan *Maksimum Square Error* (MSE).

2) Kerjakan langkah-langkah berikut selama *epoch* kurang dari maksimum *epoch* atau *MSE* lebih besar dari Error.

   a. untuk setiap pasang *node* pada jaringan lakukan langkah seperti berikut :

      Fase I : Perambatan Maju

      1. Setiap *node* input mengirim input ke layer berikutnya yaitu *hidden layer*.
      2. Setiap *node* pada hidden layer menjumlahkan sinyal input terbobot yang masuk sesuai persamaan berikut :

$$z_{in_j} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij} \quad \text{..... (2.19)}$$

      Gunakan fungsi aktivasi untuk menghitung output.

$$z_{out_j} = f\left(z_{in_j}\right) \quad (2.20)$$

Kemudian lanjutkan ke setiap *node* di layer output.

3. Setiap *node* pada output layer menjumlahkan sinyal input terbobot yang masuk sesuai persamaan berikut :

$$y_{in_k} = w_{0k} + \sum_{j=1}^{m} z_j w_{jk} \quad (2.21)$$

Gunakan fungsi aktivasi untuk menghitung output

$$y_{out_k} = f\left(y_{in_k}\right) \quad (2.22)$$

Fase II : Perambatan Mundur

4. Hitung nilai error berdasarkan nilai target dan nilai output

$$\delta_k = (t_k - y_k) f'(y_{in_k}) \quad (2.23)$$

$$\varphi_{jk} = z_j \delta_k \quad (2.24)$$

Hitung koreksi nilai bobot setiap *node* dan bias pada *hidden layer*

$$\Delta w_{jk} = a.\varphi_{jk} \quad (2.25)$$

$$\Delta w_{0k} = a.\delta_k \quad (2.26)$$

5. Setiap *node* dalam *hidden layer* menjumlahkan *error* yang masuk(yaitu dari *node* yang berasal dari layer diatasnya atau output layer)

$$\delta_{in_j} = \sum_{k=1}^{m} \delta_k \, w_{jk} \quad (2.27)$$

Tentukan nilai error dengan mengalikan delta input dan turunan aktivasinya

$$\delta_j = \delta_{in_j} f'(z_{in_j}) \quad (2.28)$$

$$\varphi_{ij} = x_i \delta_j \quad (2.29)$$

Hitung koreksi nilai bobot setiap *node* dan bias pada layer input

$$\Delta v_{ij} = a.\varphi_{ij} \quad (2.30)$$

$$\Delta v_{0j} = a.\delta_j \quad (2.31)$$

6. Setiap *node* output memperbaiki bobot dari sinyal yang masuk termasuk bobot bias

$$w_{jk}(baru) = w_{jk}(lama) + \Delta w_{jk} \quad (2.32)$$

$$w_{0k}(baru) = w_{0k}(lama) + \Delta w_{0k} \quad (2.33)$$

Begitu juga *node* pada hidden layer memperbaiki bobot dari sinyal yang masuk termasuk bobot biasnya

$$v_{ij}(baru) = v_{ij}(lama) + \Delta v_{ij} \quad \text{..... (2.34)}$$

$$v_{0j}(baru) = v_{0j}(lama) + \Delta v_{0j} \quad \text{..... (2.35)}$$

b. Hitung MSE

$$MSE = \frac{\sum(T_k - Y_k)^2}{m} \quad \text{..... (2.36)}$$

Tahap awal dari backpropagation adalah menentukan arsitektur jaringan mulai pada jumlah *node* pada *input layer*, jumlah *hidden layer* dan masing – masing *node* pada *hidden layer* dan jumlah node pada *output layer*. Menurut penelitian *Jeff Heaton*, untuk menemukan banyak *node* pada hidden layer harus memenuhi salah satu aturan berikut [HEA-08]:

1. Jumlah *node* pada *hidden layer* antara jumlah *node* pada layer input dan jumlah *node* pada layer output.
2. Jumlah *node* pada *hidden layer* harus 2/3 dari banyaknya *node* pada layer *input*, ditambah banyaknya *node* pada layer output.
3. Jumlah *node* pada *hidden layer* harus kurang dari dua kali banyaknya *node* pada layer *input*.

Masalah yang muncul apabila jumlah *node* pada *hidden layer* terlalu sedikit akan kurang tepat, maksudnya tidak cukup *node* pada *hidden layer* untuk menyampaikan signal yang sangat rumit. Sedangkan jika *node* pada *hidden layer* berlebihan akan membuat komputasi yang rumit dan terlalu banyak *source* yang digunakan. Masalah lain pada metode jaringan saraf tiruan selain pada arstitekturnya adalah mendefinisikan variabel yang digunakan untuk pelatihan dengan tujuan pelatihan lebih cepat dan efisien. Biasanya dilakukan dengan cara uji coba untuk mendapatkan beberapa variabel yang dibutuhkan. Bobot pada jaringan biasanya gunakan angka kecil random yaitu antara -0.5 hingga 0.5 karena pelatihan dimulai dari bobot kecil hingga nanti disesuaikan dengan pola yang ada.

Proses implementasi bobot untuk pengenalan pola dilakukan dengan merambatkan pola data uji pada jaringan dengan bobot hasil pelatihan dengan cara yang sama dengan Fase I yaitu perambatan maju. Hasil node pada *output layer* nantinya akan dicari kemiripan terhadap hasil node pada *output layer* data latihnya. Penilaian kemiripan menggunakan persamaan jarak *euclidean* untuk

dicari jarak terkecil pada karakter tertentu yang menunjukan karakter data uji nya. Jarak *euclidean* ditunjukan pada persamaan 2.37.

$$d(x,y) = \sqrt{\sum(x_i - y_i)^2} \quad \text{..... (2.37)}$$

Dimana d(x,y) adalah jarak x terhadap y, x_i adalah nilai atribut x ke-i dan y_i adalah nilai atribut y ke-i.

2.6 Evaluasi

Evaluasi bertujuan untuk mengukur keberhasilan sistem deteksi dan pengenalan plat nomor. Berdasarkan kebutuhan ada dua konsep evaluasi yang digunakan pada penelitian ini yaitu akurasi dan *f-measure*. Akurasi adalah perbandingan data benar/sesuai dengan target (n) dengan seluruh data yang ada (N). Persamaan akurasi dapat ditunjukan pada persamaan 2.38.

$$\text{Akurasi} = \frac{\sum n}{\sum N} x\ 100\% \quad \text{..... (2.38)}$$

F-measure adalah rata - rata harmonik dari presisi dan *recall*. *F-measure* digunakan untuk mengukur keberhasilan sistem saat mensegmentasi karakter pada plat. Presisi adalah jumlah karakter tersegmentasi (k) dibandingkan jumlah karakter yang terlihat pada plat (K) sedangkan *recall* adalah jumlah karakter tersegmentasi (k) dibandingkan dengan seluruh jumlah objek yang tersegmentasi (O). Persamaan presisi, *recall* dan *f-measure* dapat ditunjukan pada persamaan 2.39, 2.40 dan 2.41 dibawah ini.

$$\text{Presisi} = \frac{\sum k}{\sum K} \quad \text{..... (2.39)}$$

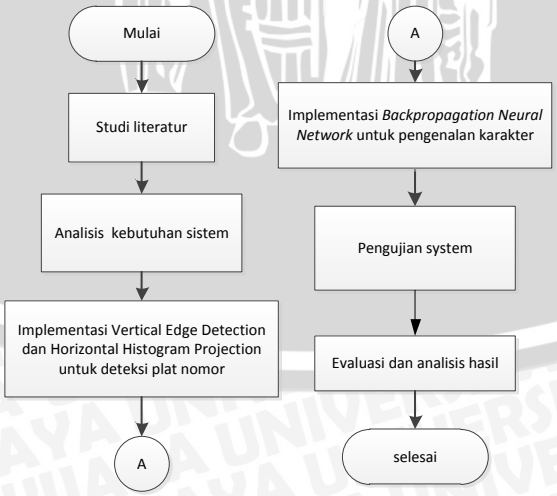
$$recall = \frac{\sum k}{\sum O} \quad \text{..... (2.40)}$$

$$f - measure = \frac{2.presisi.recal}{presisi+recal} \quad \text{..... (2.41)}$$

BAB III
METODOLOGI DAN PERANCANGAN

Bab ini akan membahas mengenai langkah - langkah yang dilakukan untuk penelitian deteksi dan pengenalan plat nomor. Gambar 3.1 adalah diagram alir metodologi penelitian yang di gunakan dan langkah - langkah pembuatan sistem deteksi dan pengenalan plat nomor dapat dijelaskan sebagai berikut :

1. Analisis permasalahan yang melatar belakangi penelitian sistem deteksi dan pengenalan plat nomor
2. Mengabil foto mobil dari pintu masuk kampus *Universitas Brawijaya* gerbang veteran sebagai data dengan jumlah kurang lebih 50 foto yang akan dideteksi dan dikenali dan menyiapkan data latih karakter untuk pengenalan karakter.
3. Melakukan analisis data gambar dan merancang sistem deteksi dan pengenalan plat nomor menggunakan *vertical edge detection* dan *backpropagation neural network.*
4. Implementasi hasil analisis dan perancangan sistem.
5. Melakukan pengujian sistem.
6. Melakukan evaluasi terhadap tingkat keberhasilan sistem dan melakukan analisis terhadap hasil pengujian.

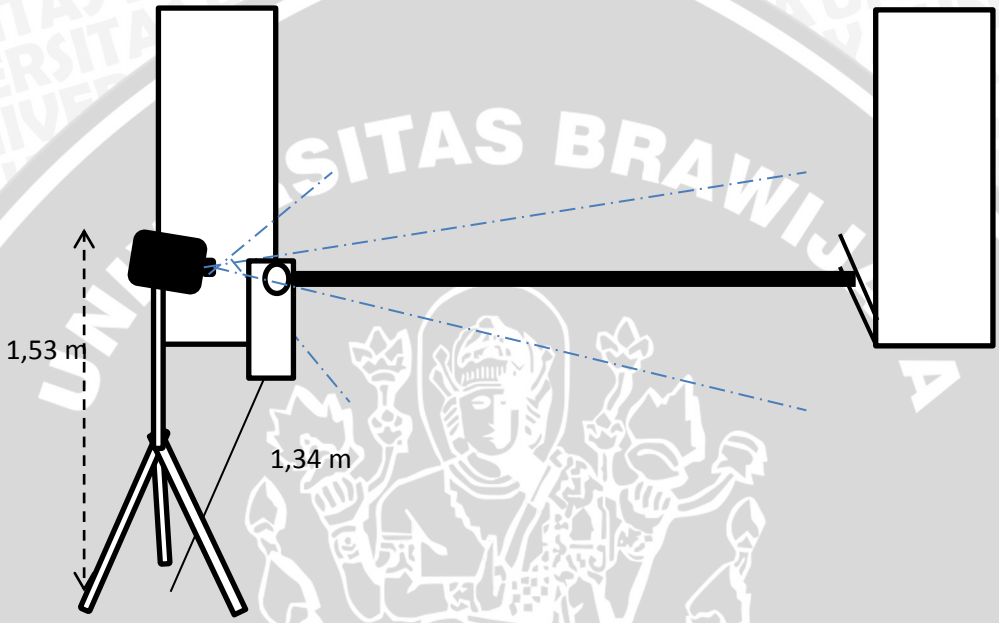


Gambar 3.1 Metode Peneletian

Sumber : Perancangan

3.1 Pengambilan Data

Data yang digunakan dalam skripsi ini didapatkan langsung dari pengambilan gambar di pintu gerbang veteran Univeritas Brawijaya. Gambar yang diambil dari posisi depan mobil dengan posisi kamera tetap. Kamera yang digunakan 14 MP dengan 18x *optical zoom*. Gambar 3.2 adalah posisi pengambilan foto mobil dan posisi kamera.



Gambar 3.2 Pengambilan Foto Mobil Dari Pintu Gerbang Veteran *Universitas Brawijaya*

Sumber : Perancangan

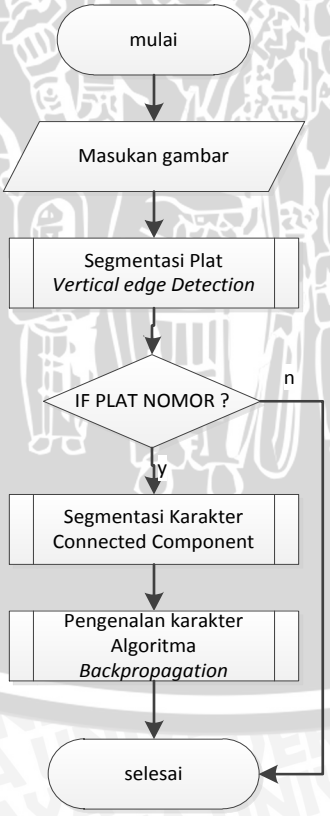
Gambar 3.2 adalah pengambilan data foto mobil di gerbang Veteran Universitas Brawijaya. Kamera berada pada jarak 1.3 meter dari portal masuk.dan Tinggi penyangga kamera adalah 1.53 M dengan kamera mengarah ke pintu masuk. Perbesaran yang dipakai oleh kamera untuk mendapatkan bagian depan mobil diambil secara *close up*. Spesifikasi foto yang diambil adalah ukuran 640x480 *pixel* dengan format warna RGB dan ekstensi file *.jpg.

Data tersebut diambil pada posisi siang hari untuk mengurangi efek dari cahaya lampu mobil jika dalam kondisi menyala. Data juga diambil pada kondisi cuaca cerah (tidak dalam kondisi hujan).

3.2 Perancangan Sistem

Berdasarkan analisis yang telah dilakukan, maka sub-bab ini akan menjelaskan mengenai rancangan proses sistem bagaimana nantinya sistem bekerja untuk mendapatkan plat nomor dari citra dan mengenali karakter. Masukan sistem untuk tahap deteksi plat nomor yang akan dibangun berupa sebuah file gambar atau 1 folder yang berisi file gambar. Proses awal dari sistem adalah menyiapkan data latih untuk pengenalan karakter dilanjutkan proses segmentasi plat nomor dan karakter.

Sebelumnya pengenalan karakter dilakukan, terlebih dulu dilakukan proses pelatihan untuk mengenali pola dari karakter. Masukan pola karakter berupa file gambar. Hasil dari proses pelatihan akan disimpan dalam file *excel* yaitu berupa pola masing - masing data latih dan bobot yang digunakan untuk proses pengujian. Gambar 3.3 menjelaskan mengenai gambaran umum alur proses sistem.



Gambar 3.3 Flowchart Secara Umum Sistem

Sumber : Perancangan

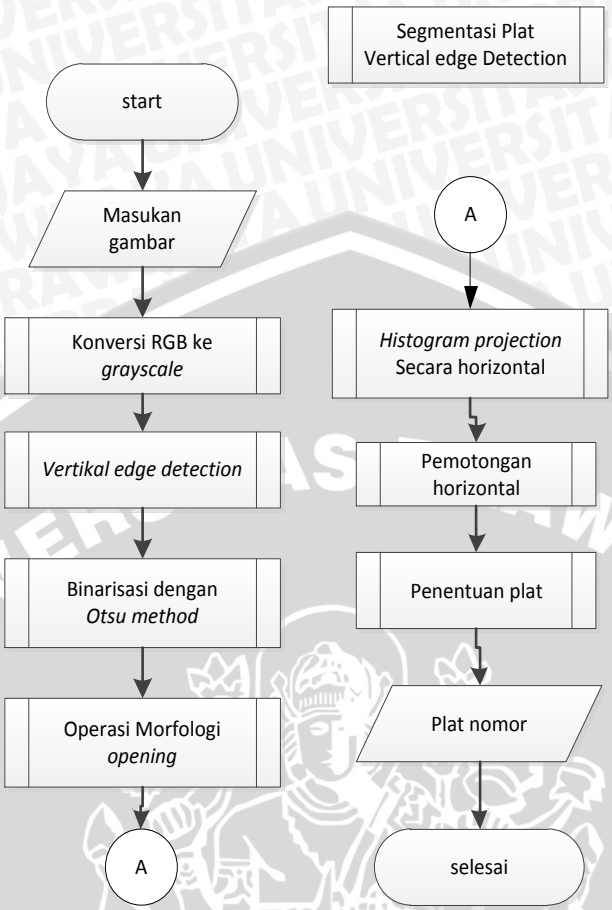
Diagram alir pada Gambar 3.3 merupakan gambaran umum proses sistem seperti yang dijelaskan sebelumnya, Setelah tahap training karakter, proses dilanjutkan pada tahap segmentasi plat yang mana berbasis pada deteksi tepi vertikal dan *histogram projection*. Potongan plat dari citra hasil segmentasi kemudian dilanjutkan pada segmentasi karakter menggunakan *connected component.* Setiap karakter yang didapatkan akan dikenali dengan *backpropagation* setelah data latih dilatihkan dengan mengimplementasikan bobot hasil pelatihan.

Detail dari masing-masing prosedur dari Gambar 3.2 akan dijelaskan lebih detail pada sub bab ini mulai dari proses segmentasi plat hingga pengenalan plat nomor.

3.2.1 Deteksi Plat Nomor

Proses deteksi plat nomor adalah proses utama sistem untuk mengambil plat nomor dan dibawa pada tahap selanjutnya. Tahap pertama pada deteksi plat nomor berbasis pada deteksi tepi vertikal pada citra adalah menkonversi citra input RGB ke *grayscale* dan melakukan deteksi tepi vertikal. Hasil deteksi tepi kemudian dipertegas untuk mendapatkan *pixel* yang benar-benar dianggap sebagai tepi dengan nilai *thershold* sesuai metode *Otsu*. Proses *Opening* dilakukan setelahnya untuk menghilangkan *noise* berupa titik kecil hasil deteksi tepi. Hasil tepi vertikal kemudian dijumlahkan tiap baris untuk dilakukan analisis dan pemotongan secara horizontal bagian dari citra yang mengandung plat nomor. Tahap terakhir adalah tahap untuk mendapatkan posisi plat yang kemudian akan diambil karakternya untuk dikenali.

Plat nomor dicirikan dengan bentuk persegi dengan karakter yang mana penyusunnya akan membuat banyak tepi vertikal yang terbentuk. Langkah selanjutnya adalah menentukan posisi plat nomor. Secara umum dengan memanfaatkan ciri tersebut posisi plat nomor dapat didapatkan dari analisis *histogram projection* secara horizontal. Gambar 3.4 berikut menjelaskan alur deteksi plat nomor.



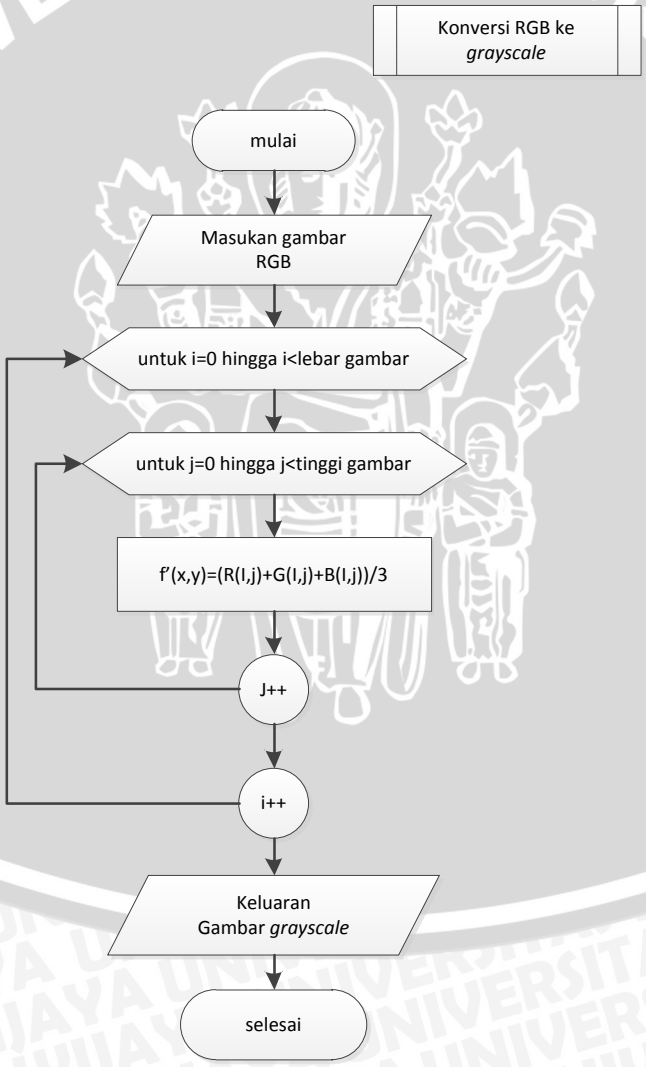
Gambar 3.4 Flowchart Deteksi Plat Nomor

Sumber : Perancangan

Gambar 3.4 adalah diagram alir untuk mencari posisi plat nomor dari gambar yang diambil di lapangan. Secara berurutan metode yang akan digunakan dimulai dari konversi format warna RGB ke *grayscale* kemudian mencari tepi vertikal dan dibinerisasikan. Hasil dari binerisasi kemudian dilakukan operasi morfologi untuk mengurangi *noise*. *Histogram projection* digunakan untuk mencari posisi plat secara horizontal dari plat pada citra biner hasil proses sebelumnya. Pemotongan plat nomor dilakukan setelah penentuan posisi horizontal dan vertikal. Hasil keluaran dari proses ini adalah potongan citra yang memuat plat nomor. Prosedur - prosedur yang dilakukan akan dibahas lebih lanjut pada sub-sub bab selanjutnya.

3.2.1.1 Konversi RGB ke *Grayscale*

Tahap pertama pada proses pencarian plat nomor adalah mengkonversi gambar berwarna atau 24 bit ke dalam gambar *grayscale* 8 bit. Tujuan konversi citra *grayscale* adalah untuk meminimalkan komputasi dengan tidak mengubah kualitas citra. Proses konversi RGB ke *grayscale* mengganti nilai masing masing *pixel* dengan rata-rata dari R, G dan B pada *pixel* tersebut. Langkah pertama adalah mendapatkan *pixel* dari citra dengan menggunakan perulangan dan melakukan manipulasi nilai *pixel* untuk mengganti nilai *pixel* sebelumnya. Prosedur konversi citra RGB ke *grayscale* dapat dijelaskan pada Gambar 3.5 dibawah ini.

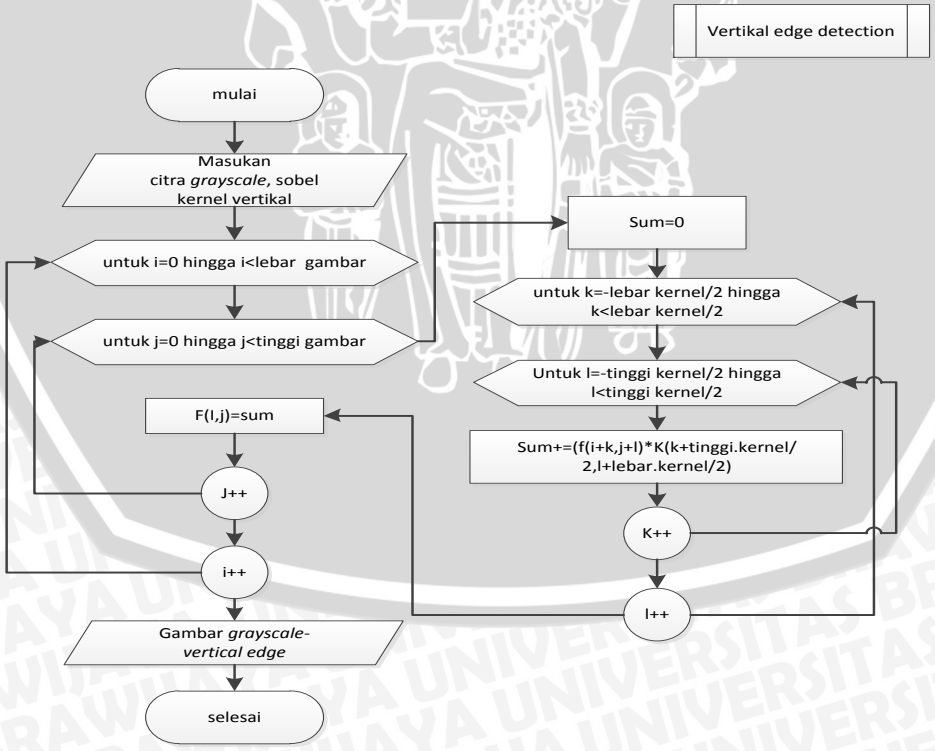


Gambar 3.5 Konversi Citra RGB Ke *Grayscale*

Sumber : Perancangan

3.2.1.2 *Vertical Edge Detection*

Tahap berikutnya adalah malakukan deteksi tepi vertikal pada gambar hasil konversi citra RGB ke *grayscale*. Deteksi tepi yang digunakan adalah kernel vertikal sobel seperti yang dijelaskan pada bab sebelumnya. Deteksi tepi vertikal akan menghapus tepi horizontal yang akan menggaggu pada deteksi plat nomor. Pada gambar mobil tepi horizontal dirasa banyak dan tidak banyak menggambarkan di mana posisi plat nomor berada. Sedangkan dengan tepi vertikal karakter pada plat akan terlihat menonjol dan dapat menunjukan posisi plat berada. Sehingga pada skripsi ini digunakan deteksi tepi vertikal untuk menunjukan fitur pada citra mobil. Deteksi tepi vertikal ini akan mengganti masing - masing nilai *pixel* dengan nilai yang didapatkan dari menghitung kali kernel dan tetangganya. Proses deteksi tepi dimulai dengan mendapatkan indek setiap *pixel*. Posisi *pixel* tertentu tersebut kemudian di sejajarkan dengan indek tengah kernel. Nilai *pixel* tersebut kemudian diganti dengan jumlah dari hasil kali *pixel* dengan kernel sesuai dengan indek masing masing *pixel*. Gambar 3.6 adalah diagram alir *vertical edge detection*.

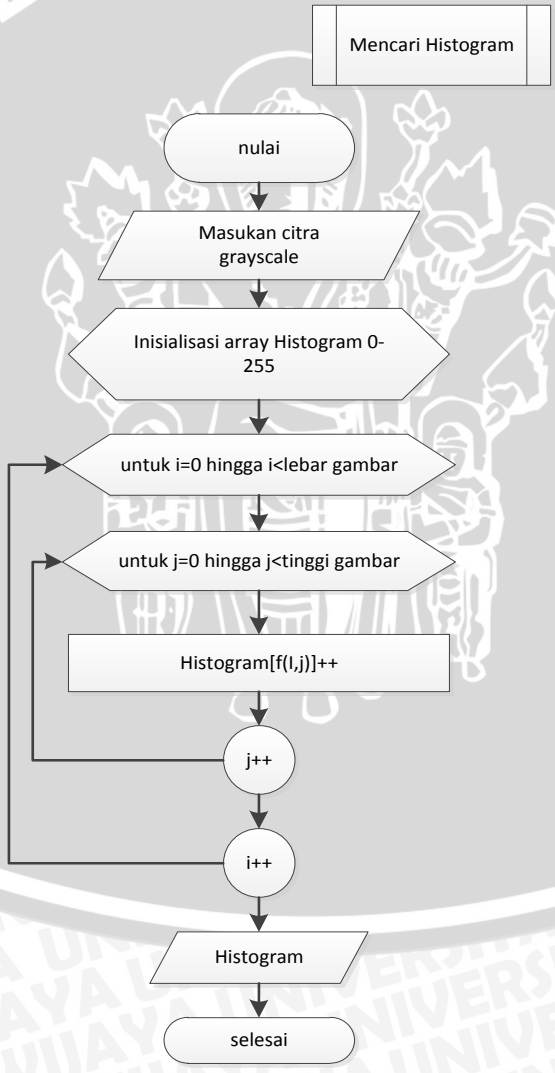


Gambar 3.6 Vertical *Edge Detection*

Sumber : Perancangan

3.2.1.3 Binerisasi *Otsu*

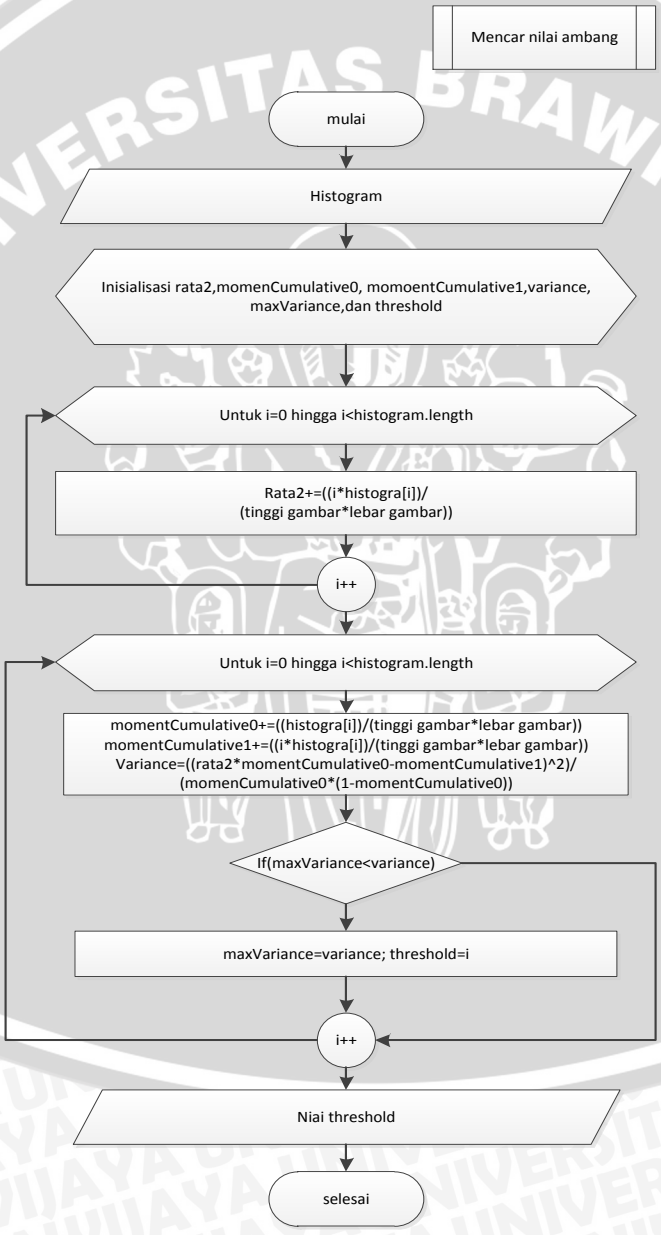
Prosedur berikunya adalah melakukan konversi ke citra biner dengan batas yang ditentukan sesuai metode *Otsu*. Konversi ke citra biner adalah untuk mendapatkan tepi citra yang kongkret berdasarkan nilai ambang. Metode *Otsu* dipilih karena dengan metode ini dapat menyesuaikan nilai ambang tergantung pada histogram citra. Tahap pertama metode *Otsu* mencari histogram pada citra. Sebelumnya diinisialisasi suatu array dengan panjang 256 karena indek warna 8-bit antara nilai 0-255 dan nilai masing masing adalah 0. Dapatkan nilai masing-masing *pixel* dan tambahkan 1 pada array dengan indek nilai *pixel* tersebut. Gambar 3.7 adalah diagram alir untuk menghitung histogram pada citra.



