DETEKSI TEMPAT KOSONG PADA LAHAN PARKIR MOBIL MENGGUNAKAN METODE *VEHICLE DETECTION* DAN *EDGE DETECTION*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Oleh:
MILYUN NI'MA SHOUMI
0610963027-96

PROGRAM STUDI ILMU KOMPUTER
JURUSAN MATEMATIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG

2011


UNIVERSITAS BRAWIJAYA

LEMBAR PENGESAHAN SKRIPSI

DETEKSI TEMPAT KOSONG PADA LAHAN PARKIR MOBIL MENGGUNAKAN METODE *VEHICLE DETECTION* DAN *EDGE DETECTION*

Oleh:
MILYUN NI'MA SHOUMI
0610963027-96

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 16 Juni 2011
dan dinyatakan memenuhi syarat untuk memperoleh gelar
Sarjana dalam bidang Ilmu Komputer

Pembimbing I	Pembimbing II
Drs. Muh. Arif Rahman, M.Kom	**Drs. Marji, MT**
NIP. 196604231991111001	**NIP. 196708011992031001**

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Abdul Rouf Alghofari, M.Sc
NIP. 196709071992031001


UNIVERSITAS BRAWIJAYA

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama	**: Milyun Ni'ma Shoumi**
Nim	**: 0610963027**
Jurusan	**: Matematika**
Penulis Skripsi berjudul	**: Deteksi Tempat Kosong Pada Lahan Parkir Mobil Menggunakan Metode *Vehicle Detection* dan *Edge Detection***

Dengan ini menyatakan bahwa:

1. **Isi dari Skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila di kemudian hari ternyata Skripsi yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, 16 Juni 2011
Yang menyatakan,

Milyun Ni'ma Shoumi
NIM. 0610963027

DETEKSI TEMPAT KOSONG PADA LAHAN PARKIR MOBIL MENGGUNAKAN METODE *VEHICLE DETECTION* DAN *EDGE DETECTION*

ABSTRAK

Antrian panjang mobil sering ditemui di beberapa fasilitas umum karena pengunjung harus berputar-putar mencari tempat parkir yang kosong. Hal ini dapat diminimalisir, salah satunya dengan menggunakan sistem informasi tempat parkir yang menunjukkan tempat parkir yang kosong atau terisi beserta jumlahnya. Pada penelitian ini disajikan dua buah metode pengolahan citra digital untuk pendeteksian tempat kosong dan terisi pada citra lahan parkir mobil, yaitu metode *vehicle detection* dan *edge detection*.

Vehicle detection merupakan metode yang digunakan untuk mendeteksi obyek pada citra dengan cara mengurangkan citra lahan parkir yang kosong dengan citra lahan parkir yang berisi mobil. Sedangkan metode *edge detection* merupakan metode yang digunakan untuk mendeteksi tepi obyek. Hasil dari kedua metode tersebut kemudian dibandingkan menggunakan AND function, sehingga didapatkan kondisi kosong atau terisi untuk setiap kotak tempat parkir. Nilai *threshold* berpengaruh pada penentuan kondisi tempat parkir. Pada penelitian ini, hasil deteksi terbaik diperoleh pendeteksian tempat terisi pada lahan parkir yang berlokasi di Malang Town Square (Matos), dengan threshold 10 dan tingkat akurasinya sebesar 99,3%.

Kata kunci: parkir mobil, deteksi tempat, vehicle detection, edge detection, AND function, threshold.


UNIVERSITAS BRAWIJAYA

VACANT PARKING LOT SPACE DETECTION USING VEHICLE DETECTION AND EDGE DETECTION METHOD

ABSTRACT

Long queues of car often encountered in some public facilities because visitors should be around to find an empty parking space. One way to minimize this case is use a parking information system which shows the location of the parking lot that is empty or occupied with their amounts. This research presents two methods of digital image processing for detection of empty space and occupied in the image of car parking area, there are vehicle detection and edge detection method.

Vehicle detection is the method used to detect objects in the image by subtracting the parking area image which an empty parking lot with the image which contains the car.While the edge detection method is a method used to detect the edge of the object. The results from these two methods were then compared using the AND function, so that obtained the condition of an empty or occupied for each box of parking lot. Threshold values affect the determination of the parking lot. In This research, the best detection result obtained by the detection of the place occupied in the parking lot located in Malang TownSquare (Matos), with a threshold of 10 and accuracy of 99.3%.

Keywords: car parking, detection, vehicle detection, edge detection, AND function, threshold.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayahnya, Tugas Akhir yang berjudul "Deteksi Tempat Kosong Pada Lahan Parkir Mobil Menggunakan Metode *Vehicle Detection* dan *Edge Detection*" ini dapat diselesaikan. Tugas Akhir ini disusun dan diajukan sebagai syarat untuk memperoleh gelar sarjana pada program studi Ilmu Komputer, jurusan Matematika, fakultas MIPA, universitas Brawijaya.

Dalam penyelesaian tugas akhir ini, penulis telah mendapat begitu banyak bantuan baik moral maupun materiil dari banyak pihak. Atas bantuan yang telah diberikan, penulis ingin menyampaikan penghargaan dan ucapan terima kasih kepada:

1. Drs. Muh Arif Rahman, M.Kom dan Drs. Marji, MT., selaku dosen pembimbing yang telah membimbing dengan baik dalam penyusunan skripsi ini.
2. Dr. Abdul Rouf Alghofari, M.Sc., selaku Ketua Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Brawijaya.
3. Drs. Marji, .MT, selaku Ketua Program Studi Ilmu Komputer, Jurusan Matematika, FMIPA Universitas Brawijaya pembimbing pendamping dalam penulisan tugas akhir.
4. Nurul Hidayat, S.Pd, M.Sc, selaku penasehat akademik.
5. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada Penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
6. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu Penulis dalam pelaksanaan penyusunan tugas akhir ini.

7. Mama, Bapak, Mas Leksa, Mbak Ula, Mas Juta, Mbak Oka, Asa, dan seluruh keluarga besar atas kasih sayang, dukungan, semangat, dan doa yang tiada henti.
8. Arie Rachmad Syulistyo yang dengan sabar selalu menemani, memberikan dukungan, bantuan, doa, dan semangat.
9. Tari, Siwi, Lia, Fafan, Rizky, Mas Adi, dan rekan-rekan di Program Studi Ilmu Komputer FMIPA Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan tugas akhir ini.
10. Dan semua pihak yang telah membantu dalam penyusunan tugas akhir ini yang tidak dapat Penulis sebutkan satu per satu

Semoga penulisan laporan tugas akhir ini bermanfaat bagi pembaca sekalian. Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan sehingga penulis mengharapkan kritik dan saran yang membangun dari pembaca.

Malang, 16 Juni 2011

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR *SOURCECODE*

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1. LATAR BELAKANG

Saat ini jumlah kendaraan bermotor, baik beroda dua atau lebih, semakin menunjukkan perkembangan yang signifikan, terlebih di kota-kota besar. Perkembangan jumlah kendaraan ini, secara tidak langsung berdampak pada kebutuhan lahan parkir di beberapa fasilitas umum, seperti perkantoran, rumah sakit, pusat perbelanjaan, kampus, atau tempat-tempat rekreasi. Fasilitas-fasilitas umum tersebut perlu melakukan perluasan lahan parkir yang dimiliki agar dapat menampung seluruh kendaraan pengunjung. Tetapi, meskipun sudah disediakan lahan parkir yang cukup luas, masih sering dijumpai antrian panjang seperti di pusat perbelanjaan atau tempat-tempat rekreasi pada saat libur akhir pekan atau libur panjang sekolah. Antrian panjang pada lahan parkir ini terjadi karena pengunjung harus berputar-putar mencari tempat kosong pada lahan parkir tersebut.

Perkembangan teknologi khususnya di bidang komputer memberikan banyak kemudahan bagi manusia. Saat ini hampir setiap kantor, perusahaan, industri, rumah tangga, atau sekolah sudah memanfaatkan komputer sebagai alat bantu utama untuk berbagai keperluan. Salah satu bentuk pemanfaatan teknologi komputer untuk mempermudah aktivitas manusia adalah untuk pengelolaan lahan parkir. Adanya sistem manajemen parkir yang telah terkomputerisasi dapat memudahkan pengunjung untuk mencari tempat kosong pada lahan parkir. Sistem manajemen parkir mengacu pada program yang dapat menghasilkan penggunaan sumber daya parkir yang lebih efisien (Litman, 2008). Saat ini telah dikembangkan beberapa teknologi tempat parkir untuk menggantikan cara tradisional yang hanya berupa tanda jumlah tempat penuh atau kosong pada papan yang terletak pada pintu masuk lahan parkir. Penyampaian informasi parkir yang telah dikembangkan diantaranya adalah melalui telpon genggam, *Personal Data Assistants* (PDAs), *Variable Message Display* (VMS), *Parking Guidance and Information System* (PGIS)

yang menampilkan jumlah tempat parkir yang kosong dan terisi, atau melaui *Urban Traffic Management and Control* (UTMC) (Idris .dkk, 2009). Contoh dari sistem yang telah dikembangkan dapat dilihat pada Gambar 1.1 dan Gambar 1.2.

Gambar 1.1. Penggunaan PGIS di Guangzhou, China (*Sumber: www.pcsvision.com.my*)

Gambar 1.2. Penggunaan VMS di Senayan City, Jakarta (*Sumber: id.wikipedia.org)*

Terdapat empat jenis kategori sistem petunjuk parkir mobil berdasarkan teknologi yang berbeda, yaitu berbasis perhitungan jumlah kendaraan, sensor kabel, sensor *wireless*, dan berbasis gambar. Diantara keempat kategori tersebut, teknologi berbasis gambar merupakan teknologi yang memiliki banyak kelebihan dibandingkan tiga teknologi lainnya. Dengan sistem ini (teknologi berbasis gambar), pengunjung dapat mengetahui secara pasti dan terperinci mengenai letak tempat yang kosong. Selain itu sistem ini tidak membutuhkan biaya yang besar untuk pengembangannya karena hanya memerlukan kamera CCTv yang sebelumnya sudah banyak dipasang di beberapa tempat untuk tujuan keamanan (Bong, Ting, dan Lai, 2008).

Pada penelitian ini digunakan gambar lahan parkir mobil yang diambil menggunakan kamera digital, sebagai pengganti CCTv, secara berkala dalam beberapa waktu yang berbeda. Untuk proses pengolahan gambar sehingga didapatkan informasi tempat kosong

pada lahan parkir mobil, digunakan beberapa teknik *image processing*. *Image processing* merupakan salah satu dari bentuk pemrosesan informasi dengan inputan berupa citra (*image*) dan keluaran yang juga berupa citra atau dapat juga bagian dari citra tersebut yang bertujuan untuk memperbaiki kualitas citra (Indra, 2010). Metode pada *image processing* yang digunakan untuk mendeteksi tempat kosong pada lahan parkir mobil adalah deteksi kendaraan atau obyek pada tempat parkir, dengan menggunakan nilai ambang batas (*threshold*) yang ditetapkan untuk deteksi piksel pada nilai tertentu. Metode yang kedua adalah mengurangi kesalahan deteksi yang mungkin timbul dikarenakan bayangan yang berasal dari kendaraan. Tanpa metode ini, sistem dapat salah mendeteksi bayangan kendaraan sebagai kendaraan yang sedang parkir. Pada metode ini digunakan teknik *median filtering* dan *sobel edge detection*. *Median filtering* merupakan filter spasial yang digunakan untuk mereduksi *noise*, dimana nilai keabuan dari titik-titik pada matriks *sub image* diurutkan dari nilai terkecil sampai dengan terbesar kemudian ditentukan nilai mediannya (Winarno, 2009). Sedangkan *sobel edge detection* merupakan salah satu metode dalam *image processing* yang berguna untuk mendeteksi tepi (*edge*) suatu obyek dalam gambar *digital* (Gonzalez, 2002). Kedua metode tersebut kemudian dibandingkan dengan operasi *AND function* untuk menentukan suatu tempat parkir kosong atau terisi kendaraan. Dengan adanya sistem informasi parkir yang dikembangkan dengan teknologi berbasis gambar ini, diharapkan dapat meminamilisir waktu pencarian tempat parkir oleh pengunjung sehingga tidak terjadi antrian panjang pada pintu masuk lahan parkir.

Berdasarkan latar belakang yang telah diuraikan, maka dilakukan penelitian dengan judul "Deteksi Tempat Kosong Pada Lahan Parkir Mobil Menggunakan Metode *Vehicle Detection* dan *Edge Detection*".

PERUMUSAN MASALAH

Perumusan masalah pada penelitian ini adalah sebagai berikut.

1. Bagaimana merancang sebuah sistem pendeteksian tempat kosong pada lahan parkir mobil menggunakan metode *vehicle detection* dan *edge detection*.
2. Bagaimana tingkat keakuratan gambar yang dihasilkan dari proses pendeteksian tempat kosong pada lahan parkir mobil dengan gambar asli.

1.2. BATASAN MASALAH

Batasan permasalahan pada penelitian ini adalah sebagai berikut.

1. Data yang digunakan untuk pengujian sistem adalah *file* citra dengan format bitmap.
2. Jenis lahan parkir pada penelitian ini adalah lahan parkir mobil terbuka yang memiliki garis pembatas di setiap tempat parkirnya.
3. Gambar lahan parkir yang digunakan untuk pengujian sistem diambil dari sisi atas depan, belakang, atau samping tempat parkir.
4. Keadaan cahaya pada saat pengambilan gambar lahan parkir turut diperhitungkan dalam pembentukan sistem.
5. Sistem hanya dapat menentukan tempat parkir terisi obyek atau tidak. Sistem tidak dapat menentukan jenis obyek yang berada di tempat parkir.

1.3. TUJUAN PENELITIAN

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut.

1. Merancang sebuah sistem pendeteksian tempat kosong pada lahan parkir mobil menggunakan metode *vehicle detection* dan *edge detection*.
2. Mengetahui tingkat keakuratan hasil deteksi pada lahan parkir.

1.4. MANFAAT PENELITIAN

Manfaat dari penelitian ini adalah sebagai berikut.

1. Dapat mempermudah pengunjung fasilitas umum untuk menemukan tempat yang kosong pada lahan parkir mobil.
2. Dapat mempersingkat waktu pencarian tempat kosong sehingga dapat mengurangi antrian kendaraan di pintu masuk lahan parkir saat padat pengunjung.
3. Dapat memudahkan petugas parkir untuk mengkoordinasikan kendaraan di lahan parkir, terlebih pada saat padat pengunjung.

1.5. METODOLOGI PENELITIAN

Metodologi yang digunakan dalam penelitian ini adalah sebagai berikut.

1. Studi Literatur
   Mempelajari teori-teori yang berhubungan dengan konsep pengolahan citra digital dan teknik-teknik yang digunakan pada penelitian ini, seperti *vehicle detection*, *sobel edge detection*, *median filtering*, dan *AND function*, dari berbagai referensi.
2. Pendefinisan dan Analisis Masalah
   Mendefinisikan dan menganalisis masalah yang terdapat pada penelitian untuk mencari solusi yang tepat.
3. Perancangan dan Implementasi Sistem
   Membuat perancangan perangkat lunak dengan analisis terstruktur dan mengimplementasikan hasil rancangan tersebut dengan membuat sebuah system deteksi tempat kosong pada lahan parkir mobil.
4. Uji Coba dan Analisis Hasil Implementasi
   Menguji perangkat lunak yang telah dibuat dan menganalisis hasil dari implementasi tersebut, apakah sudah sesuai dengan tujuan yang telah ditetapkan sebelumnya, untuk selanjutnya dievaluasi dan disempurnakan.

1.6. SISTEMATIKA PENULISAN

Untuk memberikan gambaran mengenai penelitian yang dilakukan, berikut ini dipaparkan garis besar dari keseluruhan isi laporan penelitian untuk setiap bab.

BAB I: PENDAHULUAN

Bab ini membahas mengenai latar belakang penelitian, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metode penelitian, dan sistematika penulisan.

BAB II: TINJAUAN PUSTAKA

Bab ini berisi penjelasan singkat mengenai dasar teori yang berkaitan dengan penelitian, diantaranya konsep dasar pengolahan citra digital, *sobel edge detection*, *median filtering*, dan *AND function*.

BAB III: METODE DAN PERANCANGAN

Bab ini berisi metode-metode yang digunakan dalam penelitian dan langkah-langkah yang harus dilakukan. Beberapa hal yang dibahas adalah: subyek penelitian, perangkat lunak dan perangkat keras, gambaran sistem, pengumpulan data, dan rancangan penelitian.

BAB IV: HASIL DAN PEMBAHASAN

Bab ini berisi penjelasan implementasi dari rancangan yang telah diuraikan pada Bab III dan hasil pengujian yang dilakukan. Implementasi yang dijelaskan berupa implementasi program. Selain itu bab ini juga menjelaskan penerapan aplikasi, analisis hasil uji coba berupa tingkat keakuratan yang didapatkan dari perbandingan jumlah data yang berhasil dideteksi dan dikenali, dengan jumlah data yang gagal dideteksi dan dikenali.

BAB V: PENUTUP

Bab ini berisi kesimpulan yang diperoleh dari hasil pembahasan dan saran yang diharapkan bermanfaat untuk pengembangan penelitian yang lebih lanjut.

BAB II

TINJAUAN PUSTAKA

2.1. Konsep Dasar Pengolahan Citra Digital

Citra digital adalah sebuah fungsi 2D, f(x,y), yang merupakan fungsi intensitas cahaya, dimana nilai x dan y merupakan koordinat spasial dan nilai fungsi di setiap titik (x,y) merupakan tingkat keabuan citra pada titik tersebut (Irawan, 2006). Citra digital dinyatakan dengan sebuah matriks dimana baris dan kolomnya menyatakan suatu titik pada citra tersebut dan elemen matriksnya – yang disebut sebagai elemen gambar atau piksel – menyatakan tingkat keabuan pada titik tersebut. Matriks dari citra digital berukuran NxM (tinggi x lebar), dimana:

N = jumlah baris $0 < y \leq N - 1$
M = jumlah kolom $0 \leq x \leq M - 1$
L = derajat keabuan $0 \leq f(x,y) \leq L - 1$

Berikut ini adalah gambaran matriks dari citra digital:

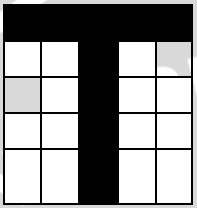
$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix} \quad (2.1)$$

Dimana indeks baris (x) dan indeks kolom (y) menyatakan suatu koordinat titik pada citra, sedangkan f(x,y) merupakan intensitas (derajat keabuan) pada titik (x,y).

Berdasarkan jenisnya, citra digital dapat dibagi menjadi 3, yaitu (Sutoyo, 2009):

1. Citra Biner (Monokrom)

Memiliki 2 buah warna, yaitu hitam dan putih. Warna hitam bernilai 1 dan warna putih bernilai 0. Untuk menyimpan kedua warna ini dibutuhkan 1 bit di memori. Contoh dari susunan piksel pada citra monokrom dapat dilihat pada gambar 2.1.



= 1 1 1 1 1
= 0 0 1 0 0
= 0 0 1 0 0
= 0 0 1 0 0
= 0 0 1 0 0

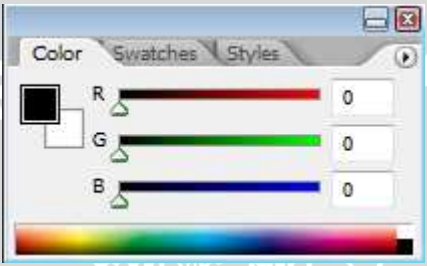
Gambar 2.1. Citra Monokrom

2. Cita Keabuan (*Grayscale*)
   Citra *grayscale* mempunyai kemungkinan warna hitam untuk nilai minimal dan warna putih untuk nilai maksimal. Banyaknya warna tergantung pada jumlah bit yang disediakan di memori untuk menampung kebutuhan warna tersebut. Semakin besar jumlah bit warna yang disediakan di memori, maka semakin halus gradasi warna yang terbentuk. Contoh:
   skala keabuan 2 bit
   jumlah kemungkinan $2^2 = 4$ warna
   kemungkinan warna 0 (minimal) sampai 4 (maksimal)
   Gambar 2.2 merupakan gradasi warna keabuan (*grayscale*).

Gambar 2.2. Gradasi Warna Keabuan
(Sumber: http://www.imagingassociates.com.au)

3. Citra Warna (*true color*)

Setiap piksel pada citra warna mewakili warna yang merupakan kombinasi tiga warna dasar, yaitu merah, hijau, dan biru (RGB = *Red, Green, Blue*). Setiap warna dasar menggunakan penyimpanan 8 bit = 1 *byte* (nilai maksimum 255 warna), jadi satu piksel pada citra warna diwakili oleh 3 *byte*. Contoh warna RGB dapat dilihat pada gambar 2.3.



Gambar 2.3. Warna RGB (*Red, Green, Blue*)

Pengolahan citra digital adalah salah satu bentuk pemrosesan informasi dengan inputan berupa citra (*image*) dan keluaran yang juga berupa citra atau dapat juga bagian dari citra tersebut. Tujuan dari pemrosesan ini adalah memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau mesin komputer. Operasi-operasi pada pengolahan citra digital secara umum dapat diklasifikasikan sebagai berikut (Munir, 2004).

1. Perbaikan kualitas citra (*image enhancement*), contohnya perbaikan kontras gelap/terang, penajaman (*sharpening*), dan perbaikan tepian obyek (*edge enhancement*)
2. Restorasi citra (*image restoration*), contohnya penghilangan kesamaran (*deblurring*)
3. Pemampatan citra (*image compression*)
4. Segmentasi citra (*image segmentation*)
5. Pengorakan citra (*image analysis*), contohnya pendeteksian tepi obyek (*edge enhancement*) dan ekstraksi batas (*boundary*)
6. Rekonstruksi citra (*image recronstruction*)

Pada penelitian ini, operasi citra digital yang digunakan adalah *image restoration* dan *image analysis*.

2.2. Relasi Dasar Antar Piksel

Pada sebuah citra, satu piksel memiliki relasi (hubungan) dengan piksel yang lain. Bentuk dari relasi dasar antar piksel, antara lain adalah (Gonzalez, 2002):

2.2.1. Ketetanggaan (*neighbors*) dari suatu piksel

Sebuah piksel *p* pada koordinat (x,y) mempunyai empat tetangga vertikal dan horisontal, dimana koordinat tersebut dinyatakan oleh: (x-1,y), (x+1,y), (x,y-1), (x,y+1).

Satuan piksel ini dinamakan 4-*neighbors* dari *p* yang dinotasikan dengan N4(p). Sedangkan empat tetangga diagonal *p* mempunyai koordinat: (x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1). Satuan piksel dinamakan 8-*neighbors* dari p, yang dinotasikan dengan ND(p). Hubungan ketetanggaan antar piksel dapat dilihat pada gambar 2.4.

(x-1,y-1)	(x-1,y)	(x-1,y+1)
(x,y-1)	(x,y)	(x,y+1)
(x+1,y-1)	(x+1,y)	(x+1,y+1)

Gambar 2.4. Hubungan Ketetanggaan Antar Piksel

2.2.2. *Adjacency, Connectivity, Region,* dan *Boundary*

Dua buah piksel dinyatakan terhubung jika kedua piksel tersebut adalah tetangga dan tingkat keabuannya mencukupi suatu ukuran *similarity* (tingkat keabuannya sama).

V merupakan satuan nilai-nilai *gray-level* yang digunakan untuk menggambarkan *adjacency*. Untuk menetapkan *V* perlu dipertimbangkan tiga jenis *adjacency*, yaitu:

- 4-*adjacency*, dua piksel *p* dan *q* dengan nilai-nilai dari *V* adalah 4-*adjacency* jika *q* adalah bagian di dalam N4(p)
- 8-*adjacency*, dua piksel *p* dan *q* dengan nilai-nilai dari *V* adalah 8-*adjacent* jika *q* adalah bagian di dalam N8(p)
- m-*adjacency* (*mixed adjacency*), dua piksel *p* dan *q* dengan nilai-nilai dari *V* adalah m-*adjacent* jika:
    - *q* adalah bagian dari N4(p), atau
    - *q* adalah bagian dari ND(p) dan yang diset tidak mempunyai nilai piksel busur lingkaran dari *V*

R merupakan suatu subset piksel di dalam sebuah citra yang dipanggil sebagai suatu *region* dari citra jika *R* merupakan suatu *connected set*.

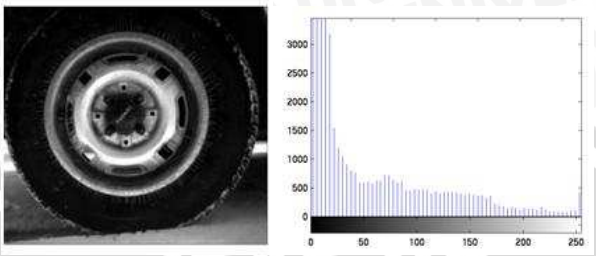
Boundary dari sebuah *region R* adalah satuan piksel di dalam *region* yang mempunyai satu atau lebih tetangga yang tidak termasuk di dalam *R*.

2.3. *Vehicle Detection*

Vehicle detection merupakan sebuah metode untuk mencari obyek pada gambar dengan cara menggabungkan tiga metode pada pengolahan citra digital, yaitu *histogram equalization*, *image subtraction*, dan binerisasi citra.

2.3.1. *Histogram Equalization*

Histogram merupakan sebuah diagram yang menunjukkan jumlah titik yang terdapat pada sebuah citra untuk setiap tingkat keabuan (Hestiningsih, 2009). Sumbu x (*absis*) pada histogram menunjukkan tingkat warna, sedangkan sumbu y (*ordinat*) menunjukkan frekuensi kemunculan titik. Gambar 2.5 merupakan contoh gambar dan histogramnya.



Gambar 2.5. Gambar ban mobil dan histogramnya
(Sumber: http://picasaweb.google.co.id)

Histogram dari sebuah citra dapat dimodifikasi untuk memperoleh histogram citra yang sesuai dengan keinginan. Salah satu cara yang dapat digunakan untuk memodifikasi histogram citra adalah perataan histogram (*histogram equalization*). *Histogram equalization* adalah sebuah proses yang mengubah distribusi nilai derajat keabuan pada sebuah citra sehingga menjadi seragam (*uniform*).

Tujuan dari *histogram equalization* adalah untuk memperoleh penyebaran histogram yang merata sehingga setiap derajat keabuan memiliki jumlah piksel yang relatif sama. Perataan histogram diperoleh dengan cara mengubah derajat keabuan sebuah piksel (r) dengan derajat keabuan yang baru (s) dengan sebuah fungsi transformasi T (Gonzalez, 2002). Secara matematis dapat ditulis seperti pada persamaan 2.2.

$$s = T(r) \qquad (2.2)$$

r dapat diperoleh kembali dari s dengan transformasi *invers* seperti pada persamaan 2.3.

$$r = T^{-1}(s), \text{ dimana } 0 \leq s \leq 1 \qquad (2.3)$$

Rumus yang digunakan untuk menghitung *histogram equalization* dapat ditulis seperti pada persamaan 2.4.

$$P_r(r_k) = \frac{n_k}{n} \; dalam \; hal \; ini \; r_k = \frac{k}{L-1}, 0 \leq k \leq L-1 \qquad (2.4)$$

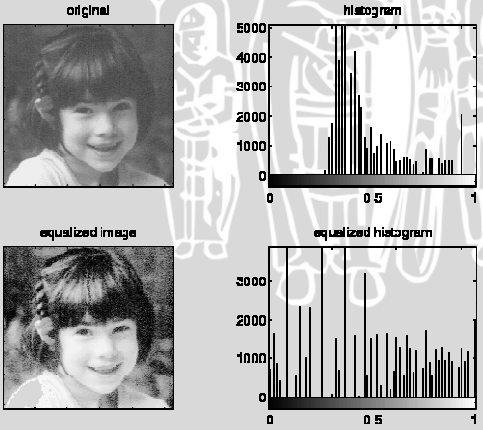
dimana n_k adalah nilai piksel pada derajat keabuan k, dan n adalah jumlah seluruh piksel pada citra. Dari perumusan tersebut dapat diartikan bahwa derajat keabuan (k) dinormalkan terhadap derajat keabuan (L-1). Nilai $r_k = 0$ menyatakan hitam, dan $r_k = 1$ menyatakan putih dalam skala keabuan yang didefinisikan.
Rumus lain yang dapat digunakan untuk menghitung *histogram equalization* pada citra dengan skala keabuan *k* bit adalah seperti pada persamaan 2.5.

$$K_o = round\left(\frac{C_i.(2^k-1)}{w.h}\right) \qquad (2.5)$$

dimana:

C_i = distribusi kumulatif dari nilai skala keabuan ke –*i* dari citra asli
round = fungsi pembulatan ke bilangan yang terdekat
K_o = nilai keabuan hasil *histogram equalization*
w = lebar citra
h = tinggi citra

Gambar 2.6. merupakan contoh histogram citra asli yang belum diratakan dan histogram citra setelah proses *equalization*.



Gambar 2.6. Citra Asli dan Citra Hasil *Equalization*
(Sumber: http://homepages.inf.ed.ac.uk)

2.3.2. *Image Subtraction*

Deteksi perbedaan dua buah citra yang hampir sama dapat dilakukan dengan operasi pengurangan atau biasa disebut dengan *image subtraction* (Munir, 2004). Untuk mencari perbedaan tersebut yang harus dilakukan adalah mengambil citra pertama dan citra kedua, kemudian salah satu citra dikurangkan terhadap citra yang lain. Secara matematis ditulis seperti pada persamaan 2.6.

$$g(x,y) = f(x,y) - h(x,y) \qquad (2.6)$$

g merupakan citra baru yang intensitas setiap pikselnya adalah selisih antara intensitas piksel *f* dan *h* (Gonzales, 2002). Contoh penerapan operasi pengurangan citra adalah untuk memperoleh suatu obyek dari dua buah citra. Pengurangan citra juga dapat digunakan untuk mendeteksi perubahan yang terjadi selama selang waktu tertentu apabila dua buah citra yang diambil adalah citra dari sudut/tempat yang sama.

2.3.3. Binerisasi dengan Metode *Otsu*

Binerisasi merupakan sebuah proses yang digunakan untuk mengubah citra dengan format skala keabuan (yang mempunyai kemungkinan nilai lebih dari 2) ke citra biner yang hanya memiliki 2 buah nilai, yaitu 0 dan 1. Operasi yang akan digunakan pada proses binerisasi adalah operasi ambang batas (*thresholding*), yaitu mengubah piksel-piksel obyek pada citra keabuan menjadi piksel-piksel dengan intensitas maksimum (255) dan mengubah piksel-piksel latar belakang pada citra keabuan menjadi piksel-piksel dengan intensitas minimum (0) pada citra biner, atau sebaliknya (obyek dengan nilai intensitas 0 dan latar belakang dengan nilai intensitas 255 pada citra biner yang dihasilkan) (Ahmad, 2009). Pada operasi *thresholding*, nilai piksel yang memenuhi syarat ambang batas dipetakan ke suatu nilai yang dikehendaki.

Terdapat dua tipe *thresholding*, yaitu: *single threshold* dan *multiple threshold* (Gonzalez, 2002). Pada *single threshold*, piksel obyek dan *background* mempunyai *gray level* yang dikelompokkan menjadi dua mode yang dominan. Salah satu cara untuk memisahkan

obyek dari *background* adalah dengan memilih nilai batas T. Titik (x,y) dimana f(x,y) > T, maka disebut *object point* dan titik tersebut merupakan *background point*.

Pada *multiple threshold*, piksel obyek dan *background*nya mempunyai *gray level* yang dikelompokkan menjadi tiga mode dominan, yaitu menggolongkan satu titik (x,y) sebagai *object class* jika T1 < (x,y) ≤ T2, *object class* yang lain jika f(x,y) > T2, dan merupakan *background* jika f(x,y) ≤ T1.

Fungsi T pada *thresholding*: T = T[x,y,p(x,y),f(x,y)], dimana f(x,y) adalah nilai *gray level* di titik (x,y) dan p(x,y) menunjukkan beberapa *local property* pada titik tersebut.

Batasan untuk citra g(x,y) dinyatakan pada persamaan 2.7.

$$g(x,y) = \begin{cases} 1 & jika\ f(x,y) > T \\ 0 & jika\ f(x,y) \leq T \end{cases} \qquad (2.7)$$

Nilai *threshold* T yang optimal dapat dicari dengan menggunakan metode *Otsu*. Tujuan dari metode *Otsu* adalah membagi histogram citra *gray level* ke dalam dua daerah yang berbeda secara otomatis. Pendekatan yang dilakukan oleh metode *Otsu* adalah dengan melakukan analisis diskriminan, yaitu menentukan suatu variabel yang dapat membedakan antara dua atau lebih kelompok yang muncul secara alami. Analisis diskriminan akan memaksimumkan variabel tersebut agar dapat membagi obyek latar depan (*foreground*) dan latar belakang (*background*).

Misalkan diberikan citra f(x, y) dan nilai threshold adalah j, dimana j berada dalam kisaran 0 ≤ j ≤ m-1 (m merupakan derajat keabuan citra). Kemudian citra dibagi menjadi dua kelompok, kelompok A dengan nilai derajat keabuan kurang dari sama dengan j dan kelompok B dengan nilai derajat keabuan lebih besar dari j. Dua kelompok tersebut disimbolkan dengan ($\alpha_0(j)$, $\mu_0(j)$) dan ($\alpha_1(j)$, $\mu_1(j)$) yang merupakan jumlah dari piksel dan nilai rata-rata derajat keabuan untuk kelompok A dan B. Rumus dari fungsi tersebut dapat dituliskan seperti pada persamaan 2.8 – 2.12 (Fang Mei, dkk, 2009).

$$\alpha_0(j) = \sum_{i=0}^{j} n_i \qquad (2.8)$$

$$\mu_0(j) = \frac{\sum_{i=0}^{j}(i.n_i)}{\alpha_0} = \frac{\mu_j}{\alpha_0} \qquad (2.9)$$

$$\alpha_1(j) = 1 - \omega_1 \qquad (2.10)$$

$$\mu_1(j) = \frac{\sum_{i=j+1}^{m-1}(i.n_i)}{\alpha_1} = \frac{\mu - \mu_j}{1 - \alpha_0} \qquad (2.11)$$

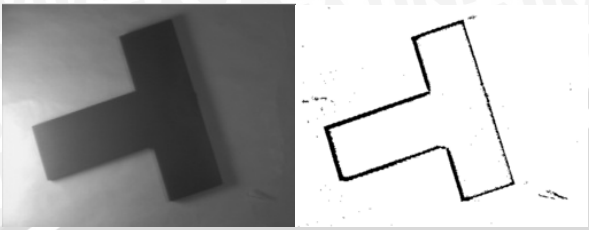
dimana

$$\mu = \sum_{i=0}^{m-1} i n_i \;, \qquad \mu_j = \sum_{i=0}^{j} i n_i \qquad (2.12)$$

n_i adalah jumlah piksel dengan nilai gray level yang ke i. Jarak varian antara dua kelompok, dilambangkan dengan $\sigma_B^2(j)$. Fungsi dari jarak varian antara dua kelompok dapat ditulis seperti pada persamaan 2.13.

$$\sigma_B^2(j) = \alpha_0(j)(\mu_0 - \mu)^2 + \alpha_1(j)(\mu_1 - \mu)^2$$
$$= \alpha_0\alpha_1(j)(\mu_0 - \mu_1)^2 \qquad (2.13)$$

Range untuk j adalah dari 0 sampai m-1, maka setiap $\sigma_B^2(j)$ dihitung menggunakan fungsi 2.13, dan nilai j menunjuk pada nilai jarak varian terbesar yang merupakan hasil dari *threshold* T. Contoh citra yang telah dikonversi menjadi citra biner dapat dilihat pada gambar 2.7.



Gambar 2.7. Citra *graysclae* dan citra hasil binerisasi
(Sumber:http://www.mathworks.com)

2.4. *Edge Detection*

Edge Detection merupakan metode untuk mencari tepian obyek pada citra dengan menggabungkan beberapa metode pada pengolahan citra digital, yaitu konversi citra warna menjadi keabuan, *histogram equalization*, *median filtering*, *sobel edge detection*, dan binerisasi. *Edge Detection* adalah metode kedua setelah *vehicle detection* yang digunakan untuk menentukan kosong atau tidaknya suatu tempat parkir. Untuk metode *histogram equalization* dan binerisasi yang digunakan pada tahap ini sama seperti yang telah dijelaskan pada sub bab 2.3.1 dan 2.3.3.

2.4.1. Konversi Citra Warna Menjadi Citra Keabuan (*Grayscale*)

Citra warna dapat diubah menjadi citra keabuan dengan cara menghitung rata-rata setiap elemen warna, yaitu *Red*, *Green*, dan *Blue*. Secara matematis, perhitungan untuk konversi citra warna menjadi citra keabuan dapat dirumuskan seperti pada persamaan 2.14.

$$f_o(x,y) = \frac{f_i^R(x,y) + f_i^G(x,y) + f_i^B(x,y)}{3} \quad \ldots\ldots\ldots(2.14)$$

dimana:

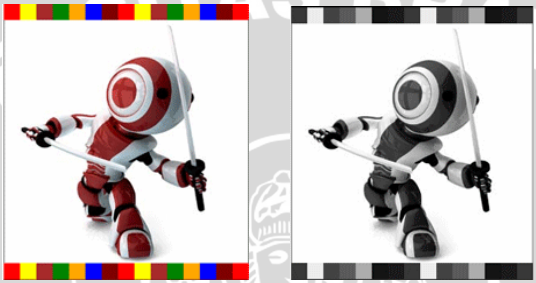
$f_o(x,y)$ = Derajat keabuan *o* di titik (x,y)

$f_i^R(x,y)$ = Nilai komponen warna merah pada derajat keabuan *i* di titik (x,y)

$f_i^G(x,y)$ = Nilai komponen warna hijau pada derajat keabuan *i* di titik (x,y)

$f_i^B(x,y)$ = Nilai komponen warna biru pada derajat keabuan *i* di titik (x,y)

Contoh gambar berwarna yang telah dikonversi menjadi citra *grayscale* dapat dilihat pada Gambar 2.8.



Gambar 2.8. Citra Warna dan Citra *Grayscale* Hasil Konversi
(Sumber: http://james.padolsey.com)

2.4.2. *Median Filtering*

Noise pada sebuah citra dapat terjadi karena karakteristik derajat keabuan (*gray-level*) atau dikarenakan adanya variabel acak yang terjadi karena karakteristik Fungsi Probabilitas Kepadatan (*Probability Density Function* (PDF)). Apabila citra yang mengandung *noise* langsung diproses dan diekstrak, maka fitur-fitur pentingnya dapat menimbulkan masalah akurasi. Jadi sebaiknya citra tersebut dibersihkan dari *noise* terlebih dahulu, dan kemudian diproses untuk diekstrak fitur-fitur pentingnya. Salah satu teknik untuk mereduksi *noise* adalah *order-statistics filters*, yang merupakan filter spasial dimana hasil responsnya didasarkan pada pengurutan nilai piksel yang dilingkupi oleh filter (Gonzalez, 2002).

Median filtering merupakan *order-statistics filter* yang paling dikenal. Cara kerja filter ini dirumuskan pada persamaan 2.15.

$$\hat{f}(x,y) = median_{(s,t)\in S_{xy}}\{g(s,t)\} \qquad (2.15)$$

Median filtering mengambil area tertentu pada citra sesuai dengan ukuran mask yang telah ditentukan (umumnya berukuran 3x3), kemudian dilihat setiap nilai piksel pada area tersebut, dan nilai tengah pada area diganti dengan nilai median (Dwayne, 2000). Cara memperoleh nilai median, yaitu nilai keabuan dari titik-titik pada matriks diurutkan dari nilai terkecil hingga yang terbesar, kemudian ditentukan nilai yang berada di tengah dari deret piksel.

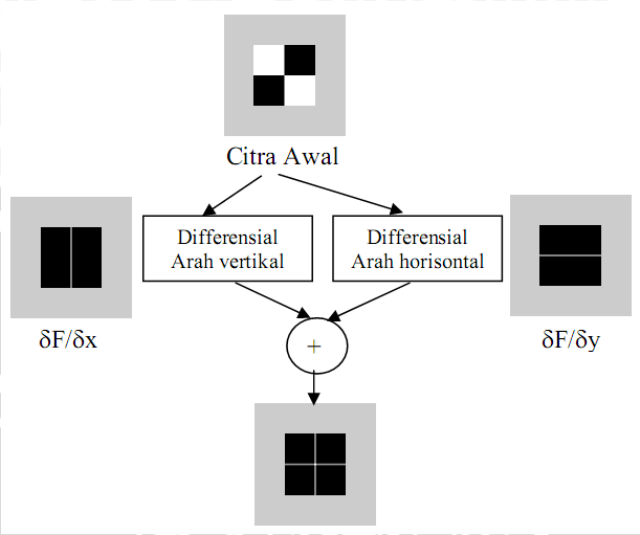
Untuk tipe-tipe *noise* tertentu, filter ini memberikan kemampuan reduksi *noise* yang sangat baik, dengan *blurring* yang lebih sedikit daripada *linear smoothing filter* untuk ukuran citra yang sama. *Median filtering* memberikan hasil yang sangat bagus untuk citra yang terkena *noise impulse bipolar* dan *unipolar*. Contoh hasil reduksi *noise* menggunakan *median filtering* dapat dilihat pada gambar 2.9.

Gambar 2.9. Citra dengan *Noise* dan Citra Hasil *Median Filtering*

(Sumber: http://homepages.inf.ed.ac.uk)

2.4.3. *Sobel Edge Detection*

Tepi obyek adalah pertemuan antara bagian obyek dengan bagian latar belakang. Dalam pengolahan citra, tepi obyek ditandai oleh titik yang nilai keabuannya memiliki perbedaan yang cukup besar dengan titik yang ada di sebelahnya (tetangganya). Sedangkan deteksi tepi (*edge detection*) adalah sebuah proses untuk menemukan perubahan intensitas yang berbeda dalam sebuah citra. Gambar 2.10 menunjukkan proses diperolehnya suatu obyek.



Gambar 2.10. Proses Deteksi Tepi Citra

Deteksi tepi dapat dilakukan dengan menghitung selisih antara dua buah titik yang bertetangga sehingga didapat besar gradien citra. Gradien adalah turunan pertama dari persamaan dua dimensi yang didefinisikan sebagai vektor seperti pada persamaan 2.16.

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \qquad (2.16)$$

Besar gradien dihitung dengan persamaan 2.17.

$$G[f(x,y)] = \sqrt{G_x^{\,2} + G_y^{\,2}} \qquad (2.17)$$

Operator Sobel merupakan salah satu operator gradien pertama yang dapat digunakan untuk mendeteksi tepi di dalam citra. Operator sobel melakukan pengukuran spasial 2D pada sebuah citra dan menguatkan daerah-daerah yang memiliki gradien spasial tinggi

untuk setiap *edge* yang bersangkutan. Operator ini menggunakan kernel ukuran 3x3 piksel untuk perhitungan gradien sehingga perkiraan gradien berada tepat di tengah matriks.

Operator sobel adalah *magnitude* dari gradien yang dihitung dengan rumusan pada persamaan 2.18.

$$M = \sqrt{S_x^2 + S_y^2} \quad \text{atau} \quad |M| = |S_x| + |S_y| \qquad (2.18)$$

dimana M adalah besar gradien di titik tengah kernel. Dua komponen dari gradien diproses dan ditambahkan dalam sebuah *single pass* dari gambar masukan dengan menggunakan operator *pseudo-convolution*. Gambar 2.11 menunjukkan matriks *pseudo-convolution*.

a_0	a_1	a_2
a_3	a_4	a_5
a_6	a_7	a_8

Gambar 2.11. Matriks *Pseudo-Convolution*

Sedangkan turunan spasial dihitung dengan persamaan 2.19 dan 2.20.

$$S_x = (a_6 + ca_7 + a_8) - (a_0 + ca_1 + a_2) \qquad (2.19)$$

$$S_y = (a_2 + ca_5 + a_8) - (a_0 + ca_3 + a_6) \qquad (2.20)$$

dimana c adalah konstanta yang bernilai 2, sedangkan S_x dan S_y diimplementasikan menjadi kernel pada gambar 2.12.

S_x =

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

S_y =

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

Gambar 2.12. Kernel S_x dan S_y

Dari kernel tersebut dapat dilihat bahwa operator sobel menggunakan pembobotan pada piksel-piksel yang lebih dekat dengan titik pusat kernel. Oleh karena itu, pengaruh piksel-piksel tetangga akan berbeda sesuai dengan letaknya terhadap titik dimana gradient dihitung. Contoh hasil deteksi tepi menggunakan operator sobel dapat dilihat pada gambar 2.13.

Gambar 2.13. Gambar Asli dan Hasil *Sobel Edge Detection*
(Sumber: http://www.pages.drexel.edu)

2.5. Operator Logika AND

Operator logika diperoleh dari aljabar Boolean, yang merupakan sebuah cara matematis untuk merepresentasikan suatu konsep tanpa mengubah arti dari konsep tersebut secara umum (codesource.net, 2010). Pada konsep aljabar Boolean hanya terdapat dua kemungkinan nilai kebenaran, yaitu benar atau salah. Untuk menyatakan nilai kebenaran dari setiap kemungkinan kombinasi yang ada, terdapat 7 jenis operator logika, yaitu AND, NAND, NOT, NOR, OR, XOR, dan XNOR. Pada operator logika AND, sebuah kondisi dinyatakan benar jika kedua subfrase bernilai benar, dan dinyatakan salah jika kedua subfrase bernilai salah. Tabel 2.1 merupakan tabel kebenaran dari operator logika AND.