Gambar 3.7 Histogram Citra

Sumber : Perancangan

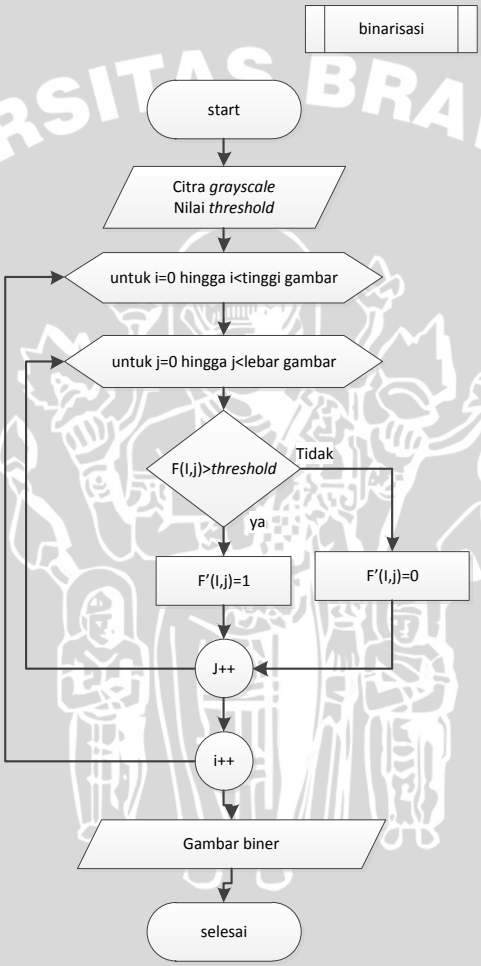
Berdasarkan histogram pada citra kemudian dicari nilai ambang dengan mencoba secara berurutan di mana nilai ambang yang dicari dapat membagi histogram menjadi 2 variansi yang sama. Untuk mencari nilai ambang maka setiap indek warna pada histogram akan dicoba dengan Persamaan 2.11 di mana dicari hasil maksimal untuk didapatkan indek warnanya sebagai nilai ambang. Gambar 3.8 adalah diagram alir metode *Otsu*.



Gambar 3.8 Metode *Otsu*

Sumber : Perancangan

Nilai threshold yang dihasilkan dari metode *Otsu* berikutnya akan digunakan untuk mengkonversi citra tepi *grayscale* menjadi citra biner. Binerisasi ini dilakukan dengan membandingkan nilai *pixel* hasil deteksi tepi dengan nilai ambang hasil metode *Otsu.* Pertama dapatkan setiap *pixel* pada citra kemudian dibandingkan dengan nilai ambang jika lebih besar maka nilainya diganti dengan 1 tetapi jika sebaliknya maka nilainya diganti dengan 0. Gambar 3.9 adalah diagram alir konversi ke citra biner.



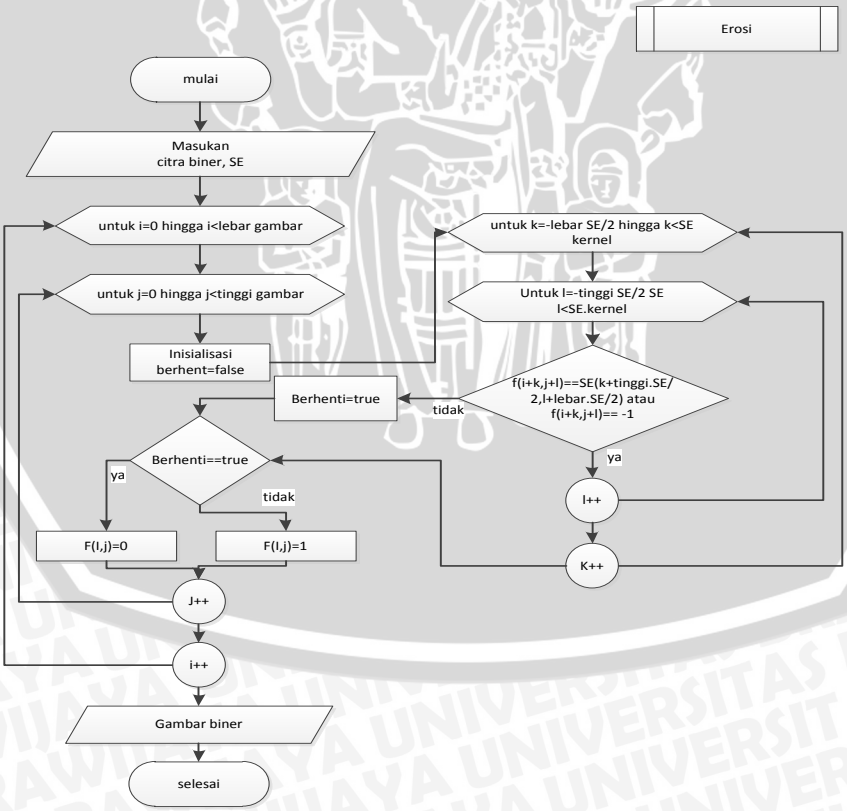
Gambar 3.9 Konversi ke Citra Biner

Sumber : Perancangan

3.2.1.4 *Morphology Operation*

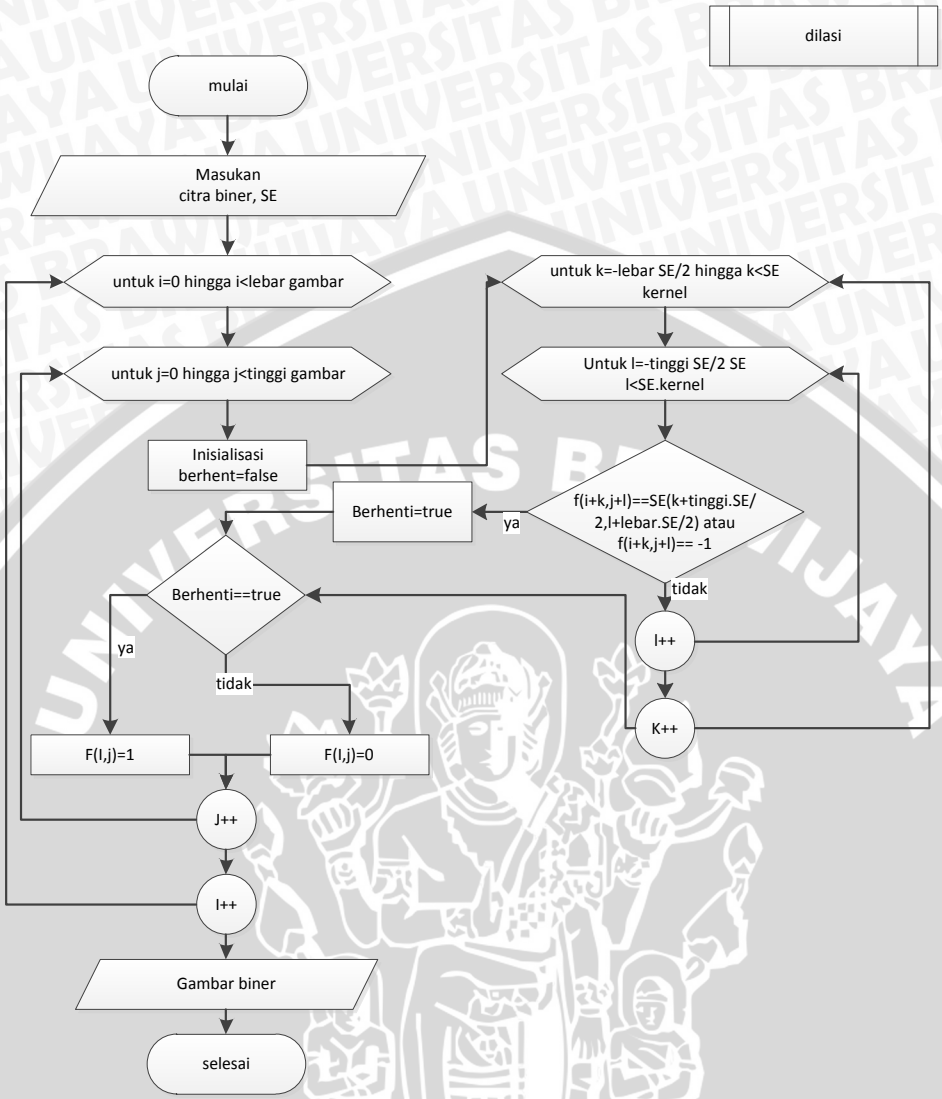
Hasil dari *vertical edge detection* dan metode *Otsu* tidak memungkiri bahwa masih terdapat *noise* pada citra hasil berupa titik yang padat mengganggu pada tahap prosesing. Operasi morfologi juga merupakan salah satu bagian dari

metode yang menjadi pembeda antara penelitian yang dilakukan dengan penlitian Amr Badr, dkk. Amr Badr, dkk melakukan prosesing *average filter* untuk memperkuat tepi pada suatu area (tergantung ukuran kernel/mask) yang dikelilingi oleh tepi lain, contohnya adalah karakter dalam plat. Sedangankan Operasi morfologi diterapkan dengan tujuan untuk mengurangi tepi lain dan mempertahankan tepi dari plat nomor itu sendiri. Operasi morfologi yang digunakan pada deteksi plat nomor adalah operasi *opening*. Operasi *opening* akan menghapus *noise* dengan tidak mengubah bentuk objek dalam citra. Operasi *opening* dilakukan dari kombinasi operasi morfologi erosi dan dilasi. Pada proses erosi didapatkan terlebih dahulu indek *pixel* yang akan dirubah nilainya. Kemudian indek *pixel* tersebut disejajarkan dengan indek tengah SE. Nilai *pixel* akan dirubah menjadi 0 jika saat pencocokan dengan SE-nya ada yang tidak sama, sebaliknya proses dilasi akan merubah nilai *pixel* menjadi 1 jika ada salah satu saja yang sama. Gambar 3.10 adalah diagram alir erosi dan Gambar 3.11 adalah diagram alir dilasi.



Gambar 3.10 *Morphology Operation* Erosi

Sumber : Perancangan



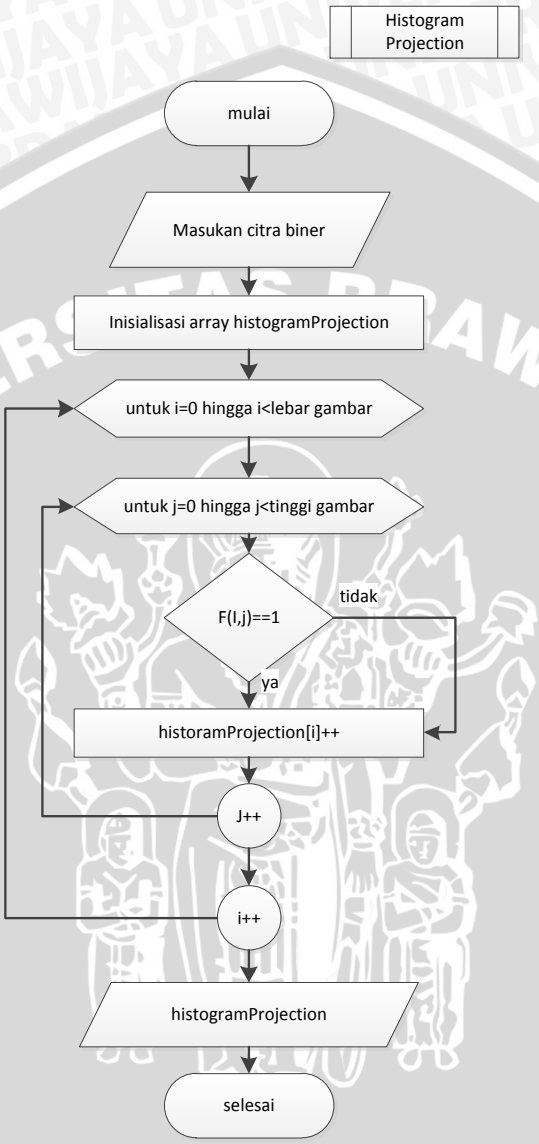
Gambar 3.11 *Morphology Operation* Dilasi

Sumber : Perancangan

3.2.1.5 Histogram Projection

Seperti yang dijelaskan pada sub bab sebelumnya bahwa tepi vetikal pada plat nomor akan terlihat lebih menonjol dari pada tepi horizontalnya. Semakin banyak karakter dalam plat akan membentuk banyak tepi vertikal. *Histogram projection* digunakan untuk menentukan posisi-y pada citra yang cenderung mirip dengan plat. Histogram projection secara horizontal akan menjumlahkan setiap *pixel* dalam 1 baris yang berwarna putih. Tahap pertama adalah melakukan inisialisasi array yang akan memuat hasil histogram sepanjang tinggi pada citra. Kemudian pada setiap baris jumlahkan nilai *pixel* hasil binarisasi. Sehingga

didapatkan *histogram horizontal projection*. Hasil dari histogram projection akan menunjukan posisi plat dengan bukit terbesar pada grafik. Gambar 3.12 adalah diagram alir untuk menghitung *histogram projection*.



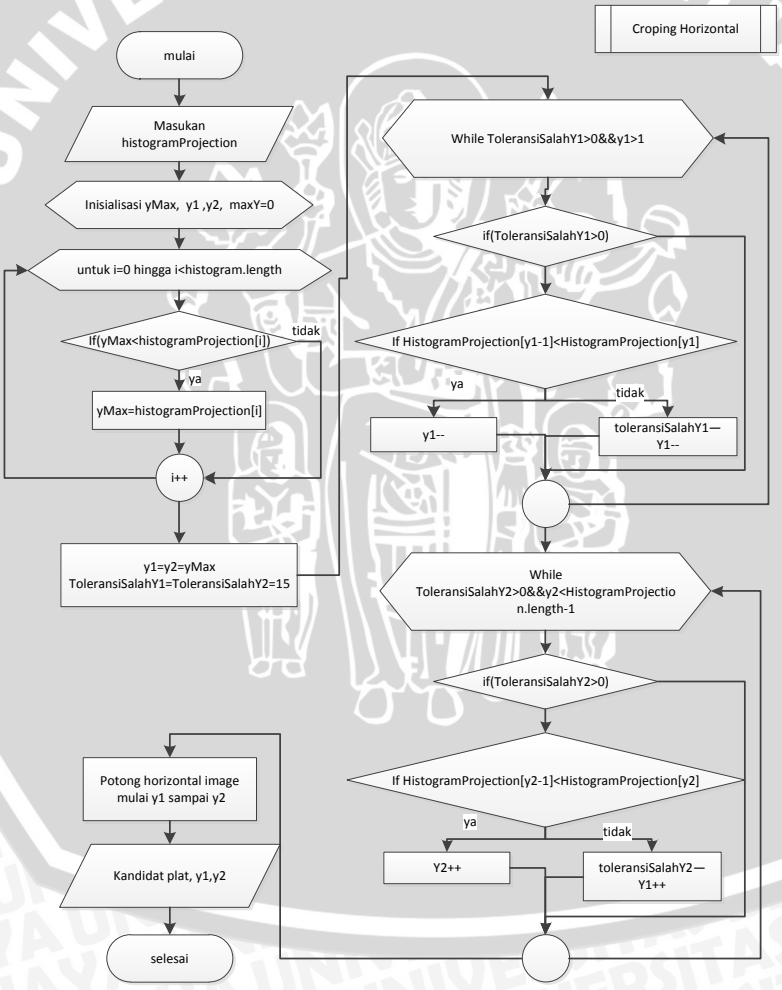
Gambar 3.12 Histogram Projection

Sumber : Perancangan

3.2.1.6 Pemotongan kandidat plat secara horizontal

Pemotongan horizontal kandidat plat pada gambar dimulai dengan menentukan posisi baris yang menunjukan titik terbesar pada grafik. Untuk menentukan posisi baris yang menunjukan titik terbesar maka setiap nilai pada histogram akan dibandingkan dengan suatu nilai yang akan dirubah setiap

didapatkan nilai yang lebih besar. Titik hasil kemudian dikembangan dengan melakukan pengecekan ke baris sebelumnya dan sesudahnya untuk menentukan apakah titik tersebut termasuk dalam bukit dari titik puncak yang ditentukan. Pada setiap bukit pada grafik tentu tidak terlihat sederhana melainkan banyak terlihat peningkatan dan penurunan pada titik tertentu terhadap titik sebelum atau sesudahnya. Sehingga untuk mendapatkan satu bukit tertinggi pada histogram projection diberikan toleransi kesalahan dalam perbandingannya. Setelah didapatkan satu bukit maka maka langkah berikunya adalah pemotongan secara horizontal dari posisi awal bukit hingga posisi akhir bukit. Gambar 3.13 adalah diagram alir pemotongan secara horizontal kandidat plat.

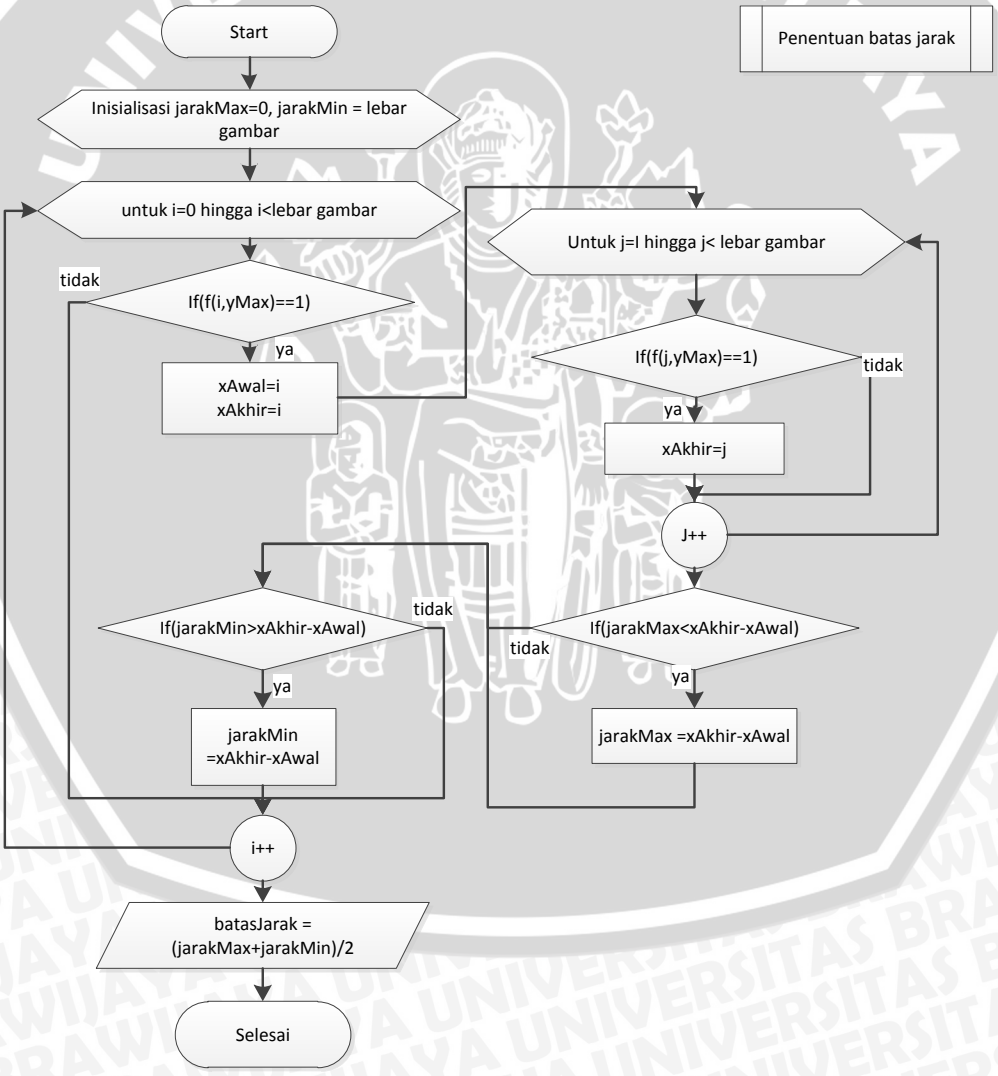


Gambar 3.13 Pemotongan Kandidat Plat Dari Gambar Secara Horizontal

Sumber : Perancangan

3.2.1.7 Pemotongan Plat Nomor

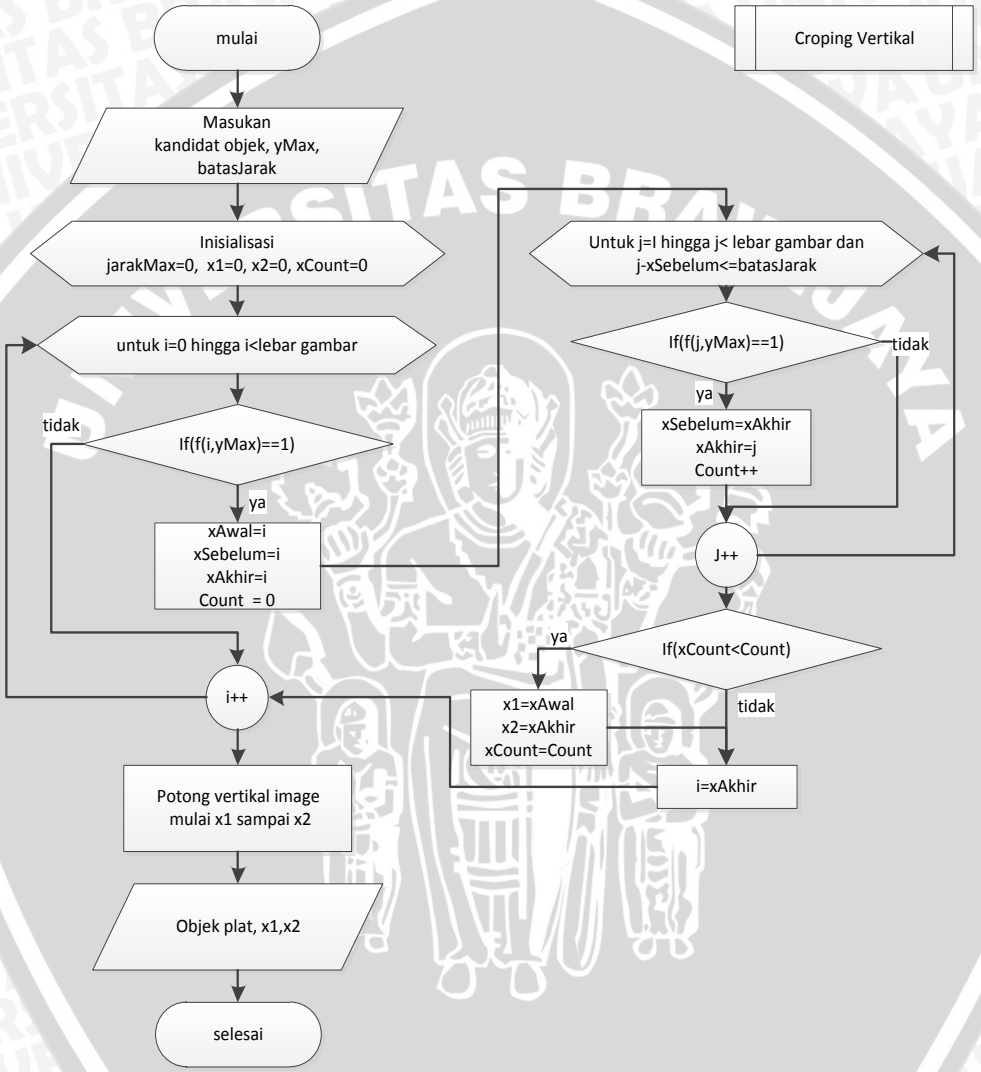
Pemotongan plat nomor dilakukan dengan memanfaatkan jarak antar titik pada baris dengan dengan nilai tertinggi pada *histogram projection*. Jarak digunakan untuk melakukan pelabelan titik - titik pada garis sebagai kelompok suatu objek dimana jika jarak antar suatu titik ke titik berikutnya lebih besar dari jarak yang ditentukan maka objek tersebut diberi label berbeda. Jarak sendiri ditentukan dengan nilai tengah range (*mid-range*) dari sebaran jarak antar titik pada garis yang ditentukan sebelumnya. Gambar 3.14 dibawah ini adalah diagram alir untuk mendapatkan batas jarak dalam pemisahan objek yang di jelaskan sebelumnya.



Gambar 3.14 Penentuan Batas Jarak Pemotongan Vertikal

Sumber : Perancangan

Hasil dari pelabelan titik - titik pada garis horizontal adalah batas titik-x awal dan titik-x akhir dari objek. Objek tersebut kemudian dilakukan segmentasi dan penentuan plat yang ditunjukan pada objek dengan jumlah titik pada garis yang terbanyak. Gambar 3.15 adalah diagram alir pemotongan plat nomor vertikal dari citra hasil pemotongan horizontal.



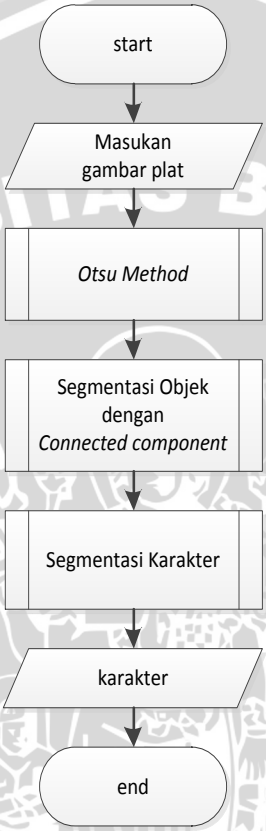
Gambar 3.15 Pemotongan Vetikal Plat Nomor

Sumber : Perancangan

3.2.2 Segmentasi Karakter dengan *connected component*

Sebelum tahap pengenalan karakter sebelumnya dilakukan dulu segmentasi karakter atau objek dalam plat yang akan dikenali polanya dengan *backpropagation neural network.* Segmentasi karakter ini memanfaatkan *Otsu*

method dan *connected component*. Setiap karakter pada plat memiliki warna yang lebih tinggi intensitasnya dari pada plat, berdasarkan ciri ini segmentasi karakter memanfaatkan *Otsu method* kemudian dilakukan segmentasi berdasarkan *connected component* dengan konsep 8-*connectivity.* Gambar 3.16 adalah flowchart dari alur proses segmentasi karakter.



Gambar 3.16 Flowchart Segmentasi Karakter

Sumber : Perancangan

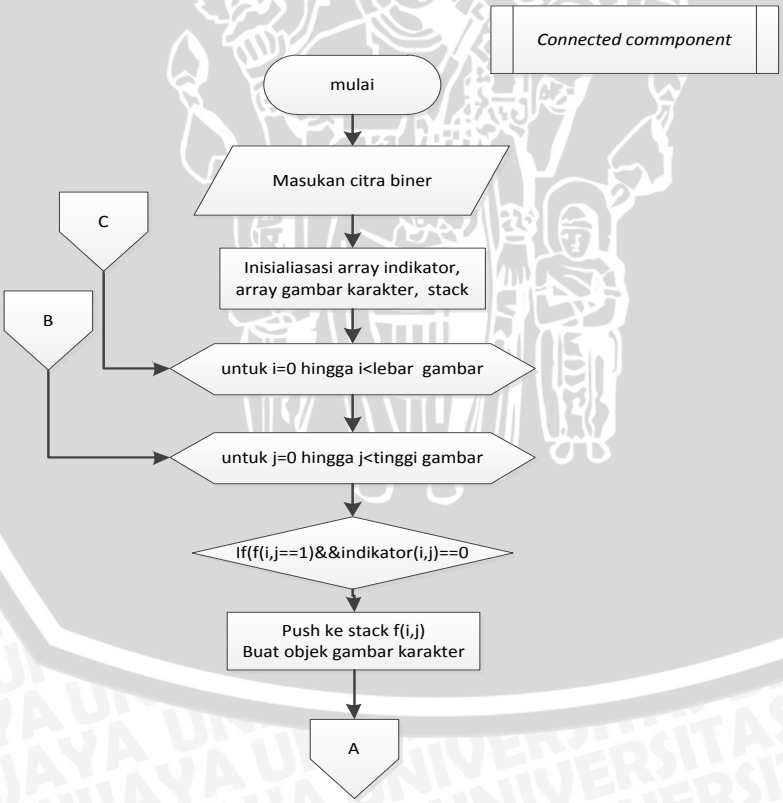
Gambar 3.16 adalah diagram alir proses segmentasi karakter dari plat nomor yang didapatkan dari proses sebelumnya. *Otsu method* yang digunakan sama dengan binarisasi sebelumnya dan telah dijelaskan sebelumnya. Hasil dari *Otsu* methode adalah citra biner. Segmentasi karakter pada gambar plat nomor dijelaskan pada sub sub bab selanjutnya.