Tabel 2.1. Tabel Kebenaran Operator Logika AND

p	Q	p ^ q
0	0	0
0	1	0
1	0	0

1	1	1

2.6. Metode Pengujian

Bagian yang cukup penting dalam pendeteksian adalah menghitung tingkat akurasi hasil pendeteksian tersebut. Pada penelitian ini tingkat akurasi digunakan dalam menghitung beberapa akurasi pendeteksian, yaitu akurasi deteksi tempat kosong, akurasi deteksi tempat yang terisi, dan akurasi seluruh hasil deteksi yang benar. Persamaan yang digunakan untuk menghitung tingkat akurasi dapat dilihat pada persamaan 2.21 (Hsu, .dkk, 2002).

$$Tingkat\ akurasi\ (\%) = 1 - \frac{Jumlah\ hasil\ deteksi\ yang\ salah}{Jumlah\ seluruh\ tempat\ parkir} \quad (2.21)$$

BAB III

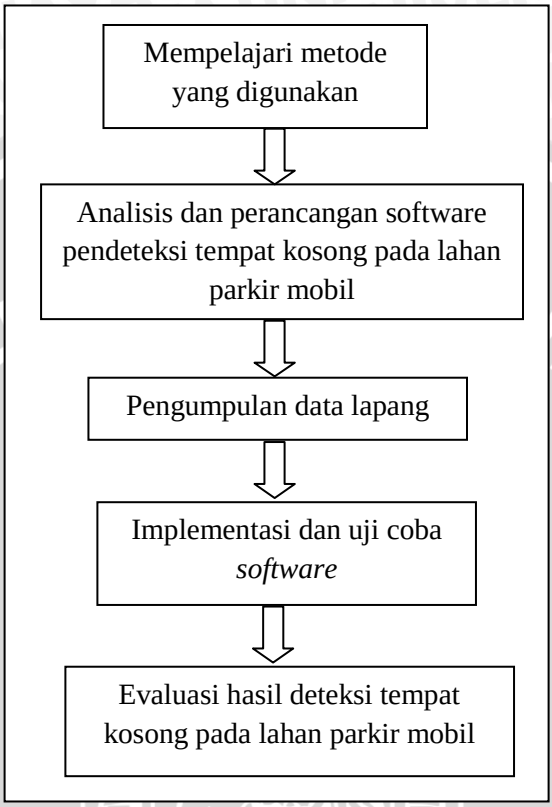
METODOLGI DAN PERANCANGAN

Bab metode dan perancangan ini berisi penjelasan mengenai metode yang digunakan dan langkah-langkah yang dilakukan dalam pembuatan perangkat lunak pendeteksi tempat kosong pada lahan parkir mobil.

Tahap-tahap yang dilakukan dalam penelitian ini adalah sebagai berikut:

1. Mempelajari metode yang digunakan pada pendeteksi tempat kosong pada lahan parkir mobil dari jurnal-jurnal terkait yang sudah ada. Metode-metode tersebut telah dipaparkan pada bab 2.
2. Menganalisa dan merancang perangkat lunak pendeteksi tempat kosong pada lahan parkir mobil dengan menggabungkan metode yang digunakan.
3. Mengumpulkan data lapang berupa gambar lahan parkir mobil terbuka di pusat perbelanjaan *Malang Town Square* (Matos), *Mall Olympic Garden* (MOG), dan data yang bersumber dari penulis jurnal berjudul "*Vacant Parking Space Detection in Static Image*", Nicholas True.
4. Membuat perangkat lunak berdasarkan analisis dan perancangan yang telah dilakukan.
5. Uji coba perangkat lunak pendeteksi tempat kosong pada lahan parkir mobil, dengan menggunakan data lapang yang telah dikumpulkan.
6. Evaluasi hasil deteksi yang dilakukan oleh perangkat lunak untuk mengetahui tingkat keakuratan gambar yang dihasilkan dengan gambar asli.

Langkah-langkah penelitian tersebut dapat diilustrasikan pada Gambar 3.1.



Gambar 3.1. Diagram Alir Tahap-Tahap Penelitian

3.1. Analisis Perangkat Lunak

Subbab analisis perangkat lunak berisi pembahasan mengenai semua hal yang diperlukan dalam proses perancangan, pembuatan, dan pengecekan perangkat lunak pada citra hasil deteksi.

3.1.1. Deskripsi Umum Perangkat Lunak

Perangkat lunak yang dikembangkan merupakan sebuah sistem yang dapat membantu *user* dalam melakukan pencarian tempat kosong pada lahan parkir mobil secara otomatis. *User* hanya

memasukkan data berupa citra lahan parkir mobil, kemudian sistem akan mengolah citra masukan tersebut hingga ditemukan tempat parkir mobil yang kosong.

Citra masukan yang digunakan pada pengujian merupakan citra lahan parkir mobil yang diambil dari tempat parkir mobil lantai 3 di pusat perbelanjaan *Malang Town Square* (Matos), *Mall Olympic Garden* (MOG), dan terdapat pula data yang diambil dari internet. Citra masukan berukuran 572 x 2275 untuk data dengan lokasi di matos, 572 x 322 untuk data dengan lokasi di MOG, dan 572 x 345 untuk data yang bersumber dari *Nicholas True*. Data citra masukan disimpan dalam file berbentuk bitmap. Metode-metode yang digunakan dalam proses pendeteksian tempat kosong pada lahan parkir mobil ini sepenuhnya berbasis pada pengolahan citra digital.

Pada saat *user* mengakses perangkat lunak, proses yang terjadi adalah sebagai berikut:

- *User* memilih citra lahan parkir mobil yang telah tersedia di dalam komputer sebagai citra masukan yang akan diproses oleh perangkat lunak.
- Proses *vehicle detection* dengan empat subproses, yaitu:
    - Sebaran warna pada citra masukan dibuat lebih merata dengan menggunakan metode *histogram equalization.*
    - *Image subtraction* dimana citra masukan dihitung perbedaan nilai pikselnya dengan citra lahan parkir yang kosong.
    - Citra hasil *image subtraction* diubah menjadi citra *grayscale.*
    - Citra *grayscale* diubah menjadi citra biner menggunakan *threshold* yang telah ditentukan.
- *Edge Detection*, pada proses ini citra masukan yang telah diubah menjadi citra *grayscale* akan ditingkatkan kualitasnya dengan metode *histogram equalization* untuk meratakan sebaran warna, *median filtering* untuk mereduksi *noise* yang ada pada citra, dan metode *sobel edge detection* untuk

mendeteksi tepian citra sehingga didapatkan hasil deteksi obyek pada citra.

- *AND function*, pada proses ini hasil deteksi pada proses *vehicle detection* dan *edge detection* akan dibandingkan sehingga didapatkan hasil akhir pendeteksian.

3.1.2. Batasan Perangkat Lunak

Batasan perangkat lunak yang dikembangkan adalah:

1. Citra lahan parkir diambil pada beberapa waktu yang berbeda, yaitu pagi, siang, sore, dan malam dengan tingkat pencahayaan yang berbeda.
2. Citra masukan yang diproses oleh perangkat lunak adalah citra yang bertipe bitmap.
3. Ukuran citra masukan adalah 572 x 275, 572 x 322, dan 572 x 345 piksel.
4. Ketersediaan tempat parkir yang berhasil dideteksi oleh perangkat lunak akan diberi tanda warna kuning untuk tempat parkir yang kosong dan warna merah untuk tempat parkir yang terisi mobil.

3.1.3. Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak didasarkan pada pembangunan sistem pendeteksi tempat kosong pada lahan parkir mobil yang baik serta dapat mudah diakses oleh user. Kebutuhan perangkat lunak meliputi hal-hal yang harus dipenuhi, yaitu:

- Sistem harus dapat mendeteksi tempat kosong pada lahan parkir mobil yang berasal dari citra masukan dalam berbagai kondisi sesuai dengan batasan dan algoritma yang telah ditentukan.
- *Interface* perangkat lunak yang dibangun bersifat *user friendly* sehingga dapat mudah dioperasikan oleh *user*.

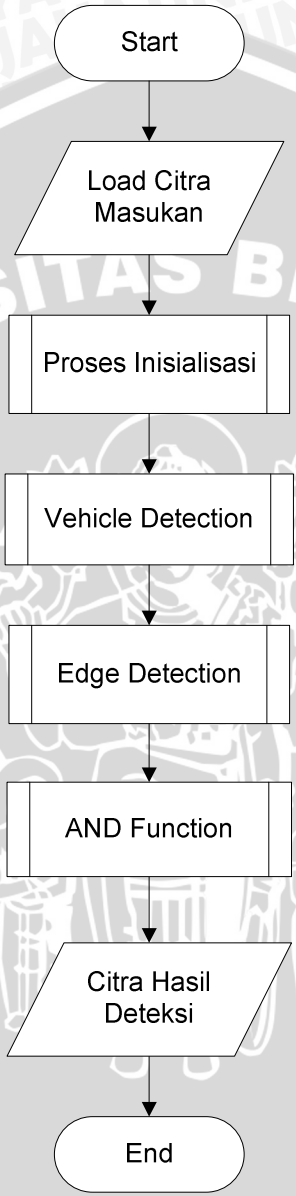
3.2. Perancangan Proses Perangkat Lunak

Berdasarkan analisis yang telah dilakukan, tahap selanjutnya adalah perancangan pembuatan perangkat lunak. Pada subbab ini dibahas proses-proses pada perangkat lunak yang saling berkaitan satu dengan yang lainnya. Selain itu dibahas juga mengenai perancangan uji coba dan perancangan antarmuka perangkat lunak.

Secara garis besar, perancangan proses pembuatan perangkat lunak terdiri dari 4 bagian, yaitu sebagai berikut.

1. Proses inisialisasi untuk mendeteksi garis pembatas di setiap tempat parkir melalui koordinat di setiap sudut yang disimpan pada file .txt
2. Proses *vehicle detection* yang merupakan metode pertama untuk mendeteksi keberadaan obyek pada tempat parkir yang terdiri dari dua subproses, yaitu *histogram equalization* dan *image subtraction*
3. Proses *edge detection* yang merupakan metode kedua untuk mendeteksi keberadaan obyek pada tempat parkir yang terdiri dari tiga subproses, yaitu *histogram equalization*, *median filtering* dan *sobel edge detection*
4. Proses yang terakhir adalah proses seleksi menggunakan *AND function* untuk mengkomparasikan dua metode sebelumnya sehingga didapatkan hasil akhir deteksi keberadaan obyek berupa mobil pada tempat parkir.

Urutan proses-proses pembuatan perangkat lunak secara umum tersebut dapat dilihat pada Gambar 3.2.



Gambar 3.2. Diagram Alir Pembuatan Perangkat Lunak

3.2.1. Proses Inisialisasi

Hasil dari tahap ini adalah koordinat setiap sudut tempat parkir yang disimpan pada file .txt. Dengan koordinat tersebut didapatkan lokasi yang tepat untuk setiap tempat parkir berupa garis pembatas. Proses yang dilakukan pada tahap ini adalah sebagai berikut.

1. Memilih koordinat titik sudut untuk setiap tempat parkir. Titik sudut merupakan perpotongan keempat garis yang ada pada kotak tempat parkir.
2. Memilih koordinat titik sudut untuk tempat parkir yang digunakan sebagai acuan dalam perhitungan *threshold*.
3. Menyimpan setiap koordinat yang telah dipilih pada file .txt.

Diagram alir proses inisialisasi sistem dapat dilihat pada Gambar 3.3.

3.2.2. Proses *Vehicle Detection*

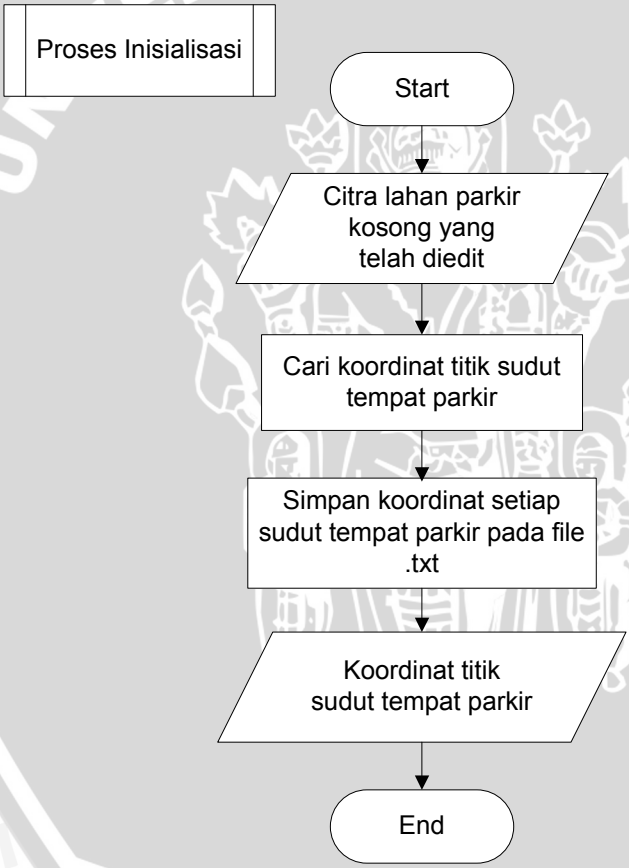
Proses *vehicle detection* memiliki tiga tahap utama, yaitu *histogram equalization, image subtraction,* dan binerisasi citra masukan. Tahap-tahap yang terdapat pada proses *vehicle detection* adalah sebagai berikut:

1. *Histogram Equalization* digunakan untuk meratakan sebaran warna RGB pada citra masukan
2. *Image subtraction* digunakan untuk mendeteksi keberadaan obyek pada citra masukan, yaitu dengan cara mengurangkan citra masukan lahan parkir yang di dalamnya terdapat obyek dengan citra referensi berupa citra lahan parkir yang kosong.
3. Citra hasil proses *image subtraction* diubah menjadi citra biner menggunakan *threshold* yang ditentukan dengan metode *Otsu thresholding*.
4. Menjumlahkan semua nilai piksel warna putih untuk setiap tempat parkir.
5. Proses seleksi dengan syarat, yaitu apabila hasil penjumlahan piksel putih bernilai lebih dari *threshold* yang telah

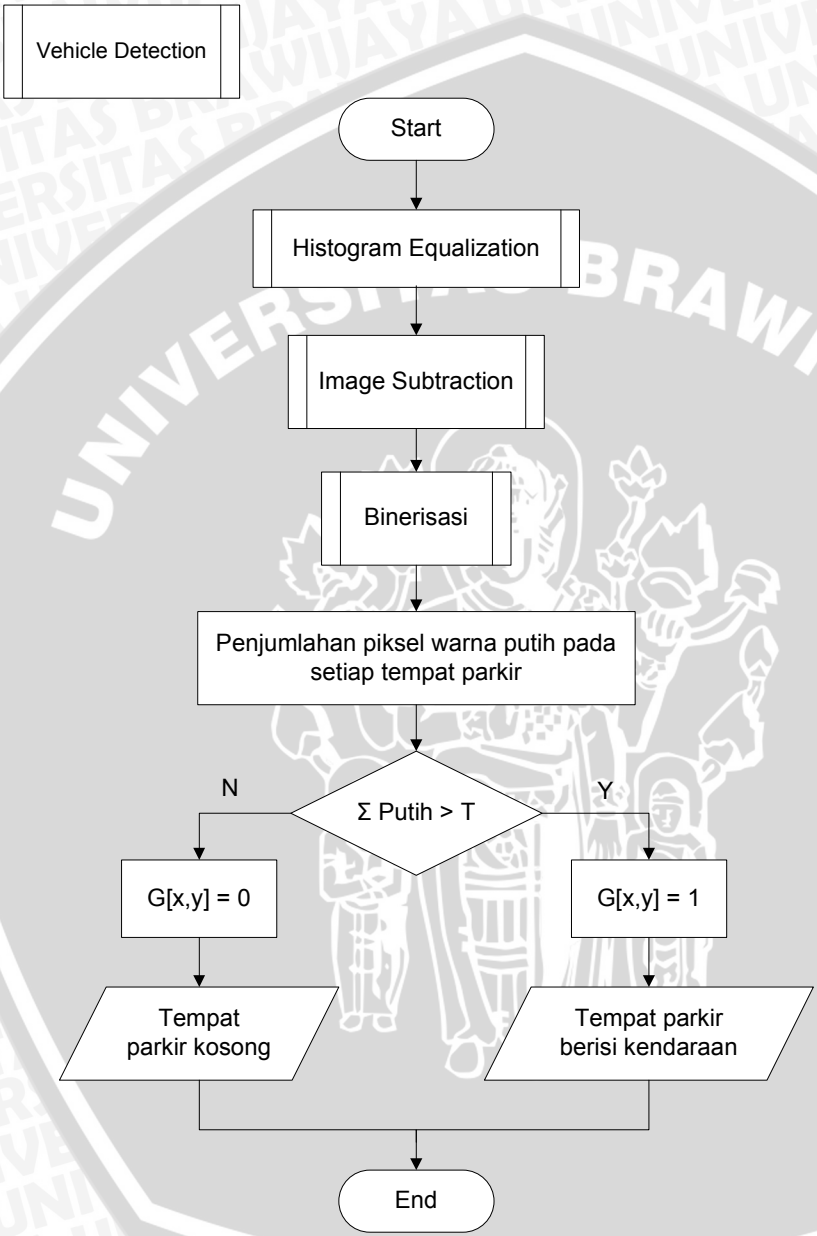
ditentukan, maka nilai dari piksel tersebut menjadi 1, dan apabila nilainya kurang dari sama dengan *threshold* maka nilainya diganti menjadi 0 (nol).

6. Dari proses seleksi tersebut dihasilkan dua nilai, yaitu nilai 1 yang dapat diartikan bahwa tempat parkir telah terisi kendaraan, dan nilai 0 yang dapat diartikan bahwa tempat parkir masih kosong.

Diagram alir proses *vehicle detection* secara umum dapat dilihat pada Gambar 3.4.



Gambar 3.3. Diagram Alir Proses Inisialisasi Sistem


Vehicle Detection
Start
Histogram Equalization
Image Subtraction
Binerisasi
Penjumlahan piksel warna putih pada setiap tempat parkir
Σ Putih > T
N
Y
G[x,y] = 0
G[x,y] = 1
Tempat parkir kosong
Tempat parkir berisi kendaraan
End

Gambar 3.4. Diagram Alir Proses *Vehicle Detection*

3.2.2.1. *Histogram Equalization*

Pada proses *histogram equalization*, nilai distribusi warna pada citra masukan dibuat rata. Citra yang terekualisasi selanjutnya digunakan dalam proses *image subtraction*. Tahap-tahap yang dilakukan pada proses ini adalah sebagai berikut.

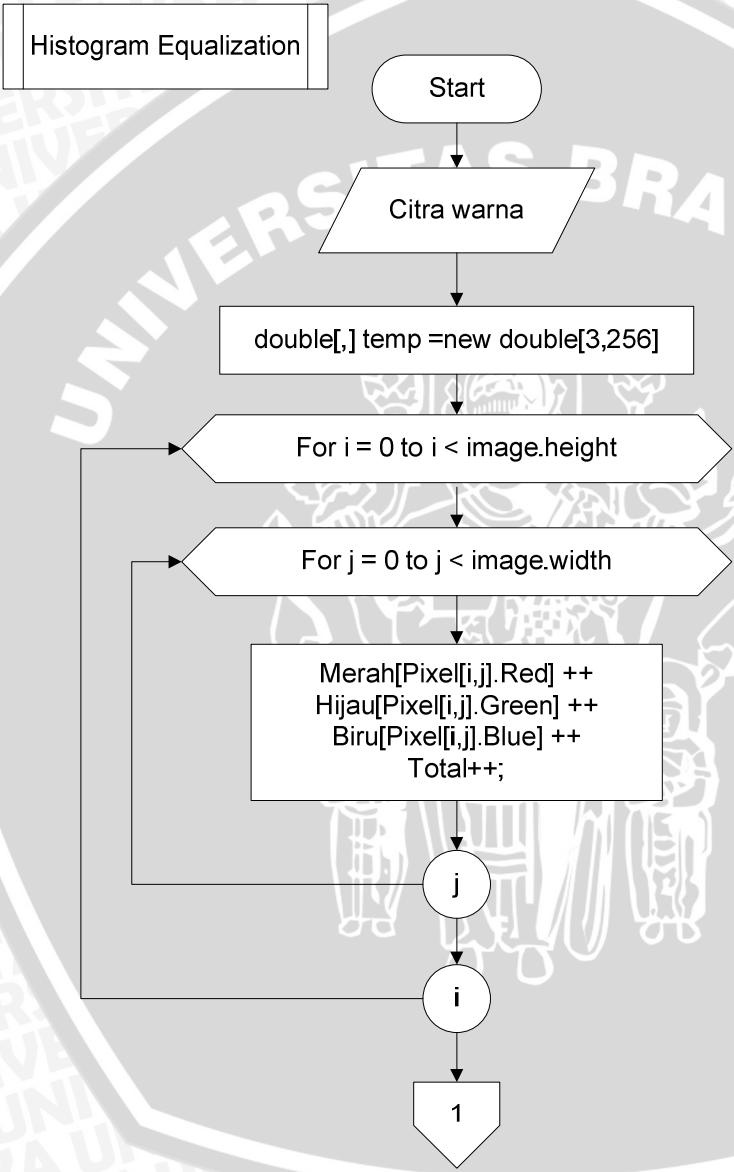
1. Masukan untuk proses *histogram equalization* adalah citra warna.
2. Dilakukan pembacaan untuk setiap nilai piksel yang terdapat pada citra, dengan perulangan berdasarkan tinggi dan lebarnya.
3. Menghitung frekuensi setiap nilai piksel untuk warna merah, hijau, dan biru.
4. Menghitung nilai keabuan setelah proses *histogram equalization* dengan perumusan 2.18.
5. Mengganti nilai piksel dengan nilai baru hasil dari proses *histogram equalization*. Diagram alir proses *histogram equalization* dapat dilihat pada gambar 3.5.

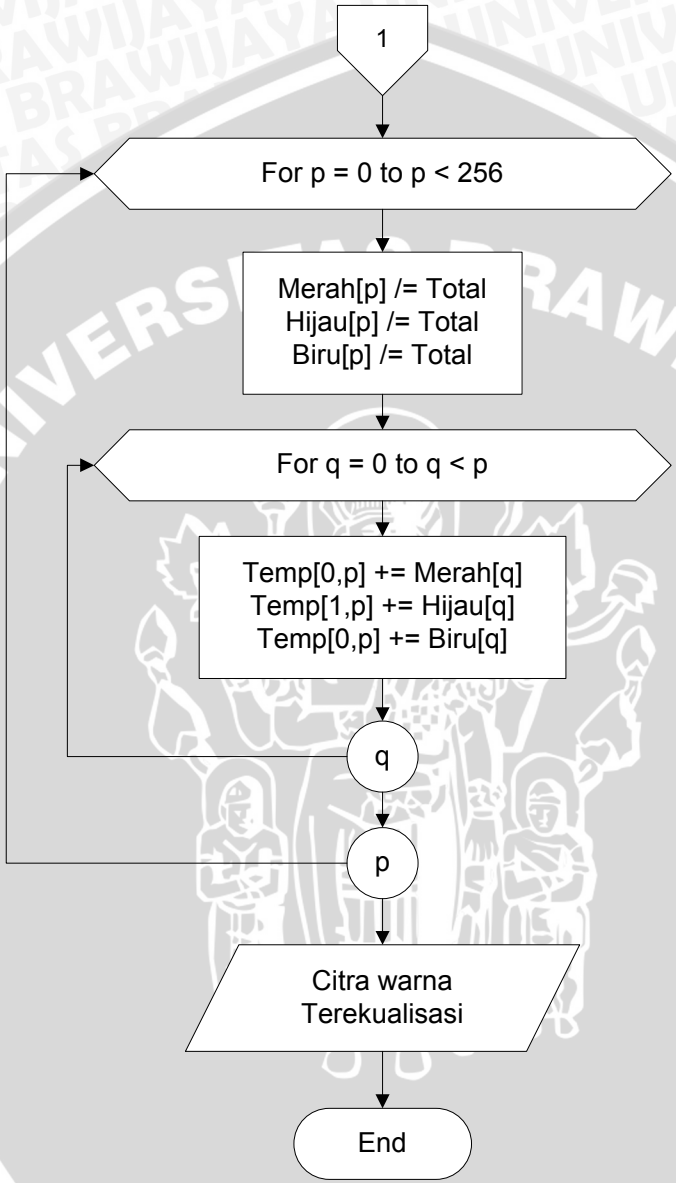
3.2.2.2. *Image Subtraction*

Proses *image subtraction* bertujuan untuk mendapatkan obyek pada citra masukan lahan parkir mobil dengan cara mengurangkan citra masukan dengan citra referensi, yaitu citra lahan parkir mobil yang kosong. Proses yang dilakukan pada tahap ini adalah sebagai berikut:

1. Citra masukan pada proses ini merupakan citra warna lahan parkir mobil yang terisi.
2. Dilakukan pembacaan untuk setiap nilai piksel yang terdapat pada citra, dengan perulangan berdasarkan tinggi dan lebarnya.
3. Piksel-piksel pada citra masukan (*data1*) dikurangi dengan piksel-piksel pada citra referensi yang telah ditentukan (*data2*), yaitu citra lahan parkir mobil yang kosong (*data3 = data1 – data2*).

Diagram alir untuk proses *image subtraction* dapat dilihat pada gambar 3.6.





Gambar 3.5. Diagram Alir Proses *Histogram Equalization*

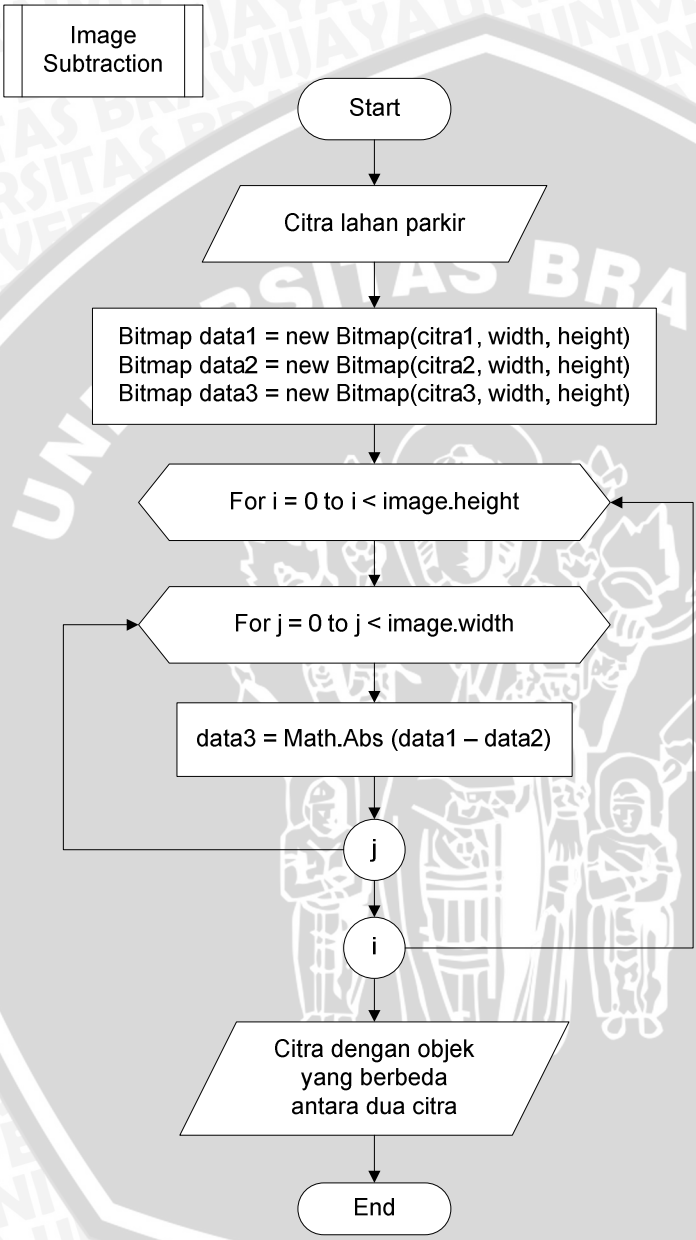


Image Subtraction
Start
Citra lahan parkir
Bitmap data1 = new Bitmap(citra1, width, height)
Bitmap data2 = new Bitmap(citra2, width, height)
Bitmap data3 = new Bitmap(citra3, width, height)
For i = 0 to i < image.height
For j = 0 to j < image.width
data3 = Math.Abs (data1 – data2)
j
i
Citra dengan objek yang berbeda antara dua citra
End

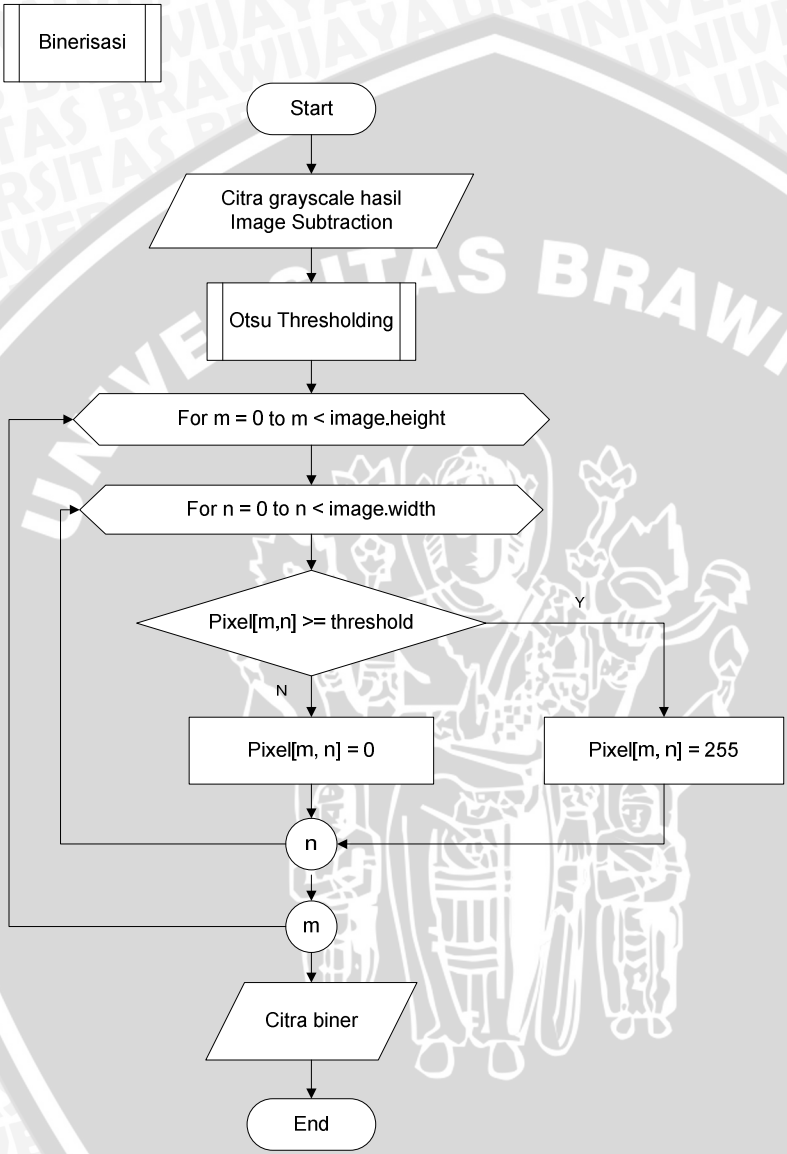
Gambar 3.6. Diagram Alir Proses *Image Subtraction*

3.2.2.3. Binerisasi

Pada proses binerisasi citra, hasil dari proses *image subtraction* dikonversi menjadi citra biner yang hanya memiliki dua kombinasi warna, yaitu warna hitam yang ditandai dengan nilai 0 dan warna putih yang ditandai dengan nilai 0. Pada proses binerisasi digunakan *threshold* yang telah ditentukan untuk membedakan antara obyek dan *background*. Penentuan nilai *threshold* pada proses binerisasi ini menggunakan metode *Otsu Thresholding*. Warna hitam digunakan untuk merepresentasikan *background* citra, sedangkan warna putih digunakan untuk merepresentasikan obyek. Tahap-tahap pada proses binerisasi adalah sebagai berikut.

1. Masukan citra tempat parkir hasil dari proses *image subtraction*.
2. Menentukan nilai *threshold* T dengan metode *Otsu Thresholding*.
3. Dilakukan pembacaan untuk setiap nilai piksel yang terdapat pada citra, dengan perulangan berdasarkan tinggi dan lebarnya.
4. Membandingkan nilai keabuan (*grayscale*) setiap piksel pada citra dengan nilai *threshold* yang telah ditentukan. Jika nilai keabuan lebih besar dari nilai *threshold*, maka nilai piksel diganti dengan 255. Sebaliknya, jika nilai keabuan kurang dari nilai *threshold* maka nilai piksel diganti 0.

Diagram alir proses binerisasi dapat dilihat pada Gambar 3.7.



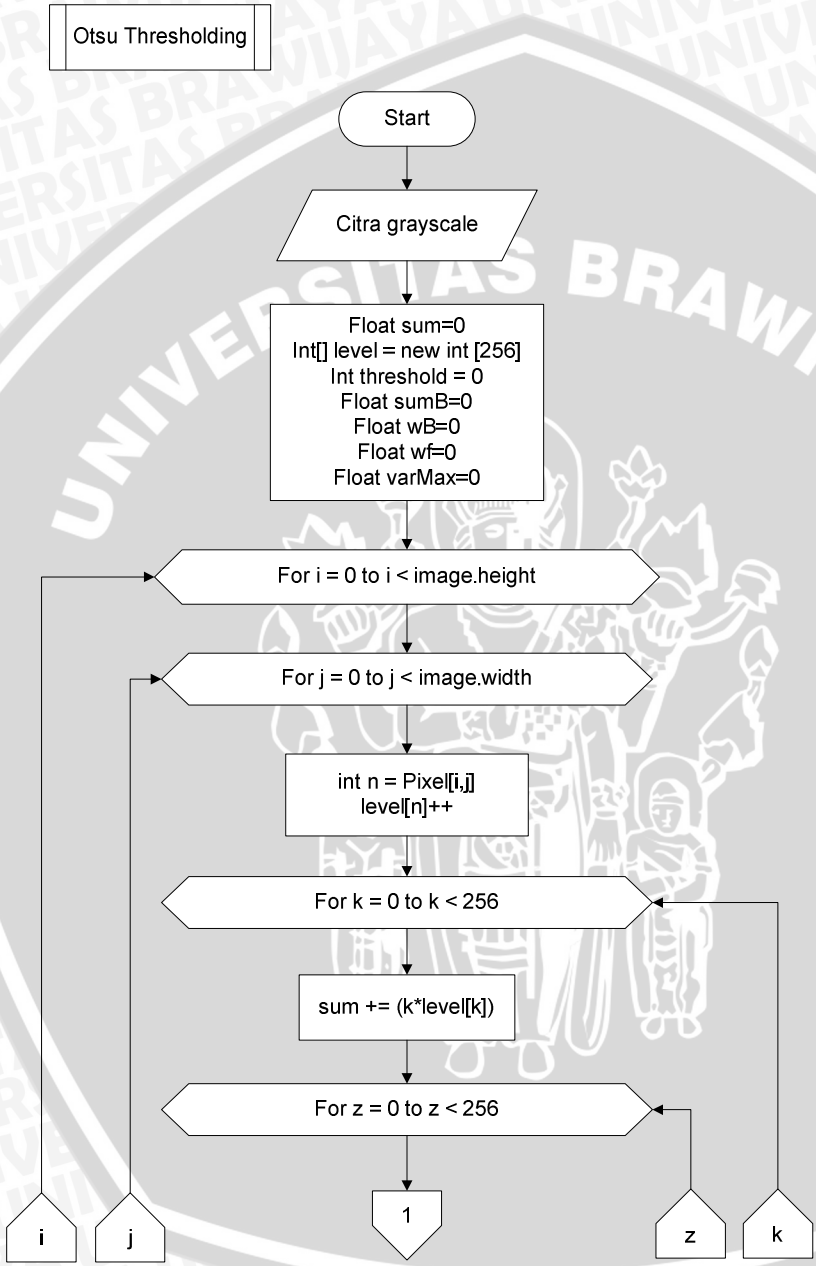
Gambar 3.7. Diagram Alir Proses Binerisasi

3.2.2.3.1. *Otsu Thresholding*

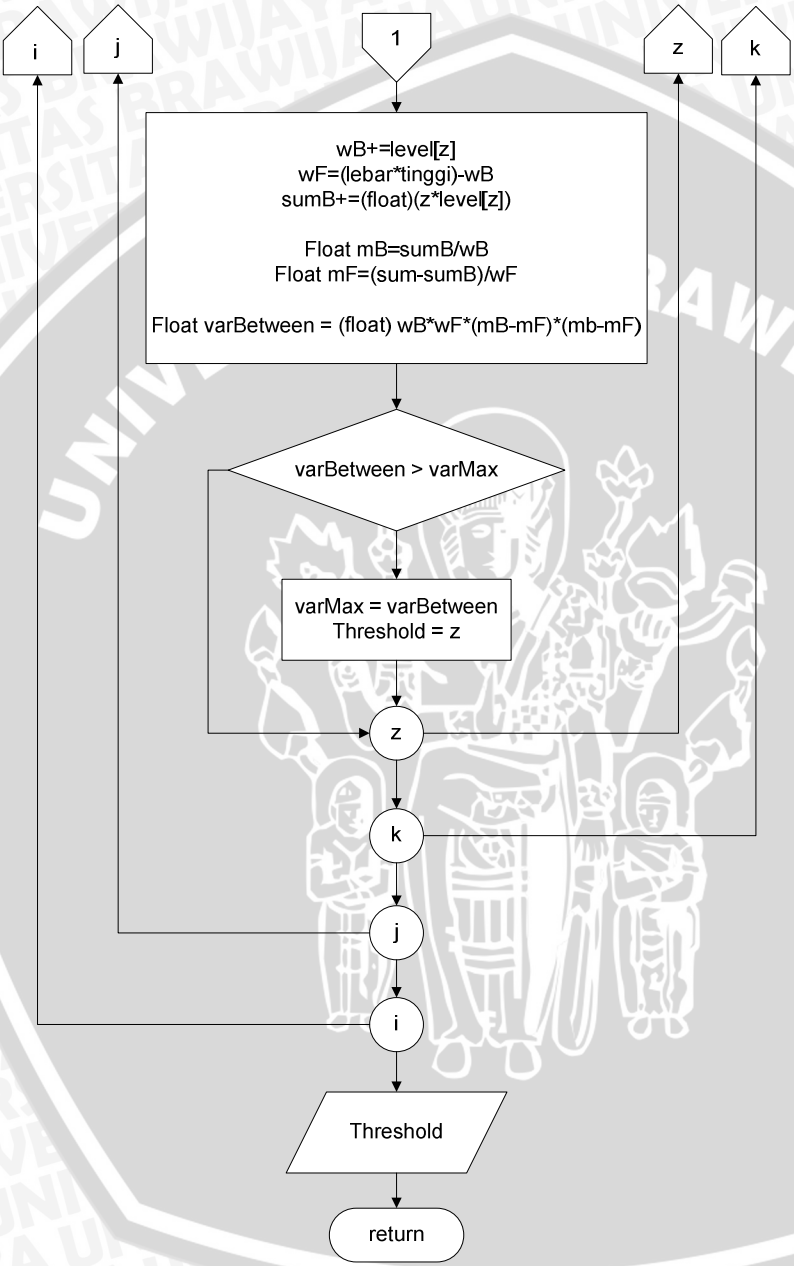
Pada proses ini akan ditentukan nilai thresholding yang digunakan untuk mengkonversi citra keabuan menjadi citra biner. Tahap-tahap pada proses *Otsu thresholding* adalah sebagai berikut.

1. Masukan pada proses ini adalah citra *grayscale* hasil dari proses *image subtraction*.
2. Dilakukan pembacaan untuk setiap nilai piksel yang terdapat pada citra, dengan perulangan berdasarkan tinggi dan lebarnya.
3. Dilakukan perhitungan frekuensi setiap nilai piksel keabuan.
4. Menghitung bobot (*weight*) untuk *background* dan *foreground* citra.
5. Menghitung nilai rata-rata *gray level* dari piksel *background* dan *foreground*.
6. Menghitung jarak varian antar dua kelompok (*background* dan *foreground*).
7. Dilakukan pengecekan apakah terdapat nilai maksimum baru yang dapat digunakan sebagai *threshold*. Hasil dari pengecekan tersebut adalah nilai *threshold* yang digunakan sebagai batas untuk mengkonversi citra keabuan menjadi citra biner.

Diagram alir proses *Otsu thresholding* dapat dilihat pada Gambar 3.8.


Otsu Thresholding
Start
Citra grayscale
Float sum=0
Int[] level = new int [256]
Int threshold = 0
Float sumB=0
Float wB=0
Float wf=0
Float varMax=0
For i = 0 to i < image.height
For j = 0 to j < image.width
int n = Pixel[i,j]
level[n]++
For k = 0 to k < 256
sum += (k*level[k])
For z = 0 to z < 256
1
i
j
z
k


UNIVERSITAS BRAWIJAYA


i
j
1
z
k
wB+=level[z]
wF=(lebar*tinggi)-wB
sumB+=(float)(z*level[z])
Float mB=sumB/wB
Float mF=(sum-sumB)/wF
Float varBetween = (float) wB*wF*(mB-mF)*(mb-mF)
varBetween > varMax
varMax = varBetween
Threshold = z
z
k
j
i
Threshold
return

Gambar 3.8. Diagram Alir Proses *Otsu Thresholding*

3.2.3. *Edge Detection*

Edge Detection merupakan metode kedua untuk mendeteksi keberadaan obyek pada tempat parkir. Secara umum, tahap-tahap yang terjadi pada *edge detection* adalah sebagai berikut.

1. Citra masukan pada proses *edge detection* merupakan citra warna lahan parkir mobil.
2. Dilakukan proses *grayscale* untuk mengubah citra warna menjadi citra keabuan.
3. Citra keabuan ditingkatkan kualitasnya menggunakan *histogram equalization* agar nilai piksel pada citra lebih merata.
4. Kemudian dilakukan reduksi *noise* pada citra masukan menggunakan metode *median filtering*.
5. Proses selanjutnya adalah pendeteksian tepi obyek menggunakan *sobel edge detection* sehingga dapat ditentukan obyek pada citra.
6. Citra hasil *sobel edge detection* diubah menjadi citra biner melalui proses binerisasi sehingga citra hanya terdiri dari piksel bernilai 1 dan 0.

Diagram alir proses *edge detection* dapat dilihat pada Gambar 3.9.