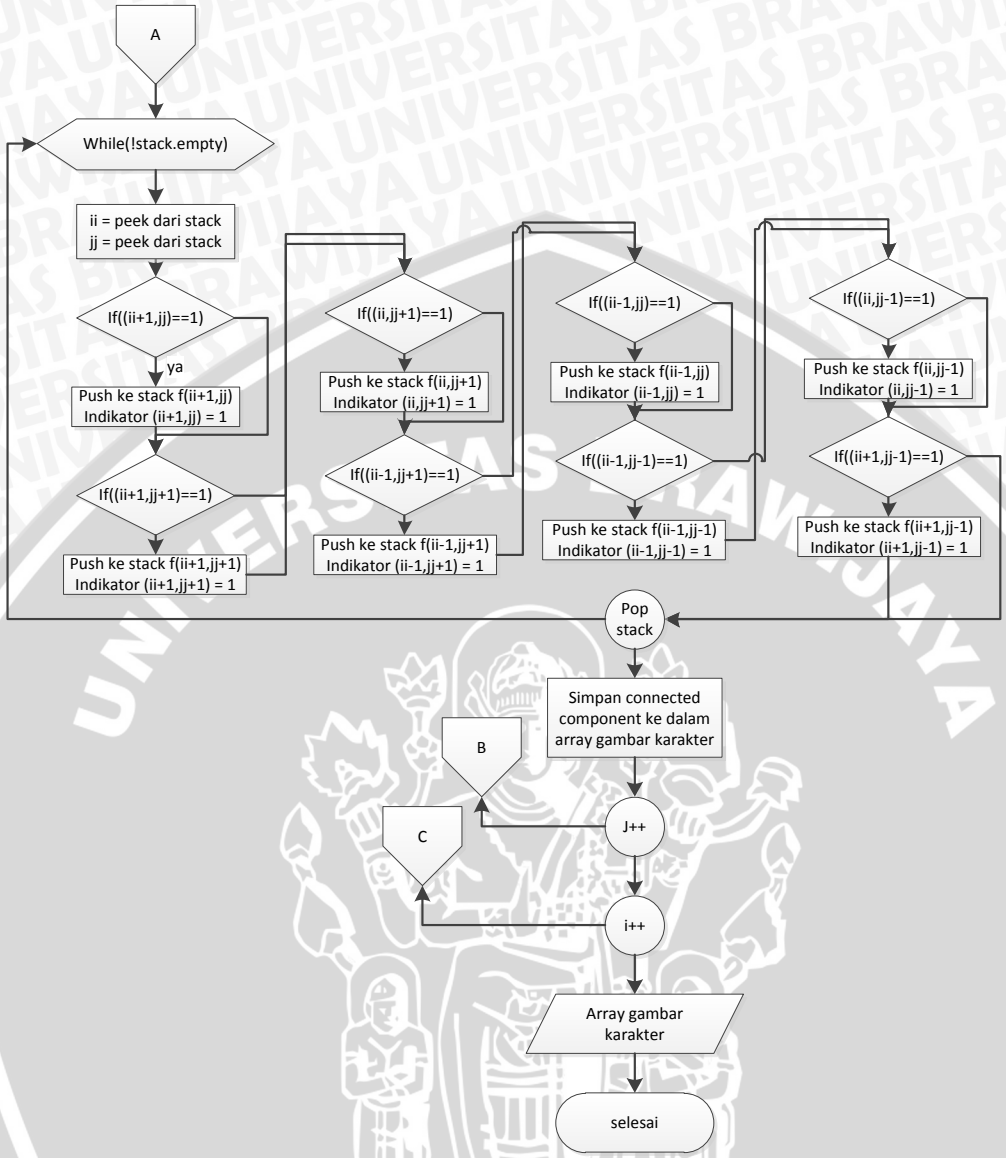
3.2.2.1 Segmentasi Objek

Metode segmentasi yang paling sering digunakan adalah *connected component*. Hasil dari proses binarisasi *Otsu* digunakan untuk mempertegas objek

dalam citra. Kemudian digunakan *connected component* untuk mendapatkan setiap bagian dari objek dalam citra yang terhubung oleh warna putih. Tahap awal pada *connected component* adalah mencari nilai putih pada citra. Kemudian *pixel* tersebut disimpan indeknya untuk ditelusuri.

Penelusuran dilakukan dengan memanfaatkan struktur data *stack*. Setiap indek yang didapatkan disimpan dalam *stack*. Pada posisi awal jika ditemukan warna putih. indek disimpan kemudian diambil dalam perulangan hingga *stack* kosong lagi. Dalam perulangan, *stack* akan diisi lagi oleh indek *pixel* yang berwarna putih yang dicari dari tetangganya mulai bagian sebelah kiri berputar searah jarum jam. Setiap *pixel* yang ditemukan akan ditelusuri kembali ke perulangan berikutnya hingga pada suatu titik tidak ditemukan tetangga yang berwarna putih. Diambil lagi indek *pixel* dalam *stack* jika masih ada dan melakukan tahapan yang sama seperti sebelumnya. Gambar 3.17 adalah diagram alir untuk mendapatkan *connected component.*



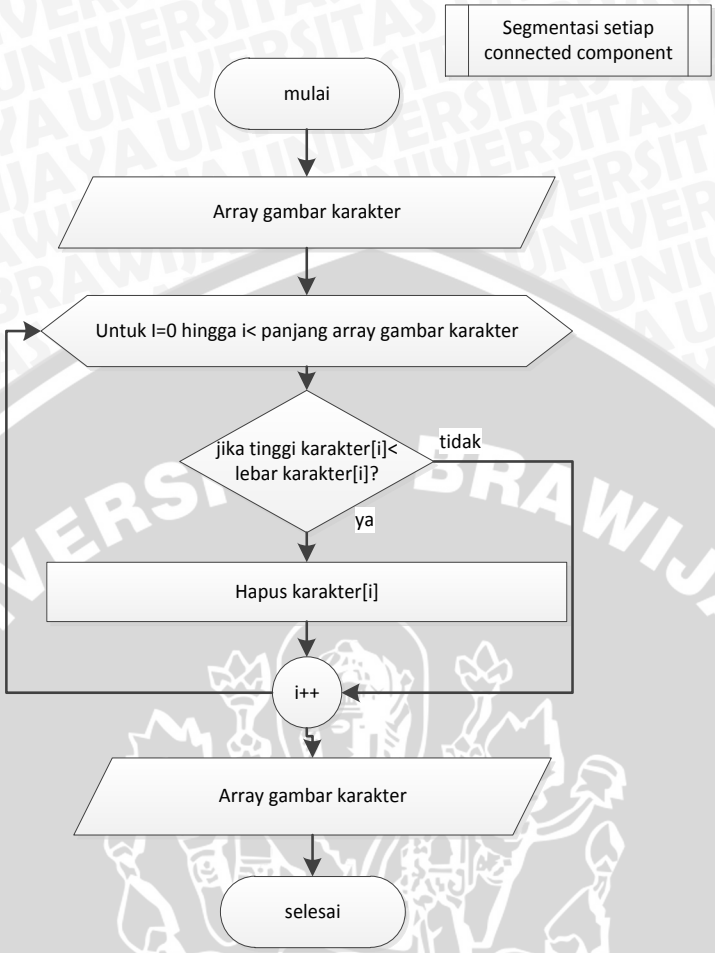


Gambar 3.17 *Connected Component*

Sumber : Perancangan

3.2.2.2 Segmentasi Karakter

Hasil dari segmentasi objek yang didapatkan tentu tidak semuanya adalah karakter bisa saja berupa *noise* atau objek lain. Segmantasi karakter dilakukan dengan menyaring objek yang terdeteksi dari hasil *connected component*. Karakter biasanya memiliki tinggi yang lebih panjang dari pada lebarnya. Karakteristik dari karakter ini kemudian dimanfaatkan setidaknya untuk mengurangi objek lain yang tersegmentasi dari proses sebelumnya. Gambar 3.18 adalah diagram alir proses seleksi objek hasil segmentasi.



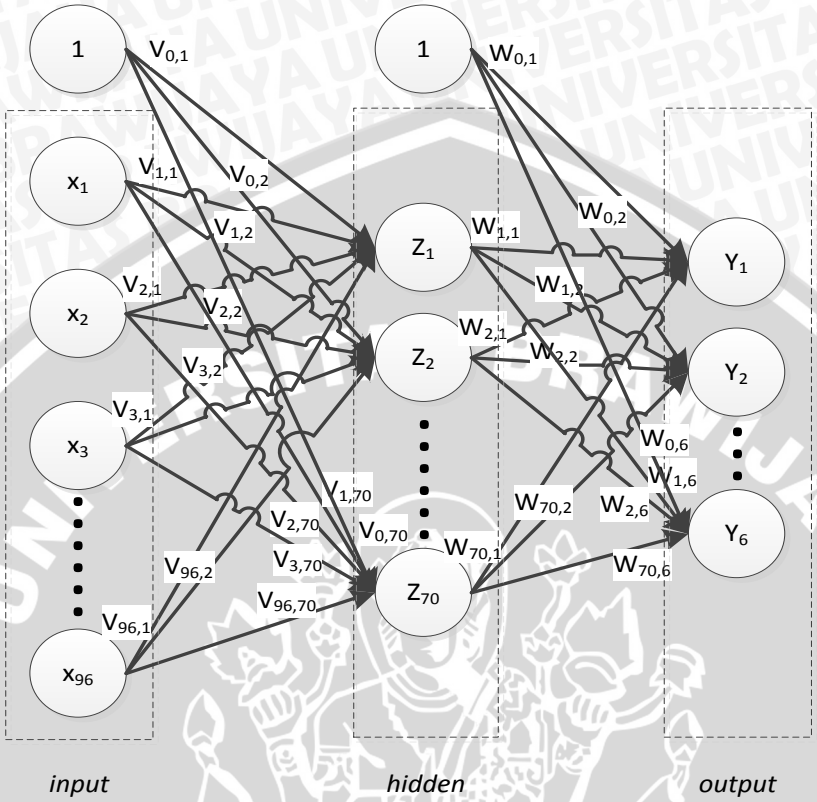
Gambar 3.18 Seleksi Gambar Karakter

Sumber : Perancangan

3.2.3 Pengenalan karakter dengan *backpropagation neural network*

Proses ini dilakukan setelah proses training untuk mengenali pola dari setiap karakter yang menjadi masukan sistem. Proses training meliputi ekstraksi fitur dari gambar karakter dan proses pleatihan metode *backpropagation neural network.* Langkah awal sebelum pelatihan *backpropagation* adalah mendeskripsikan arsitektur jaringan yang akan digunakan. Jumlah node pada layer *input* adalah 96 node didapatkan dari jumlah fitur yang akan diekstrak dari hasil segmentasi karakter dan data latih yaitu 12 untuk ukuran tinggi dan 8 untuk ukuran lebar. Jumlah Node pada layer *output* adalah 6 node sesuai dengan jumlah karakter pada bilangan biner dari total kelas yaitu 36 (26 huruf dan 10 angka) dan jumlah node pada *hidden layer* adalah 70 sesuai dengan aturan kedua penentuan

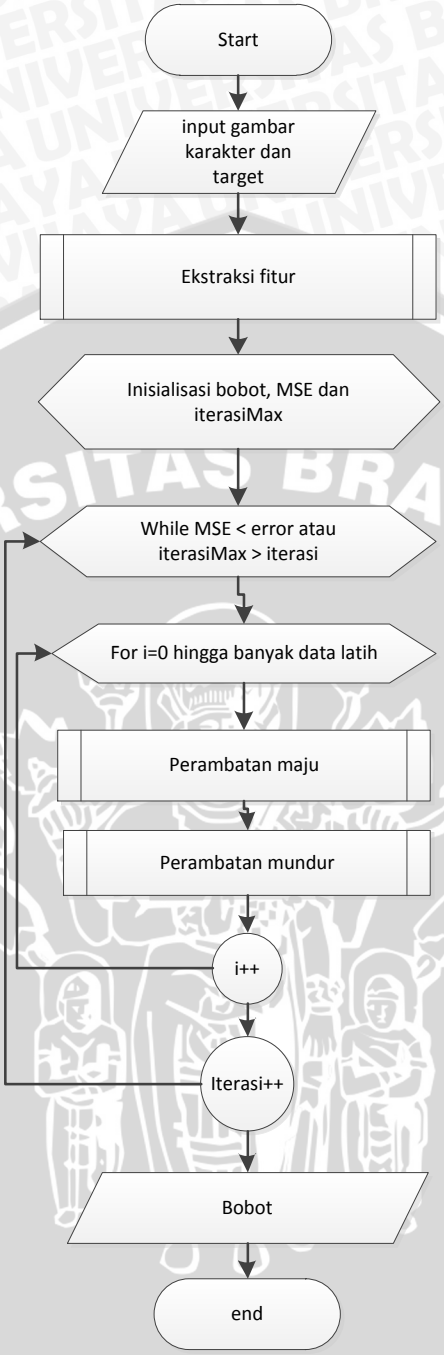
jumlah node oleh *Jeaf Heaton*. Gambar 3.19 adalah gambar arsitektur *Backpropagation* yang digunakan untuk pengenalan.



Gambar 3.19 Arsitektur *Backpropagation Neural Network*

Sumber : Perancangan

Proses pelatihan *backpropagation* telah dijelaskan pada bab sebelumnya, dimulai dengan membawa sinyal pada layer *input* hingga layer *output* dilanjutkan dengan membawa *error* dan koreksi bobot pada layer *output* hingga ke layer *input*. Bobot akan dilatihkan dan diperbaiki secara berurutan pada setiap data dan akan diulang hingga kondisi berhenti terpenuhi, yaitu nilai MSE dan jumlah iterasi. Gambar 3.20 berikut adalah flowchart proses training data latih dengan *backpropagation.*



Gambar 3.20 Training *Backpropagation Neural Network*

Sumber : Perancangan

Hasil dari proses training adalah bobot neural network yang siap digunakan untuk mengenali pola karakter pada plat nomor yang akan diuji. Pada proses pengujian setiap karakter input dari plat diekstraksi fitur kemudian nilai fitur dilewatkan pada neural network dan dihitung tingkat kemiripannya dengan

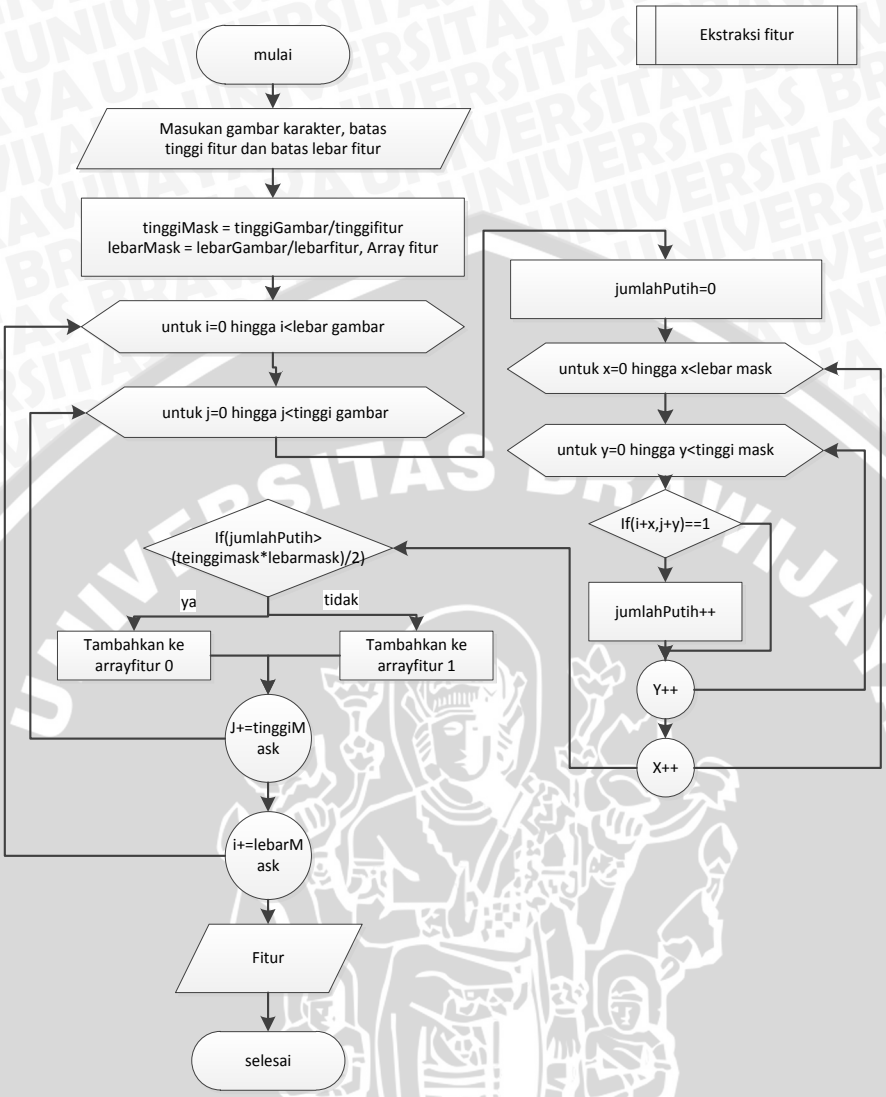
target di data latih. Gambar 3.21 berikut adalah flowchart pencarian pola dari data latih.

Gambar 3.21 Testing *Backpropagation Neural Network*

Sumber : Perancangan

3.2.3.1 Ekstraksi Fitur

Tahap awal dari *backpropagation neural network* adalah ekstraksi fitur. Langkah awal dari ekstraksi fitur adalah menentukan jumlah fitur yang akan digunakan untuk melakukan pengenalan pola. Ekstraksi fitur dari citra dilakukan dengan membagi citra dalam beberapa baris dan kolom. Untuk setiap kotak hasil pembagian akan mewakili nilai 1 jika diketahui nilai 1 lebih dominan dalam kotak tersebut dan sebaliknya. Gambar 3.22 adalah diagram alir untuk ekstrksi fitur.



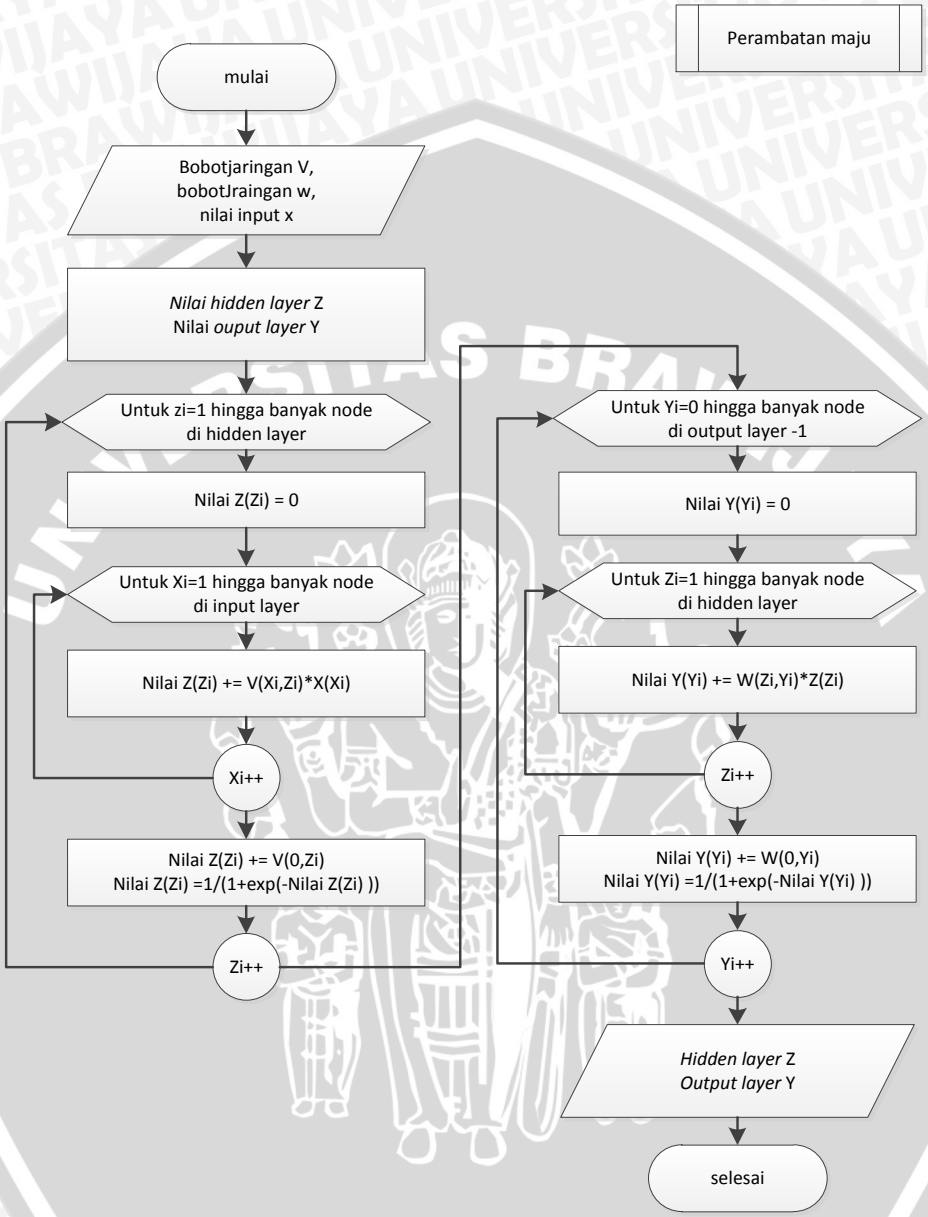
Gambar 3.22 Ekstraksi Fitur Citra

Sumber : Perancangan

3.2.3.2 Perambatan Maju

Ekstraksi fitur akan menentukan input dari jaringan. Perambatan maju akan membawa sinyal input dari layer *input* ke layer *output*. *Bakcpropagation* memiliki target pada layer output yang mana akan dibandingkan dengan nilai yang dihasilkan oleh jaringan melalui proses perambatan maju. Secara teknis telah dijelaskan pada bab sebelumnya. Tahap pertama data pada layer input disampaikan pada *hidden layer* dengan menjumlahkan hasil kali *node* pada layer input dengan bobot yang berhubungan dengan *node* pada *hidden layer* tersebut. Kemudian diaktifasi dengan fungsi aktifasi sigmoid biner. Hasil pada *hidden layer*

kemudian disampaikan pada output layer dengan cara yang sama. Gambar 3.23 adalah diagram alir proses perambatan maju.



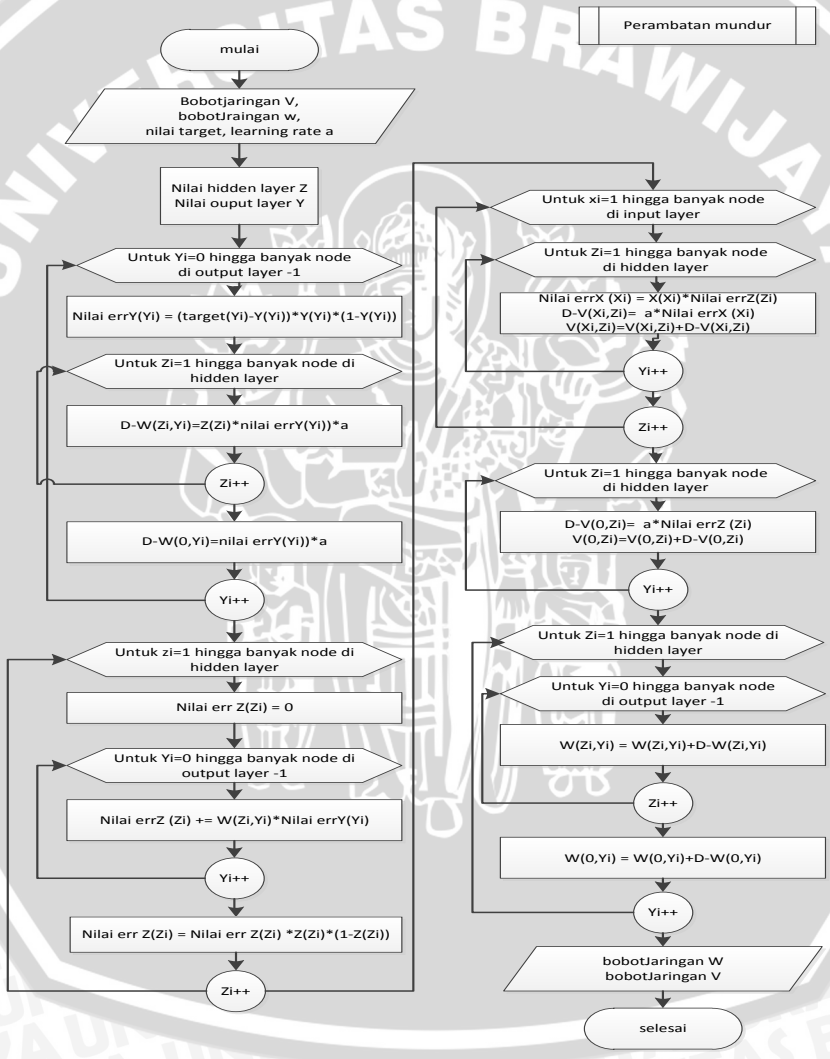
Gambar 3.23 Perambatan Maju

Sumber : Perancangan

3.2.3.3 Perambatan Mundur

Perambatan mundur dilakukan untuk perbaikan bobot jika hasil dari perambatan maju belum memenuhi kondisi berhenti. Perambatan mundur membawa hasil evaluasi untuk menghitung perbaikan bobot dengan persamaan

yang telah dijelaskan pada bab sebelumnya. Evaluasi bobot pada output layer dihitung dengan persamaan 2.24 sebesar *learning rate* yang diinisialisasi sebelumnya. Kemudian evalusi juga disampaikan pada *hidden layer* dengan cara yang sama seperti pada perambatan maju. hasil penjumlahan kemudian diaktifasi dengan persamaan 2.29 dan bobot pada hidden layer diperbarui sebesar *learning rate* nya sesuai persamaan 2.30. hasil perbaruan masing masing bobot kemudian ditambahkan pada bobot sebelumnya. Gambar 3.24 adalah diagram alir perambatan mundur.

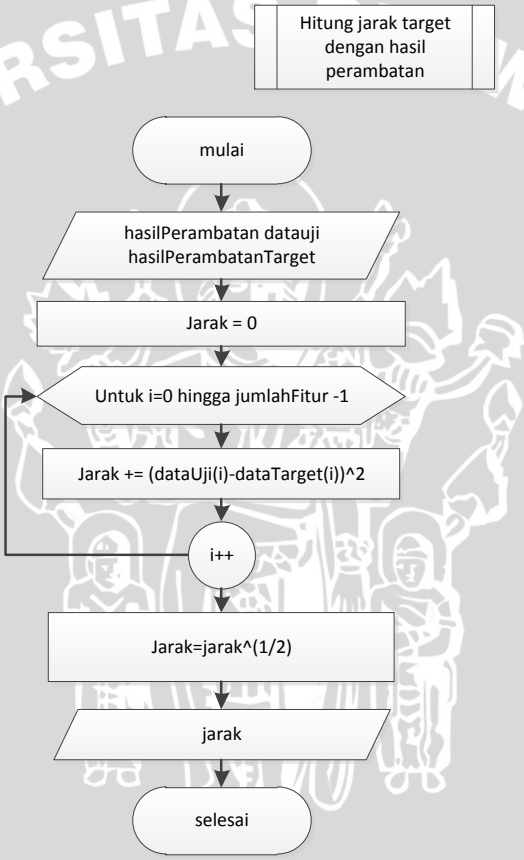


Gambar 3.24 Perambatan Mundur

Sumber : Perancangan

3.2.3.4 Jarak Euclidean

Penentuan pola pada data uji tergantung pada nilai pada layer output yang dihasilkan input data uji saat melewati jaringan. Hasil output data uji kemudian dibandingkan dengan hasil output dari data latih dengan persamaan jarak *euclidean*. Semakin kecil jarak yang dihasilkan maka semakin mirip data uji dengan data latihnya. Jarak *euclidean* dihitung dari akar jumlah kuadrat dari masing masing selisih antara target dengan hasil perambatan maju. Gambar 3.25 adalah diagram alir perhitungan *euclidean distance* data uji dengan data latihnya.



Gambar 3.25 *Euclidean Distance*

Sumber : Perancangan

3.3 Perhitungan Manual

3.3.1 Deteksi Plat nomor dan pengenaan karakter

Deteksi plat nomor pada skripsi ini dilakukan pada citra, sehingga untuk menjelaskan proses pencarian plat nomor diberikan contoh citra seperti ditunjukan pada Gambar 3.26.

Gambar 3.26 Contoh Citra

Bagian citra pada kotak yang dibuat akan dijadikan sebagai contoh studi kasus untuk diselesaikan dengan metode yang digunakan. Bagian diatas kemudian diambil dan dirubah ukurannya menjadi 12x9 *pixel* untuk menyederhanakan perhitungan manual. Gambar 3.27 dibawah ini adalah hasil ekstraksi nilai RGB dari contoh.

Index	R G B										
24 25 29	22 23 27	25 24 29	28 27 31	28 28 30	29 29 31	35 36 41	44 45 50	50 51 55	54 55 59	51 50 56	47 56 52
24 25 29	22 23 27	21 20 25	51 50 55	133 138 144	122 127 133	47 48 53	49 50 55	45 46 50	111 110 123	129 132 139	102 101 107
23 22 27	22 23 27	16 17 22	51 50 56	133 138 144	122 127 133	48 51 56	14 13 19	19 19 21	16 16 18	100 103 110	129 132 139
20 19 24	22 21 26	15 16 21	46 47 52	168 173 179	172 177 183	52 55 60	16 15 21	24 24 26	20 20 22	100 103 110	144 147 154
29 18 23	22 23 27	16 14 19	52 50 55	127 134 142	183 190 198	60 63 70	44 45 50	23 23 23	19 19 19	78 79 84	136 137 142
29 18 23	22 23 27	17 15 20	55 53 58	100 107 115	112 119 127	53 56 63	44 45 50	23 23 23	21 21 21	47 48 53	88 89 94
16 16 18	17 17 19	13 12 17	58 57 62	100 107 115	112 119 127	19 18 23	44 45 50	20 19 24	22 21 26	14 13 19	111 110 116
23 23 25	20 20 22	17 16 21	61 60 65	71 72 76	54 55 59	64 65 68	44 45 50	69 6 73	76 75 80	80 79 85	107 106 112
42 42 44	32 32 34	21 22 26	41 40 45	79 80 85	79 80 85	73 74 78	44 45 50	60 61 63	52 53 58	48 47 53	37 36 42

Gambar 3.27 Hasil Ekstraksi Nilai RGB dari Contoh Citra

Langkah berikutnya adalah melakukan konveri citra berwarna (RGB) ke citra *grayscale* dengan persamaan 2.1 diatas. Contoh perhitungan pada *pixel* f'(0,0) sebagai berikut :

$$f'(0,0) = \frac{R(0,0) + G(0,0) + B(0,0)}{3} = \frac{(24 + 25 + 29)}{3} = 26$$

Sehingga didapatkan citra *grayscale* seperti pada Gambar 3.28 berikut.

26	24	26	29	29	30	37	46	52	56	52	52
26	24	22	52	138	127	49	51	47	115	133	103
24	24	18	52	138	127	52	15	20	17	104	133
21	23	17	48	173	177	56	17	25	21	104	148
23	24	16	52	134	190	64	46	23	19	80	138
23	24	17	55	107	119	57	46	23	21	49	90
17	18	14	59	107	119	20	46	21	23	15	112
24	21	18	62	73	56	66	46	49	77	81	108
43	33	23	42	81	81	75	46	61	54	49	38

Gambar 3.28 Hasil Konversi RGB Ke *Grayscale* dari Contoh Citra

Hasil konversi citra contoh gambar RGB ke grayscale sendiri dapat ditunjukan pada gambar 3.29 dibawah ini.

Gambar 3.29 Contoh Citra Hasil Konversi RGB ke *Grayscale*

Langkah berikutnya untuk mendeteksi posisi plat nomor adalah dengan mencari tepi vertikal dari gambar di mana dalam satu baris semakin banyak tepi vertikal akan menunjukan semakin besar kemungkinan bahwa pada baris tersebut terdapat plat nomor.

Mask yang digunakan untuk deteksi tepi adalah mask sobel vertikal. Perhitungan dengan menggunakan suatu mask dilakukan dengan persamaan 2.4 yaitu *sum of product.* Gambar 3.30 adalah mask sobel vertikal

1	0	-1
2	0	-2
1	0	-1

Gambar 3.30 sobel mask vertikal

Misal untuk menghitung *pixel* f'(0,0) adalah sebagai berikut :

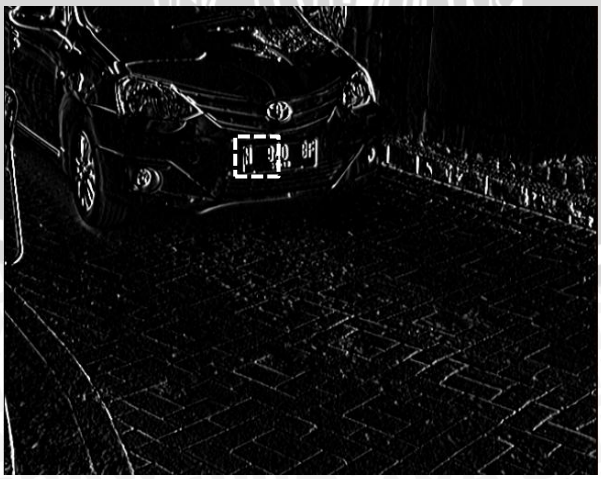
$$f'(0,0) = (1 * 0) + (0 * 0) + (-1 * 0) + (2 * 0) + (0 * 26) + (-2 * 24) + (1 * 0) + (0 * 26) + (-1 * 24) = -72$$

Sehingga didapatkan citra hasil deteksi tepi vertikal seperti pada Gambar 3.31 berikut.

0	4	0	0	0	72	43	0	0	0	20	238
0	14	0	0	0	255	247	22	0	0	0	255
0	19	0	0	0	255	255	97	0	0	0	255
0	20	0	0	0	255	255	135	19	0	0	255
0	24	0	0	0	255	255	148	77	0	0	255
0	22	0	0	0	255	255	109	101	0	0	194
0	17	0	0	0	232	229	49	41	0	0	161
0	34	0	0	0	108	127	45	0	0	0	227
0	45	0	0	0	20	80	44	0	0	1	180

Gambar 3.31 Hasil Deteksi Tepi Vertikal dari Contoh Citra

Hasil deteksi tepi vertikal pada citra sendiri dapat ditunjukan pada Gambar 3.32 dibawah ini.



Gambar 3.32 Contoh Citra Hasil Deteksi Tepi Vertikal

Untuk mempertegas tepi vertikal sehingga nanti dapat dicari nilai intensitas tepi tertinggi pada horizontal histogram projection maka perlu dilakukan binerisasi. Metode binerisasi yang diusulkan dalam skripsi ini adalah *Otsu methode*. *Otsu methode* akan mencari nilai ambang tergantung pada histogram citra tersebut. Misal pada contoh diatas diketahui histogram yang tidak bernilai 0 (hanya sebagai contoh) dan probabilitas masing masing intensitas seperti pada Tabel 3.1. Kolom 'i' pada Tabel 3.1 adalah intensitas-i pada citra sedangkan frekuensi adalah jumlah intensitas-i pada citra. Kolom 'pi' adalah probabilitas untuk intensitas-i sedangkan kolom 'i.pi' adalah hasil kali intensitas-i dengan probabilitasnya. Contoh untuk menghitung perhitungan probabilitas digunakan persamaan 2.10, untuk intensitas-0 ditunjukan sebagai berikut:

- frekuensi(0) $= \text{count}(0)$
  $= 58$
- P_i $= \frac{58}{12*9}$
  $= 0.537$

Tabel 3.1 Contoh Histogram dan Probabilitas

i	frekuensi	pi	i.Pi
0	58	0.537	0
1	1	0.0093	0.009
4	1	0.0093	0.037
14	1	0.0093	0.13
17	1	0.0093	0.157
18	1	0.0093	0.167
19	1	0.0093	0.176
20	2	0.0185	0.37
21	2	0.0185	0.389
22	1	0.0093	0.204
24	1	0.0093	0.222
35	1	0.0093	0.324
40	1	0.0093	0.37
44	1	0.0093	0.407
45	1	0.0093	0.417
46	1	0.0093	0.426
47	1	0.0093	0.435
49	1	0.0093	0.454

73	1	0.0093	0.676
75	1	0.0093	0.694
80	1	0.0093	0.741
97	1	0.0093	0.898
100	1	0.0093	0.926
107	1	0.0093	0.991
108	1	0.0093	1
128	1	0.0093	1.185
135	1	0.0093	1.25
147	1	0.0093	1.361
160	1	0.0093	1.481
179	1	0.0093	1.657
193	1	0.0093	1.787
226	1	0.0093	2.093
229	1	0.0093	2.12
231	1	0.0093	2.139
237	1	0.0093	2.194
248	1	0.0093	2.296
255	13	0.1204	30.69
jumlah	108	μ_T	60.9

Methode Otsu akan mencari nilai variansi maksimum di mana akan membagi histogram menjadi 2 berdasarkan tingkat variannya. Misalkan dicoba untuk tingkat intensitas 109,127 dan 135. Mula mula dihitung rata-rata, momen kumulatif ke-0 dan momen kumulatif ke-1 dengan persamaan masing - masing Persamaan 2.14, Persamaan 2.12 dan Persamaan 2.13. Kemudian di tentukan titik yang memiliki varians tertinggi dengan persamaan 2.11, berikut adalah contoh perhitungannya :

$$\omega(107) = \sum_{i=1}^{107} P_i = 0.537 + 0.009 + .. + 0.009 = 0.7685$$

$$\mu(107) = \sum_{i=1}^{107} i.P_i = 0 + 0.009 + 0.037 + .. + 0.991 = 9.620$$

$$\sigma_B^2(107^*) = \frac{[\mu_T.\omega(109) - \mu_6]^2}{\omega(109)[1-\omega(6)]} = \frac{(60.9*0.76 - 9.62)^2}{0.76*(1-0.76)} = 7764.949$$

$$\omega(108) = \sum_{i=1}^{108} P_i = 0.537 + 0.009 + .. + 0.009 = 0.778$$

$$\mu(108) = \sum_{i=1}^{108} i.P_i = 0 + 0.009 + 0.037 + .. + 1 = 10.62$$

$$\sigma_B^2(108^*) = \frac{[\mu_T.\omega(127) - \mu_6]^2}{\omega(127)[1-\omega(6)]} = \frac{(60.9*0.778 - 10.62)^2}{0.778*(1-0.778)} = 7805.65$$

$$\omega(128) = \sum_{i=1}^{128} P_i = 0.537 + 0.009 + .. + 0.009 = 0.787$$

$$\mu(128) = \sum_{i=1}^{128} i.P_i = 0 + 0.009 + 0.037 + .. + 1.250 = 11.80$$

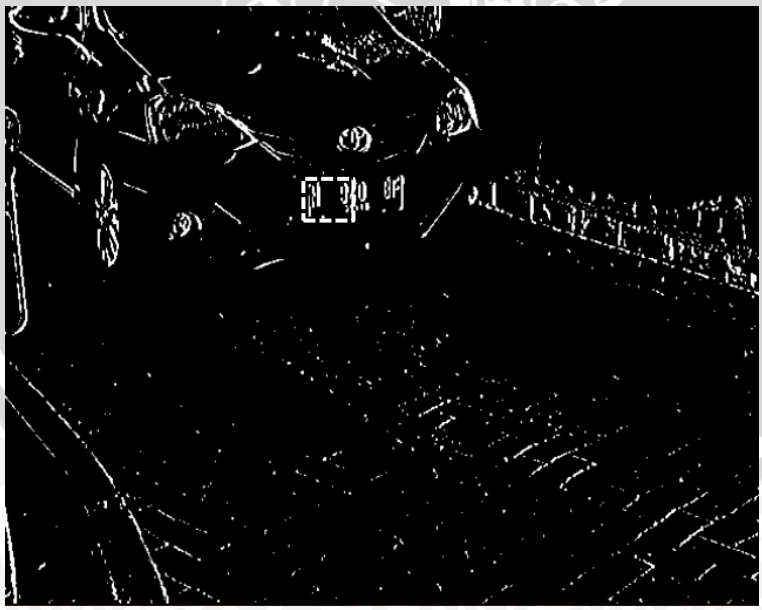
$$\sigma_B^2(128^*) = \frac{[\mu_T.\omega(135) - \mu_6]^2}{\omega(135)[1-\omega(6)]} = \frac{(60.926 * 0.78 - 10.620)^2}{0.78 * (1-0.78)} = 7779.127$$

Percobaan dilakukan untuk semua intensitas hingga didapatkan variance ($\sigma_B^2(k^*)$) maksimal dan nilai ambang k. Setelah dilakukan percobaan dari contoh diatas didapatkan nilai ambang yaitu 109. Sehingga didapat bentuk citra biner seperti pada Gambar 3.33.

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

Gambar 3.33 Hasil Binarisasi metode *Otsu* dari Contoh Citra

Hasil binarisasi metode Otsu pada citra sendiri dapat ditunjukan pada Gambar 3.34 dibawah ini



Gambar 3.34 Contoh Citra Hasil Binarisasi Otsu

Posisi plat nomor didukung oleh banyaknya jumlah vertikal *edge*. Hasil dari deteksi tepi vertikal pada gambar diatas masih suit untuk dilakukan analisis karena tepi dari objek lain berupa *noise* dapat menggaggu analisis pada *histogram projection*. Sehingga perlu dilakukan proses *opening* untuk mengurangi tepi dari objek lain yang menggaggu. Gambar 3.36 berikut adalah hasil dari proses *erosi* dengan *structuring element* seperti pada Gambar 3.35 dibawah ini.

1	1

Gambar 3.35 *Structuring Element* Yang Digunakan Untuk Proses Opening

Opening adalah hasil dari proses erosi dan dilasi yang dilakukan secara berurutan. Gambar 3.35 berikut adalah hasil dari proses erosi.

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

Gambar 3.36 Hasil Operasi Erosi dari Contoh Citra

Hasil dari proses erosi kemudian dilakukan proses dilasi untuk mengembalikan bentuk citra seperti semula dan mengurangi *noise* yang ada akibat deteksi tepi. Gambar 3.37 adalah hasil operasi dilasi pada citra.

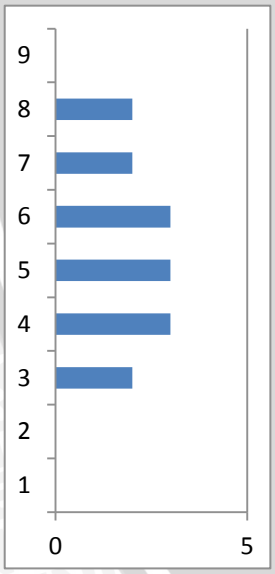
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	1	1	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0

Gambar 3.37 Hasil Operasi Dilasi dari Contoh Citra

Hasil dari operasi *opening* pada citra sendiri dapat dilihat pada gambar 3.38 dibawah ini.

Gambar 3.38 Contoh Citra Hasil Operasi Opening

Hasil Operasi *opening* diatas kemudian dilakukan proyeksi jumlah intensitas pada setiap baris untuk menemukan jumlah tepi secara horizontal yang disebut dengan *histogram projection* secara horizontal. Gambar 3.39 adalah *histogram projection* horizontal yang didapatkan.



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

Gambar 3.39 Hasil *Histogram Projection* dari Contoh Citra

Hasil dari *vertical edge detection* dan *histogram projection* adalah kandidat plat nomor yang dipotong secara horizontal dari citra asal. Sehingga hasinya ditunjukan pada Gambar 3.40.

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

Gambar 3.40 Hasil Pemotongan Gambar Secara Horizontal dari Contoh Citra

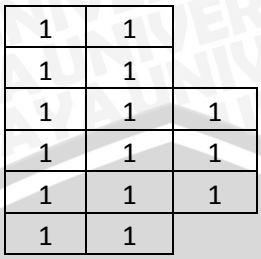
Pemotongan plat vertikal dari hasil pemotongan horizontal dilakukan dengan menganalisis titik pada baris yang memiliki tepi terbanyak pada histogram projection secara horizontal. Sehingga didapatkan baris-3 dari Gambar 3.39 diatas. Misal ditentukan suatu objek dari hasil pemotongan horizontal memiliki jarak tidak lebih dari 2. Sehinga hasil pemotongan plat vertikal ditunjukan pada gambar 3.41.

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

Gambar 3.41 Hasil Pemotongan Vertikal dari Contoh Citra

Tahap berikutnya adalah mendapatkan karakter dari gambar dan pengenalan karakter. Sebelumnya citra diambil dari pemotongan citra *grayscale* dengan posisi yang telah didapatkan pada proses segmentasi plat nomor. Untuk mendapatkan karakter dari plat dilakukan *thresholding* dengan metode *Otsu* dan *connected component.* Pada contoh hasil segmentasi plat nomor diatas, hasil *thresholding* dengan *Otsu* akan menghasilkan citra yang sama dengan sebelumnya. Hasil *connected component* juga akan sama dengan menghilangkan semua pixel bernilai 0 yang berada disekitar objek. karena tidak ada nilai 1 hasil

thresholding yang dipisahkan dengan nilai 0. Sehingga karakter yang didapatkan sama dengan Gambar 3.42 seperti dibawah ini.



Gambar 3.42 Hasil Pemotongan Karakter dari Contoh Citra

3.3.2 Pelatihan dan pengenalan karakter

Pelatihan karakter dimulai dari inisialisasi awal. Jumlah *node* untuk layer input adalah 25, untuk *hidden layer* 2 dan untuk *output layer* adalah 2. *Learning rate* yang diberikan adalah 0.3 dan maksimum iterasi adalah 1. Langkah awal membangkitkan nilai random antara -0.5 hingga 0.5 untuk bobot-V dan bobot-W. dan hasilnya ditunjukan pada Tabel 3.2. Tabel Bobot-V pada Tabel 3.2 berupa matrik dimana jumlah baris sebanyak jumlah *node* pada layer input ditambah dengan 1 baris sebagai bias dan jumlah kolom sebanyak *node* pada *hidden layer*. Sedangkan pada Tabel Bobot-W pada Tabel 3.2 juga berupa matrik dimana jumlah baris sebanyak jumlah *node* pada *hidden layer* ditambah 1 baris sebagai bias dan jumlah kolom sebanyak jumlah *node* pada *output layer*.

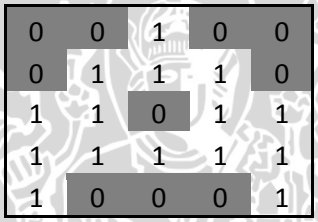
Tabel 3.2 Inisialisasi Bobot Secara Random

bobot v		
input\hidden	1	2
1	-0.14	-0.16
2	0.416	0.354
3	-0.03	-0.6
4	0.41	-0.3
5	-0.24	0.232
6	0.49	0.194
7	-0.02	-0.03
8	-0.41	0.156
9	-0.31	-0.37
10	-0.23	0.413
11	0.047	0.023
12	0.379	0.217
13	0.202	0.014

bobot w		
hidden\output	1	2
1	-0.5	0.4269
2	-0.4	-0.1551
bias	-0.4	0.15493

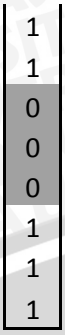
14	-0.15	-0.33
15	0.481	-0.1
16	0.237	-0.27
17	-0.45	0.004
18	0.449	-0.06
19	-0.43	0.125
20	-0.34	-0.33
21	0.272	0.306
22	0.194	0.451
23	-0.5	-0.24
24	-0.36	0.128
25	-0.18	0.439
bias	-0.43	0.23

Misal diberikan citra A dengan ukuran 5x5 *pixel* seperti pada Gambar 3.43.



Gambar 3.43 Data latih jaringan

Sehingga nilai fitur untuk masing-masing *node* pada layer *input* adalah seperti pada Gambar 3.44.



Gambar 3.44 Nilai *Node* pada *Input Layer*

Untuk setiap data latih lakukan prambatan maju dan perambatan mundur untuk pembaruan bobot. Misal untuk data-A pada iterasi-1, Tahap pertama lakukan perambatan maju cari nilai z_{in} dan z_{out} dengan persamaan masing – masing Persamaan 2.19 dan Persamaan 2.20. Misal untuk data-A pada hidden layer di *node* ke-1 seperti berikut :

$$z_{in1} = v_{01} + \sum_{i=1}^{n} x_i v_{i1}$$

$$z_{in1} = 0.43 + (-0.14 * 0) + (0.416 * 0) + (-0.03 * 1) + \cdots + (-0.18 * 1)$$

$$z_{in1} = -0.025$$

$$z_{out_1} = \frac{1}{1+\exp(-1*-0.025)} = 0.49375032550049$$

Sehingga didapatkan hasil nilai *input* dan *output* untuk *node* pada hidden layer seperti pada Tabel 3.3.

Tabel 3.3 Nilai *Input* Dan *Output* Pada *Hidden Layer*

Hidden Layer	Z_in	Zout
1	-0.025	0.493750326
2	-1.05	0.259225101

Tahap berikutnya mencari nilai pada layer *output* yang sebelumnya target telah ditentukan. Misal target karakter-A adalah 00. Berikut adalah contoh perhitungan *node* ke-1 pada layer output dengan Persamaan 2.21 dan Persamaan 2.22 :

$$y_{in_1} = w_{01} + \sum_{j=1}^{p} z_j w_{j1}$$

$$y_{in_1} = -0.4 + (-0.5 * 0.493750326) + (-0.4 * 0.259225101)$$

$$y_{in_1} = -0.750565203077383$$

$$y_{out_1} = \frac{1}{1+\exp(-1*-0.750565203077383)} = 0.32069815837779$$

Sehingga didapatkan nilai input dan output untuk layer *output* seperti pada Tabel 3.4.

Tabel 3.4 Hasil Input Dan Output Pada *Output Layer*

outputlayer	Y_in	Y_out
1	-0.750565203	0.320698158
2	0.325506201	0.580665566

Tahap kedua adalah melakukan perambatan mundur untuk perbaikan bobot dengan Persamaan 2.23 dan Persamaan 2.24. Perbaikan bobot diawali dengan menghitung nilai error. Contoh nilai error untuk *node* ke-1 pada layer output dan bobot-W dari nilai sebagai berikut :

$\delta_1 = (t_1 - y_1) f'(y_{in_1})$

$\delta_1 = (t_1 - y_1) f(y_{in_1})(1 - f(y_{in_1}))$

$\delta_1 = (0 - 0.320698158)(0.320698158)(1 - 0.320698158)$

$\delta_1 = -0.0698643662648333$

$\varphi_{11} = z_1 \delta_1$

$\varphi_{11} = 0.493750326 * -0.0698643662648333$

$\varphi_{11} = -0.0344955535841469$

Sehingga hasil error pada output layer ditunjukan pada Tabel 3.5.

Tabel 3.5 Hasil Perhitungan Error

outputlayer	Y_in	Y_out	target	error
1	-0.750565203	0.320698158	0	-0.06986
2	0.325506201	0.580665566	0	-0.14139

Jadi perubahan bobot-W dapat ditunjukan seperti berikut sesuai dengan Persamaan 2.25 dan Persamaan 2.26 :

$\Delta w_{jk} = a.\varphi_{jk}$ (untuk bias φ_{jk} diganti dengan δ_j)

$\Delta w_{11} = 0.3 * -0.0344955535841469$

$\Delta w_{11} = -0.014142643$

Kemudian hitung juga nilai error yang masuk ke hidden layer denga Persamaan 2.27 dan nilai error pada hidden layer dengan Persamaan 2.28 dan Persamaan 2.29. Contoh pada node ke-1 di hidden layer.

$\delta_{in_1} = \sum_{j=1}^{m} \delta_k \, w_{j1}$

$\delta_{in_1} = (-0.06986 * -0.5) + (-0.14139 * 0.4269)$

$\delta_{in_1} = -0.750565203077383$

$\delta_1 = \delta_{in_1} f'(z_{in_1})$

$\delta_1 = \delta_{in_1} f(z_{in_1})(1 - f(z_{in_1}))$

$\delta_1 = (-0.750565203077383) * (0.493750326) * (1 - 0.493750326)$

$\delta_1 = -0.00635559959598903$

$\varphi_{ij} = x_i \delta_j$

$\varphi_{ij} = 0 * -0.00635559959598903$

$\varphi_{ij} = 0$

Sehingga hasil error pada hidden layer ditunjukan seperti pada Tabel 3.6.

Tabel 3.6 Hasil Perhitungan Evaluasi Error

Hidden Layer	δ_in	nilai error hidden
1	-0.025426371	-0.0063556
2	0.049875031	0.009577375

Jadi pembaruan bobot-V dapat dicari dengan Persamaan 2.30 untuk node dan 3.31 untuk bias, berikut adalah contohnya :

$\Delta v_{ij} = a.\varphi_{ij}$ (untuk bias φ_{ij} diganti dengan δ_i)

$\Delta v_{11} = 0.3 * 0$

$\Delta v_{11} = 0$

Langkah terakhir untuk tahap perambatan mundur adalah pembaruan bobot-V dan bobot-W dengan Persamaan 2.32 untuk node dan Persamaan 2.33 untuk bias dan Persamaan 2.34 untuk node dan 2.35 untuk bias. sehingga didapatkan pembaruan bobot seperti Tabel 3.7.

Tabel 3.7 Pembaruan Bobot-V Dan Bobot-W

bobot v(baru)		
input\hidden	1	2
1	-0.14	-0.16
2	0.416	0.354
3	-0.03190668	-0.597126788
4	0.41	-0.3
5	-0.24	0.232

bobot w (baru)		
hidden\output	1	2
1	-0.510348666	0.405956883
2	-0.405433179	-0.166095399
bias	-0.42095931	0.112513588

6	0.49	0.194
7	-0.02190668	-0.027126788
8	-0.41190668	0.158873212
9	-0.31190668	-0.367126788
10	-0.23	0.413
11	0.04509332	0.025873212
12	0.37709332	0.219873212
13	0.202	0.014
14	-0.15190668	-0.327126788
15	0.47909332	-0.097126788
16	0.23509332	-0.267126788
17	-0.45190668	0.006873212
18	0.44709332	-0.057126788
19	-0.43190668	0.127873212
20	-0.34190668	-0.327126788
21	0.27009332	0.308873212
22	0.194	0.451
23	-0.5	-0.24
24	-0.36	0.128
25	-0.18190668	0.441873212
bias	0.42809332	-0.227126788

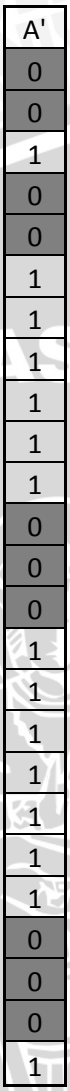
Pengenalan karakter pada jaringan saraf tiruan dilakukan dengan memasukan pola uji pada jaringan dengan bobot hasil pelatihan sebelumnya untuk mendapatkan hasil pada *output layer*. Sehingga pola uji dapat dikenali dengan membandingkan kemiripan nilai *layer ouput* pola uji dengan nilai *layer ouput* pola yang telah dilatihkan. Perbandingan yang dilakukan pada penelitian ini menggunakan persamaan jarak *euclidean*, jarak paling kecil menunjukan bahwa semakin mirip pola yang diujikan dengan pola tertentu yang dilatihkan.

Misal diberikan huruf A dengan pola yang dimodifikasi dari pola data latihnya seperti ditunjukan pada Gambar 3.45.

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

Gambar 3.45 Data latih jaringan

Sehingga nilai fitur untuk masing-masing *node* pada layer *input* adalah seperti pada Gambar 3.46.



Gambar 3.46 Nilai *Node* pada *Input Layer*

Selanjut nilai node pada layer input diteruskan ke *hidden layer* hingga ke *output layer* dengan cara yang sama dengan bobot hasil pelatihan sebelumnya seperti pada tahap perambatan maju saat pelatihan jaringan. Sehingga masing – masing nilai pada hidden layer dan output layer dapat ditunjukan pada Tabel 3.8 di bawah ini.

Tabel 3.8 Nilai *hidden layer* dan *Output Layer*

Hidden Layer	Z_in	Zout
1	0.006	0.501499996
2	-0.33	0.418240623

Output layer	Y_in	Y_out
1	-0.818046247	0.306178545
2	0.304151227	0.575457004

Hasil *output layer* pola uji kemudian dibandingkan dengan masing masing hasil *ouput layer* pola yang dilatihkan untuk dicari jarak paling kecil. Kemiripan yang dicari di tunjukan dengan persamaan jarak *euclidean*, berikut adalah contoh perhitungan nilai jarak.

$$d(x,y) = \sqrt{\sum(x_i - y_i)^2}$$

$$d(x,y) = \sqrt{(0.320 - 0.306)^2 + (0.580 - 0.575)^2}$$

$$d(x,y) = \sqrt{0.00021 + 0.000027}$$

$$d(x,y) = 0.0154$$

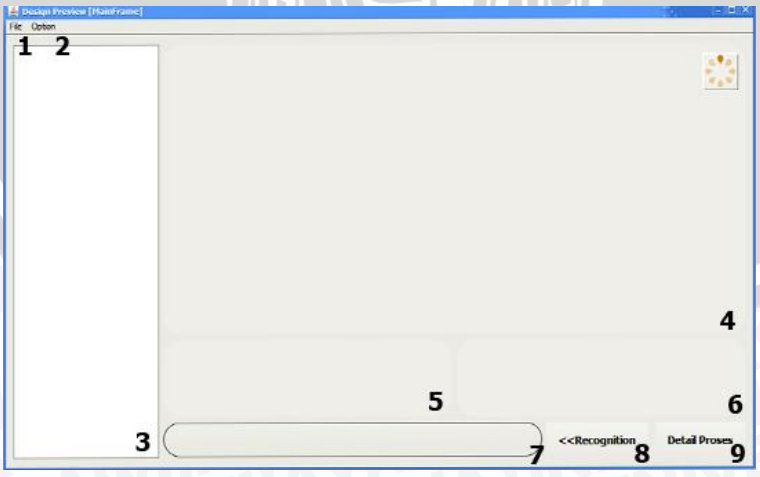
Jarak antara pola data uji dengan pola data latih dengan persamaan *eulidean* adalah 0.0154, karena tidak terdapat data latih lain sebagai pembanding maka jarak tersebut adalah jarak terkecil dan pola data uji dikenali sebagai karakter A.

3.4 Perancangan User Interface

Rancangan user interface dari deteksi dan pengenalan plat nomor dengan *Vertical Edge Detection* terdiri dari empat halaman yaitu halaman utama, detail pemrosesan citra, management data training dan hasil pengenalan karakter.

3.4.1 Perancangan Halaman Utama

Rancangan user interface halaman utama dari deteksi dan pengenalan plat nomor dapat dilihat pada Gambar 3.47.



Gambar 3.47 Perancangan Halaman Utama

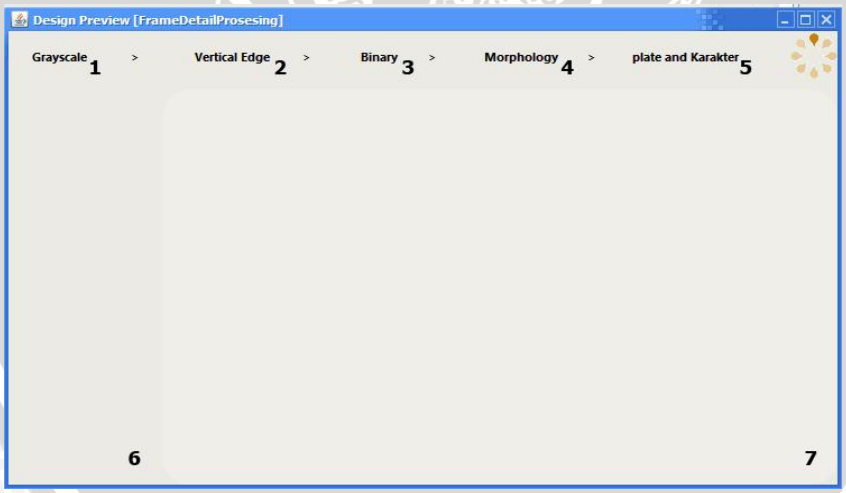
Sumber : Perancangan

Keterangan :

1. Bagian untuk input file gambar yang akan diproses untuk dideteksi dan dikenali plat nomornya.
2. Bagian untuk mengatur parameter dan managemen data latih untuk pengenalan karakter.
3. List seleksi jika memubuka satu folder sekaligus.
4. Hasil *open single file* atau setelah memilih pada list di nomor 3.
5. Hasil deteksi plat nomor.
6. Hasil segmentasi karakter.
7. Hasil pengenalan karakter.
8. Tombol untuk memproses dan menampilkan detail pengenalan karakter.
9. Tombol untuk menampilkan detail prosesing pada citra untuk deteksi plat nomor

3.4.2 Perancangan Detail Pemrosesan Citra

Rancangan user interface detail pemrosesan citra dari deteksi dan pengenalan plat nomor dapat dilihat pada Gambar 3.48.