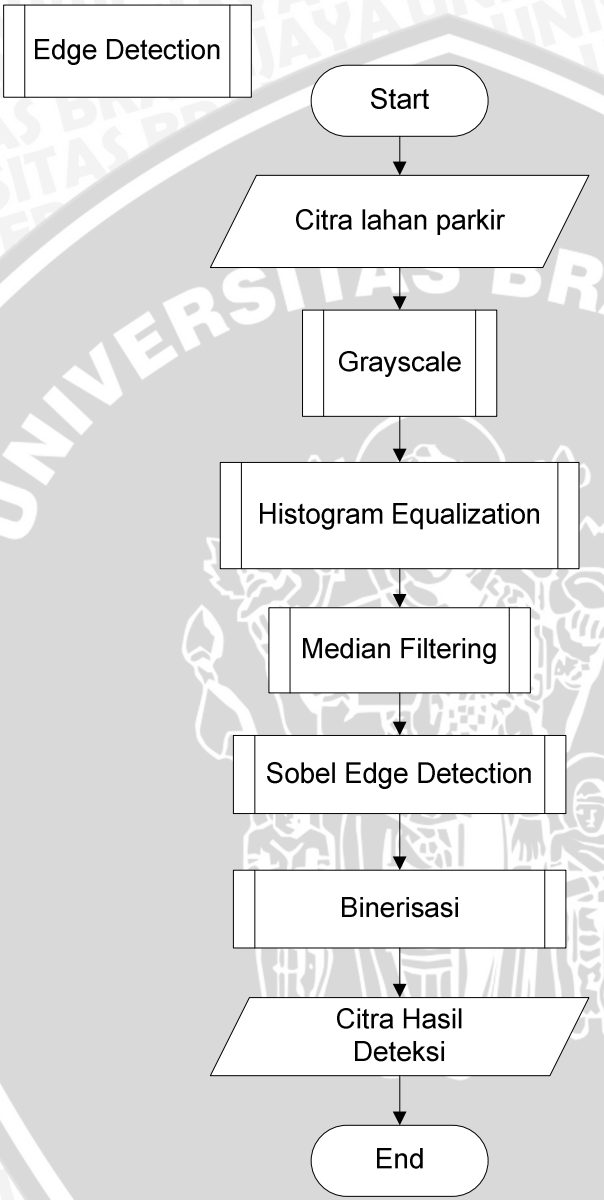
3.2.3.1. Proses *Grayscale*

Pada proses *grayscale*, citra masukan diubah menjadi citra keabuan. Proses yang dilakukan pada tahap ini adalah sebagai berikut:

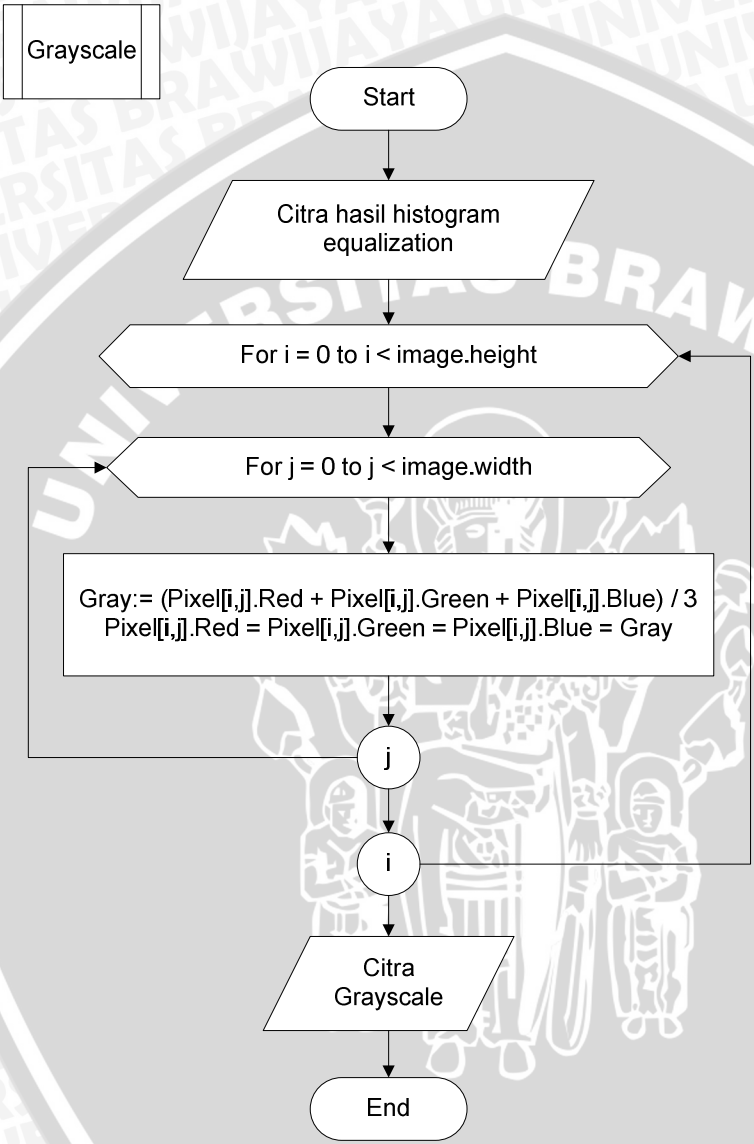
1. Dilakukan pembacaan untuk setiap nilai piksel yang terdapat pada citra, dengan perulangan berdasarkan tinggi dan lebarnya.
2. Masing-masing nilai warna merah, hijau, dan biru pada setiap piksel yang diwakili oleh *Pixel[i,j].Red*, *Pixel[i,j].Green*, dan *Pixel[i,j].Blue* diambil untuk kemudian dihitung menggunakan perumusan 2.13,

sehingga didapatkan hasil akhir untuk proses ini berupa citra keabuan.

Diagram alir proses *grayscale* dapat dilihat pada Gambar 3.10.



Gambar 3.9. Diagram Alir Proses *Edge Detection*



Gambar 3.10. Diagram Alir Proses *Grayscale*

3.2.3.2. *Histogram Equalization*

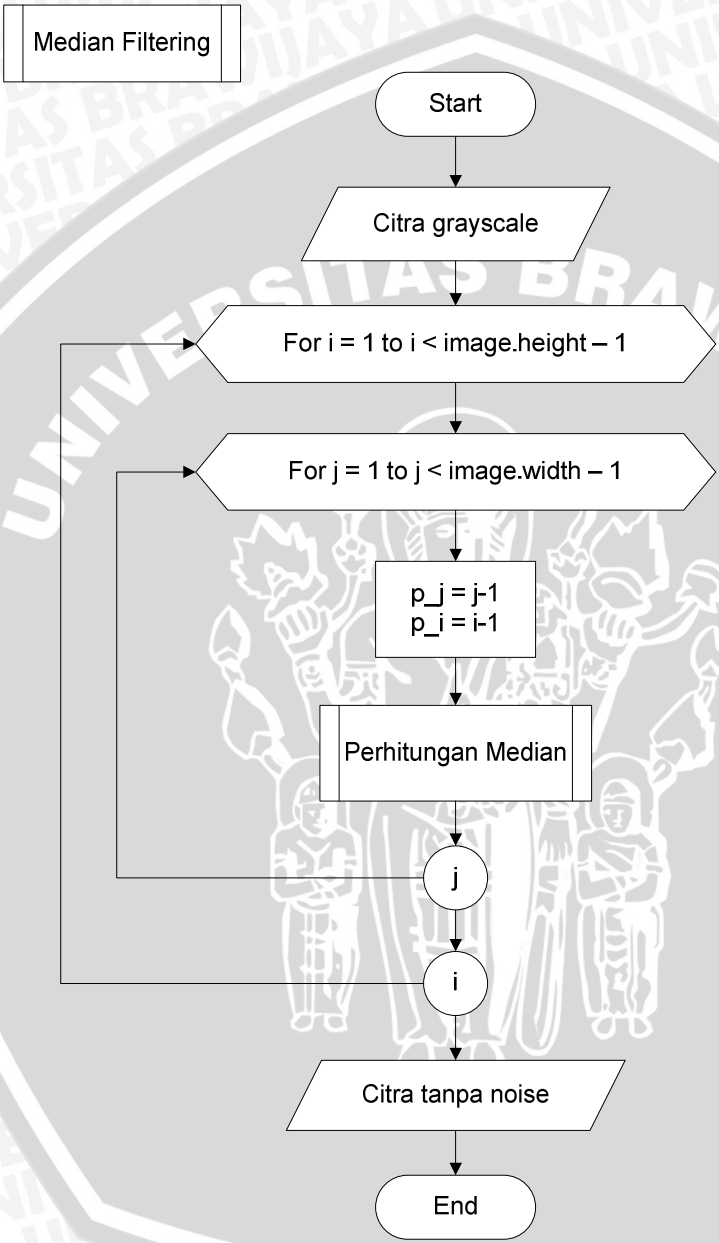
Proses *histogram equalization* pada edge detection ini hampir sama dengan proses *histogram equalization* pada *vehicle detection* (subbab 3.2.2.1.), yang membedakan hanyalah citra masukan pada proses ini adalah citra keabuan (*grayscale*), sehingga piksel yang nantinya diproses adalah piksel keabuan.

3.2.3.3. *Median Filtering*

Pada proses ini sistem akan mereduksi *noise* yang terdapat pada citra keabuan. *Noise* pada citra masukan lahan parkir dapat berupa bayangan mobil atau manusia. Tahap-tahap yang terdapat pada proses *median filtering* adalah sebagai berikut.

1. Citra masukan berupa citra keabuan lahan parkir hasil dari proses *grayscale* dan *histogram equalization.*
2. Dilakukan pembacaan untuk setiap nilai piksel yang terdapat pada citra, dengan perulangan berdasarkan tinggi dan lebarnya.
3. Inisialisasi terhadap variabel *p_j* yang bernilai *j-1* dan variabel *p_i* yang bernilai *i-1*. Hal ini dilakukan agar pembacaan piksel ketika proses *median filtering* dengan mask 3x3 dimulai dari koordinat (0, 0).
4. Proses perhitungan *median filtering* yang mengambil setiap kernel 3x3 pada matriks citra.
5. Hasil dari proses *median filtering* merupakan citra dengan *noise* yang sudah direduksi.

Diagram alir proses median filtering dapat dilihat pada gambar 3.11.



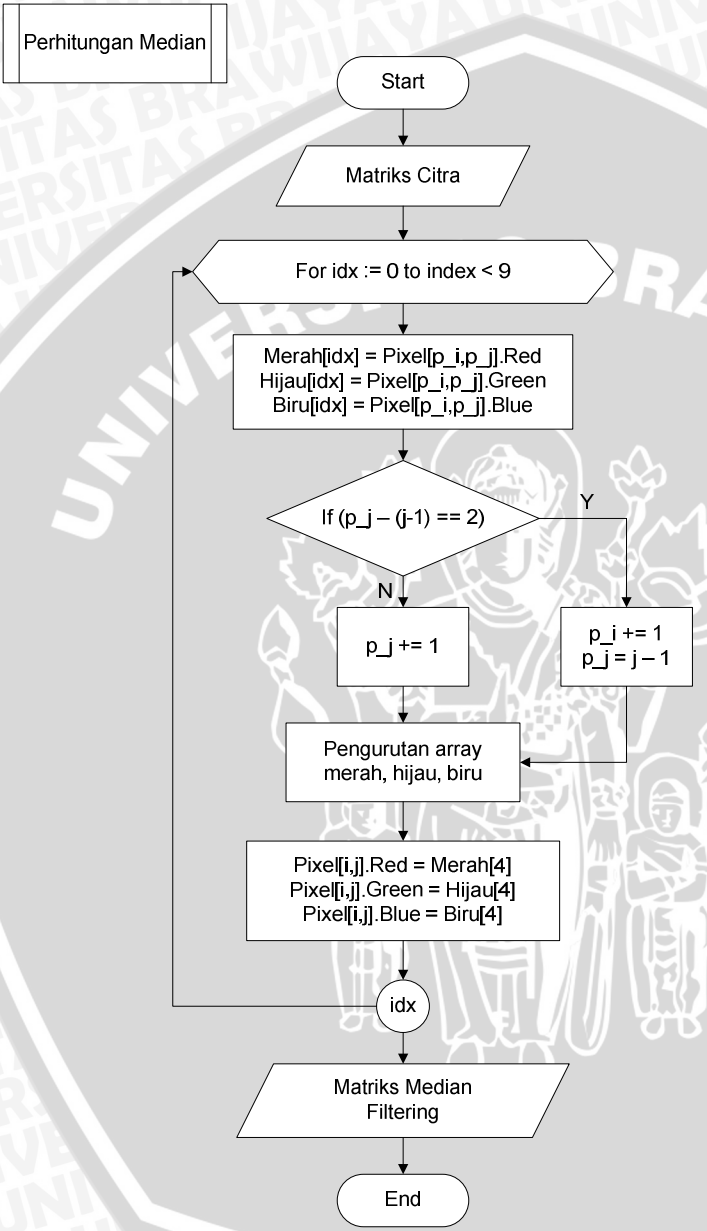
Gambar 3.11. Diagram Alir Proses *Median Filtering*

3.2.3.3.1. Perhitungan *Median Filtering*

Pada proses ini sistem akan mengambil setiap kernel 3x3 pada matriks citra dimulai dari koordinat (0,0). Tahap-tahap yang terdapat pada proses perhitungan median adalah sebagai berikut.

1. Masukan pada proses ini adalah matriks citra yang telah dilakukan pembacaan setiap pikselnya pada proses *median filtering*.
2. Dibuat tiga buah *array*, yaitu *merah[idx]*, *hijau[idx]*, dan *biru[idx]* untuk menyimpan nilai-nilai piksel merah, hijau, dan biru.
3. Selanjutnya dilakukan seleksi dengan syarat *(p_j-(j-1)==2)*. Jika menemui kondisi yang memenuhi syarat tersebut, maka pembacaan dengan mask 3x3 bergeser ke baris selanjutnya (*p_i += 1*) dan kolom pertama (*p_j = j-1*). Namun jika tidak memenuhi syarat tersebut, maka pembacaan bergeser pada kolom selanjutnya dan masih pada baris yang sama (*p_j += 1*).
4. Apabila semua piksel telah dibaca, dilakukan pengurutan terhadap *array* merah, hijau, dan biru. Kemudian diambil nilai tengah dari nilai *array* pada matriks yang telah diurutkan.
5. Nilai tengah yang sudah didapatkan dimasukkan pada matriks citra, yaitu pada indeks ke empat.
6. Hasil dari proses perhitungan median adalah matriks yang setiap nilai tengah pada kernel 3x3 nya sudah diganti.

Diagram alir proses perhitungan *median* dapat dilihat pada Gambar 3.12.


Perhitungan Median
Start
Matriks Citra
For idx := 0 to index < 9
Merah[idx] = Pixel[p_i,p_j].Red
Hijau[idx] = Pixel[p_i,p_j].Green
Biru[idx] = Pixel[p_i,p_j].Blue
If (p_j – (j-1) == 2)
Y
N
p_j += 1
p_i += 1
p_j = j – 1
Pengurutan array
merah, hijau, biru
Pixel[i,j].Red = Merah[4]
Pixel[i,j].Green = Hijau[4]
Pixel[i,j].Blue = Biru[4]
idx
Matriks Median
Filtering
End

Gambar 3.12. Diagram Alir Proses Perhitungan *Median*

3.2.3.4. *Sobel Edge Detection*

Setelah citra keabuan difilter dengan *median filtering*, proses selanjutnya adalah mendeteksi tepian citra menggunakan operator sobel seperti pada rumusan 2.22 dan 2.23. Proses deteksi tepi dengan operator sobel ini bertujuan untuk memisahkan obyek dengan bayangan yang terbentuk oleh obyek. Proses-proses yang terdapat pada tahap ini adalah sebagai berikut:

1. Citra masukan berupa citra keabuan.
2. Inisialisasi operator sobel yang terdiri dari arah vertikal dan horisontal (Gx dan Gy).
3. Dilakukan pembacaan untuk setiap nilai piksel yang terdapat pada citra, dengan perulangan berdasarkan tinggi dan lebarnya.
4. Proses konvolusi matriks
5. Hasil dari proses ini merupakan citra biner.

Diagram alir proses sobel edge detection dapat dilihat pada Gambar 3.13.

3.2.3.4.1. Korelasi Matriks

Proses korelasi matriks merupakan proses pengalian kernel matriks 3x3 pada citra uji dengan operator sobel. Tahap-tahap yang terdapat pada proses konvolusi matriks adalah sebagai berikut.

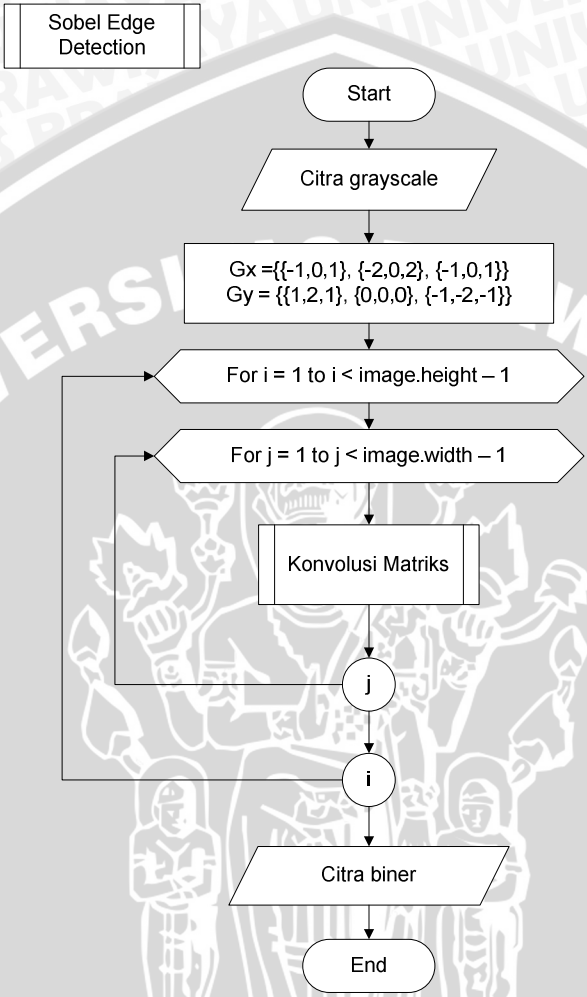
1. Inisialisasi variabel x dan y yang digunakan untuk menampung nilai hasil perhitungan *sobel edge detection*.
2. Perulangan untuk proses konvolusi matriks.
3. Perkalian antara operator sobel dengan kernel pada citra keabuan.
4. Dilakukan seleksi dari nilai x dan y, dengan syarat jika hasil dari (*sqrt(x*x + y*y)*) lebih besar dari *threshold* yang telah ditentukan, maka nilai piksel diubah menjadi 255, dan apabila kurang dari *threshold* maka

nilai piksel diubah menjadi 0. Sehingga didapatkan citra hasil sobel *edge detection* dalam bentuk biner.

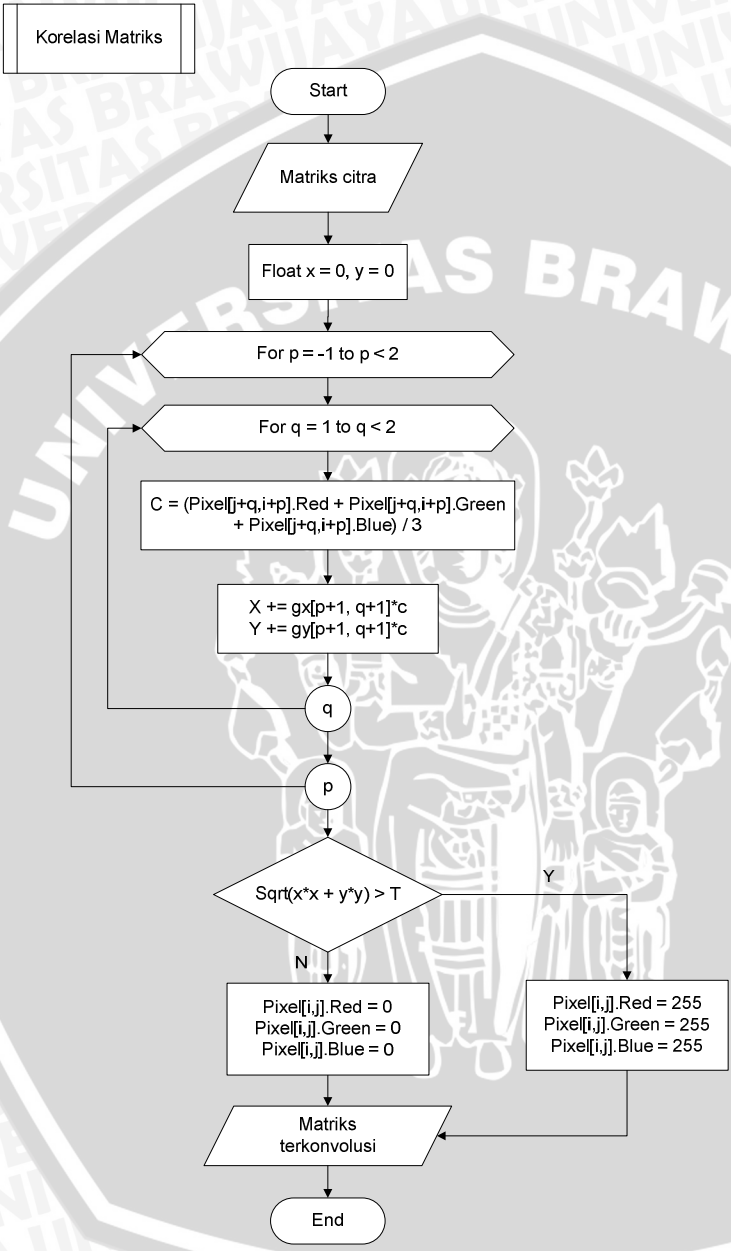
Diagram alir proses konvolusi matriks dapat dilihat pada Gambar 3.14.

3.2.3.5. Binerisasi

Proses binerisasi pada *edge detection* ini sama dengan proses binerisasi pada *vehicle detection* (subbab 3.2.2.4.).



Gambar 3.13. Diagram Alir Proses *Sobel Edge Detection*



Gambar 3.14. Diagram Alir Proses Korelasi Matriks

3.3. Contoh Perhitungan

Subbab ini memaparkan contoh perhitungan sederhana secara manual agar dapat lebih mudah memahami cara kerja sistem perangkat lunak yang dibangun untuk mendeteksi tempat kosong pada lahan parkir mobil.

Tahapan perhitungan untuk proses *vehicle detection* adalah sebagai berikut:

1. Matriks citra warna berukuran 8 x 8 dapat dilihat pada tabel 3.1.

Tabel 3.1. Matriks Citra Warna 8 x 8

(i,j)	0	1	2	3	4	5	6	7
0	31 33 22	42 44 33	46 48 37	41 43 32	114 116 103	119 121 108	84 86 73	111 113 100
1	42 44 33	44 46 35	64 66 55	82 84 73	92 94 83	76 78 65	6 8 0	61 63 50
2	188 189 181	186 187 179	193 194 186	157 158 150	177 178 170	156 158 147	191 193 182	231 233 222
3	137 138 132	134 135 127	34 35 29	177 178 170	156 157 149	173 174 166	152 153 145	150 152 141
4	255 255 251	255 255 250	73 74 69	178 179 173	97 98 92	211 212 206	186 187 181	159 160 152
5	141 142 137	134 135 130	125 126 121	153 154 149	137 138 133	139 140 134	152 153 148	150 151 145
6	116 116 114	118 118 116	151 151 149	120 120 118	132 132 130	137 138 133	145 145 143	118 119 114
7	123 123	129 129	139 139	135 135	134 134	120 120	140 140	129 129

	121	127	137	133	132	118	138	127

2. Pengurangan dengan matriks citra referensi berukuran 8 x 8. Matriks citra referensi dapat dilihat pada tabel 3.2.

Tabel 3.2. Matriks citra referensi

(i,j)	0	1	2	3	4	5	6	7
0	20 15 21	27 35 23	39 43 27	19 31 28	105 110 93	104 115 101	72 66 53	98 95 93
1	30 31 29	27 34 29	57 53 45	75 79 68	78 85 63	67 58 54	4 5 0	41 51 35
2	157 172 147	154 163 159	167 174 163	138 142 120	147 164 140	124 147 121	175 182 164	197 189 200
3	111 118 123	119 116 110	20 15 17	147 158 150	135 129 110	157 163 152	141 132 121	110 131 119
4	200 211 213	212 203 215	39 65 47	154 162 134	85 82 81	199 187 179	147 156 162	119 127 134
5	111 121 119	109 117 102	107 98 115	114 127 113	120 117 119	117 123 98	129 113 124	134 126 128
6	103 97 103	98 98 91	110 123 107	98 106 94	123 119 116	119 115 121	126 124 129	104 102 103
7	108 112 106	117 105 112	115 109 121	124 109 115	102 118 107	99 97 94	114 106 117	104 110 106

Apabila nilai hasil pengurangan antar piksel bernilai minus maka nilai warna pada citra hasil diubah menjadi 0. Contoh pengurangan nilai piksel pada koordinat (0,0) adalah sebagai berikut:

- Nilai *Red*: 31 – 20 = 11

- Nilai *Green*: 33 – 15 = 18
- Nilai *Blue*: 22 – 21 = 1

Pengurangan untuk nilai warna *Red*, *Green*, dan *Blue* tersebut dilakukan hingga koordinat (7, 7). Matriks citra hasil pengurangan dua buah matriks citra tersebut dapat dilihat pada tabel 3.3.