Gambar 3.48 Perancangan Detail Pemrosesan Image

Sumber : Perancangan

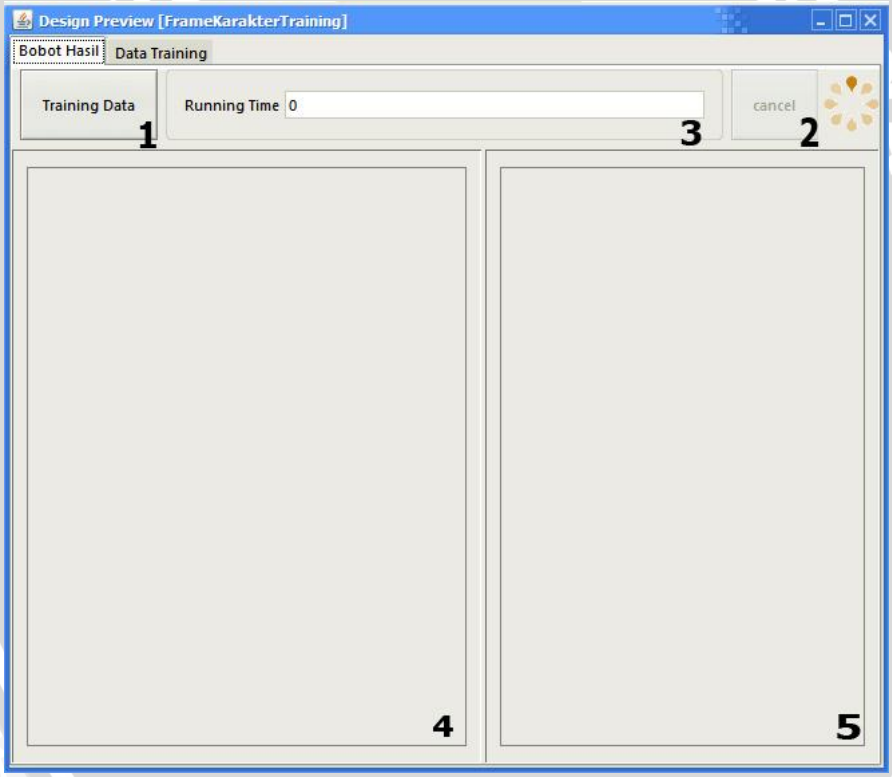
Keterangan :

1. Tombol untuk menunjukan hasil konversi citra RGB ke *grayscale*.
2. Tombol untuk menunjukan hasil *vertical edge detection.*
3. Tombol untuk menunjukan hasil Binarisasi dengan *Otsu method.*

4. Tombol untuk menunjukan hasil operasi morfologi yaitu opening.
5. Hasil dari deteksi plat nomor.
6. Grafik *histogram projection* yang muncul setelah tombol operasi morfologi .
7. Hasil citra hasil prosesing.

3.4.3. Perancangan Halaman Managemen Data Training

Rancangan user interface untuk melakukan pengolahan pada data training dari deteksi dan pengenalan plat nomor terdiri dari dua panel. Panel pertama ditunjukan pada Gambar 3.49 untuk proses pelatihan dan hasil bobot. Panel kedua ditunjukan pada Gambar 3.50 untuk managemen data training yaitu menambahkan dan menghapus data training.



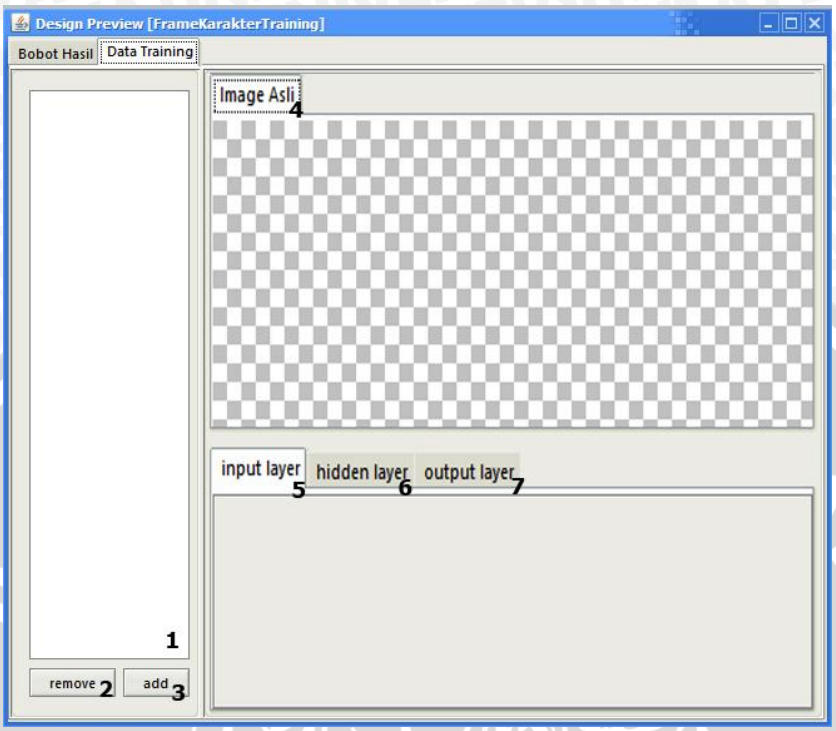
Gambar 3.49 Perancangan Managemen Data Training Untuk Hasil Bobot

Sumber : Perancangan

Keterangan :

1. Tombol untuk memulai proses pelatihan.
2. Tombol melakukan pembatalan proses training, hanya aktif jika proses telah dijalankan.
3. Waktu eksekusi pelatihan terakhir

4. Bobot-v yang dihasilkan dari proses pelatihan.
5. Bobot-W yang dihasilkan dari proses pelatihan.



Gambar 3.50 Perancangan Managemen Data Training Untuk Detail Data

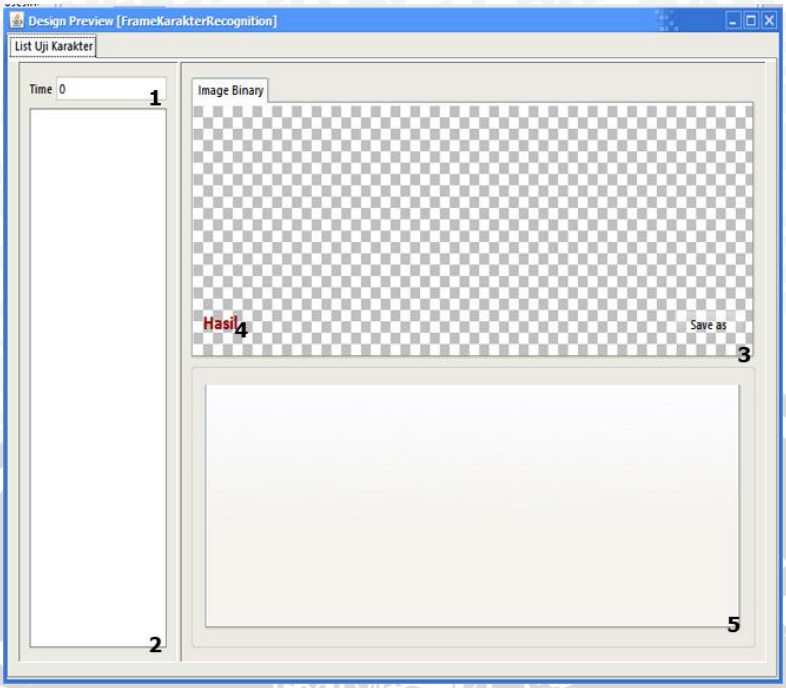
Sumber : Perancangan

Keterangan :

1. List untuk data training.
2. Hapus data training yang dipilih dari list.
3. Tambah data training.
4. Gambar data training yag dipilih dari list.
5. Variabel masukan untuk jaringan dari data training yang dipilih.
6. Nilai setiap unit pada *hidden layer*.
7. Nilai setiap unit pada *output layer*.

3.4.4. Perancangan Detail Pengenalan Karakter

Rancangan user interface untuk melakukan pengujian atau pengenalan karakter dari deteksi dan pengenalan plat nomor dapat dilihat pada Gambar 3.51.



Gambar 3.51 Perancangan Detail Pengenalan Karakter

Sumber : Perancangan

Keterangan :

1. Waktu eksekusi untuk pengenalan karakter.
2. List karakter hasil segmentasi.
3. Gambar karakter hasil segmentasi
4. Hasil prediksi dari sistem.
5. Detail perhitungan dan perbandingan jarak antara data uji dengan data latihnya.

3.5 Perancangan Uji Coba dan Evaluasi

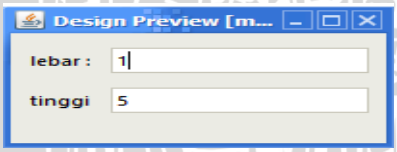
Perancangan Uji coba dilakukan untuk menggambarkan skenario pengujian terhadap sistem deteksi dan pengenalan plat nomor dengan metode vertical edge detection dan backpropagation neural network. Pengujian akan dilakukan pada beberapa uji coba antara lain :

1. Uji coba mengetahui perbesaran optimal terkait pengaruh vertical edge dari objek lain terhadap histogram projection dan *Structuring Element* yang optimal untuk mengurangi *noise* dari objek lain terhadap tingkat akurasi untuk segmentasi plat nomor.

2. Pengukuran hasil segmentasi karakter.
3. Uji coba untuk mengetahui pengaruh *stop condition* yaitu MSE dan banyak data latih pada tiap kelas terkait dengan tingkat akurasi yang dihasilkan
4. Uji coba untuk mencari variabel *learning rate* terbaik untuk hasil akurasi terbesar.

3.5.1 Uji Coba Mengetahui Kombinasi Terbaik Perbesaran Kamera dengan SE

Pengujian akan dilakukan pada program untuk mengetahui perbesaran yang optimal untuk mendapatkan tingkat akurasi terbaik pada deteksi plat nomor dengan metode *vertical edge detection*. Deteksi plat nomor yang menggunakan metode berbasis pada deteksi tepi ini akan sangat dipengaruhi oleh kekomplekan tepi pada citra. Sehingga dilakukan uji coba untuk mengetahui perbesaran terbaik pada kasus pengambilan gambar yang telah ditentukan sebelumnya. Deteksi tepi juga menerapkan operasi morfologi untuk mengurangi *noise* dari tepi yang tidak dibutuhkan. Sehingga diuji juga kombinasi *SE* pada operasi morfologi pada deteksi plat nomor untuk mencari *SE* optimal pada kasus perbesaran kamera tertentu. Inputan parameter uji coba dapat dilihat pada Gambar 3.52. Tabel rancangan uji coba perbesaran dengan kombinasi ukuran SE dapat dilihat pada Tabel 3.9.

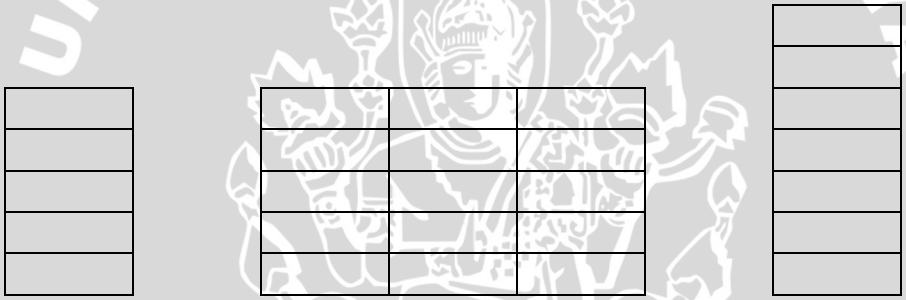


Gambar 3.52 Input Parameter Uji Coba Structuring Element

Tabel 3.9 Skenario Percobaan Perbesaran Pengambilan Gambar

	Perbesaran 2.1x			Perbesaran 3.4x			Perbesaran 4.7x			Perbesaran 5.7x		
Ukuran SE	1x5	3x5	1x7	1x5	3x5	1x7	1x5	3x5	1x7	1x5	3x5	1x7
Jumlah Data												
Data Benar												
Akurasi												

Perbesaran kamera yang akan diuji adalah 2.1x, 3.4x, 4.7x, dan 5.7x. Perbesaran yang diambil adalah perbesaran yang tersedia pada kamera dengan jarak minimal 1x karena jika jarak antar perbesaran terlalu kecil tidak terlalu menunjukan perubahan. Perbesaran yang terlalu besar akan sulit mengambil data dilapangan karena ruang tangkap kamera akan menjadi kecil sehingga perbesaran di batasi hingga perbesaran 5.7x. Selain perbesaran kamera parameter lain yang digunakan sebagai pengujian adalah ukuran SE. ukuran SE yang digunakan adalah persegi panjang dengan nilai ganjil. Ukuran SE yang dipilih 1x5, 3x5 dan 1x7. SE yang dipilih berupa persegi panjang vertikal untuk menghapus atau setidaknya mengurangi *noise* dan tepi objek lain yang berbentuk miring dan menyesuaikan dengan metode deteksi tepi vertikal yang digunakan. Gambar 3.53 adalah contoh ukuran SE yang digunakan sebagai bahan uji coba.



Gambar 3.53 Bentuk SE yang digunakan untuk Uji Coba

Tepi objek yang berbentuk miring ini akibat deteksi tepi yang dilakukan mencari posisi tepi lokal. Diharapkan dengan ukuran tersebut telah dapat diambil suatu kesimpulan mengenai ukuran SE yang optimal untuk segmentasi plat nomor.

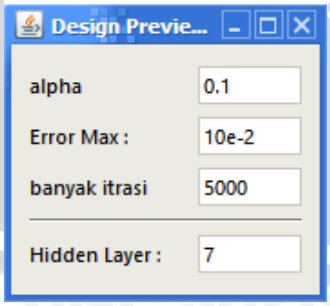
3.5.2 Uji Coba Pengaruh MSE dan Banyak Data Latih pada Akurasi

Pembelajaran pada jaringan saraf tiruan pada dasarnya adalah menentukan bobot pada arsitektur jaringan yang telah ditentukan dimana bobot yang ada dapat mengklasifikasikan antara kelas satu dengan kelas lainya berdasarkan target dan nilai yang dihasilkan oleh jaringan. Besarnya MSE yang digunakan sebagai *stop condition* pada jaringan menunjukan sejauh mana jaringan dapat memisahkan antar kelas yang dilatihkan. Pengujian ini dilakukan untuk mengetahui pengaruh MSE terhadap akurasi pengenalan data uji dan membandingkan dengan banyaknya *epoch* yang dibutuhkan untuk proses pelatihan.

Klasifikasi juga dipengaruhi oleh banyaknya data latih untuk mencirikan kelas yang ditujukan oleh data latih tersebut selain oleh metode klasifikasi yang digunakan. Banyaknya data latih akan mempengaruhi tingkat akurasi pada proses klasifikasi. Pengujian yang dilakukan ini juga menentukan pengaruh banyaknya data latih pada besar akurasi dan sejauh mana jaringan dapat mengklasifikan pola yang ada.

Besar MSE digunakan untuk pengujian ini adalah 10^{-1}, 10^{-2}, 10^{-3},10^{-4},dan 10^{5}. Besar MSE dinaikan secara bertahap dengan pengali 10^{-1} dengan harapan dapat menunjukan perubahan yang signifikan dan dirasa cukup untuk digunakan sebagai pengujian pada pengujian pengaruh MSE pada tingkat akurasi. Banyak data latih yang diujikan adalah 2, 3, 5 dan 7 yang masing masing data latih telah dapat menunjukan pola dari kelas yang diwakili.

Selain MSE sebagai variabel input pada *backpropagation neural network*, Arsitektur jaringan juga perlu dideskripsikan untuk skenario pengujian yang dilakukan. Banyak *node* pada *hidden layer* ditentukan sebelumnya dengan jumlah 70 *node*, karena jumlah tersebut memenuhi ketiga aturan penentuan jumlah *node* yang telah dijelaskan pada bab sebelumnya. Variabel lain yang perlu diinisialisasi sebelum proses pengujian adala besar *learning rate*. Untuk pengujian MSE, *learning rate* yang digunakan adalah 0.01 karena dengan nilai tersebut laju perubahan bobot dilakukan selambat mungkin hingga *stop conditon* terpenuhi. Gambar 3.54 adalah inputan parameter uji coba pada jaringan dan Tabel 3.10 adalah tabel uji coba kombinasi MSE dan *learning rate*.



Gambar 3.54 Input Parameter Uji Coba Pelatihan Jaringan

Tabel 3.10 Skenario Percobaan Pengaruh MSE dan Banyak Data Latih

		Data Latih			
		2	3	5	7
MSE 10^{-1}	Benar				
	Akurasi				
	Epoch				
MSE 10^{-2}	Benar				
	Akurasi				
	Epoch				
MSE 10^{-3}	Benar				
	Akurasi				
	Epoch				
MSE 10^{-4}	Benar				
	Akurasi				
	Epoch				
MSE 10^{-5}	Benar				
	Akurasi				
	Epoch				

3.5.3 Uji Coba Mencari Variabel *Learning rate* Terbaik

Learning rate adalah variabel yang mengatur laju perubahan bobot pada jaringan. Uji coba yang dilakukan berikutnya adalah dengan mencari nilai *learning rate* terbaik terkait pada akurasi yang dihasilkan. Pengujian akan dilakukan terhadap variabel yang dipilih pada pengujian sebelumnya yaitu penentuan nilai MSE dan jumlah data latih. Variabel yang akan diuji adalah 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.4 dan 0.45. Alasan variabel *learning rate* tersebut dipilih adalah untuk melihat perubahan akurasi yang dihasilkan sedikit demi sedikit dan tidak membuat MSE semakin kecil atau malah berlebihan karena *learning rate* berhubungan dengan kecepatan untuk mencapai *stop condition*. Tabel 3.11 adalah skenario uji coba yang dilakukan.

Tabel 3.11 Skenario Percobaan *Learning rate*

Learning rate	Akurasi (%)
0.05	
0.10	

0.15	
0.20	
0.25	
0.30	

BAB IV
IMPLEMENTASI

Bab ini akan membahas mengenai implementasi perangkat lunak yang dibangun untuk deteksi dan pengnalan plat nomor berdasarkan analisis data perancangan yang lakukan pada bab sebelumnya. Bab ini juga akan membahas mengenai spesifikasi dan lingkungan sistem, batasan sistem, implementasi algoritma dalam sistem dan implementasi antar muka.

4.1 Lingkungan Implementasi

Lingkungan implementasi yang akan dijelaskan dalama sub bab ini terkait dengan lingkungan implementasi perangkat lunak dan perangkat keras yang digunakan dalam mengimlementasikan sistem yang digunakan dalam penelitian ini.

4.1.1 Lingkungan Perangkat Keras

Perangkat keras yang digunakan dalam mengembangkan perangkat lunak dalam penelitian ini memiliki spesifikasi sebagai berikut :

1. Prosesor Intel(R) Pentium(R) Dual CPU T 2330@1.60GHz(2 CPUs)
2. Memori 2 GB
3. Harddisk 80 GB
4. Monitor 14”

4.1.2 Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan dalam mengembangkan sistem dan penelitian ini terdiri dari :

1. Sistem Operasi Windows XP 32 bit
2. JDK 7.45 32-bit
3. NetBeans IDE 7.3.1

4.2 Implementasi Program

Implementasi program akan menjelaskan implementasi prosedur-prosedur dari perancangan yang telah dirancang sebelumnya kedalam sistem yang mana menggunakan bahasa pemrograman JAVA. Secara garis besar sistem dibangun dengan 3 *package* utama yang masing masing *package* akan

mengimplementasikan tentang pengolahan citra, jaringan saraf tiruan menggunakan *Backpropagation* dan implementasi antar muka user. Penyimpanan data latih karakter yang akan digunakan adalah berupa file excel dan gambar dari data latih. Untuk menjelaskan rancangan umum sistem sebelumnya akan ditunjukan class diagram sebagai dasar implementasi ke sistem. Gambar 4.1 akan menjelaskan class diagram yang dibangun pada sistem.

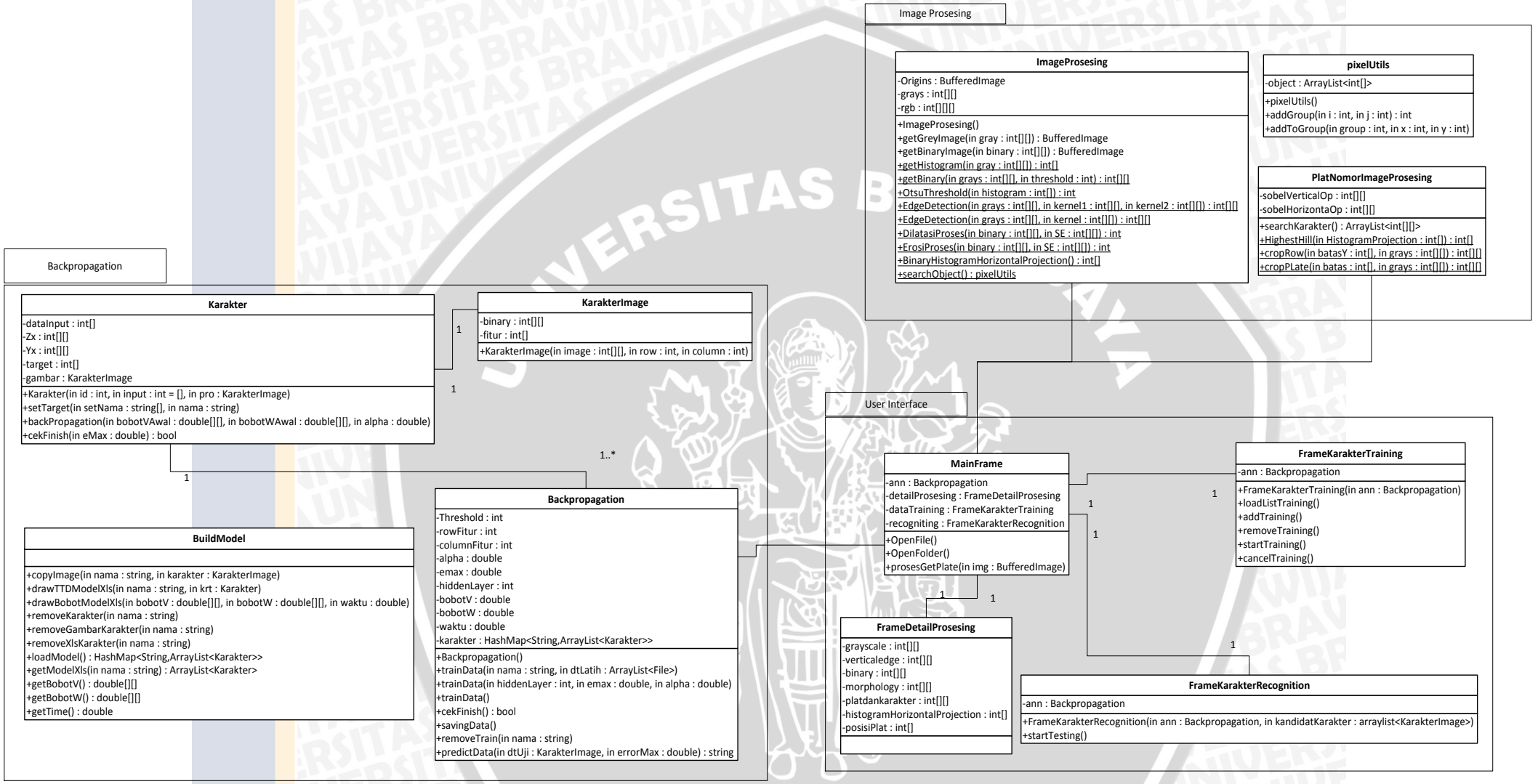
Gambar 4.1 juga menggambarkan attribut dan prosedur – prosedur penting yang ada pada sistem. Keterangan dan implementasi source code setiap prosedure pada setiap class akan dijelaskan pada subbab ini. Struktur data dan tipe data lain seperti File dan BufferedImage diambil dengan mengimport class dari JAVA.

4.2.1 Masukan Citra Mobil

Input sistem dapat berupa *single file dari* citra mobil maupun melakukan load citra bersama-sama pada satu folder untuk dilakukan seleksi langsung pada sistem deteksi plat nomor. Input citra mobil ini dilakukan untuk deteksi plat nomor dan melakukan segmentasi karakter pada plat nomor untuk dikenali atau *recogniting*. Pada proses ini juga citra berupa file akan diubah ke dalam BufferedImage untuk dapat membedakan antara file gambar dengan file lain. Implementasi fungsi ini akan dijelaskan seperti pada *Source code* 4.1 dan *Source code* 4.2 berikut,

```
int i=fileBrowserImage1.showOpenDialog(this);
 if(i==0){
   try {
      BufferedImage ori =ImageIO.read(fileBrowserImage1.
        getSelectedFile());
      prosesGetPlate(ori);
   } catch (IOException ex) {}
 }
```

***Source Code* 4.1 Fungsi Pembacaan *Single File* Citra Mobil**



Gambar 4.1 Class Diagram Sistem

```
int i=fileBrowser1.showOpenDialog(this);
if(i==0){
 new Thread(new Runnable() {
   public void run() {
      listItem = new Vector<listItemImage>();
         File[] data = fileBrowser1.getSelectedFile().listFiles();
         for (int j = 0; j < data.length; j++) {
           if(data[j].getName().endsWith("JPG"))
              try {
                listItem.add(new listItemImage(
                ImageIO.read(data[j]),(j+1)+"."));
              } catch (IOException ex) {}
         }
         listGambar.setListData(listItem);
    }
  }).start();
}
```

***Source Code* 4.2 Fungsi Pembacaan Satu Folder Citra Mobil**

Hasil pembacaan folder akan disajikan pada suatu list pada *interface* sehingga user dapat melakukan seleksi langsung pada sistem dan melakukan deteksi plat nomor dengan *event* klik dari list. *Source code* 4.3 akan menunjukan *event* klik dari list tersebut dan sebagai pemicu untuk proses berikutnya.

```
if(listItem!=null)
 prosesGetPlate(((listItemImage)listGambar.getSelectedValue())
   .getGambar());
```

***Source Code* 4.3 Fungsi *Event* dari List**

4.2.2 Deteksi Plat Nomor

Proses berikutnya dari sistem adalah melakukan deteksi plat nomor dari citra yang dipilih. Proses ini akan di-*handle* oleh sebuah *thread* yang telah dibuat untuk menjalankan suatu proses sehingga proses ini tidak akan mengganggu proses lain termasuk *user interface*. Prosedur deteksi plat nomor adalah salah satu bagian utama dari sistem dan akan memanfaatkan prosedur lain untuk pengolahan citra digital. Awal dari proses adalah menampilkan animasi *progress* pada *user interface*. Selanjutnya adalah melakukan operasi citra seperti yang dijelaskan pada Gambar 3.4 sebelumnya yaitu konversi citra RGB ke *grayscale*. *vertical edge detection*, binarisasi dengan metode *Otsu*, histogram projection dan penentuan plat dengan pemotongan horizontal dan vertikal. *Source code* 4.4 berikut akan menjelaskan implementasi dari proses deteksi plat nomor dari citra dengan *vertical edge detection*.

```
public void prosesGetPlate(BufferedImage o){
  final BufferedImage ori = o;
  if(as!=null)
    as.stop();
  as = new Thread(new Runnable() {
    public void run() {
      loading.setIcon(new  javax.swing.ImageIcon(getClass().
       getResource("/image/load.gif")));
      ImagePanelOri.setBackgroundImage(new ImageIcon(ori));
      Current = new FrameDetailProsesing();
      ImageProsesing oriProses = new ImageProsesing(ori);
      Current.grayscale = oriProses.Gray();
      Current.bluring = Current.grayscale;
      Current.verticaledge = ImageProsesing.EdgeDetection(
       Current.bluring,PlatNomorImageProsesing.sobelVerticalOp);
      Current.binary = ImageProsesing.getBinaryFromOtsuThreshold(
       Current.verticaledge);
      Current.morphology = ImageProsesing.ErosiProses(
       Current.binary, moConstant.matriks);
      Current.morphology = ImageProsesing.DilatasiProses(
       Current.morphology, moConstant.matriks);
      int[][] morphology = Current.morphology;
      Current.histogramHorizontalProjection = ImageProsesing
         .BinaryHistogramVerticalProjection(Current.morphology);
         //ambil kandidat posisi-Y
         int [] posisiY = PlatNomorImageProsesing.HighestHill(
           Current.histogramHorizontalProjection);
         Current.posisiPlat[0] = posisiY[0];
         Current.posisiPlat[1] = posisiY[1];
         morphology = PlatNomorImageProsesing
          .cropRow(posisiY, morphology);
         morphology = ImageProsesing.DilatasiProses(morphology,
           PlatNomorImageProsesing.SEMorphologicalFilterDilation);
         int [] morphologyProjection = ImageProsesing
          .BinaryHistogramVerticalProjection(morphology);
         int [][] posisiX = PlatNomorImageProsesing
          .ObjeckByDistance(morphology,morphologyProjection);
         int max = 0;
         for(int i=0;i<posisiX.length;i++){
           if(max<(posisiX[i][0]-posisiX[i][1])){
             max=(posisiX[i][0]-posisiX[i][1]);
             Current.posisiPlat[2] = posisiX[i][0];
             Current.posisiPlat[3] = posisiX[i][1];
           }
          }
          //segmentasi karater
         Current.morphology = PlatNomorImageProsesing
          .drawKandidatPlate(posisiY,posisiX,Current.morphology);
         int [][] plateGray = PlatNomorImageProsesing
          .cropPLate(Current.posisiPlat,Current.grayscale);
         platBiners = ImageProsesing
          .getBinaryFromOtsuThreshold(plateGray);
         karakterSegmentation = ImageProsesing
          .searchObject(platBiners);
         Current.platdankarakter = platBiners;
         int [][] karakter = PlatNomorImageProsesing
          .searchKarakter(platBiners,karakterSegmentation);
```

```
            imagePanelPlatNomor.setBackgroundImage(new ImageIcon(
              ImageProsesing.getGreyImage(plateGray)));
            imagePanelKarakter.setBackgroundImage(new ImageIcon(
              ImageProsesing.getBinaryImage(karakter)));
            loading.setIcon(new javax.swing.ImageIcon(
              getClass().getResource("/image/load.png")));
        }
    });
    as.start();
}
```