Tabel 3.3. Matriks citra hasil pengurangan

(i,j)	0	1	2	3	4	5	6	7
0	11 18 1	15 9 10	7 5 10	22 12 4	9 6 10	15 6 7	12 20 20	13 18 7
1	12 13 4	17 12 6	7 13 10	7 5 5	14 9 20	9 20 11	2 3 0	20 12 15
2	31 17 34	32 23 20	26 20 23	19 16 30	30 14 30	32 11 26	16 11 18	34 44 22
3	26 20 9	15 19 17	14 20 12	30 20 20	21 28 39	16 11 14	11 21 24	40 21 22
4	55 44 38	43 52 35	34 9 22	24 17 39	12 16 11	12 25 27	39 31 19	40 33 18
5	30 21 18	25 18 28	18 28 6	39 27 36	17 21 14	22 17 36	23 40 24	16 25 17
6	13 19 11	20 20 25	41 28 42	22 14 24	9 13 14	18 23 12	19 21 14	14 17 11
7	15 11 15	12 24 15	24 30 16	11 26 18	32 16 25	21 23 24	26 34 21	25 19 21

3. Citra hasil pengurangan diubah menjadi citra keabuan dengan perumusan 2.14. Misal perhitungan untuk piksel pada koordinat (1,6) adalah sebagai berikut:

$$G(1,6) = \frac{2+3+0}{3} = 1,67$$

Matriks citra keabuan dapat dilihat pada tabel 3.4.

Tabel 3.4. Matriks citra keabuan

(i,j)	0	1	2	3	4	5	6	7
0	10	11,33	7,33	12,67	8,33	9,33	17,33	12,67
1	9,67	11,67	10	5,67	14,3	13,3	1,67	15,67
2	27,3	25	23	21,67	24,67	23	15	33,33
3	18,33	17	15,33	23,33	29,33	13,67	18,67	27,67
4	45,67	43,33	21,67	26,67	13	21,33	29,67	30,33
5	23	23,67	17,33	34	17,33	25	29	19,33
6	14,33	21,67	37	20	12	17,67	18	14
7	13,67	17	23,33	18,33	24,33	22,67	27	21,67

4. Mengubah citra keabuan menjadi citra biner, dengan *threshold* sebagai berikut.

$$T = \frac{1,67 + 45,67}{2} = 23,67$$

Dengan *threshold* yang telah ditentukan, dilakukan proses seleksi untuk setiap nilai piksel. Matriks citra 8x8 hasil operasi *thresholding* dapat dilihat pada tabel 3.5.

Tabel 3.5. Matriks citra biner

(i,j)	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	1	1	0	0	1	0	0	1
3	0	0	0	0	1	0	0	1
4	1	1	0	1	0	0	1	1
5	0	1	0	1	0	1	1	0
6	0	0	1	0	0	0	0	0
7	0	0	0	0	1	0	1	0

Dari matriks citra biner tersebut dapat dihitung jumlah piksel berwarna putih yang direpresentasikan degan nilai 1, yaitu sebanyak 3. Maka dapat dihitung prosentase jumlah nilai piksel putih pada citra biner yaitu:

$$Jumlah\ piksel\ putih = \frac{18}{64}\ x\ 100\% = 28{,}1\%$$

5. Dilakukan proses seleksi dengan membandingkan presentase jumlah piksel putih dengan nilai *threshold*, dan didapatkan bahwa presentase jumlah piksel putih lebih besar dari nilai *threshold* (28,1% > 4,5%). Sehingga dapat disimpulkan bahwa terdapat obyek berupa mobil pada citra masukan lahan parkir mobil.

Tahapan perhitungan untuk *edge detection* adalah sebagai berikut:

1. Matriks citra warna yang berukuran 8 x 8 diubah menjadi citra keabuan dengan perhitungan sebagai berikut:

Piksel di koordinat (0,0): $f(0,0) = \frac{31 + 33 + 22}{3} = 29$

Perhitungan tersebut dilakukan hingga piksel yang berada di koordinat (8,8). Hasil konversi matriks citra warna menjadi citra keabuan dapat dilihat pada tabel 3.6.

Tabel 3.6. Matriks Citra Keabuan *Edge Detection*

(i,j)	0	1	2	3	4	5	6	7
0	29	40	44	39	111	116	81	108
1	40	42	62	80	90	73	5	58
2	186	184	191	155	175	154	189	229
3	136	132	33	175	154	171	150	148
4	254	253	72	177	96	210	185	157
5	140	133	124	152	136	138	151	149
6	115	117	150	119	131	136	144	117
7	122	128	138	143	133	119	139	128

2. *Noise* yang terdapat pada citra direduksi menggunakan *median filter*. Misal untuk mask 3x3 yang pertama, yaitu matriks yang meliputi koordinat (0,0); (0,1); (0,2); (1,0); (1,1); (1,2); (2,0); (2,1); (2,2), yang nilai-nilainya adalah:

29	40	44
40	42	62
186	184	191

Nilai-nilai piksel tersebut diurutkan dari nilai yang terkecil ke nilai yang terbesar sebagai berikut:

29 40 40 42 44 62 184 186 191

Dari deret nilai yang telah terurut tersebut kemudian ditentukan nilai tengahnya, yaitu pada indeks ke-4 (indeks dimulai dari 0) yang bernilai 44. Nilai tengah ini selanjutnya diletakkan pada koordinat piksel (1,1). Hal ini dilakukan hingga penggantian nilai piksel di koordinat (6,6). Matriks citra hasil dari proses *median filtering* dapat dilihat pada tabel 3.7.

Tabel 3.7. Matriks Citra Hasil *Median Filtering*

(i,j)	0	1	2	3	4	5	6	7
0	29	40	44	39	111	116	81	108
1	40	44	62	90	111	111	108	58
2	186	132	132	154	154	154	150	229
3	136	184	175	155	171	171	171	148
4	254	133	133	136	154	151	151	157
5	140	133	133	131	136	138	149	149
6	115	128	133	136	136	136	138	117
7	122	128	138	143	133	119	139	128

Warna hijau pada matriks menunjukkan nilai piksel yang diganti setelah proses *median filtering*, sedangkan warna kuning menunjukkan nilai piksel yang tidak mengalami perubahan. Nilai piksel yang mengalami perubahan adalah nilai tengah matriks kernel berukuran 3x3.

3. Peningkatan kualitas citra menggunakan *histogram equalization*. Tahap-tahap pada proses *histogram equalization* adalah sebagai berikut:
    - Menghitung frekuensi dan distribusi kumulatif dari nilai skala keabuan pada matriks citra yang *noise*nya telah direduksi menggunakan *median filtering*. Daftar frekuensi dan pehitungan distribusi kumulatif dapat dilihat pada tabel 3.8.

Tabel 3.8. Frekuensi dan Distribusi Kumulatif Skala Keabuan

Skala Keabuan	Frek	Distribusi Kumulatif	Skala Keabuan	Frek	Distribusi Kumulatif
29	1	1	133	6	26 + 6 = 32
39	1	1 + 1 = 2	136	6	32 + 6 = 38
40	2	2 + 2 = 4	138	3	38 + 3 = 41
44	2	4 + 2 = 6	139	1	41 + 1 = 42
58	1	6 + 1 = 7	140	1	42 + 1 = 43
62	1	7 + 1 = 8	143	1	43 + 1 = 44
81	1	8 + 1 = 9	148	1	44 + 1 = 45
90	1	9 + 1 = 10	149	2	45 + 2 = 47
108	2	10 + 2 = 12	150	1	47 + 1 = 48
111	3	12 + 3 = 15	151	2	48 + 2 = 50
115	1	15 + 1 = 16	154	4	50 + 4 = 54
116	1	16 + 1 = 17	155	1	54 + 1 = 55
117	1	17 + 1 = 18	157	1	55 + 1 = 56
119	1	18 + 1 = 19	171	3	56 + 3 = 59
122	1	19 + 1 = 20	175	1	59 + 1 = 60
128	3	20 + 3 = 23	184	1	60 + 1 = 61
131	1	23 + 1 = 24	186	1	61 + 1 = 62
132	2	24 + 2 = 26	229	1	62 + 1 = 63
			254	1	63 + 1 = 64

    - Menghitung nilai keabuan hasil *histogram equalization* menggunakan rumus 2.5 subbab 2.4.1 Berikut ini merupakan contoh perhitungan untuk skala keabuan 2 dan 30:

- Skala keabuan 2

$$K_o = round\left(\frac{1\,x\,(2^8-1)}{8\,x\,8}\right) = round\left(\frac{255}{64}\right) = 4$$

- Skala keabuan 30

$$K_o = round\left(\frac{2\,x\,(2^8-1)}{8\,x\,8}\right) = round\left(\frac{510}{64}\right) = 8$$

Hasil perhitungan untuk seluruh nilai skala keabuan dapat dlihat pada tabel 3.9.

Tabel 3.9. Nilai Skala Keabuan Awal dan Hasil Ekualisasi

Keabuan Awal	Frek	Keabuan Hasil	Keabuan Awal	Frek	Keabuan Hasil
29	1	4	133	6	127
39	1	8	136	6	151
40	2	16	138	3	163
44	2	24	139	1	167
58	1	28	140	1	171
62	1	32	143	1	175
81	1	36	148	1	179
90	1	40	149	2	187
108	2	48	150	1	191
111	3	60	151	2	199
115	1	64	154	4	215
116	1	68	155	1	219
117	1	72	157	1	223
119	1	76	171	3	235
122	1	80	175	1	239
128	3	92	184	1	243
131	1	96	186	1	247
132	2	104	229	1	251
			254	1	255

Matriks citra setelah proses *histogram equalization* dapat dilihat pada tabel 3.10.

Tabel 3.10. Matriks Citra Hasil Histogram Ekualisasi

(i,j)	0	1	2	3	4	5	6	7
0	4	16	24	8	60	68	36	48
1	16	24	32	40	60	60	48	28
2	247	104	104	215	215	215	191	251
3	151	243	239	219	235	235	235	179
4	255	127	127	151	215	199	199	223
5	171	127	127	96	151	163	187	187
6	64	92	127	151	151	151	163	72
7	80	92	163	175	127	76	167	92

4. Deteksi tepi obyek pada citra menggunakan operator *Sobel*. Misal untuk deteksi kernel 3x3 yang pertama, meliputi koordinat (0,0);(0,1);(0,2); (1,0);(1,1);(1,2); (2,0);(2,1);(2,2), yang nilainya dapat dilihat pada tabel 3.11.

Tabel 3.11. Kernel Pertama pada Citra

(a_1) 4	(a_2) 16	(a_3) 24
(a_4) 16	(a_5) 24	(a_6) 32
(a_7) 247	(a_8) 104	(a_9) 104

Nilai-nilai piksel tersebut kemudian dimasukkan pada perhitungan dengan perumusan matematis pada 2.19 dan

2.20. Sehingga nilai Gx dan Gy untuk kernel 3x3 yang pertama adalah:

G_x = (247 + 2*104 + 104) – (4 + 2*16 + 24) = 499
G_y = (24 + 2*32 + 104) – (4 + 2*16 + 247) = -91

Nilai Gx dan Gy digunakan untuk mencari nilai tengah kernel yang pertama dengan perhitungan sebagai berikut:

$$M = \sqrt{{G_x}^2 + {G_y}^2} = \sqrt{(499)^2 + (-91)^2} = 507$$

Jika nilai M yang dihasilkan lebih dari 255, maka nilai piksel tersebut diganti menjadi 0. Perhitungan tersebut dilakukan hingga pada kernel 3x3 terakhir yang meliputi koordinat (5,5);(5,6);(5,7); (6,5);(6,6);(6,7); (7,5);(7,6);(7,7). Matriks citra setelah proses *sobel edge detection* dapat dilihat pada tabel 3.12.

Tabel 3.12. Matriks Citra Hasil *Sobel Edge Detection*

(i,j)	0	1	2	3	4	5	6	7
0	4	16	24	8	60	68	36	48
1	16	0	0	0	0	0	0	28
2	247	0	0	0	0	0	0	251
3	151	122	87	217	113	47	59	179
4	255	0	0	0	0	0	161	223
5	171	0	41	172	253	207	0	187
6	64	207	206	174	64	240	251	72
7	80	92	163	175	127	76	167	92

5. Proses binerisasi untuk mengubah nilai piksel sehingga hanya terdiri dari dua nilai, yaitu 1 dan 0 dengan *threshold* sebagai berikut.

$$T = \frac{0 + 255}{2} = 127.5$$

Matriks citra hasil konversi menjadi citra biner dapat dilihat pada tabel 3.13.

Tabel 3.13. Matriks Citra Biner

(i,j)	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	1
3	1	0	0	1	0	0	0	1
4	1	0	0	0	0	0	1	1
5	1	0	0	1	1	1	0	1
6	0	1	1	1	0	1	1	0
7	0	0	1	1	0	0	1	0

Dari matriks citra biner tersebut dapat dihitung jumlah piksel berwarna putih yang direpresentasikan degan nilai 1, yaitu sebanyak 21. Maka dapat dihitung prosentase jumlah nilai piksel putih pada citra biner yaitu:

$$Jumlah\,piksel\,putih = \frac{21}{64}\ x\ 100\% = 32.81\%$$

6. Dilakukan perbandingkan antara presentase jumlah piksel putih dengan nilai *threshold* antara bayangan dan mobil yang ditentukan melalui beberapa uji coba. Misal didapatkan *threshold* sebesar 10% maka prosentase jumlah piksel putih lebih besar dari nilai *threshold* (32.81% > 10%). Sehingga dapat disimpulkan bahwa terdapat obyek berupa mobil pada citra masukan lahan parkir mobil.

7. Proses seleksi dengan *AND function* antara hasil dari *vehicle detection* dan *pre-processing*. Pada proses *vehicle detection* didapatkan bahwa terdapat obyek pada tempat parkir sehingga direpresentasikan dengan nilai 1, begitu pula pada *pre-processing* dihasilkan nilai 1. Dari kedua hasil tersebut dimasukkan pada tabel kebenaran *AND*, sehingga menghasilkan nilai 1 (1 & 1 → 1). Dengan demikian dapat disimpulkan bahwa pada citra masukan tersebut terdapat obyek berupa mobil.

3.4. Perancangan Uji Coba

Pada sub bab ini dilakukan perancangan uji coba perangkat lunak pendeteksi tempat parkir mobil yang kosong. Pengujian dilakukan dengan membandingkan antara citra asli dan citra keluaran yang dihasilkan oleh perangkat lunak, serta menghitung tingkat keakuratan dari citra hasil pendeteksian.

3.4.1. Citra Uji

Citra yang diuji adalah citra yang memiliki spesifikasi sebagai berikut:

1. 35 buah citra tempat parkir mobil
2. Citra warna 24 bit, RGB

3. Tipe file citra uji adalah .bmp
4. Citra berukuran 572 x 275, 572 x 322, dan 572 x 345
5. Citra uji didapatkan dari pengambilan gambar secara manual di parkir mobil lantai 3 pusat perbelanjaan *Malang Town Square* (Matos), parkir mobil bagian depan *Mall Olympic Garden* (MOG) menggunakan kamera digital. Terdapat pula citra uji yang bersumber dari penulis jurnal berjudul "*Vacant Parking Space Detection in Static Image*", Nicholas True.

3.4.2. Lingkungan Pengujian

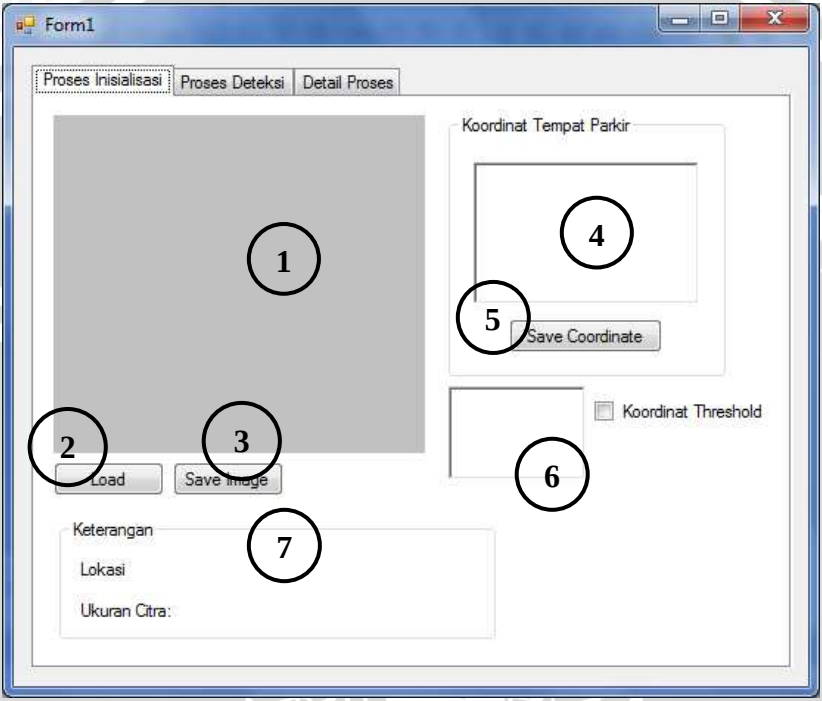
Perangkat lunak yang digunakan antara lain adalah *Microsoft Excel* sebagai media penyimpanan koordinat setiap tempat parkir dalam bentuk file bertipe .csv, *MS Paint* dan *Adobe Photoshop* untuk manipulasi citra uji dan citra referensi pada proses inisialisasi. Sedangkan perangkat keras yang digunakan dalam pengujian sama dengan yang digunakan dalam proses implementasi.

3.5. Perancangan *User Interface*

Rancangan tampilan untuk aplikasi pendeteksi tempat parkir mobil terdiri dari satu halaman utama dengan tiga buah *tab control*, yaitu *tab* Proses Inisialisasi, *tab* Proses Deteksi, dan *tab* Detail Proses. Antarmuka *tab* Proses Inisialisasi secara garis besar terdiri dari tombol untuk memilih citra masukan yang akan diolah, keterangan citra, tombol untuk memproses citra, panel untuk citra masukan dan citra keluaran, text box untuk menampilkan koordinat titik sudut, serta tombol *save* citra hasil proses dan *save* koordinat. Tampilan dari perancangan *user interface* untuk tab Proses Inisialisasi dapat dilihat pada gambar 3.15.

Antarmua *tab* Proses Deteksi secara garis besar terdiri dari panel untuk citra masukan dan citra keluaran, keterangan untuk citra masukan dan citra keluaran, tombol untuk memilih citra masukan yang akan diolah, dan tombol untuk memproses citra masukan. Tampilan dari perancangan *user interface* untuk tab Proses Deteksi dapat dilihat pada gambar 3.16.

Sedangkan antarmuka tab Detail Proses secara garis besar terdiri dari panel untuk citra masukan dan citra keluaran, tombol untuk memilih citra masukan yang akan diproses, dan beberapa radio button untuk melihat setiap proses. Tampilan dari perancangan *user interface* untuk tab Detail Proses dapat dilihat pada gambar 3.17.

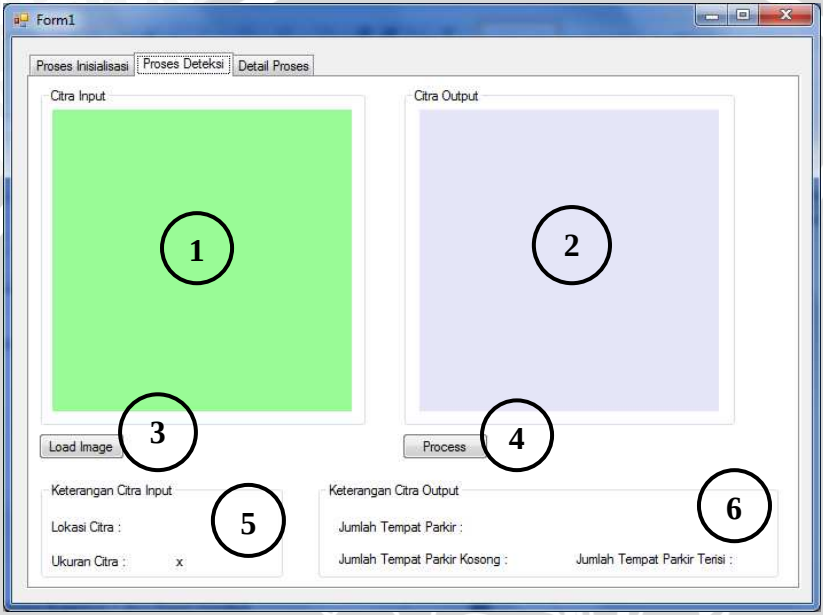


Gambar 3.15. Rancangan *User Interface Tab* Proses Inisialisasi

Keterangan gambar:

1. Tempat untuk menampilkan citra masukan
2. Tombol untuk memilih citra masukan yang akan diproses
3. Tombol untuk menyimpan citra
4. Tempat untuk menampilkan koordinat titik sudut
5. Tombol untuk menyimpan koordinat titik sudut pada file .txt

6. Tempat untuk menampilkan koordinat titik sudut kotak tempat parkir yang akan digunakan sebagai acuan
7. Informasi mengenai citra yang akan diproses, berupa lokasi dan ukuran citra.