***Source Code* 4.4 Proses Fungsi Deteksi Plat Nomor pada Citra Mobil**

4.2.2.1 Konversi citra RGB ke *grayscale*

Konversi citra RGB ke citra *grayscale* dilakukan dengan mencari nilai rata-rata dari ketiga nilai penyusun citra RGB sesuai dengan persamaan 2.1. nilai *pixel* diekstrak dari BufferedImage yang merupakan struktur data dari java untuk merepresntasikan citra Menjadi array 2D agar lebih mudah untuk memanipulasi nilai *pixel*nya. *Source code* 4.5 berikut adalah implementasi proses konversi citra RGB ke *grayscale*.

```
Origins = buff;
grays = new int[Origins.getWidth()][Origins.getHeight()];
rgb = new int[3][Origins.getWidth()][Origins.getHeight()];
for (int i = 0; i < Origins.getWidth(); i++) {
  for (int j = 0; j < Origins.getHeight(); j++) {
    int c = Origins.getRGB(i, j);
    int r = (c & 0x00ff0000) >> 16;
    int g = (c & 0x0000ff00) >> 8;
    int b = c & 0x000000ff;
    grays[i][j] = (int) (((r) + (b) + (g)) / 3);
    rgb[0][i][j] = r;
    rgb[1][i][j] = g;
    rgb[2][i][j] = b;
  }
}
```

Source Code* 4.5 Proses Fungsi Konversi Citra RGB ke *Grayscale

4.2.2.2 *Vertikal Edge Detection*

Deteksi tepi dilakukan untuk mendapatkan tepi dari citra. Sedangkan pada deteksi plat nomor sendiri dilakukan untuk mencari posisi plat nomor yang dicirikan pada kepadatan tepi pada plat. Tepi pada plat didukung oleh adanya karakter pada plat sehingga menonjolkan ciri plat dengan tepi yang lebih banyak dari objek lain. Deteksi tepi vertikal ini bertujuan mengurangi tepi pada objek lain tatapi tetap akan membuat plat lebih menonjol karena tepi vertikal pada plat nomor lebih dominan. Deteksi tepi vertikal yang digunakan pada sistem ini

menggunakan sobel kernel vertikal. *Source code* 4.6 dibawah ini adalah implementasi deteksi tepi vertikal pada citra.

```
public static int[][] EdgeDetection(int value[][],
  float kernel1[][]) {
 int hasil[][] = new int[value.length][value[0].length];
 int jarak = (kernel1.length - (1)) / 2;
 for (int i = jarak; i < ((kernel1.length % 2 == 1) ?
    value.length - jarak : value.length - jarak - 1); i++) {
  for (int j = jarak; j < ((kernel1.length % 2 == 1) ?
    value[i].length - jarak : value[i].length - jarak - 1); j++) {
   float sum1 = 0;
   for (int x = -jarak; x <= ((kernel1.length % 2 == 1) ?
     jarak : jarak + 1); x++) {
    for (int y = -jarak; y <= ((kernel1.length % 2 == 1) ?
      jarak : jarak + 1); y++) {
     sum1 = sum1 + ((value[i + x][j + y] *
      kernel1[x + jarak][y + jarak]));
     }
   }
   if(sum1>255)
       hasil[i][j] =255;
     else if(sum1<0)
       hasil[i][j] =0;
     else
       hasil[i][j] = (int)sum1;
    }
  }
 return hasil;
}
```

***Source Code* 4.6 Proses Fungsi Deteksi Tepi Vertikal**

4.2.2.3 Binarisasi *Otsu*

Binarisasi pada deteksi plat nomor digunakan untuk mempertegas tepi citra hasil deteksi plat nomor. Binarisasi dengan metode *Otsu* akan menentukan nilai thershold berdasarkan histogram sehingga nilai threshold dapat ditentukan secara objektif. *Source code* 4.7 dibawah ini adalah hasil implementasi penentuan nilai threshold dengan metode *Otsu.*

```
public static int OtsuThreshold(int[][] value) {
   int [] histogram =  getHistogram(value);
   double variance =0, maxvariance = 0;
   double firstCm = 0, zeroCm = 0;
   double tmead = 0;
   double luas = (value.length*value[0].length);
   int thershold = 0;
   for(int i=0;i<histogram.length;i++){
      double lala = (i*histogram[i])/luas;
      tmead += lala;
   }
   for(int i=0;i<histogram.length;i++){
      zeroCm += (histogram[i]/luas);
      firstCm += ((i*histogram[i])/luas);
```

```
        variance = Math.pow(tmead * zeroCm - firstCm,2)/
                   (zeroCm*(1-zeroCm));
        if(maxvariance<variance){
           maxvariance = variance;
           thershold = i;
        }
     }
     return thershold;
}
```

Source Code* 4.7 Proses Fungsi Penentuan Nilai Threshold Dengan Metode *Otsu

4.2.2.4 *Morphological Operation*

Operasi morphologi yang dilakukan untuk menghilangkan *noise* pada citra hasil binarisasi. *Noise* berupa tepi yang terlalu kecil sehingga dapat untuk diabaikan dalam penentuan plat nomor. Operasi morphologi yang digunakan adalah *opening* dengan bentuk *Struktur element* persegi panjang vertikal dengan panjang dan lebar yang akan ditentukan pada proses pengujian. *Opening* dilakukan dengan mengkombinasi 2 operasi morphologi dasar yaitu erosi dan dilasi seperti yang dijelaskan pada bab sebelumnya. *Source code* 4.8 dan *source code* 4.9 masing masing adalah implementasi fungsi erosi dan dilasi dengan paramter berupa array 2D dari hasil binarisasi citra.

```
public static int[][] ErosiProses(int[][] value, float[][] kernel)
{
  int hasil[][] = new int [value.length][value[0].length];
  int jarak = (kernel.length - (1)) / 2;
  for (int i = 0; i <  value.length; i++) {
    for (int j = 0; j < value[i].length; j++) {
      boolean black = false;
      for (int x = -jarak; x <= ((kernel.length % 2 == 1) ?
        jarak : jarak + 1); x++) {
        for (int y = -jarak; y <= ((kernel.length % 2 == 1) ?
          jarak : jarak + 1); y++) {
          if(i + x<0||i + x>=value.length||
            j + y<0||j + y>=value[i].length)
            continue;
          if (kernel[x + jarak][y + jarak] == -1)
            continue;
          if (value[i + x][j + y] !=
            kernel[x + jarak][y + jarak]){
            black = true;
            break;
          }
        }
        if (black)
          break;
      }
```

```
        hasil[i][j] = ((black) ? 0 : 1);
      }
    }
  return hasil;
}
```

***Source Code* 4.8 Implementasi Fungsi Erosi pada Citra Biner**

```
public static int[][] DilatasiProses(int[][]value,float[][]kernel)
{
 int hasil[][] = new int [value.length][value[0].length];
 int jarak = (kernel.length - (1)) / 2;
 for (int i = 0; i <  value.length; i++) {
   for (int j = 0; j < value[i].length; j++) {
     boolean putih = false;
     for (int x = -jarak; x <= ((kernel.length % 2 == 1) ?
       jarak : jarak + 1); x++) {
       for (int y = -jarak; y <= ((kernel.length % 2 == 1) ?
         jarak : jarak + 1); y++) {
         if(i + x<0||i + x>=value.length||j + y<0||j +
           y>=value[i].length)
           continue;
         if (kernel[x + jarak][y + jarak] == -1)
           continue;
         if (value[i + x][j + y] ==
           kernel[x + jarak][y + jarak]) {
           putih = true;
           break;
         }
       }
       if (putih)
         break;
     }
     hasil[i][j]=((putih) ? 1 : 0);
   }
 }
 return hasil;
}
```

***Source Code* 4.9 Implementasi Fungsi Dilasi pada Citra Biner**

4.2.2.5 *Horizontal Histogram Projection*

Proyeksi histogram secara horizontal diharapkan dapat menunjukan posisi horizontal dari plat mobil. Proyeksi melakukan mengakumulasi nilai intensitas pada citra biner. Hasilnya adalah sebuah histogram hasil proyeksi. Pada metode yang diusulkan plat nomor diasumsikan berada pada bukit tertinggi yang ada pada histogram. Sehingga posisi horizontal diambil dari titik (y) pada awal bukit hingga titik (y) diakhir bukit. Proses masih akan dilakukan pada titik tertinggi pada bukit tersebut sehingga masing – masing titik akan disimpan dalam memori untuk proses ditahap berikutnya. Source code 4.10 adalah implementasi fungsi horizontal projection.

```
public static int [] BinaryHistogramHorizontalProjection (
  int [][]value){
  int hasil [] = new int [value.length];
  for(int i=0;i<value.length;i++){
    int sum = 0;
    for(int j=0;j<value[0].length;j++)
      if(value[i][j]==1)
        sum++;
      hasil[i] = sum;
  }
  return hasil;
}
```

***Source Code* 4.10 Implementasi Fungsi Horizontal Projection pada Citra Biner**

4.2.2.6 Pemotongan Horizontal

Tahap selanjutnya untuk pada metode yang diajukan adalah menentukan titik horizontal dan memotong citra yang dianggap terdapat plat nomor didalamnya. Seperti yang dijelaskan sebelumnya pemotongan citra diambil dengan mencari bukit tertinggi pada *histogram projection* yang dihasilkan pada tahap sebelumnya. Source code 4.11 adalah implementasi fungsi pencariaan titik potong histogram yang dimaksud sebelumnya.

```
public static int [] HighestHill(int HistogramProjection []){
  int panjangMax = 0;
  int posMax = 0;
  for (int i=0;i<HistogramProjection.length;i++){
    if (panjangMax < HistogramProjection[i]) {
      posMax=i;
      panjangMax = HistogramProjection[i];
    }
  }
  boolean posEnd = false;
  boolean posStart = false;
  int kesalahanEnd=15;
  int ends=posMax;
  int kesalahanStart=15;
  int starts = posMax;
  while(kesalahanEnd>0&&kesalahanStart>0&&
    ends>1&&starts<HistogramProjection.length-1){
    if(kesalahanEnd>0&&HistogramProjection[ends-1]<
      HistogramProjection[ends]){
      ends--;
    }else if(kesalahanEnd>0){
      kesalahanEnd--;
      ends--;
    }
    if(kesalahanStart>0&&
      HistogramProjection[starts+1]<HistogramProjection[starts]){
      starts++;
    }else if(kesalahanStart>0){
```

```
        starts++;
        kesalahanStart--;
      }
    }
   return new int[]{starts,ends};
}
```

Source Code* 4.11 Implementasi Penentuan Titik Potong Horizontal dari Image Berdasarkan *Histogram Projection

4.2.2.7 Pemotongan Vertikal

Pemotongan vertikal dilakukan dengan memisahkan objek berdasarkan kedekatan posisi titik putih pada suatu row yang memiliki puncak tertinggi pada *histogram projection*. Pada row yang diambil dari citra ditentukan titik awal dan titik akhir dari objek kemudian dipilih objek yang paling panjang untuk ditujukan sebagai citra yang mengandung plat nomor. Source code 4.12 adalah implementasi untuk penentuan titik vertikal yang dianggap sebagai plat nomor.

```
public static int [][] ObjeckByDistance (int [][] array,
   int HistogramProjection []){
   int [][] posisiX = new int [0][2];
   int posMax = 0;
   int max = 0;
   for (int i=0;i<HistogramProjection.length;i++){
     if(max<HistogramProjection[i]){
       max = HistogramProjection[i];
       posMax=i;
     }
   }
   max = 0;
   int jarakMax = 50;
   int start = 0;
   int end = 0;
   for(int i=0;i<array.length;i++){
     if(array[i][posMax]==1){
       start = i;
       end = i;
       int posSebelum = i;
       for(int j=i;j<array.length;j++){
         if(array[j][posMax]==1){
           if(j-posSebelum<=jarakMax){
             posSebelum = end;
             end=j;
           }
           else
            break;
         }
       }
       int [][] temp = new int [posisiX.length+1][2];
       for(int j=0;j<posisiX.length;j++){
         temp[j][0] = posisiX[j][0];
         temp[j][1] = posisiX[j][1];
```

```
        }
        temp[posisiX.length][0] = end;
        temp[posisiX.length][1] = start;
        posisiX = temp;
        i=end;
      }
    }
   return posisiX;
}
```

***Source Code* 4.12 Implementasi Penentuan Titik Potong Vertikal dari Image**

4.2.3 Segmentasi Karakter

Setelah plat nomor dideteksi, maka tahap berikutnya adalah melakukan segmentasi objek dan seleksi karakter. Segmentasi karakter dilakukan dengan mencari *connected component* dengan aturan 8-*Connectivity* dan seleksi rasio dari objek yang didapatkan. Rasio yang dimaksud adalah malakukan pemilihan objek yang terdeteksi dengan panjang tinggi yang lebih besar dari lebarnya karena karakter pada plat memiliki tinggi lebih panjang dari lebarnya. Source code 4.13 adalah fungsi untuk mencari titik sudut objek (berwarna putih) dengan *connected component.*

```
public static pixelUtils searchObject(int value[][]) {
  pixelUtils util = new pixelUtils();
  boolean cek[][] = new boolean[value.length][];
  for (int j = 0; j < value.length; j++) {
    cek[j] = new boolean [value[j].length];
    for (int i = 0; i < value[j].length; i++) {
      cek[j][i] = false;
    }
  }
  for (int j = 0; j < value.length; j++) {
    for (int i = 0; i < value[j].length; i++) {
      if (!cek[j][i]) {
        if (value[j][i] != 0) {
          int group = util.addGroup(j, i);
          addToMember(value,cek,util, group, j, i);
        }
      }
    }
  }
  return util;
}

protected static void addToMember(int [][] value, boolean[][] cek,
  pixelUtils util, int group, int i, int j) {
  Stack<Integer> st = new Stack<Integer>();
  st.push(i + (j * value.length));
  while (!st.empty()) {
    int posisi = st.peek();
    int xi = posisi % value.length;
    int xj = (posisi - xi) / value.length;
```

```
cek[xi][xj] = true;
//tengah kanan
if (xi + 1 < value.length) {
  if (value[xi + 1][xj] != 0 && !cek[xi + 1][xj]) {
    xi++;
    st.push(xi + (xj * value.length));
    util.addToGroup(group, xi, xj);
    continue;
  }
}
//tengah kiri
if (xi - 1 >= 0) {
  if (value[xi - 1][xj] != 0 && !cek[xi - 1][xj]) {
    xi--;
    st.push(xi + (xj * value.length));
    util.addToGroup(group, xi, xj);
    continue;
  }
}
//bawah kanan
if (xi + 1 < value.length && xj + 1 < value[xj].length) {
  if (value[xi + 1][xj + 1] != 0 && !cek[xi + 1][xj + 1]) {
    xi++;
    xj++;
    st.push(xi + (xj * value.length));
    util.addToGroup(group, xi, xj);
    continue;
  }
}
//tengah bawah
if (xj + 1 < value[xj].length) {
  if (value[xi][xj + 1] != 0 && !cek[xi][xj + 1]) {
    xj++;
    st.push(xi + (xj * value.length));
    util.addToGroup(group, xi, xj);
    continue;
  }
}
//kiri bawah
if (xi - 1 >= 0 && xj + 1 < value[xj].length) {
  if (value[xi - 1][xj + 1] != 0 && !cek[xi - 1][xj + 1]) {
    xi--;
    xj++;
    st.push(xi + (xj * value.length));
    util.addToGroup(group, xi, xj);
    continue;
  }
}
//kiri atas
if (xi - 1 >= 0 && xj - 1 >= 0) {
  if (value[xi - 1][xj - 1] != 0 && !cek[xi - 1][xj - 1]) {
    xi--;
    xj--;
    st.push(xi + (xj * value.length));
    util.addToGroup(group, xi, xj);
    continue;
  }
```

```
    }
    //tengah atas
    if (xj - 1 >= 0) {
      if (value[xi][xj - 1] != 0 && !cek[xi][xj - 1]) {
        xj--;
        st.push(xi + (xj * value.length));
        util.addToGroup(group, xi, xj);
        continue;
      }
    }
    //kanan atas
    if (xi + 1 < value.length && xj - 1 >= 0) {
      if (value[xi + 1][xj - 1] != 0 && !cek[xi + 1][xj - 1]) {
        xi++;
        xj--;
        st.push(xi + (xj * value.length));
        util.addToGroup(group, xi, xj);
        continue;
      }
    }
    st.pop();
    }
  }
```

Source Code* 4.13 Implementasi Fungsi Mencari *Connected Component

Pencarian karakter kemudian dilanjutkan dengan menseleksi objek hasil segmentasi berdasarakan rasio objek tersebut. Dari *source code* diatas objek hasil segmentasi disimpan pada struktur data *pixel*Util yang dibuat kemudian untuk implementasi seleksi ditunjukan pada *source code* 4.14.

```
public static ArrayList<int [][]> searchKarakter(pixelUtils
  util,int [][]value){
  ArrayList<int [][]> karakter = new ArrayList<int [][]>();
  for (int gr = 0; gr < util.getGroup().size(); gr++) {
    int[] batas = util.getObject(gr);
    int y1=0,y2=0;
    boolean awal=false;
    float panjang = batas[1]-batas[0];
    float lebar = batas[3]-batas[2];
    if(lebar/panjang>batasBawahKarakter){
      if(!awal){
        awal=true;
        y1=batas[3];
        y2=batas[2];
      }
    if((y1>=batas[2]&&y2<=batas[2])||(y1>=batas[3]&&
      y2<=batas[3])||(batas[3]>=y1&&batas[2]<=y1)||
      (batas[3]>=y2&&batas[2]<=y2)){
      int subImage [][]= new int[batas[1] - batas[0]]
        [batas[3] - batas[2]];
      if(batas[3]-batas[2]<=0||batas[1]-batas[0]<=0)
        continue;
      for (int p = 0; p < subImage.length; p++) {
        for (int q = 0; q < subImage[p].length; q++) {
          subImage[p][q] = value[p+batas[0]][q + batas[2]];
```

```
            }
          }
          karakter.add(subImage);
        }
      }
    }
    return karakter;
}
```

***Source Code* 4.14 Implementasi Seleksi Karakter dari Objek**

4.2.4 Pengenalan karakter

Setelah didapatkan karakter dari hasil segmentasi karakter. Tahap berikutnya adalah pengenalan karakter dengan *backpropagation neural network.* Secara mendasar ada 2 proses pada suatu metode klasifikasi yaitu pelatihan dan pengujian. Selayaknya metode klasifikasi *backpropagation neural network* juga menerapkan 2 tahap dasar tersebut.

4.2.4.1 Tahap Pelatihan

Tahap pelatihan digunakan untuk mencari bobot pada jaringan yang menghasilkan *node* pada layer output sesuai atau mendekati target. Tahap pelatihan dimulai dengan melakukan inisialisasi bobot. Inisialisasi bobot dilakukan dengan menerapkan metode *Nguyen Widrow*. *Source code* 4.15 adalah implementasi inisialisasi bobot dengan bilangan random -0.5 sampai 0.5.

```
double [][] bobotVAwal = new double[(int)(rowFitur*columnFitur)+1]
   [(int)hiddenLayer];
Random acak = new Random();
for(int i=0;i<bobotVAwal.length;i++)
   for(int j=0;j<bobotVAwal[0].length;j++)
      bobotVAwal[i][j]=Math.random()-0.5;
String bit = Integer.toBinaryString(karakter.size());
double   [][]   bobotWAwal   =   new   double[(int)hiddenLayer+1]
[bit.length()];
   for(int i=0;i<bobotWAwal.length;i++)
     for(int j=0;j<bobotWAwal[0].length;j++)
       bobotWAwal[i][j]=Math.random()-0.5;
//setBobot
bobotV = bobotVAwal;
bobotW = bobotWAwal;
```

***Source Code* 4.15 Implementasi Inisialisasi Bobot**

Bobot hasil inisialisasi kemudian dibawah ke tahap berikutnya untuk dilatihkan pada jaringan. Bobot akan diperbaiki pada masing masing pelatihan data pada jaringan dan akan diulang kembali hingga masing masing data memiliki hasil keluaran yang sama atau mendekati dengan target. *Source code* 4.16 akan menunjukan proses perulangan yang diujikan pada masing masing data.

```
int i=0;
do{
  for(String ttds : karakter.keySet()){
    for(Karakter ttd : karakter.get(ttds)){
      ttd.backPropagation(bobotV, bobotW, alpha);
    }
  }
  i++;
}while(cekFinish()||i<maxIter);
```

***Source Code* 4.16 Implementasi Perulangan pada Tiap Tiap Data Latih**

Sedangkan implementasi untuk tahap perambatan maju dan perambatan mundur dilakukan pada masing masing data latih. *source code* 4.17 adalah implementasi fungsi perambatan maju dan perambatan mundur pada data latih.

```
public double[][] backPropagation(double[][] bobotVAwal,
 double[][] bobotWAwal, double alpha) {
 //mulai backpropagation
 //langkah maju -> hitung Zi
 double [][] nilaiZ = new double [Backpropagation.hiddenLayer][2];
 for(int i=0;i<nilaiZ.length;i++){
   double jumlah = 0;
   for(int j=1;j<=dataInput.length;j++)
   jumlah = jumlah +(bobotVAwal[j][i]*dataInput[j-1]);
   jumlah+=bobotVAwal[0][i];
   nilaiZ[i][0] = jumlah;
   nilaiZ[i][1] = 1/(1+Math.exp(-1*jumlah));
 }
 Zx=nilaiZ;
 //langkah maju -> hitung yi
 double nilaiY [][]= new double[target.length][2];
 for(int i=0;i<nilaiY.length;i++){
   double jumlah = 0;
   for(int j=1;j<=nilaiZ.length;j++)
   jumlah = jumlah +(bobotWAwal[j][i]*nilaiZ[j-1][1]);
   jumlah+=bobotWAwal[0][i];
   nilaiY[i][0] = jumlah;
   nilaiY[i][1] = 1/(1+Math.exp(-1*jumlah));
 }
 Yx = nilaiY;
 //langkah mundur -> bobot w
 double deltaW1[] = new double [target.length];
 for(int i=0;i<nilaiY.length;i++)
   deltaW1[i] = (target[i]-nilaiY[i][1])* nilaiY[i][1]*
     (1-nilaiY[i][1]);
 double [][] bobotWbaru =
   new double[bobotWAwal.length][bobotWAwal[0].length];
 for(int i=0;i<bobotWbaru[0].length;i++)
   bobotWbaru[0][i] = bobotWAwal[0][i]+(alpha*deltaW1[i]*1);
 for(int i=1;i<bobotWbaru.length;i++)
   for(int j=0;j<bobotWbaru[0].length;j++)
     bobotWbaru[i][j] = bobotWAwal[i][j]+
      (alpha*deltaW1[j]*nilaiZ[i-1][1]);
 //langkah mundur -> bobot V
 double []deltaV1 = new double [Backpropagation.hiddenLayer];
 for(int i=0;i<deltaV1.length;i++){
```

```
    deltaV1[i] = 0;
    for(int j=0;j<target.length;j++)
      deltaV1[i]+=(deltaW1[j]*bobotWAwal[i+1][j]);
    deltaV1[i]=deltaV1[i]*nilaiZ[i][1]*(1-nilaiZ[i][1]);
  }
  double bobotVbaru [][] = new
    double[dataInput.length+1][Backpropagation.hiddenLayer];
  for(int i=0;i<bobotVbaru[0].length;i++)
    bobotVbaru[0][i] = bobotVAwal[0][i]+ (alpha*deltaV1[i]);
    for(int i=1;i<bobotVbaru.length;i++)
      for(int j=0;j<bobotVbaru[0].length;j++)
    bobotVbaru[i][j] = bobotVAwal[i][j]+
      (alpha*deltaV1[j]*dataInput[i-1]);
    //setBobotW
    for(int i=0;i<bobotWbaru.length;i++)
      for(int j=0;j<bobotWbaru[0].length;j++)
        bobotWAwal[i][j]=bobotWbaru[i][j];
    //setBobotV
    for(int i=0;i<bobotVbaru.length;i++)
      for(int j=0;j<bobotVbaru[0].length;j++)
        bobotVAwal[i][j]=bobotVbaru[i][j];
    return nilaiY;
}
```

***Source Code* 4.17 Implementasi Perambatan Maju dan Perambatan Mundur Untuk Perbaikan Bobot**

4.2.4.2 Tahap Pengujian

Tahap pengujian digunakan untuk menentukan atau mengklasifikasi data uji berdasarkan kemiripan *output* hasil proses perambatan maju pada jaringan. Hasil dari setiap *node* pada layer *output* kemudian dibandingkan dengan mencari jarak terhadap nilai *output* dari data latih. Penentuan klasifikasi akan ditentukan dengan mencari jarak terkecil pada data latih yang ada. Source code 4.18 adalah implementasi klasifikasi karakter dengan menggunakan *backpropagation neural network.*

```
public String predictData(KarakterImage ss,double error){
    double min=1000;
    String hasil = "";
    double nilaiZ[][] = compareToSearchNilaiZ(bobotV, ss.fitur);
    double nilaiY[] = compareToSearchNilaiY(bobotW, nilaiZ);
    for(String ttd : karakter.keySet()){
       double mean = 1000;
       for(Karakter a : karakter.get(ttd)){
         double meanJarak =0;
         for(int i=0;i<nilaiY.length;i++){
           double jarak = Math.pow(nilaiY[i]-a.Yx[i][1],2);
           meanJarak+=jarak;
         }
         meanJarak = Math.sqrt(meanJarak);
         if(mean>meanJarak)
           mean=meanJarak;
```

```
            }
            if(min>mean){
                min=mean;
                hasil=ttd;
            }
        }
        if(min<=error)
            return hasil;
        return null;
    }
```

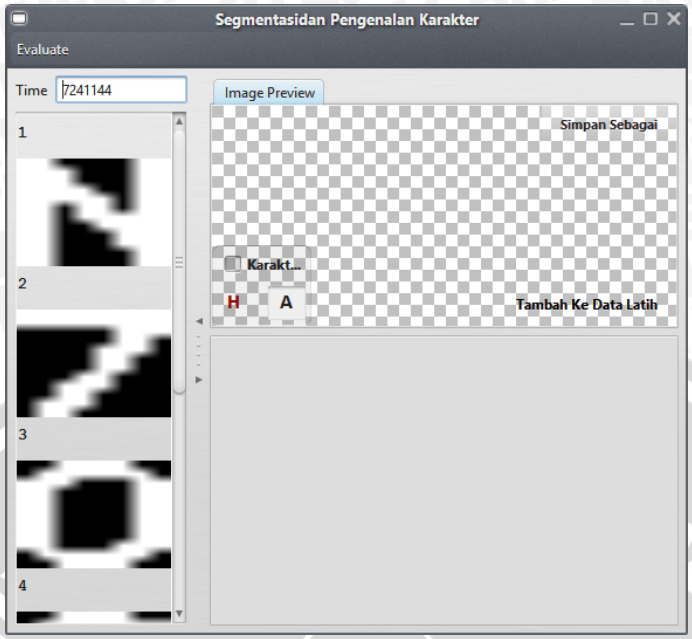
Source Code* 4.18 Implementasi Klasifikasi Karakter dengan *Backpropagation Neural Network

4.3 Implementasi Antarmuka

Antar muka sistem terdiri dari 3 *frame* utama. *Frame* pertama adalah tampilan utama untuk seleksi dan hasil deteksi plat nomor. *Frame* kedua adalah tampilan untuk manajemen data training dengan menggunakan algoritma *backpropagation neural network* Dan frame ke tiga adalah tampilan hasil recognition. Fungsi masukan citra mobil baik berupa *single file* maupun satu folder dilakukan di *frame* pertama. *Frame* pertama juga sebagai frame utama untuk menjebatani user masuk ke frame kedua dan ketiga juga ke menu lainya. Gambar 4.2 adalah tampilan frame utama sistem.

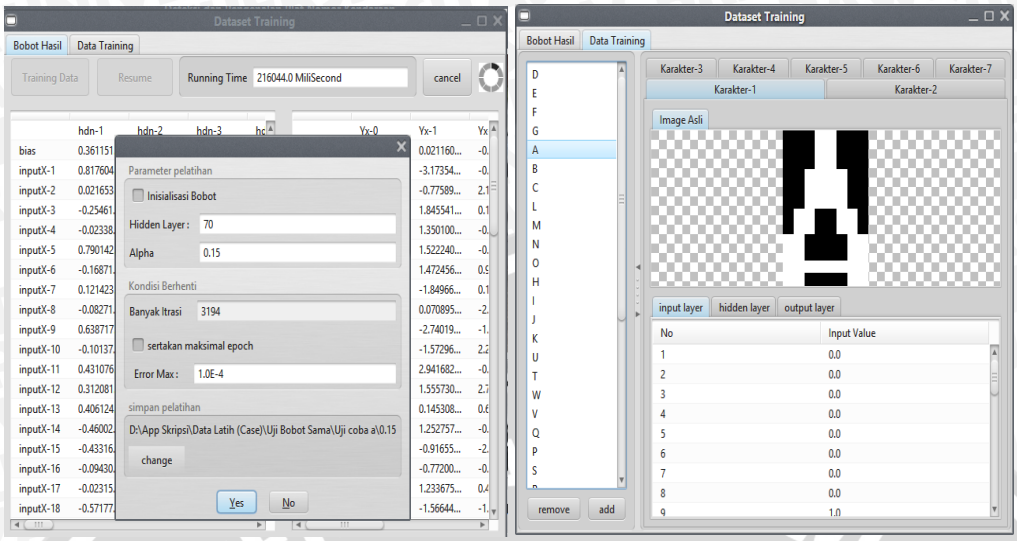
Gambar 4.2 Tampilan *Frame* Utama dari Sistem

Gambar 4.2 pada bagian kiri menunjukan bagian seleksi dari gambar saat membuka file folder. Seleksi dipicu dengan aksi klik dari user diarea bagian kiri dan akan memproses untuk deteksi plat nomor dan segmentasi karakter di bagian kanan pada *frame* utama tersebut. Setelah segmentasi berhasil, user dapat memproses ke tahap pengenalan dengan aksi klik ke button *recogniting* dibagian kanan bawah. Kemudian menu pengenalan pola akan muncul seperti pada Gambar 4.3 berikut.



Gambar 4.3 Tampilan *Frame Recognition* dari Sistem

Seperti pada frame utama, pada bagian kiri menu recognition merupakan seleksi karakter yang didapatkan dari segmentasi karakter. Kemudian pada bagian kanan atas adalah hasil ekstraksi fitur dan bagian kanan bawah adalah detail perhitungan setelah di masukan ke dalam jaringan dan hasil perbandingan dengan data latih yang ada. Pengenalan diambil dari nilai jarak terkecil dari hasil perhitungan. Kembali ke frame utama, Frame utama juga sebagai media menuju ke frame lainya. Frame yang dimaksud adalah menagement data latih dan aksi untuk melakukan pelatihan. Ada dua panel yang disediakan pada frane pelatihan. Yang pertama adalah tampilan bobot jaringan dan kedua adalah tampilan management data latih dan nilai *output* hasil pelatihan di jaringan. Gambar 4.4 adalah tampilan adalah tampilan management data latihnya dan fungsi pelatihannya.