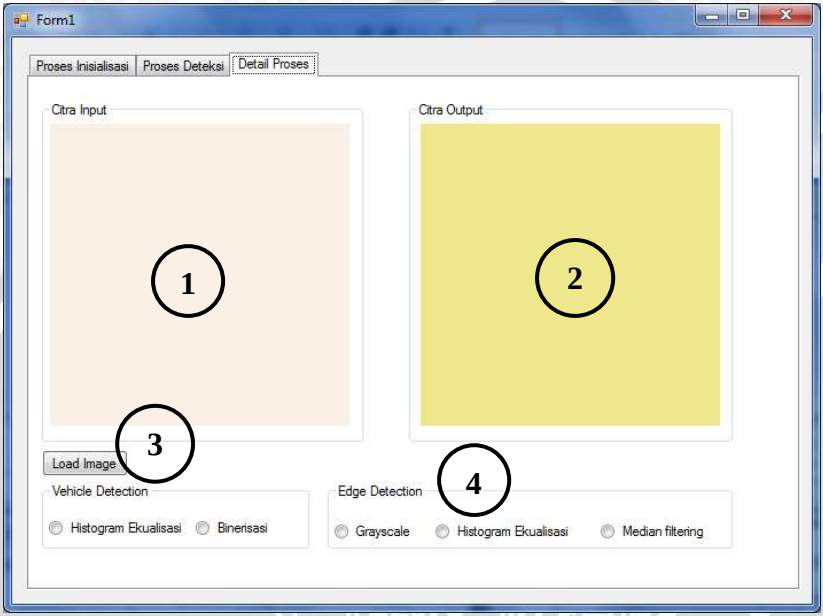


Gambar 3.16. Rancangan *User Interface Tab* Proses Deteksi

Keterangan gambar:

1. Tempat untuk menampilkan citra masukan
2. Tempat untuk menampilkan citra hasil dari proses deteksi
3. Tombol untuk memilih citra masukan yang akan diproses
4. Tombol untuk melakukan proses deteksi
5. Informasi mengenai citra masukan yang akan diproses, berupa lokasi dan ukuran citra

6. Informasi mengenai citra hasil proses deteksi, berupa jumlah tempat parkir, jumlah tempat parkir kosong, dan jumlah tempat parkir yang terisi.



Gambar 3.17. Rancangan *User Interface Tab* Detail Proses

Keterangan gambar:

1. Tempat untuk menampilkan citra masukan
2. Tempat untuk menampilkan citra hasil dari proses pengolahan citra masukan
3. Tombol untuk memilih citra masukan yang akan diproses
4. Beberapa *radio button* untuk memproses citra masukan, diantaranya histogram ekualisasi, binerisasi, konversi citra menjadi keabuan (*grayscale*), dan *median filtering*.

3.6. Pengujian Kualitas Citra Hasil Deteksi

Rancangan tabel pengujian keakuratan citra hasil deteksi tempat parkir terdiri dari 5 buah kolom, yaitu:

1. Jumlah tempat parkir yang ada pada citra masukan.
2. *Threshold* yang digunakan dalam pengujian.
3. Prosentase tingkat keberhasilan pendeteksian tempat yang terisi.
4. Prosentase tingkat keberhasilan pendeteksian tempat yang kosong.
5. Prosentase tingkat keberhasilan pendeteksian tempat yang kosong dan terisi.

Tabel pengujian tingkat keakuratan hasil deteksi tempat parkir dapat dilihat pada tabel 3.14.

Tabel 3.14. Tabel Pengujian

Jumlah Tempat	***Threshold***	**Isi (%)**	**Kosong(%)**	**Kosong dan Isi (%)**

BAB IV

IMPLEMENTASI DAN PEMBAHASAN

Pada bab implementasi dan pembahasan ini akan dijelaskan mengenai proses-proses yang telah dirancang pada bab tiga. Hal-hal yang akan dibahas antara lain adalah implementasi dari proses-proses yang telah dirancang beserta antar muka program, dan bagian-bagian *source code* yang ada dalam implementasi program. Pada bab ini juga dilakukan analisa terhadap hasil yang diperoleh dari deteksi tempat kosong pada lahan parkir mobil.

4.1. Lingkungan Implementasi

Proses implementasi merupakan tahapan penerapan rancangan sistem pada bahasa pemrograman yang dapat dimengerti oleh komputer. Lingkungan implementasi yang dijabarkan pada bab ini meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

4.1.1. Lingkungan Perangkat Keras

Pada penelitian ini, perangkat keras yang digunakan adalah sebuah *notebook* dengan spesifikasi sebagai berikut.

1. Processor Intel® Pentium® Dual CPU T2370 @ 1.73 GHz
2. RAM 1.00 GB
3. *Hardisk* dengan kapasitas 120 GB
4. Monitor 15”
5. *Keyboard*
6. *Mouse*
7. Kamera digital

Perangkat-perangkat keras ini digunakan sebagai pendukung dalam penelitian ini. *Notebook* berfungsi sebagai perangkat pembangun sistem, sedangkan kamera digital digunakan untuk memperoleh data citra lahan parkir mobil.

4.1.2. Lingkungan Perangkat Lunak

Perangkat lunak yang digunakan pada pengembangan sistem deteksi tempat kosong pada lahan parkir mobil ini adalah sebagai berikut.

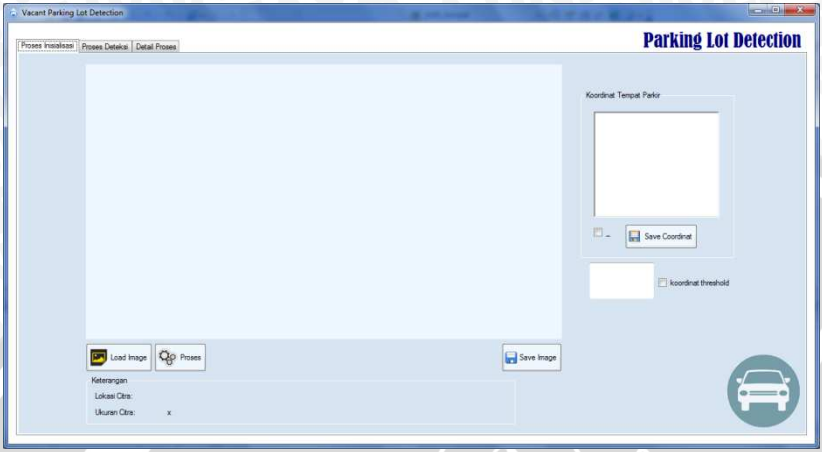
1. Sistem Operasi *Microsoft Windows 7 Ultimate.*
2. *Microsoft Visual C# 2005 Express Edition* sebagai *software development* dalam implementasi rancangan sistem.
3. *Corel Draw* dan *Adobe Photoshop* untuk mengedit data citra.
4. *Notepad* sebagai media penyimpanan koordinat sudut tempat parkir.

4.2. Implementasi Antarmuka

Berdasarkan perancangan pada sub bab 3.5, maka dibuatlah antar muka untuk pendeteksian tempat kosong pada lahan parkir. Tampilan utama pada implementasi ini adalah sebuah form yang terdiri dari tiga *tab control*, yaitu tab untuk proses inisialisasi, tab untuk deteksi tempat kosong, dan tab untuk melihat detail proses. Tampilan utama dari implementasi ini dapat dilihat pada Gambar 4.1.

4.2.1. Antarmuka Proses Inisialisasi

Pada *tab* proses inisialisasi terdapat tombol *Load Image* untuk menginputkan citra tempat parkir yang akan dideteksi garis pembatasnya. Citra masukan pada proses inisialisasi ini adalah citra tempat parkir yang kosong.

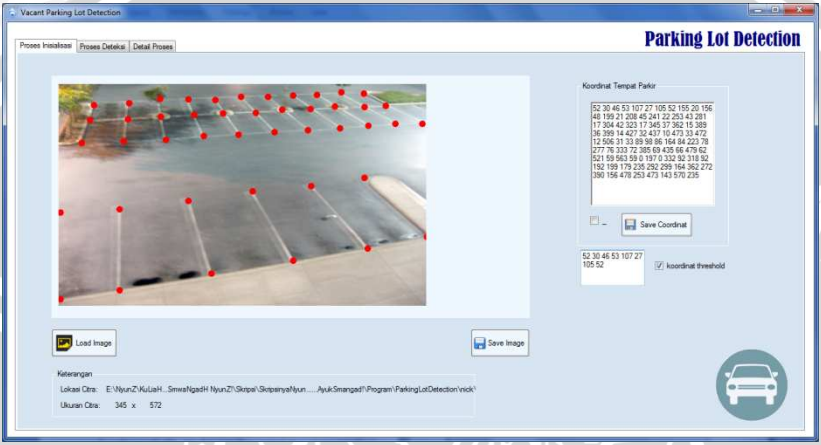


Gambar 4.1. Implementasi *Interface*

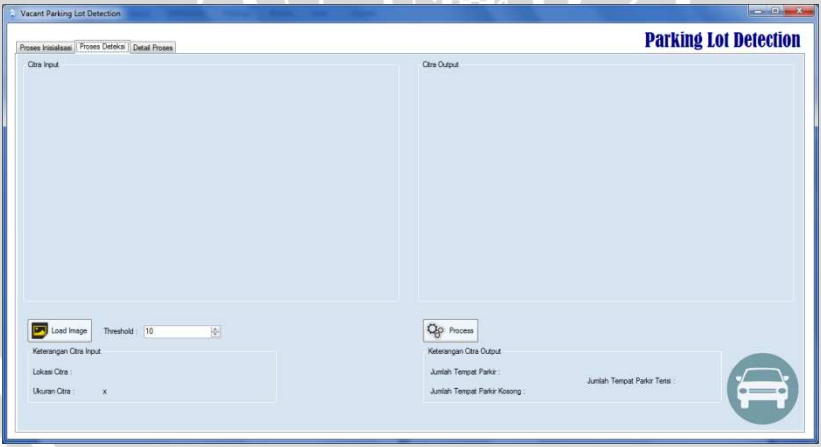
Dari citra tempat parkir yang telah dipilih, kemudian dicari koordinat titik sudut untuk setiap kotak tempat parkir dengan cara menekan titik sudut-titik sudut pada citra hasil proses inisialisasi, maka koordinat titik sudut tempat parkir yang ditekan akan tertulis pada *text box*. Untuk memilih kotak parkir yang akan dijadikan acuan dalam penentuan *threshold*, dilakukan dengan mencentang *check box* koordinat *threshold* dan menekan keempat titik sudut kotak tempat parkir acuan. Kotak tempat parkir yang dijadikan acuan sebaiknya adalah kotak yang berukuran sedang, tidak terlalu besar dan tidak terlalu kecil. Setelah semua koordinat sudut dipilih, kemudian *user* menekan tombol *Save Coordinate* untuk menyimpan koordinat pada file .txt, dan menekan tombol *Save Image* untuk menyimpan gambar hasil proses inisialisasi yang nantinya akan digunakan pada proses deteksi. Hasil dari proses inisialisasi beserta koordinat titik sudut yang telah dipilih dapat dilihat pada gambar 4.2.

4.2.2. Antar Muka Proses Deteksi

Pada *tab* proses deteksi terdapat tombol *Load Image* untuk menginputkan citra tempat parkir yang akan dideteksi, dan tombol Proses untuk melakukan deteksi tempat kosong pada lahan parkir mobil. Tampilan antarmuka untuk *tab* proses deteksi dapat dilihat pada gambar 4.3.

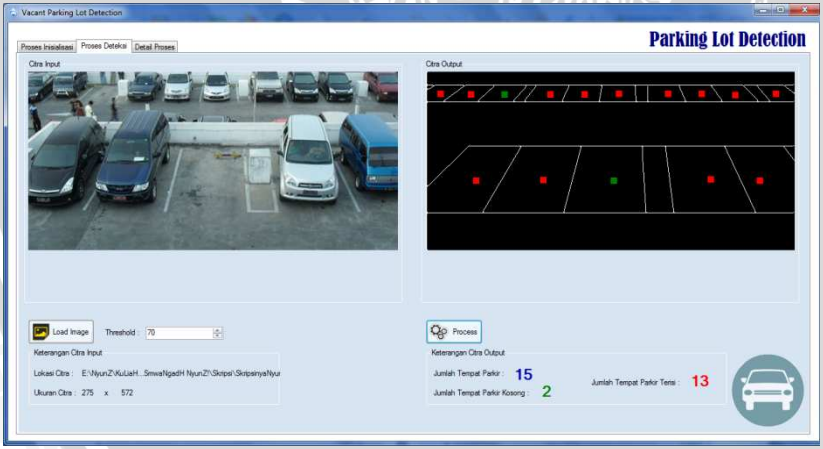


Gambar 4.2. *Interface Tab* Proses Inisialisasi



Gambar 4.3. *Interface Tab* Proses Deteksi

Citra masukan pada proses deteksi ini adalah citra lahan parkir mobil yang terisi dalam beberapa kondisi. Setelah menekan tombol *Load Image* dan memilih citra masukan, citra yang dipilih akan ditampilkan di sebelah kiri pada bagian Citra Input. Kemudian *user* memilih salah satu *threshold* yang tersedia sebagai batas untuk menentukan kondisi tempat parkir, dan selanjutnya menekan tombol Proses untuk melakukan pendeteksian tempat kosong pada lahan parkir mobil. Citra hasil proses deteksi akan ditampilkan di sebelah kanan pada bagian Citra *Output*. Selain itu juga terdapat keterangan mengenai citra hasil deteksi pada bagian Keterangan Citra Output, berupa jumlah seluruh tempat parkir, jumlah tempat parkir yang kosong, dan jumlah tempat parkir yang terisi. Pada citra hasil deteksi, tempat parkir yang kosong atau terisi akan diberi tanda pada setiap kotaknya. Untuk tempat parkir yang kosong akan diberi tanda berwarna hijau, sedangkan tempat parkir yang terisi akan diberi tanda berwarna merah. Hasil deteksi tempat kosong pada lahan parkir mobil dapat dilihat pada gambar 4.4.

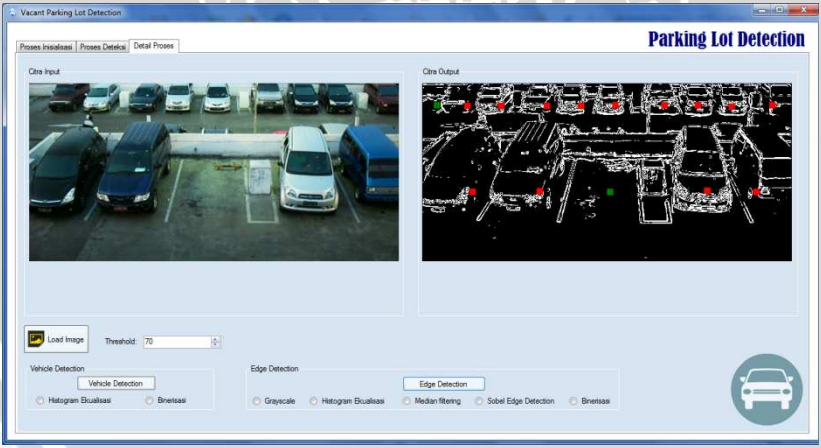


Gambar 4.4. *Interface* Hasil Proses Deteksi

Proses pendeteksian ini melewati beberapa proses, antara lain proses *histogram equalization*, proses konversi citra menjadi *grayscale*, *image subtraction*, binerisasi, *median filtering*, *sobel edge detection*, proses penghitungan jumlah piksel warna putih, proses *thresholding* untuk penentuan kondisi tempat parkir, proses AND *function*, dan proses menggambar kotak sebagai tanda kondisi tempat parkir (kosong atau terisi).

4.2.3. Antar Muka Detail Proses

Pada *tab* detail proses terdapat tombol *Load Image* untuk menginputkan citra yang ingin diolah, tombol untuk melihat hasil dari proses *vehicle detection* dan *edge detection*, *numeric up down* untuk pilihan *threshold* sebagai batas untuk menentukan kondisi tempat parkir, dan beberapa *radio button* untuk setiap proses, yaitu *histogram equalization*, konversi citra ke *grayscale*, *image subtraction*, *median filtering*, *sobel edge detection*, dan binerisasi. Tampilan antarmuka untuk *tab* detail proses dapat dilihat pada gambar 4.5.



Gambar 4.5. *Interface Tab* Detail Proses

Setelah menekan tombol *Load Image* dan memilih citra masukan, citra yang dipilih akan ditampilkan di sebelah kiri pada bagian Citra Input. Kemudian *user* memilih *threshold*, dan menekan salah satu tombol atau *radio button* yang tersedia. Citra hasil proses pengolahan akan ditampilkan di sebelah kanan pada bagian Citra *Output*.

4.3. Implementasi Program

Berdasarkan perancangan proses yang terdapat pada bab tiga, maka pada subbab ini akan dijelaskan mengenai implementasi proses-proses tersebut. Secara garis besar proses dikelompokkan menjadi empat tahap, yaitu tahap inisialisasi, tahap *vehicle detection*, *edge detection*, tahap penentuan status tempat parkir, dan tahap AND *function*.

4.3.1. Implementasi Tahap Inisialisasi

Tahap inisialisasi menghasilkan koordinat setiap titik sudut pada tempat parkir. Proses pemilihan setiap titik sudut dituliskan pada *Source Code* 4.1.

```
private void pbInisialisasi_MouseClick(object sender,
MouseEventArgs e)
{
   if (checkBox1.Checked)
   { tbKoordinat_thres.AppendText(e.X +" "+ e.Y +" "); }
   else
   rtb.AppendText(e.X + " " + e.Y + " ");
}
```

***Sorce Code* 4.1. Proses Pemilihan Koordinat Titik Sudut**

Jika *checkBox1* dicentang, maka koordinat akan dituliskan pada *text box tbKoordinat_thres*. Koordinat pada *text box* ini menunjukkan koordinat titik sudut kotak tempat parkir yang digunakan sebagai acuan dalam menentukan *threshold*.

Pada proses selanjutnya, koordinat titik sudut yang sudah dipilih disimpan pada file .txt. Nama file untuk penyimpanan koordinat titik sudut kotak parkir ini bergantung pada citra masukan pada proses inisialisasi. Proses penyimpanan koordinat titik sudut untuk setiap kotak tempat parkir dituliskan pada *Source Code* 4.2.

```
//simpan koordinat ke notepad
private void SaveToFile(string input1, string input2)
{
    try
    {
        if (lokasi == lokasi_matos)
        {
            FileInfo fi = new FileInfo
                          ("koordinat_matos.txt");

            FileStream fs = fi.Create();
            StreamWriter sw = new StreamWriter(fs);
            sw.WriteLine(input1);
            sw.WriteLine(input2);
            sw.Close();
            MessageBox.Show("koordinat sudah
                             disimpan");
        }
        else
        {
            FileInfo fi = new FileInfo
                          ("koordinat_nick.txt");
            FileStream fs = fi.Create();
            StreamWriter sw = new StreamWriter(fs);
            sw.WriteLine(input1);
            sw.WriteLine(input2);
            sw.Close();
            MessageBox.Show("koordinat sudah
                             disimpan");
        }
    }
    catch (Exception ex)
    { MessageBox.Show(ex.Message); }
}
```

***Source Code* 4.2. Proses PenyimpananKoordinat Titik Sudut**

4.3.2. Implementasi Tahap *Vehicle Detection*

4.3.2.1. *Histogram Equalization*

Pada proses *histogram ekualization*, sebaran warna RGB (*Red*, *Green*, dan *Blue*) pada citra masukan lahan parkir akan dibuat lebih merata. Implementasi proses *histogram equalization* dapat dilihat pada *Source Code* 4.3.

```
public void ekualisasi()
{
   double[] merah = new double[256];
   double[] hijau = new double[256];
   double[] biru = new double[256];
   double[,] temp = new double[3, 256];
   int total = 0;

   //hitung frekuensi
   for (int i = 0; i < tinggi; i++)
   {
      for (int j = 0; j < lebar; j++)
      {
         merah[Pixel[i, j].Merah]++;
         hijau[Pixel[i, j].Hijau]++;
         biru[Pixel[i, j].Biru]++;
         total++;
      }
   }
   //hitung nilai ekualisasi
   for (int p = 0; p < 256; p++)
   {
      merah[p] /= total;
      hijau[p] /= total;
      biru[p] /= total;
      for (int q = 0; q < p; q++)
      {
         temp[0, p] += merah[q];
         temp[1, p] += hijau[q];
         temp[2, p] += biru[q];

      }
      // ubah domain cdf dari [0..1] ke [0..255]
      temp[0, p] *= 255;
      temp[1, p] *= 255;
      temp[2, p] *= 255;
   }
    //ganti nilai piksel dengan nilai ekualisasi
   for (int i = 0; i < tinggi; i++)
   {
      for (int j = 0; j < lebar; j++)
      {
         red[i, j] = (byte)temp[0, Pixel[i, j].Merah];
         green[i, j] = (byte)temp[1, Pixel[i, j].Hijau];
         blue[i, j] = (byte)temp[2, Pixel[i, j].Biru];
      }    }    }
```

Source Code* 4.3. Proses *Histogram Equalization

4.3.2.2. *Image Subtraction*

Pada proses *image subtraction*, citra referensi yang telah ditentukan sebelumnya, yaitu citra lahan parkir mobil yang masih kosong (citra keabuan), akan dikurangkan dengan citra hasil proses konversi ke *grayscale* untuk mendapatkan objek yang diharapkan berupa mobil. Citra referensi yang digunakan bergantung pada citra masukan yang dipilih oleh *user*. Sistem secara otomatis akan memilih citra referensi sesuai dengan citra masukan. Proses *image subtraction* dituliskan pada *Source Code* 4.4.

```
public int[,] subtraction(int[,] data1, int[,] data2)
{
   subtract_result = new int[data1.GetLength(0),
                             data1.GetLength(1)];
   for (int i = 0; i < data1.GetLength(0); i++)
   {
      for (int j = 0; j < data1.GetLength(1); j++)
      {
         subtract_result[i, j] = (byte) System.Math.Abs
                                 (data1[i, j] - data2[i, j]);
      }
   }
           return subtract_result;
}
```

Source Code* 4.4. Proses *Image Subtraction

4.3.2.3. Binerisasi dengan *Otsu Thresholding*

Pada proses binerisasi, citra hasil dari proses *image subtraction* diubah menjadi citra biner yang hanya memiliki nilai piksel 0 dan 1 menggunakan metode *Otsu* untuk menentukan *threshold*nya. Jika nilai piksel pada suatu koordinat *(i,j)* lebih besar atau sama dengan *threshold* maka nilai piksel tersebut akan diubah menjadi 255, dan jika kurang dari *threshold* maka nilai pikselnya diubah menjadi 0. Proses binerisasi dengan metode *Otsu* dituliskan pada *Source Code* 4.5.

```
public int otsu()
{
   float sum = 0;
   int[] level = new int[256];
   int threshold = 0;

   for (int i = 0; i < tinggi; i++)
   {
      for (int j = 0; j < lebar; j++)
      {
         int n = Pixel[i, j].Merah;
         level[n]++;
      }
   }

   for (int k = 0; k < 256; k++) sum += (k * level[k]);

   float sumB = 0;
   float wB = 0;
   float wF = 0;
   float varMax = 0;

   for (int z = 0; z < 256; z++)
   {
      wB += level[z]; //weight background
      wF = (lebar * tinggi) - wB; //weight foreground

      sumB += (float)(z * level[z]);

      float mB = sumB / wB;//mean background
      float mF = (sum - sumB) / wF;

      //between class variance
      float varBetween = (float)wB*wF*(mB-mF)*(mB-mF);

      if (varBetween > varMax)
      {
         varMax = varBetween;
         threshold = z;
      }
    }
    return threshold;
}
        //end otsu
```

```
public void threshold(int[,] thres_data)
{
  int threshold = otsu();

  for (int i = 0; i < tinggi; i++)
  {
     for (int j = 0; j < lebar; j++)
     {
        if (thres_data[i, j] >= threshold)
           { thres_data[i, j] = 255; }
        else
           { thres_data[i, j] = 0; }
     }
  }
}
```