Gambar 4.4 Tampilan Management Data Latih

Panel pop-up pada Gambar 4.4 adalah panel yang muncul saat tombol pelatihan di klik. Panel tersebut akan meminta parameter yang digunakan saat proses pelatihan.

BAB V
PENGUJIAN DAN ANALISIS

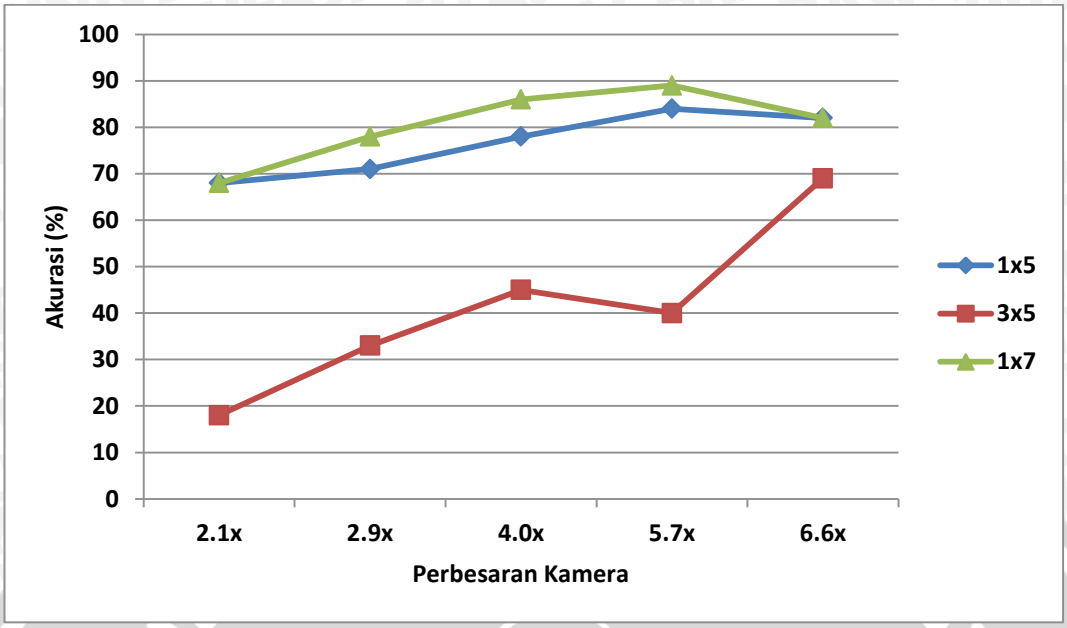
Bab ini mambahas tentang tahapan pengujian dan analisis hasil sistem deteksi dan pengenalan plat nomor dengan *vertical edge detection* dan *backpropagation neural network*. Pengujian yang akan dilakukan dengan mengevaluasi hasil uji coba sesuai dengan perancangan uji coba sebelumnya dan mengevaluasi akurasi deteksi dan pengenalan plat nomor *vertical edge detection* dan *backpropagation neural network*.

5.1 Hasil dan Analisis Uji Coba Kombinasi Perbesaran Kamera Dengan SE

Hasil dari deteksi plat nomor adalah pemotongan bagian citra yang mengandung plat nomor. Uji coba ini dilakukan untuk menemukan kombinasi perbesaran kamera dengan *Structuring Element* (SE) pada operasi *opening* yang optimal. Hasilnya akan dibandingkan dengan pengamatan langsung oleh mata manusia apakah benar atau tidak untuk dihitung tingkat akurasinya. Semakin kecil perbesaran kamera akan membuat ukuran plat pada citra 640x480 *pixel* menjadi semakin kecil dan membuat objek lain semakin banyak. Sedangkan SE merupakan variabel yang digunakan pada operasi morfologi untuk deteksi plat nomor. Variasi ukuran SE untuk menemukan ukuran terbaik pada uji coba variasi kamera yang dilakukan. Tabel 5.1 adalah hasil uji coba yang dilakukan terhadap data yang diambil dengan rata-rata waktu eksekusi 0.887 second.

Tabel 5.1 Hasil Uji Coba SE pada akurasi deteksi plat nomor

	Perbesaran 2.1x			Perbesaran 3.4x			Perbesaran 4.7x			Perbesaran 5.7x			Perbesaran 6.6x		
Ukuran SE	1x5	3x5	1x7	1x5	3x5	1x7	1x5	3x5	1x7	1x5	3x5	1x7	1x5	3x5	1x7
Jumlah Data	38	38	38	42	42	42	73	73	73	59	59	59	56	56	56
Data Benar	26	7	26	30	14	33	57	33	63	50	24	53	46	39	46
Akurasi (%)	68	18	68	71	33	78	78	45	86	84	40	89	82	69	82

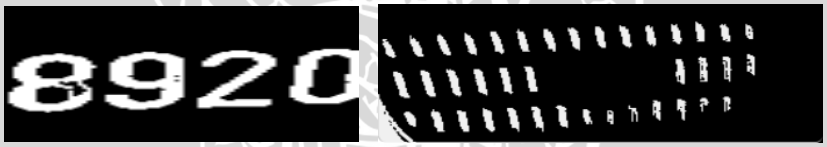


Gambar 5.1 Grafik Uji Coba Kombinasi Perbesaran Kamera Dengan *Structuring Element*

Gambar 5.1 adalah grafik representasi akurasi deteksi plat nomor dengan metode *vertical edge detection* dan *histogram projection* terhadap uji coba kombinasi perbesaran kamera dengan ukuran *structuring element.* Gambar 5.1 diatas terlihat bahwa perbesaran terbaik ditunjukan pada perbesaran 5.7x dan terburuk adalah 2.1x. Deteksi plat nomor yang berbasis pada deteksi tepi memang sangat dipengaruhi oleh *edge* dari objek lain yang dapat menggaggu analisis *histogram projection*. Hal ini yang mempengaruhi akurasi pada uji coba perbesaran kamera. Semakin kecil perbesaran pada kamera maka objek lain juga semakin banyak termuat pada gambar dan ukuran plat juga akan terlihat semakin kecil. Hal ini akan membuat akurasi menurun seperti yang ditunjukan pada grafik Gambar 5.1. Perbesaran yang terlalu kecil bisa jadi memunculkan sekumpulan objek lain yang memiliki ciri yang sama seperti plat nomor yang dicari. Contohnya seperti warna hitam putih pada trotoar jalan. Semakin besar perbesaran yang diberikan pada kamera maka penampakan objek lain selain plat nomor akan terkurangi. Akibatnya akurasi untuk mendapatkan posisi plat akan meningkat, seperti peningkatan yang ditunjukan pada perbesaran 2.1x sampai 5.7x. Sedangkan jika perbesaran ditingkatkan berlebihan maka akan menyulitkan pengambilan gambar di lapangan, karena ruang tangkap gambar pada kamera

akan semakin kecil. Hal ini juga dipersulit pada posisi berhenti mobil yang sering tidak pada posisi yang diinginkan saat pengembilan gambar, bisa jadi berhenti pada posisi berlebihan atau berhenti sebelum menempati posisi pengambilan gambar.

Kesulitan pada deteksi plat nomor pada perbesaran kamera yang berlebihan adalah munculnya ruang kosong antara kode wilayah dan nomor registrasi terlalu jauh sehingga sistem membaca masing masing sebagai 1 objek yang terpisah seperti yang ditunjukan pada akurasi perbesaran 6.6x. Hal ini akan menunjukan hasil deteksi plat secara tidak penuh (hanya terpotong sebagian pada plat). Selain hal tersebut perbesaran pada kamera yang berlebihan juga dapat menghasilkan kesalahan pada posisi lain yang juga menjadi lebih besar seiring perbesaran pada kamera dan posisi fokus kamera, seperti lampu mobil atau bemper mobil yang dipenuhi oleh tepi. Gambar 5.2 adalah contoh deteksi plat nomor yang salah. Gambar 5.2 sebelah kiri adalah contoh pemotongan plat salah akibat ruang kosong yang terlalu lebar antara kode wilayah dan nomor registrasi, sedangkan sebelah kanan adalah kesalahan deteksi plat nomor akibat objek lain yang memiliki tepi lebih menonjol dari pada plat (bemper).



Gambar 5.2 Contoh Pemotongan Plat yang Salah

Penelitian ini juga menerapkan operasi morfologi yaitu opening untuk mengurangi *noise* dengan mempertahankan bentuk asli dari objek. Perbesaran juga akan sangat mempengaruhi pada operasi morfologi yang dilakukan. Semakin kecil perbesaran maka objek yang dicari yaitu plat akan terlihat semakin kecil karena ukuran citra tetap yaitu 640x480 *pixel*. Opening akan mengurangi *noise* tergantung pada SE yang ditentukan atau bisa jadi akan mengurangi bagian pada objek yang dicari jika bagian pada objek tersebut lebih kecil dari ukuran SE membuat objek tidak terlalu menonjol pada *histogram projection*. Akibatnya bisa jadi kesalahan pada deteksi plat saat di aplikasikan. Sehingga perlu dilakukan uji coba untuk menentukan ukuran SE yang terbaik untuk diimplementasikan.

Gambar 5.1 menunjukan hasil kurasi percobaan ukuran SE. akurasi terbaik ditunjukan pada ukuran 7x1 dan terburuk adalah 3x5. Hal ini menunjukan bahwa lebar tepi vertikal karakter hasil deteksi tepi kebanyakan kurang dari lebar SE yang diujikan, sehingga untuk tetap mempertahankan fitur karakter ukuran lebar 1 *pixel* pada SE-lah yang terbaik untuk diimplementasikan pada hasil deteksi tepi vertikal. Sedangkan untuk tinggi tepi objek lain dapat diminimalisasi dengan ukuran SE yang cukup tinggi yaitu 7, hal ini menunjukan bahwa objek lain yang dihilangkan tidak sepenuhnya berbantuk vertikal melainkan melengkung sehingga dengan ukuran SE 7x1 dapat memberikan hasil terbaik untuk mengurangi *noise* hasil deteksi tepi. Deteksi tepi vertikal dengan melibatkan suatu *mask* akan menghasilkan tepi secara lokal yang ditandai jarak intensitasnya tergantung pada ukuran *mask* yang digunakan. Sehingga pada deteksi tepi vertikal yang digunakan memungkinkan terdapat tepi diagonal yang dapat dihilangkan atau meminimalkan dengan ukuran SE vertikal tertentu, pada hasil pengujian ini didapatkan ukuran SE terbaik yaitu 7x1.

5.2 Hasil dan Analisis Segmentasi Karakter

Metode Segmentasi yang digunakan berbasis pada binerisasi dengan metode *Otsu* dan analisis pada *connected component*. Kekurangan dari metode ini akan terbatas pada intensitas tertentu. Karakter dengan intensitas warna rendah tidak akan tersegmentasi. Contohnya pada plat nomor berwarna kuning dan putih yang karakternya berwarna putih. Kelebihan dengan binerisasi ini dapat mensegmentasi bentuk karakter sesuai dengan area warna penyusun karakter tersebut. Sehingga meminimalkan penggabungan dua atau lebih objek yang berdekatan tetapi tidak sama. Jika dibandingkan dengan deteksi tepi biasa bentuk karakter yang tersegmentasi akan berada pada tepi bagian luar dari karakter. Sehingga memungkinkan penggabungan antara objek yang berdekatan.

Pengujian yang dilakukan ini untuk mecari nilai *f-measure* metode yang digunakan untuk segmentasi karakter. Uji coba dilakukan untuk semua data yang telah diambil dilapangan dengan variabel SE terbaik yang dilakukan pada pengujian sebelumnya yaitu 7x1 dan data yang berhasil dideteksi plat nomornya.

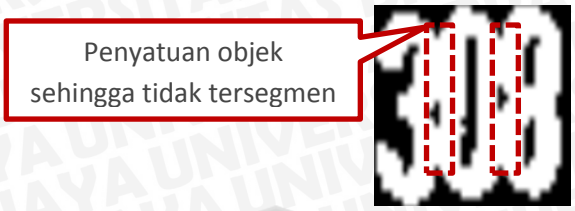
Tabel 5.2 di bawah ini adalah hasil pengukuran *f-measure* hasil segmentasi lebih detail dapat dilihat pada lampiran.

Tabel 5.2 Hasil Uji Segmentasi Karakter

Perbesaran	Jumlah Karakter	Jumlah Karakter Tersegmentasi	Jumlah Selain Karakter Tersegmentasi	F-Measure
2.1x	147	129	16	0.883
3.4x	185	167	23	0.89
4.7x	431	335	38	0.833
5.7x	325	312	31	0.934
6.6x	253	167	183	0.766

Tabel 5.2 diatas adalah hasil pengukuran keberhasilan segmentasi karakter yang diterapkan. Hasil pengukuran dilakukan secara global dengan jumlah karakter dari seluruh plat yang terdeteksi dan mengukur *f-measure* secara keseluruhan. Hasil segmentasi yang didapatkan pada Tabel 5.2 menunjukan hampir semua hasil *f-measure* memiliki besar yang hampir sama yaitu sekitar 0.85 dengan kesalahan yang sama pula kecuali pada perbesaran 6.6x. Penurunan nilai *f-measure* pada segmentasi karakter dengan perbesaran kamera 6.6x disebabkan efek blur pada citra akibat pergerakan mobil saat pengambilan. Efek blur pada citra membuat tepi objek pada citra yang harusnya terpisah menjadi menyatu, sehingga saat filtrasi objek yang seharusnya karakter tidak tersaring sebagai karakter.

Hasil segmentasi karakter dengan binarisasi *Otsu* jika ditinjau dari sudut presisinya. Kegagalan dalam mensegmentasi karakter terjadi akibat adanya karakter yang terhubung dengan karakter lain atau dengan frame plat nomornya. Sehingga dengan metode *connected component* akan dianggap 1 objek yang tidak dapat dikatakan karakter karena tinggi objek tidak lebih panjang dari lebarnya. Gambar 5.3 dibawah ini adalah contoh penggabungan karakter.



Gambar 5.3 Contoh Penggabungan Karakter

Nilai *f-measure* juga dipengaruhi oleh banyaknya kesesuaian karakter yang didapatkan atau biasa disebut *recall*. Nilai *recall* sendiri dipengaruhi oleh banyaknya objek lain yang dianggap sebagai karakter pada program. Pada kasus segmentasi ini, kebanyakan objek lain yang terdeteksi adalah bagian frame pada plat nomor dan bagian tepi dari bemper mobil jika segmentasi plat terlalu panjang. Gambar 5.4 berikut adalah contoh objek lain yang tersegmen selain plat nomor.

Gambar 5.4 Contoh Objek Lain Yang Ikut Tersegmentasi

5.3 Hasil dan Analisis Uji Coba MSE dan Jumlah Data Latih

Uji coba jumlah data latih dan MSE terkait metode yang digunakan ini dilakukan untuk mengukur tingkat akurasi *backpropagation neural network*. Variabel jumlah data latih yang digunakan pada uji coba dilakukan untuk mengukur sejauh mana metode yang digunakan dapat memisahkan pola antar kelas. Sedangkan variabel MSE sebagai *stop condition* pada saat pembelajaran jaringan di uji untuk mencari pengaruh MSE pada tingkat akurasi dan menentukan nilai MSE terbaik untuk pelatihan. Uji coba dilakukan pada variabel perbesaran dan ukuran SE pada deteksi plat nomor dengan akurasi terbaik yaitu pada perbesaran kamera 5.7x dan dengan MSE 1x7. Pengujian juga hanya dilakukan pada hasil segmentasi karakter yang tidak cacat, maksudnya tidak terpotong sehingga masih dapat diartikan. Dari analisis segementasi didapatkan banyak karakter yang akan diuji pada pengujian ini adalah 309 karakter.

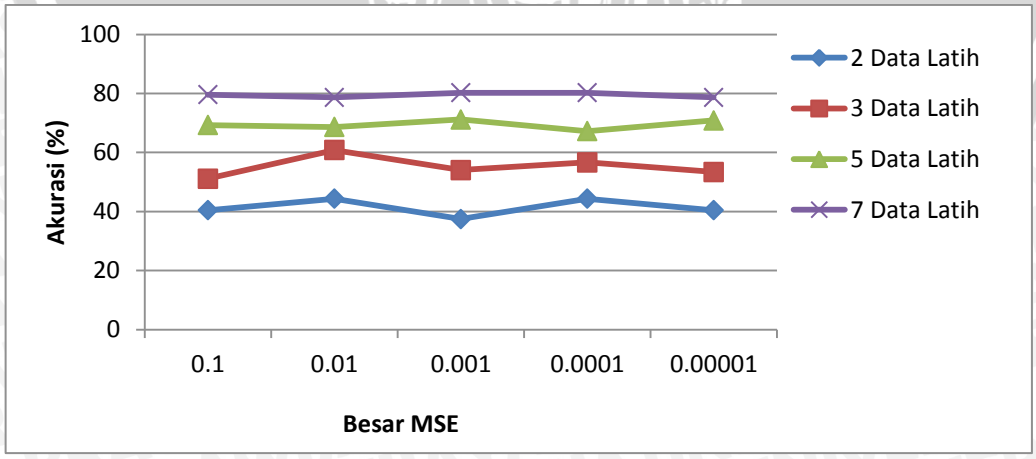
Jumlah kelas yang dilatihkan adalah 36 terdiri dari 26 karakter huruf dan 10 karakter angka. Data latih diambil sebagian dari hasil segmentasi karakter yang di rasa dapat mewakili kelasnya sebanyak variabel jumlah data latih yang diujikan dan jika tidak terpenuhi data latih dibuat secara manual pada aplikasi pengolahan

image, lebih detail mengenai pola data latih telah di lampirkan pada lampiran. Alasan sebagian data latih diambil dari hasil segmentasi adalah untuk mendapatkan pola data latih yang proporsional untuk diujikan pada hasil segmentasi karakter. Tabel 5.3 hasil pengukuran akurasi yang dilakukan.

Tabel 5.3 Hasil Uji Coba Besar MSE dan Jumlah Data Latih pada Akurasi

MSE		Data Latih			
		2	3	5	7
10^{-1}	Benar	125	158	214	246
	Akurasi	40.453	51.132	69.255	79.611
	Epoch	617	493	1234	501
10^{-2}	Benar	137	188	212	243
	Akurasi	44.336	60.841	68.608	78.64
	Epoch	2736	1809	1453	18403
10^{-3}	Benar	119	167	220	248
	Akurasi	37.425	54.045	71.197	80.258
	Epoch	13716	9764	8757	5888
10^{-4}	Benar	137	175	208	248
	Akurasi	44.336	56.634	67.213	80.258
	Epoch	89708	68058	63656	3968
10^{-5}	Benar	125	165	219	243
	Akurasi	40.453	53.398	70.873	78.640
	Epoch	684476	37938	20291	32717

Tabel 5.3 di atas adalah hasil pengujian yang dilakukan, untuk hasil akurasi dapat direpresentasikan dalam grafik seperti pada Gambar 5.5 berikut.



Gambar 5.5 Grafik Uji Coba Besar MSE dan Jumlah Data Latih pada Akurasi

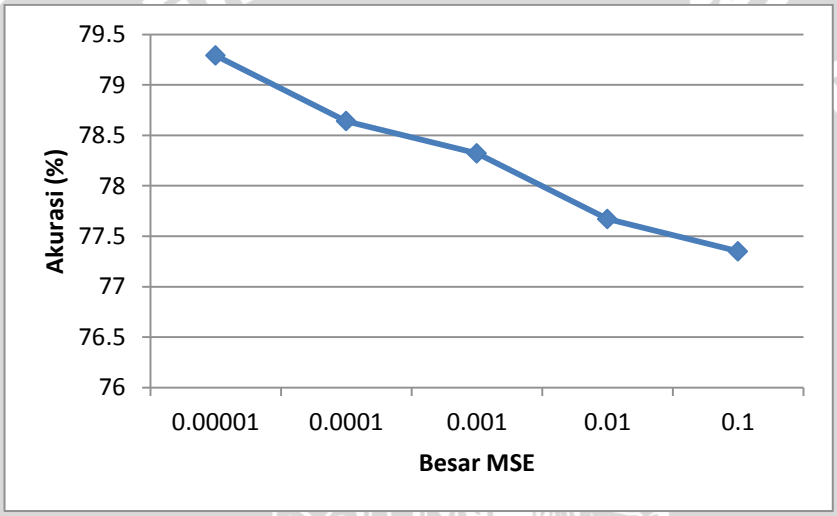
Gambar 5.5 di atas adalah grafik akurasi pengenalan karakter hasil uji coba besar MSE dan jumlah data latih. Representasi grafik hasil uji coba pada Gambar 5.5 terlihat paling mencolok adalah perbedaan jumlah data latih pada hasil akurasi. Gambar 5.5 menunjukan semakin bertambah data latih maka semakin terjadi peningkatan pada akurasi. Terlihat pada variabel 2 data latih akurasi maksimal hanya 44.336% kemudian naik pada variabel 3 data latih dengan akurasi maksimal yaitu 60.841% berikutnya yaitu variabel 5 dan 7 data latih masing - masing dengan akurasi maksimal yaitu 71.197% dan 80.258%. Hal ini menunjukan bahwa metode *backpropagation neural network* dapat memisahkan pola antar kelas dengan baik meski terdapat banyak variasi pola pada data latih. Sehingga semakin banyak data latih seharusnya akurasi klasifikasi akan semakin baik, tergantung pada pola data latih apakah telah mewakili pola data ujinya dan jumlah *epoch* yang dibutuhkan saat pelatihan juga semakin bertambah.

Gambar 5.5 juga merupakan hasil representasi uji coba MSE. Akurasi yang dihasilkan pada uji coba MSE berubah – ubah dan tidak menunjukan grafik peningkatan atau penurunan secara global. Hal ini menunjukan MSE pada uji coba ini tidak berpengaruh pada besar akurasi yang dihasilkan. Perubahan akurasi pada setiap hasil pelatihan pada uji coba MSE ini disebabkan bobot yang dihasilkan pada setiap pelatihan diinisialisasi secara random antara -0.5 dan 0.5, sehingga hasil akirnya pun belum tentu bisa lebih baik meski *stop condition*-nya bisa lebih mendekati target. Mengenai hal tersebut kemudian dilakuan uji coba kembali dengan melakukan inisialisasi bobot yang sama sebelumnya pada data dengan jumlah data latih terbaik yaitu 7 data latih dan dengan skenario pengujian yang sama. Hasilnya dapat ditunjukan pada Tabel 5.4 dan direpresentasikan dalam bentuk grafik pada Gambar 5.6.

Tabel 5.4 Hasil Uji Coba Besar MSE dengan Inisialisasi Bobot Sama

MSE		7 Data Latih
10^{-1}	Benar	239
	Akurasi	77.35
	epcoh	436
10^{-2}	Benar	240

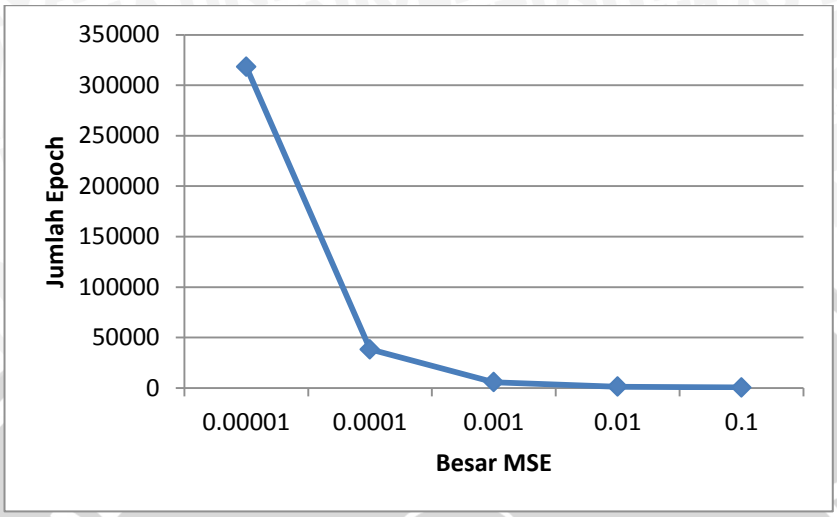
	Akurasi	77.67
	epcoh	1241
10^{-3}	Benar	242
	Akurasi	78.32
	epcoh	5734
10^{-4}	Benar	243
	Akurasi	78.64
	epcoh	38275
10^{-5}	Benar	245
	Akurasi	79.29
	epcoh	318323



Gambar 5.6 Grafik Uji Coba Besar MSE pada Akurasi dengan Inisialisasi Bobot Sama

Gambar 5.6 diatas adalah hasil uji coba besar MSE pada 7 data latih dengan inisialisasi bobot sebelumnya untuk mendapatkan hasil perbandingan yang sesuai pada variabel MSE. Dari hasil pengujian yang ditunjukan pada Gambar 5.6 diatas didapatkan akurasi terbaik ditunjukan pada MSE 10^{-5} dan terburuk pada MSE 10^{-1}. Hal ini menunjukan semakin besar nilai MSE maka akurasi juga semakin turun karena jaringan belum dapat memisahkan pola antar kelas dengan baik. Sehingga nilai perbandingan *ouput* yang dihasilkan saat pengujian tidak menghasilkan jarak yang jauh antar kelas, sehingga bisa saja terjadi kesalahan. Sebaliknya dari hasil pengujian didapatkan semakin kecil nilai MSE akurasi semakin meningkat, hal ini menunjukan semakin kecil nilai MSE jaringan akan semakin dapat memisahkan pola antar kelas sehingga memberikan hasil akurasi

yang semakin meningkat. Gambar 5.7 dibawah ini adalah grafik uji coba besar MSE terhadap banyaknya *epoch* yang dibutuhkan untuk pelaihan.



Gambar 5.7 Grafik Uji Coba Besar MSE pada Akurasi

Gambar 5.7 diatas adalah grafik pengaruh MSE pada banyaknya *epoch* yang dibutuhkan untuk uji coba. Gambar 5.7 diatas menunjukan grafik peningkatan jumlah *epoch* pada setiap penurunan nilai MSE. Hal ini menunjukan bahwa nilai MSE berbanding terbalik dengan jumlah *epoch*, semakin kecil nilai MSE semakin banyak *epoch* saat pelatihan dan semakin besar pula akurasi yang dihasilkan saat pengujian. Dari analisis grafik pada Gambar 5.6 dan Gambar 5.7 dipilih nilai MSE 10^{-4} sebagai MSE yang optimal dan sebagai variabel pengujian berikutnya karena dengan nilai MSE 10^{-4} didapatkan hasil yang lebih baik dibanding nilai MSE 10^{-5} ditinjau dari peningkatan *epoch* dan peningkatan akurasi. Peningkatan akurasi dari MSE 10^{-4} ke MSE 10^{-5} hanya menaikan sekitar 1% akurasi sedangkan *epoch* pelatihan sekitar 300,000. Sehingga dengan nilai MSE 10^{-4} adalah nilai MSE yang optimal dan cukup untuk dijadikan variabel uji ke uji coba berikutnya berikutnya.

5.4 Hasil dan Analisis Uji Coba *Learning Rate* pada Tingkat Akurasi

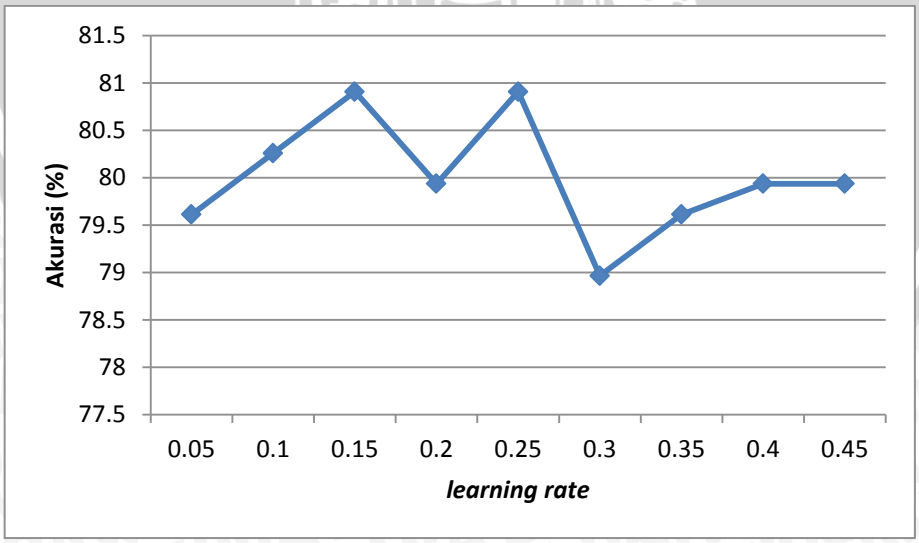
Uji coba *learning rate* digunakan untuk mengetahui pengaruh laju perubahan bobot pada ketepatan pengenalan karakter. Uji coba *learning rate* dilakukan dengan mencoba variabel *learning rate* seperti skenario uji coba yang telah ditentukan dengan variabel tetap yang diambil dari hasil skenario

sebelumnya yaitu MSE 10^{-4} dan 7 data latih pada setiap kelas. Pengujian juga dilakukan dengan menginisialisasi bobot yang sama. Tabel 5.5 adalah hasil uji coba *learning rate*.

Tabel 5.5 Hasil Uji Coba *Learning Rate*

Learning rate	Jumlah Benar	Akurasi (%)	*Epoch*
0.05	246	79.611	8480.0
0.10	248	80.258	4255.0
0.15	250	80.906	3194.0
0.20	247	79.935	2412.0
0.25	250	80.906	2082.0
0.30	244	78.964	1687.0
0.35	246	79.611	1588.0
0.40	247	79.935	1402.0
0.45	247	79.935	1102.0

Tabel 5.5 diatas adalah hasil pengukuran akurasi pengenalan karakter yang dilakukan pada data yang sama pada uji coba MSE dan banyak data latih sebelumnya. Gambar 5.8 di bawah ini adalah hasil representasi grafik dari hasil pengujian.



Gambar 5.8 Grafik Uji Coba Besar *Learning Rate* pada Akurasi

Gambar 5.8 adalah grafik hasil uji coba pengaruh *learning rate* terhadap akurasi. Percobaan dilakukan dengan melakukan inisialisasi bobot yang sama untuk setiap variabel yang diuji. Hasil didapatkan akurasi terbaik adalah 80.906% dengan variabel *learning rate* 0.15 dan 0.25 dengan jumlah *epoch* masing – masing 3195 dan 208 dan hasil terburuk didapatkan 78.964 pada *learning rate* 0.3 dengan jumlah epoch 1687. Grafik pada Gambar 5.8 menunjukan naik-turun akurasi tetapi dengan penaikan atau penurunan yang tidak begtu jauh. Simpangan maksimalnya yaitu 1.942 dan membuat grafik diatas tidak menunjukan suatu pola untuk diambil suatu kesimpulan. Bila ditinjau dari persamaan 2.25, 2.26, 2.30 dan 2.31, *learning rate* mengatur seberapa besar koreksi kesalahan bobot dapat diterima untuk pembaruan bobot. Semakin besar *learning rate* yang digunakan saat pelatihan maka semakin sedikit pula *epoch* yang dibutuhkan untuk pelatihan. Sebagai pengali koreksi bobot, *learning rate* juga memberikan variasi pada perubahan bobot untuk proses pembelajaran yang membuat hasil pengenalan menjadi dinamis. Sehingga untuk penentuan *learning rate* pada kasus ini dengan target MSE dan data latih tertentu perlu dilakukan uji coba untuk mendapatkan variabel *learning rate* sehingga akurasi klasfikasi dapat dimaksimalkan.

BAB VI
PENUTUP

6.2 Kesimpulan

Kesimpulan yang diambil dari skripsi ini adalah :

1. *Vertical edge detection* dan *histogram projection* untuk deteksi plat nomor didapatkan akurasi terbaik yaitu 89% pada perbesaran 5.7x dengan ukuran *Structuring Element* 1x7. Akurasi 89% ini dapat diterima sehingga metode ini dapat diterapkan pada sistem parkir untuk deteksi plat nomor dengan rata rata waktu eksekusi 0.887 second.
2. Segmentasi karakter hasil uji coba dengan metode binarisasi *Otsu* yang dilakukan didapatkan hasil *f-measure* dengan nilai baik yaitu sebesar 0.934. Metode ini memiliki kelemahan tidak dapat mensegmentasi untuk karakter dengan warna hitam (plat nomor kendaraan umum). Sehingga untuk implementasinya dilapangan tergantung pada kebijakan yang berlaku pada sistem parkir yang ada.
3. Hasil pengenalan karakter dengan metode *backpropagation neural network* yang digunakan didapatkan hasil akurasi maksimal 80.906% pada jumlah data latih masing masing karakter 7, MSE 10^{-4}, dan *learning rate* 0.25. Akurasi metode *backpropagation* yang dihasilkan akan sangat tergantung pada pola data latihnya. Meskipun sedikit pola yang dilatihkan akan sangat efisien jika pola yang diujikan telah dipelajari sehingga dengan pola yang sedikit telah cukup untuk mengenali data ujinya. Semakin kecil MSE yang digunakan sebagai *stop condition* akan berdampak pada peningkatan akurasi dan peningkatan jumlah *epoch* saat pelatihan, Sehingga dalam penentuan MSE perlu dipertimbangkan juga jumlah *epoch* saat pelatihan dengan akurasi yang dihasilkan untuk mengurangi waktu pelatihan. Pengaruh *learning rate* pada jaringan adalah percepatan perubahan bobot untuk pencapaian minimal *stop condition* yaitu MSE. Sehingga semakin besar nilai *learning rate* yang diinputkan maka jumlah *epoch* yang butuhkan untuk pelatihan akan semakin sedikit. Sedangkan pengaruh *learning rate* pada akurasi pengenalan tidak

menentu karena pada grafik uji coba tidak terlihat suatu pola. Sehingga untuk menentukan variabel *learning rate* perlu dilakukan uji coba untuk mendapatkan hasil yang memuaskan.

6.2 Saran

Saran untuk penelitian berikutnya dari skripsi ini adalah :

1. Mengimplementasikan metode *edge-based algorithm* pada deteksi plat nomor karena kelebihan metode ini adalah dari segi waktu eksekusi yang dapat mendukung pengembangan secara *real time* (berbasis video).
2. Metode lain untuk pengolahan citra yang dibutuhkan untuk kasus tertentu pada permasalahan ini adalah pengurangan efek blur pada citra yang bisa terjadi akibat pergerakan mobil yang cepat. Salah satunya yang bisa diambil adalah IHS – RGB atau Brovey.
3. Metode Segmentasi lain untuk segmentasi karakter yang dapat digunakan adalah analisis *histogram projection.*
4. Penelitian berikutnya juga dapat dikembangkan dengan mengubah ekstraksi fitur untuk pengenalan karakter pada metode *backpropagation neural network*. Karena berdasarkan analisis hasil segmentasi, akan didapatkan karakter yang bisa jadi berbeda jauh dari karakter yang dilatihkan. Misal karakter data ujinya *counter-italic* sedangkan karakter data latihnya *italic*. Penelitian berikutnya dapat menerapkan ekstraksi fitur secara global misal *histogram projection*, atau penerapan dengan mengubah bentuk karakter terlebih dahulu seperti metode *thining*.
5. Penelitian berikutnya juga dapat mengubah metode pengenalan karakter dengan syarat mempertimbangkan waktu pengenalan karakter seperti SVM atau KNN. Karena banyak kelas yang dilatih adalah 36 karakter yang terdiri dari 26 karakter huruf dan 10 karakter angka.

DAFTAR PUSTAKA

[ANA-10] Ananto, Suharyoko. 2010 "Rancang Bangun Sistem Pengenalan Pola Tanda Tangan Menggunakan Metode Counter Propagation Neural Network." Skripsi Manajemen Informatika dan teknik komputer. 9-10.

[ANO-13] anonimous. 2013. Mahasiswa *Universitas Brawijaya*. Diakses 20 Mei 2014.www.ub.ac.id/tentang/profil-universitas/mahasiswa.

[BAD-11] Badr AMR,Mohamed M, Abdel Wahab, dkk.2011."Automatic Number Plate Recognition System".Annals of the University of Craiova.64-65.

[DAN-05] Danian, Zheng, Zhao Yannan, dan Wang Jiaxin.2005. "An Efficient Method Of License Plate Location." Science Direct. 3-7.

[DAR-10] Darma, Putra.2010. "Pengolahan Citra Digital". Yogyakarta: penerbit andi.

[DON-14] Dong, ZhengHao, dan FengXin.2014. "Research on License Plate Recognition Algorithm based on Support Vector Machine ." ACADEMY PUBLISHER 9.2-8.

[FAJ-10] Fajar, Astuti Hermawati, dan Koesdijarto Roenadi. "A Real-Time License Plate Detection System For Parking Access." DIKTI.

[HEA-08] Heaton, Jeff.2008."Introduction to Neural Networks with Java, Second Edition." United States : Heaton Research, Inc.

[IDH-08] Idhawati, Hestiningsih. "Pengolahan Citra." 2008.

[IKH-12] Ikhwan, Ruslianto, dan Harjoko Agus. "Pengenalan Karakter Plat Nomor Mobil Secara Real Time." IJEIS. 10. 2-10.

[LAU-95] Laurene, Fausett.1995. "Faundamentals of Neural Network : Architectures, Algoritms and Application." New Jersey: Prentice Hall.

[MAH-11] Mahmood, Ashoori-Lalimi, dan Ghofrani Sedigheh. 2011. "An Efficient Method for Vehicle License Plate Detection in Complex Scenes." Science Research. 2-4.

[MAR-10] Maria, Petrou, dan Petrou Costas.2010." Image Processing: The Fundamentals. 2nd." West Sussex: wiley.

[MOS-13] Md.Mostafa, Kamal Sarker, Yoon Sook, Lee Jaehwan, dan Sun Park Dong. "Novel License Plate Detection Method Based on Heuristic Energy Map." J-KICS 38C.3-12.

[NIC-00] Nick, Efford.2000. "Digital Image Processing : A Practical Introduction Using Java." Harlow: Addison Wesley.

[RAF-02] Rafael, C. Gonzale, dan E. Wood Richard.2002."Digital Image Processing. 3rd." new jersey: prentice hall.

[REZ-13] Reza, Azad, dan Reza Shayegh Hamid. 2013. "New Method for Optimization of License Plate Recognition system with Use of Edge Detection and Connected Component." International Conference on Computer and Knowledge Engineering. Ferdowsi University of Mashhad.2-5.

[RIA-01] Riadi.2001."Jaringan Saraf Tiruan untuk Pengenalan Tanda Tangan." Intitut Pertanian Bogor : Jurusan Ilmu Komputer.2-5.

[ROD-09] Rodolfo P, dos Santos, Clemente Gabriela S, Ing Ren Tsang, dan D.C. Calvalcanti George. "Text Line Segmentation Based on Morphology and Histogram Projection." IEEE. 2-4.

[RON-13] Ronak, P Patel, M Patel Narendra, and Brahmbhatt Keyur. "Automatic Licenses Plate Recognition."IJCSMC 2, no. 4.2-6.

[RUD-06] Rudi, Tri Handoyo. "Peningkatan Keamanan Database dengan Menggunakan Teknologi Pengenalan Tanda Tangan Berbasis Mod Anfis." Sekolah Tinggi Manajemen Informatika dan Teknik Komputer Surabaya, 2006.

[SHA-13] Shan, Du.2013."Automatic License Plate Recognition (ALPR): A State-of-the-Art Review." IEEE. 3-6.

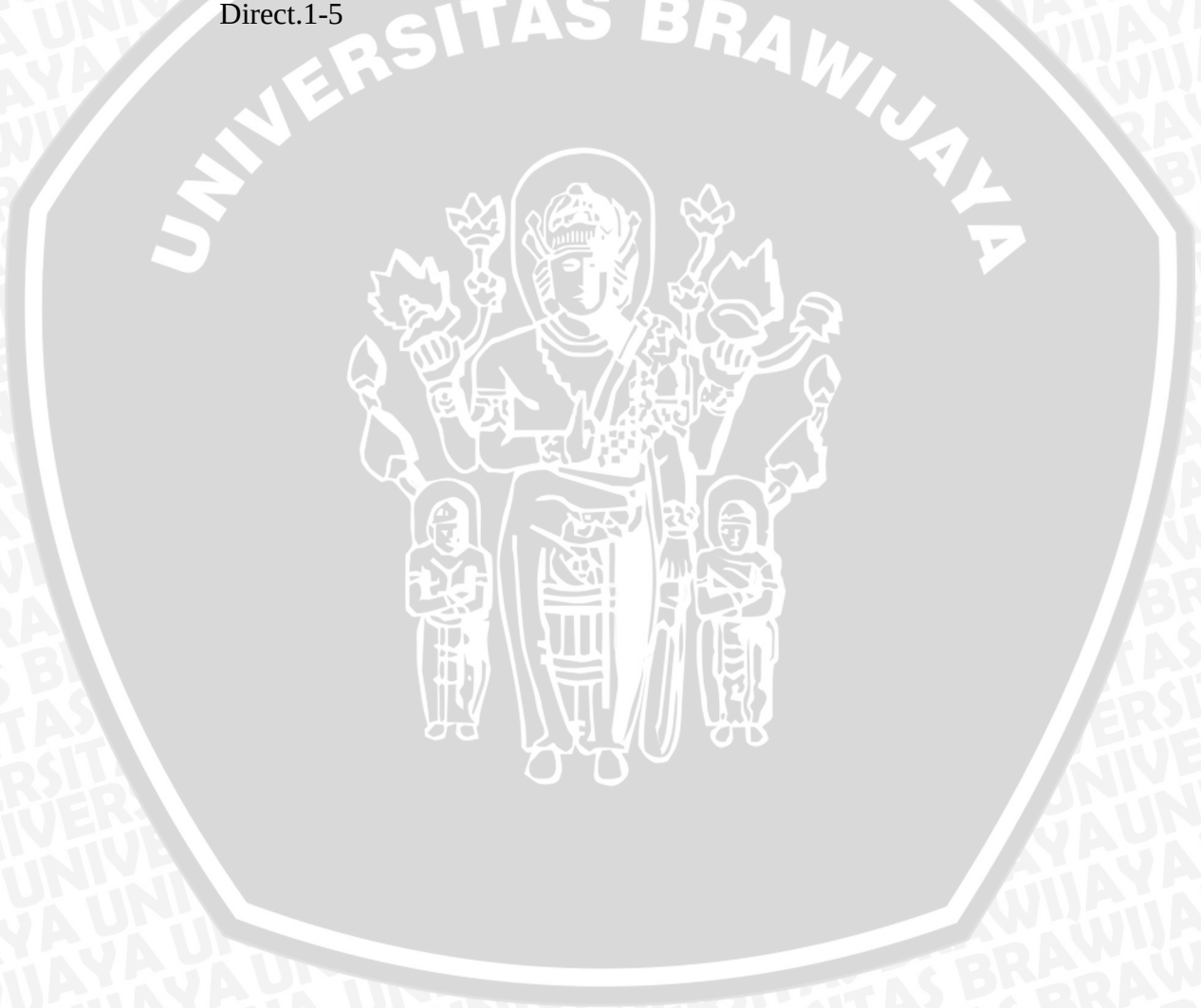
[SRI-10] Sri, Cahyaningsih. 2010. "Deteksi Osteoporosis dengan Tresholding Metode *Otsu* Pada Citra X-Ray Tulang Rahang." Universitas Islam Negeri Maulana Malik Ibrahim Malang.

[SYA-10] Syafiie, Nur Luthfie. 2010."Implementasi Jaringan Saraf Tiruan Backpropagation Pada Aplikasi Pengenalan Wajah Dengan Jarak

Yang Berbeda Menggunakan MATLAB 7." Universitas Gunadarma : Jurusan Teknik Informatika.2-5.

[TRA-04] Tran, Duc Duan, Anh Duc Duong, dan Le Hong Du Tran. 2004."Combining Hough Transform and Contour Algorithm for detecting Vehicles License-Plates." International Symposium on Intelligent Multimedia.2-4.

[VAH-09] Vahid, Abolghasemi, dan Ahmadyfard Alireza. 2009."An edge-based color-aided method for license plate detection." Science Direct.1-5



LAMPIRAN

1. Contoh Data Pengambilan Gambar di Lapangan

No	Gambar
Hasil Pengambilan dengan Perbesaran Kamera 2.1x	
1	
2	
3	

4	
5	
Hasil Pengambilan dengan Perbesaran Kamera 3.4x	
1	
2	

3	
4	
5	

Hasil Pengambilan dengan Perbesaran Kamera 4.7x

1	

2	
3	
4	
5	

Hasil Pengambilan dengan Perbesaran Kamera 5.7x	
1	
2	
3	
4	

5	

Hasil Pengambilan dengan Perbesaran Kamera 6.6x

1	
2	
3	

4	
5	

2. Hasil Deteksi Plat nomor dan Segmentasi karakter

No	Hasil segmentasi plat	Jumlah Objek	Jumlah Karakter tersegmen	Jumlah Karakter tdk tersegmen	Presisi	*Recall*	*F-Measure*
Hasil Evaluasi pada Perbesaran Kamera 2.1x							
1	-	0	0	0	0	0	0
2	-	0	0	0	0	0	0
3		7	6	0	1	0.85714	0.92307692
4	-	0	0	0	0	0	0
5		7	7	0	1	1	1
6	-	0	0	0	0	0	0
7		8	7	0	1	0.875	0.93333333
8	-	0	0	0	0	0	0

9	[redacted]	6	6	1	0.85714	1	0.92307692
10	[redacted]	7	7	0	1	1	1
11	[redacted]	6	6	0	1	1	1
12	[redacted]	6	6	0	1	1	1
13	[redacted]	5	4	3	0.57143	0.8	0.66666667
14	[redacted]	7	7	0	1	1	1
15	[redacted]	7	7	0	1	1	1
16	[redacted]	10	7	0	1	0.7	0.82352941
17	-	0	0	0	0	0	0
18	-	0	0	0	0	0	0
19	[redacted]	7	7	0	1	1	1
20	[redacted]	6	6	0	1	1	1
21	[redacted]	8	7	0	1	0.875	0.93333333
22	-	0	0	0	0	0	0
23	-	0	0	0	0	0	0
24	[redacted]	5	5	2	0.71429	1	0.83333333
25	-	0	0	0	0	0	0
26	[redacted]	8	6	0	1	0.75	0.85714286
27	-	0	0	0	0	0	0
28	[redacted]	3	3	3	0.5	1	0.66666667
29	-	0	0	0	0	0	0
30	[redacted]	8	7	0	1	0.875	0.93333333
31	-	0	0	0	0	0	0
32	-	0	0	0	0	0	0
33	[redacted]	5	5	1	0.83333	1	0.90909091

34	-	0	0	0	0	0	0
35	[redacted]	7	5	2	0.71429	0.71429	0.71428571
36		0	0	0	0	0	0
37	[redacted]	2	1	6	0.14286	0.5	0.22222222
38	[redacted]	10	7	0	1	0.7	0.82352941
Jumlah Keseluruhan data		145	129	18	0.87755	0.88966	0.88356164
Hasil Evaluasi pada Perbesaran Kamera 3.4x							
1	[redacted]	9	7	0	1	0.77778	0.875
2	[redacted]	9	7	0	1	0.77778	0.875
3	[redacted]	7	7	0	1	1	1
4	[redacted]	7	7	0	1	1	1
5	[redacted]	7	7	0	1	1	1
6	[redacted]	6	6	0	1	1	1
7	[redacted]	6	6	0	1	1	1
8	[redacted]	7	7	0	1	1	1
9	-	0	0	0	0	0	0
10	-	0	0	0	0	0	0
11	-	0	0	0	0	0	0
12	-	0	0	0	0	0	0
13	[redacted]	2	2	5	0.28571	1	0.444444
14	[redacted]	6	6	0	1	1	1
15	-	0	0	0	0	0	0
16	-	0	0	0	0	0	0
17	[redacted]	8	8	0	1	1	1
18	[redacted]	7	7	0	1	1	1

19	[redacted]	7	7	0	1	1	1
20	-	0	0	0	0	0	0
21	[redacted]	9	7	0	1	0.77778	0.875
22	-	0	0	0	0	0	0
23	[redacted]	6	6	0	1	1	1
24	-	0	0	0	0	0	0
25	-	0	0	0	0	0	0
26	-	0	0	0	0	0	0
27	-	0	0	0	0	0	0
28	[redacted]	8	8	0	1	1	1
29	-	0	0	0	0	0	0
30	-	0	0	0	0	0	0
31	[redacted]	7	7	0	1	1	1
32	[redacted]	11	6	0	1	0.54545	0.705882
33	[redacted]	7	7	0	1	1	1
34	[redacted]	11	7	0	1	0.63636	0.777778
35	[redacted]	12	7	0	1	0.58333	0.736842
36	[redacted]	7	7	0	1	1	1
37	[redacted]	5	5	2	0.71429	1	0.833333
38	-	0	0	0	0	0	0
39	[redacted]	7	7	7	0.5	1	0.666667
40	-	0	0	0	0	0	0
41	[redacted]	9	7	0	1	0.77778	0.875
42	[redacted]	3	2	4	0.33333	0.66667	0.444444

Jumlah Keseluruhan data		190	167	18	0.9027	0.87895	0.890667
Hasil Evaluasi pada Perbesaran Kamera 4.7x							
1	[redacted]	9	7	0	1	0.77778	0.875
2	[redacted]	3	2	5	0.4	0.66667	0.5
3	[redacted]	7	7	0	1	1	1
4	[redacted]	8	6	0	1	0.75	0.857142857
5	[redacted]	7	7	0	1	1	1
6	[redacted]	8	7	0	1	0.875	0.933333333
7	[redacted]	7	7	0	1	1	1
8	-	0	0	0	0	0	0
9	[redacted]	6	4	2	0.66666667	0.66667	0.666666667
10	-	0	0	0	0	0	0
11	[redacted]	3	2	6	0.25	0.66667	0.363636364
12	-	0	0	0	0	0	0
13	[redacted]	6	6	0	1	1	1
14	-	0	0	0	0	0	0
15	[redacted]	7	7	0	1	1	1
16	-	0	0	0	0	0	0
17	[redacted]	8	6	0	1	0.75	0.857142857
18	[redacted]	6	6	0	1	1	1
19	[redacted]	8	8	0	1	1	1
20	[redacted]	6	6	0	1	1	1
21	[redacted]	8	7	0	1	0.875	0.933333333
22	[redacted]	9	8	0	1	0.88889	0.941176471
23	[redacted]	4	1	6	0.14285714	0.25	0.181818182

24	[redacted]	9	7	0	1	0.77778	0.875
25	-	0	0	0	0	0	0
26	[redacted]	9	8	0	1	0.88889	0.941176471
27	[redacted]	7	7	0	1	1	1
28	-	0	0	0	0	0	0
29	-	0	0	0	0	0	0
30	[redacted]	7	7	0	1	1	1
31	[redacted]	5	5	2	0.71428571	1	0.833333333
32	-	0	0	0	0	0	0
33	[redacted]	3	2	5	0.28571429	0.66667	0.4
34	[redacted]	6	6	0	1	1	1
35	[redacted]	7	7	0	1	1	1
36	-	0	0	0	0	0	0
37	[redacted]	8	8	0	1	1	1
38	[redacted]	1	1	6	0.14285714	1	0.25
39	[redacted]	4	4	6	0.4	1	0.571428571
40	[redacted]	3	3	3	0.5	1	0.666666667
41	[redacted]	6	6	0	1	1	1
42	[redacted]	4	4	6	0.4	1	0.571428571
43	[redacted]	5	5	1	0.83333333	1	0.909090909
44	-	0	0	0	0	0	0
45	[redacted]	8	7	0	1	0.875	0.933333333
46	[redacted]	3	3	5	0.375	1	0.545454545
47	[redacted]	9	6	0	1	0.66667	0.8
48	[redacted]	9	7	0	1	0.77778	0.875

49	N 1282 AD	7	7	0	1	1	1
50	L 1889 CE	7	7	0	1	1	1
51	B 1 2 [redacted]	6	6	8	0.42857143	1	0.6
52	[redacted] 1504 [redacted]	7	5	7	0.41666667	0.71429	0.526315789
53	-	0	0	0	0	0	0
54	N 800 BX	6	6	0	1	1	1
55	[redacted]	4	2	5	0.28571429	0.5	0.363636364
56	N 1048 [redacted]	7	7	0	1	1	1
57	N 1462 AD	7	7	0	1	1	1
58	[redacted] 1593 JJ	8	7	0	1	0.875	0.933333333
59	L 444 RI	7	6	0	1	0.85714	0.923076923
60	[redacted] A	1	1	6	0.14285714	1	0.25
61	[redacted] L 1870	5	5	1	0.83333333	1	0.909090909
62	[redacted] B 1648 SYD	10	8	0	1	0.8	0.888888889
63	N 1 2 BK [redacted]	8	7	1	0.875	0.875	0.875
64	N 659 BL	6	6	0	1	1	1
65	[redacted] N	1	1	6	0.14285714	1	0.25
66	N 1590	5	5	2	0.71428571	1	0.833333333
67	-	0	0	0	0	0	0
68	B 6 AJ	4	4	3	0.57142857	1	0.727272727
69	S [redacted] WJ	3	3	4	0.42857143	1	0.6
70	N 1634 CI	7	7	0	1	1	1
71	[redacted] 1435 GU	9	7	0	1	0.77778	0.875
72	B 8010 KX	8	7	0	1	0.875	0.933333333
73	N 1601 AA	7	7	0	1	1	1

Jumlah Keseluruhan data		373	335	96	0.77726218	0.89812	0.833333333
Hasil Evaluasi pada Perbesaran Kamera 5.7x							
1	N 941 BE	6	6	0	1	1	1
2	N 504	4	4	2	0.666667	1	0.8
3	DK 1464 IC	8	8	0	1	1	1
4	N 1004 CC	7	7	0	1	1	1
5	L 1538 KA	7	7	0	1	1	1
6	S 1335 JD	7	7	0	1	1	1
7	N 1826 CH	7	7	0	1	1	1
8	N 1274 X	6	6	1	0.857143	1	0.9230769
9	N 1274 XA	7	7	0	1	1	1
10	N 1723 AV	7	7	0	1	1	1
11		0	0	0	0	0	0
12		0	0	0	0	0	0
13	N 900 BE	6	6	0	1	1	1
14		0	0	0	0	0	0
15	AG 900 O	6	6	0	1	1	1
16	N 1141 AY /	11	7	0	1	0.636364	0.7777778
17	N 1627 BJ	9	7	0	1	0.777778	0.875
18	N 1410 KC	7	7	0	1	1	1
19	N 931 BL	6	6	0	1	1	1
20	B1959TFF	8	8	0	1	1	1
21		0	0	6	0	0	0
22		0	0	0	0	0	0
23	N 775 NN	6	6	0	1	1	1

24	N 775 NN	6	6	0	1	1	1
25	-	0	0	0	0	0	0
26	N 993 BC	8	6	0	1	0.75	0.8571429
27	B 1182 CKF	9	8	0	1	0.888889	0.9411765
28	N 1303 BN	7	7	0	1	1	1
29	AG 1877 KE	9	8	0	1	0.888889	0.9411765
30	AG 1877 KE	10	8	0	1	0.8	0.8888889
31	D 534 FS	8	6	0	1	0.75	0.8571429
32	N 1172 CR	7	7	0	1	1	1
33	B 18 K	4	4	4	0.5	1	0.6666667
34	W 671 XM	6	6	0	1	1	1
35	S 795 HJ	8	6	0	1	0.75	0.8571429
36	-	0	0	0	0	0	0
37	-	0	0	0	0	0	0
38	-	0	0	0	0	0	0
39	N 1366 GH	7	7	0	1	1	1
40	N 900 BE	6	6	0	1	1	1
41	N 9859 CB	8	7	0	1	0.875	0.9333333
42	N 1310 KC	8	7	0	1	0.875	0.9333333
43	N 1586 BK	8	7	0	1	0.875	0.9333333
44	-	0	0	0	0	0	0
45	B 1 BOS	7	5	0	1	0.714286	0.8333333
46	-	0	0	0	0	0	0
47	N 703 AF	6	6	0	1	1	1
48	N 703 AF	6	6	0	1	1	1

49	[redacted]	9	7	0	1	0.777778	0.875
50	[redacted]	7	7	0	1	1	1
51	[redacted]	7	7	0	1	1	1
52	[redacted]	0	0	0	0	0	0
53	[redacted]	10	7	0	1	0.7	0.8235294
54	[redacted]	7	7	0	1	1	1
55	[redacted]	10	8	0	1	0.8	0.8888889
56	[redacted]	7	7	0	1	1	1
57	[redacted]	9	6	0	1	0.666667	0.8
58	[redacted]	7	7	0	1	1	1
59	[redacted]	7	7	0	1	1	1
Jumlah Keseluruhan data		343	312	13	0.96	0.909621	0.9341317
Hasil Evaluasi pada Perbesaran Kamera 6.6x							
1	-	0	0	0	0	0	0
2	[redacted]	2	2	5	0.28571	1	0.444444
3	-	0	0	0	0	0	0
4	[redacted]	6	6	0	1	1	1
5	[redacted]	4	1	6	0.14286	0.25	0.181818
6	-	0	0	0	0	0	0
7	[redacted]	5	5	2	0.71429	1	0.833333
8	-	0	0	0	0	0	0
9	[redacted]	7	7	0	1	1	1
10	[redacted]	5	5	2	0.71429	1	0.833333
11	[redacted]	7	7	0	1	1	1
12	[redacted]	8	8	0	1	1	1

13	-	0	0	0	0	0	0
14	N 1721 BA	7	7	0	1	1	1
15	N 321 BE	5	5	0	1	1	1
16	B 9 BX	4	4	0	1	1	1
17	N	1	1	5	0.166667	1	0.285714
18	N 1	2	2	5	0.285714	1	0.444444
19	C	2	1	5	0.166667	0.5	0.25
20	N 746 BD	6	6	0	1	1	1
21	N G	2	2	4	0.333333	1	0.5
22	N 1423 A	6	6	1	0.857143	1	0.923077
23	N 1818	5	5	2	0.714286	1	0.833333
24	P 689 V	6	5	1	0.833333	0.833333	0.833333
25	AG 648	6	5	2	0.714286	0.833333	0.769231
26	N 679	5	5	1	0.833333	1	0.909091
27	AE 1445 BD	8	8	0	1	1	1
28	N 1497 BQ	7	7	0	1	1	1
29	-	0	0	0	0	0	0
30	N 1167 A	7	7	0	1	1	1
31	N 721	4	4	2	0.666667	1	0.8
32	N 10	3	3	7	0.3	1	0.461538
33	N 302 DK	6	6	0	1	1	1
34	N 1 7	4	3	4	0.428571	0.75	0.545455
35	L 1785 WS	7	7	0	1	1	1
36	-	0	0	0	0	0	0

37	-	0	0	0	0	0	0
38	-	0	0	0	0	0	0
39	[redacted]	5	3	4	0.428571	0.6	0.5
40	[redacted]	6	5	2	0.714286	0.833333	0.769231
41	-	0	0	0	0	0	0
42	-	0	0	0	0	0	0
43	-	0	0	0	0	0	0
44	-	0	0	0	0	0	0
45	-	0	0	0	0	0	0
46	-	0	0	0	0	0	0
47	[redacted]	5	5	1	0.833333	1	0.909091
48	-	0	0	0	0	0	0
49	-	0	0	0	0	0	0
50	[redacted]	1	1	5	0.166667	1	0.285714
51	[redacted]	5	3	3	0.5	0.6	0.545455
52	[redacted]	3	2	5	0.285714	0.666667	0.4
53	[redacted]	2	2	5	0.285714	1	0.444444
54	[redacted]	5	3	3	0.5	0.6	0.545455
55	-	0	0	0	0	0	0
56	[redacted]	4	3	4	0.428571	0.75	0.545455
Jumlah Keseluruhan data		183	167	86	0.660079051	0.912568306	0.766055046

3. Karakter Data Latih pada Jaringan

Karakter	Karakter-1	Karakter-2	Karakter-3	Karakter-4	Karakter-5	Karakter-6	Karakter-7
A							
B							
C							
D							
E							
F							
G							
H							
I							
J							
K							
L							
M							
N							
O							
P							
Q							

R							
S							
T							
U							
V							
W							
X							
Y							
0							
1							
2							
3							
4							
5							
6							
7							
8							
9							