Source Code* 4.5. Proses Binerisasi dengan *Otsu Thresholding

4.3.3. Implementasi Tahap *Edge Detection*

4.3.3.1. Konversi ke *Grayscale*

Citra hasil dari proses *histogram equalization* terlebih dahulu dikonversi menjadi citra keabuan sebelum menuju proses pengurangan dua citra. Proses konversi citra berwarna menjadi citra keabuan (*grayscale*) dapat dilihat pada Source Code 4.6.

```
public void grayscale()
{
   for (int i = 0; i < tinggi; i++)
   {
      for (int j = 0; j < lebar; j++)
      {
         byte grey = (byte)((Pixel[i, j].Merah +
                     Pixel[i, j].Hijau +
                     Pixel[i, j].Biru) / 3);
         gray_results[i,j] = grey;
      }
   }
}
```

Source Code* 4.6. Proses Konversi ke *Grayscale

4.3.3.2. *Median Filtering*

Pada proses *median filtering*, citra masukan yang telah melalui proses konversi ke *grayscale* dan *histogram equalization* akan direduksi *noise*nya dengan cara mengambil mask 3x3 pada citra yang ditampung pada variabel satu dimensi yaitu *red*. Kemudian nilai-nilai piksel pada kernel tersebut diurutkan dan diambil nilai tengahnya (*red[4]*) untuk dimasukkan pada kernel 3x3. Implementasi proses *median filtering* dapat dilihat pada *Source Code* 4.7.

```
public void median()
{
   byte[] red = new byte[9];

   for (int i = 1; i < tinggi - 1; i++)
   {
      for (int j = 1; j < lebar - 1; j++)
      {
         int t_j = j - 1;
         int t_i = i - 1;

         for (int index = 0; index < 9; index++)
         {
            red[index] = (byte)gray_results[t_i, t_j];
            if (t_j - (j - 1) == 2)
            {
               t_i += 1;
               t_j = j - 1;
            }
            else { t_j += 1; }
         }
         for (int m = 0; m < 8; m++)
         {
            for (int n = m + 1; n < 9; n++)
            {
               if (red[m] > red[n])
               {
                  byte temp = red[m];
                  red[m] = red[n];
                  red[n] = temp;
               }  }  }
           median_results[i, j] = red[4];
       }     }
}
```

Source Code* 4.7. Proses *Median Filtering

4.3.3.3. *Sobel Edge Detection*

Hasil proses *median filtering* selanjutnya dideteksi tepi objeknya menggunakan metode *Sobel Edge Detection*. Deteksi tepi sobel memiliki dua operator *mask* yang berbentuk matriks konvolusi berukuran 3x3, yaitu secara vertikal (*dy*) dan secara horizontal (*dx*). Proses *sobel edge detection* dituliskan pada *Source Code* 4.8.

```
public void sobel()
{
   int[,] dx = new int[,] { { -1, 0, 1 }, { -2, 0, 2 },
                            { -1, 0, 1 } };
   int[,] dy = new int[,] { { 1, 2, 1 }, { 0, 0, 0 },
                            { -1, -2, -1 } };

   for (int i = 1; i < tinggi - 1; i++)
   {
      for (int j = 1; j < lebar - 1; j++)
      {
         double gx = 0;
         double gy = 0;

         //mengambil mask 3x3 & mengalikan dgn operator
         for (int p = -1; p < 2; p++)
         {
            for (int q = -1; q < 2; q++)
            {
               gx += gray_results[i + p, j + q] *
                     dx[p + 1, q + 1];

               gy += gray_results[i + p, j + q] *
                     dy[p + 1, q + 1];
            }
         }
         sobel_results[i, j] = (byte)(Math.Sqrt(gx * gx
                                + gy * gy));
      }
   }
   //memasukkan nilai hasil sobel
   for (int m = 1; m < tinggi - 1; m++)
   {
      for (int n = 1; n < lebar - 1; n++)
      {
         gray_results[m, n] = sobel_results[m, n];
      }   }   }
```

Source Code* 4.8. Proses *Sobel Edge Detection

Citra hasil dari proses *sobel edge detection* ini selanjutnya diubah menjadi citra biner seperti pada proses 4.3.2.4.

4.3.4. Implementasi Tahap Penentuan Status Tempat Parkir

4.3.4.1. Pembacaan Koordinat pada File

Citra biner hasil dari tahap *vehicle detection* dan *edge detection* selanjutnya dihitung jumlah piksel warna putihnya untuk setiap kotak tempat parkir. Pada proses ini, sebelumnya sistem akan membaca koordinat yang telah tersimpan pada file .txt untuk selanjutnya dilakukan perhitungan berdasarkan koordinat tersebut. Proses pembacaan koordinat pada file .txt dapat dilihat pada *Source Code* 4.9.

```
public void ReadFromFile(string direktori_txt)
{
   try
   {
      string temp = null;
      data = new ArrayList();

      StreamReader read = new StreamReader
                          (direktori_txt);
      char[] d ={ ' ' };

      temp = read.ReadLine();
      string[] temp_item2 = temp.Split(d);

      for (int m = 0; m < koordinat_thres.Length; m++)
      {
         koordinat_thres[m] = Convert.ToInt16
                              (temp_item2[m]);
      }

      while (((temp = read.ReadLine()) != null))
      {
         string[] temp_item = temp.Split(d);

         for (int i = 0; i < temp_item.Length - 1; i++)
         {
```

```
                data.Add(temp_item[i]);
            }
        }
                    read.Close();
    }
    catch (Exception ex)
    { MessageBox.Show(ex.Message); }
}
```

***Source Code* 4.9. Proses Pembacaan Koordinat dari File**

4.3.4.2. Perhitungan Jumlah Warna Putih

Setelah melakukan pembacaan koordinat pada file .txt, dilakukan perhitungan jumlah nilai piksel berwarna putih untuk setiap kotak menggunakan batasan (*x1, y1*) dan (*x4, y4*). Setiap kotak yang akan dihitung nilai piksel putih di dalamnya, terlebih dahulu dilakukan pengecekan terhadap panjang dan tinggi kotak tersebut. Kemudian dibandingkan dengan panjang dan tinggi kotak acuan untuk penentuan *threshold*. Implementasi tahap penentuan status tempat parkir dapat dilihat pada *Source Code* 4.10.

```
public ArrayList sum_white(int[,] data_proses)
{
    int idx = 0;
    int panjang_thres = 0;
    int tinggi_thres = 0;
    jmlh_tempat = 0;
    jmlh_putih = new ArrayList();

    if (koordinat_thres[0] <= koordinat_thres[2])
        { koordinat_thres[0] = koordinat_thres[2]; }
    if (koordinat_thres[4] >= koordinat_thres[6])
        { koordinat_thres[4] = koordinat_thres[6]; }
    if (koordinat_thres[1] <= koordinat_thres[5])
        { koordinat_thres[1] = koordinat_thres[5]; }
    if (koordinat_thres[3] <= koordinat_thres[7])
        { koordinat_thres[7] = koordinat_thres[3]; }
    if (koordinat_thres[1] >= koordinat_thres[5])
        { koordinat_thres[5] = koordinat_thres[1]; }
    if (koordinat_thres[3] >= koordinat_thres[7])
        { koordinat_thres[3] = koordinat_thres[7]; }
```

```
    panjang_thres = Math.Abs(koordinat_thres[4] –
                             koordinat_thres[0]);
    tinggi_thres = Math.Abs(koordinat_thres[3] –
                            koordinat_thres[1]);

    for (int i = 0; i < index.Count - 1; i++)
    {
      for (int m = (int)index[i] + 1; m < (int)
           index[i + 1] - 7; m += 4)
      {
        if ((m + 7) < data.Count)
        {
          jmlh_tempat++;
          int x1 = Convert.ToInt16((string)(data[m]));
          int y1 = Convert.ToInt16((string)(data[m+1]));
          int x2 = Convert.ToInt16((string)(data[m+2]));
          int y2 = Convert.ToInt16((string)(data[m+3]));
          int x3 = Convert.ToInt16((string)(data[m+4]));
          int y3 = Convert.ToInt16((string)(data[m+5]));
          int x4 = Convert.ToInt16((string)(data[m+6]));
          int y4 = Convert.ToInt16((string)(data[m+7]));

          if (x1 <= x2)
          { x1 = x2; }
          if (x3 >= x4)
          { x3 = x4; }
          if (y1 <= y3)
          { y1 = y3; }
          if (y2 <= y4)
          { y4 = y2; }
          if (y1 >= y3)
          { y3 = y1; }
          if (y2 >= y4)
          { y2 = y4; }

          int panjang_awal = Math.Abs(x4 - x1);
          int tinggi_awal = Math.Abs(y4 - y1);

    if ((panjang_awal > panjang_thres) &&
        (tinggi_awal > tinggi_thres))
        {
          int selisih_panjang = Math.Abs((panjang_awal –
                                panjang_thres) / 2);
          int selisih_tinggi = Math.Abs((tinggi_awal –
                               tinggi_thres) / 2);

          x1 = x1 + selisih_panjang;
```

```
            x4 = x4 - selisih_panjang;
            y1 = y1 + selisih_tinggi;
            y4 = y4 - selisih_tinggi;
        }

        idx = 0;
        for (int k = x1; k <= x4; k++)
        {
            for (int j = y1; j <= y4; j++)
            {
                if (data_proses[j, k] == 255)
                {
                    idx++;
                }
            }
        }
            jmlh_putih.Add(idx);
      }
    }
  }
  return jmlh_putih;
}
```

***Source Code* 4.10. Perhitungan Jumlah Warna Putih**

4.3.4.3. Penentuan Kondisi Tempat Parkir

Setelah diketahui jumlah piksel warna putih untuk setiap kotak, selanjutnya dilakukan pengecekan jumlah warna putih terhadap *threshold* yang dipilih oleh user. Kotak parkir yang memiliki jumlah piksel warna putih lebih besar dari *threshold*, maka kotak tersebut diartikan telah terisi (diberi nilai 1). Sebaliknya, jika kotak parkir yang memiliki jumlah piksel warna putih kurang dari *threshold*, maka kotak tersebut masih kosong (diberi nilai 0). Implementasi tahap penentuan status tempat parkir dapat dilihat pada *Source Code* 4.11.

```
public ArrayList cond(ArrayList putih, double thres)
{
   kondisi = new ArrayList();
   for (int p = 0; p < putih.Count; p++)
   {
     if ((int)putih[p] > thres)
     {
       kondisi.Add(1);
     }
     else if ((int)putih[p] <= thres)
     {
       kondisi.Add(0);
     }
   }
   return kondisi;
}
```

***Source Code* 4.11. Penentuan Kondisi Tempat Parkir**

4.3.5. Implementasi Tahap *AND Function*

4.3.5.1. AND *Function*

Masing-masing proses *vehicle detection* dan *edge detection* memiliki hasil berupa status untuk setiap kotak tempat parkir yang diberi nilai 1 jika tempat terisi dan 0 jika tempat kosong. Dari hasil tersebut, nilai dari status setiap kotak tempat parkir dikomparasikan menggunakan *AND Function*, dimana kondisi benar (nilai 1) akan terpenuhi jika dua kondisi yang dibandingkan bernilaia benar semuanya. Implementasi tahap *AND Function* dapat dilihat pada *Source Code* 4.12.

```
public void AND(ArrayList data1, ArrayList data2)
{
   int idx = 0;
   for (int n = 0; n < data.Count; n += 4)
   {
      if ((n + 7) < data.Count)
      {
         if (((int)data1[idx] == 0) &&
             ((int)data2[idx] == 0))
            { kondisi.Add(0); }

         else if (((int)data1[idx] == 0) &&
                  ((int)data2[idx] == 1))
                { kondisi.Add(0); }

         else if (((int)data1[idx] == 1) &&
                  ((int)data2[idx] == 0))
                { kondisi.Add(0); }
         else if (((int)data1[idx] == 1) &&
                  ((int)data2[idx] == 1))
                { kondisi.Add(1); }
      }
    }
}
```

Source Code* 4.12. Proses *AND Function

4.3.5.2. Penggambaran Tanda Status Tempat Parkir

Setiap kotak tempat parkir yang telah memiliki nilai 1 atau 0 hasil dari proses *AND Fuction*, selanjutnya diberi tanda berwarna hijau jika kotak tempat parkir bernilai 0 dan tanda berwarna merah jika kotak tempat parkir bernilai 1. Warna hijau menunjukkan tempat parkir yang kosong dan warna merah menunjukkan tempat parkir yang terisi. Proses penggambaran tanda status pada tempat parkir dapat dilihat pada *Source Code* 4.13.

```
public Bitmap draw_rect(PictureBox output)
{
   int idx = 0;
   Bitmap b = new Bitmap(output.Image);
   Graphics graph = Graphics.FromImage(b);

   for (int n = 0; n < data.Count; n += 4)
   {
    if ((n + 7) < data.Count)
    {
       int x1 = Convert.ToInt16((string)(data[n]));
       int y1 = Convert.ToInt16((string)
                (data[n + 1]));
       int x4 = Convert.ToInt16((string)
                (data[n + 6]));
       int y4 = Convert.ToInt16((string)
                (data[n + 7]));

       //gambar tanda kotak merah
       if ((int)kondisi[idx] == 1)
       {
         graph.DrawRectangle(new Pen(Color.Red, 5),
                    new Rectangle(((x1 + x4) / 2),
                    ((y1 + y4) / 2), 5, 5));
         tempat_isi++;
       }

       //gambar tanda kotak hijau
       else if ((int)kondisi[idx] == 0)
       {
          graph.DrawRectangle(new Pen
                 (Color.Green, 5),
                 new Rectangle(((x1 + x4) / 2),
                 ((y1 + y4) / 2), 5, 5));
          tempat_kosong++;
       }
       idx++;
    }
  }
  return b;
}
```

***Source Code* 4.13. Proses Penggambaran Tanda Status**

4.4. Implementasi Uji Coba

Pada subbab ini akan dilakukan pembahasan mengenai pengujian yang telah dilakukan pada sistem dan hasil evaluasi dari hasil deteksi yang dikeluarkan oleh sistem.

4.4.1. Hasil Pengujian Deteksi

Nilai dari parameter yang ditentukan berpengaruh terhadap tingkat akurasi hasil deteksi. Parameter tersebut adalah *threshold* yang digunakan sebagai batasan penentuan status tempat parkir, kosong atau terisi. Nilai *threshold* yang digunakan pada pengujian adalah 10, 30, dan 70. Pengujian ini dilakukan dengan menggunakan 35 gambar lahan parkir. Selain itu, pengujian juga dilakukan dengan menggunakan *threshold* 20, 40, 50, dan 60 untuk lebih mengetahui detail perbedaan tingkat akurasi yang dipengaruhi oleh *threshold*. Dari 35 gambar ini terdiri dari 12 gambar dengan lokasi di *Malang Town Square* (Matos), 5 gambar dengan lokasi di *Mall Olympic Garden* (MOG), dan 18 gambar yang bersumber dari penulis jurnal Nicholas True. Citra lahan parkir mobil yang digunakan pada pengujian ini dapat dilihat pada lampiran 1. Pada proses pendeteksian tempat kosong pada lahan parkir mobil ini terdapat dua kemungkinan hasil, yaitu benar atau salah. Kondisi benar merupakan kondisi dimana proses mendapatkan tempat kosong dengan benar, sebalikanya kondisi salah jika proses tidak mendapatkan tempat kosong dengan benar.

4.4.1.1. Hasil Pengujian Data Lokasi Matos

Hasil deteksi tempat kosong pada lahan parkir mobil terhadap data gambar percobaan yang berlokasi di matos dapat dilihat pada tabel 4.1.

Tabel 4.1. Hasil Deteksi Lokasi Matos

Jumlah Tempat	*Threshold*	Isi	Kosong	Kosong dan Isi
15	10	99,4%	60,3%	93,8%
15	20	99,4%	70,1%	96%
15	30	98,8%	70,8%	87,7%
15	40	96,8%	70,8%	95,4%
15	50	96,8%	70,8%	95,4%
15	60	95%	83,3%	94,9%
15	70	92,6%	83,3%	92,8%

Dari tabel 4.1 didapatkan hasil keakuratan proses deteksi tempat parkir mobil di lokasi matos, dengan jumlah tempat parkir sebanyak 15 dan *threshold* 10, adalah sebesar 99,4% untuk akurasi deteksi tempat parkir yang terisi, 60,3% untuk akurasi deteksi tempat yang kosong, dan 93,8% untuk akurasi deteksi keseluruhan lahan parkir, baik yang terisi maupun kosong. Dari hasil deteksi dengan *threshold* 30 didapatkan tingkat akurasi sebesar 98,8% untuk deteksi tempat parkir yang kosong, 70,8% untuk akurasi deteksi tempat yang kosong, dan 87,7% untuk akurasi deteksi keseluruhan lahan parkir, baik yang terisi maupun kosong. Sedangkan untuk hasil deteksi dengan *threshold* 70 didapatkan tingkat akurasi sebesar 92,6% untuk tempat parkir yang terisi, 83,3% untuk akurasi tempat parkir yang terisi, dan 92,8% untuk akurasi kesuluruhan lahan parkir, baik yang terisi maupun kosong.

4.4.1.2. Hasil Pengujian Data yang Bersumber Dari Jurnal

Hasil deteksi pada lahan parkir mobil terhadap data gambar percobaan yang berasal dari jurnal dapat dilihat pada tabel 4.2.

Tabel 4.2. Hasil Deteksi Data yang Bersumber Dari Jurnal

Jumlah Tempat	*Threshold*	Isi	Kosong	Kosong dan Isi
25	10	85,7%	54,2%	74,2%
25	20	80%	60,6%	72%
25	30	78,4%	69,7%	74,4%
25	40	74,6%	69,3%	72%
25	50	66,4%	71,4%	67,3%
25	60	63,8%	73,6%	67,3%
25	70	62,6%	77,8%	67,1%

Dari tabel 4.2 didapatkan hasil keakuratan proses deteksi tempat parkir mobil dengan data yang bersumber dari jurnal, dengan jumlah tempat parkir sebanyak 25 dan *threshold* 10, adalah sebesar 85,7% untuk akurasi deteksi tempat parkir yang terisi, 54,2% untuk akurasi deteksi tempat yang kosong, dan 74,2% untuk akurasi deteksi keseluruhan lahan parkir, baik yang terisi maupun kosong. Dari hasil deteksi dengan *threshold* 30 didapatkan tingkat akurasi sebesar 78,4% untuk deteksi tempat parkir yang kosong, 69,7% untuk akurasi deteksi tempat yang kosong, dan 74,4% untuk akurasi deteksi keseluruhan lahan parkir, baik yang terisi maupun kosong. Sedangkan untuk hasil deteksi dengan *threshold* 70 didapatkan tingkat akurasi sebesar 62,6% untuk tempat parkir yang terisi, 77,8% untuk akurasi tempat parkir yang terisi, dan 67,1% untuk akurasi kesuluruhan lahan parkir, baik yang terisi maupun kosong.

4.4.1.3. Hasil Pengujian Data Lokasi MOG

Hasil deteksi pada lahan parkir mobil terhadap data gambar yang berlokasi di MOG dapat dilihat pada tabel 4.3.

Tabel 4.3. Hasil Deteksi Lokasi MOG

Jumlah Tempat	*Threshold*	Isi	Kosong	Kosong dan Isi
27	10	74,6%	62,8%	68,2%
27	20	64%	76%	71,2%
27	30	63,4%	93,6%	72,6%
27	40	61,8%	93,6%	71,2%
27	50	55,2%	98%	70,4%
27	60	55,4%	98%	70,4%
27	70	51,2%	98%	66,6%

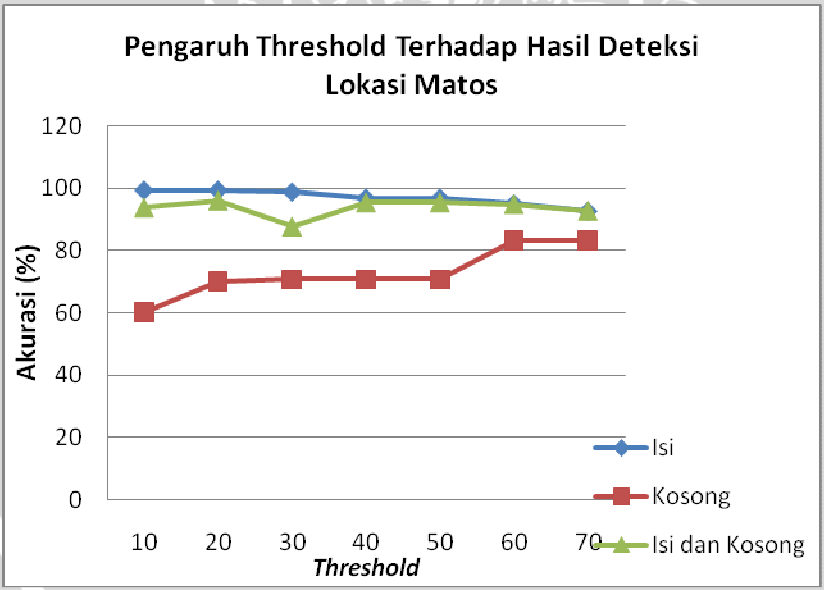
Dari tabel 4.3 didapatkan hasil keakuratan proses deteksi tempat parkir mobil dengan data yang bersumber dari jurnal, dengan jumlah tempat parkir sebanyak 27 dan *threshold* 10, adalah sebesar 74,6% untuk akurasi deteksi tempat parkir yang terisi, 62,8% untuk akurasi deteksi tempat yang kosong, dan 68,2% untuk akurasi deteksi keseluruhan lahan parkir, baik yang terisi maupun kosong. Dari hasil deteksi dengan *threshold* 30 didapatkan tingkat akurasi sebesar 63,4% untuk deteksi tempat parkir yang kosong, 93,6% untuk akurasi deteksi tempat yang kosong, dan 72,6% untuk akurasi deteksi keseluruhan lahan parkir, baik yang terisi maupun kosong. Sedangkan untuk hasil deteksi dengan *threshold* 70 didapatkan tingkat akurasi sebesar 51,2% untuk tempat parkir yang terisi, 98% untuk akurasi tempat parkir yang terisi, dan 66,6% untuk akurasi kesuluruhan lahan parkir, baik yang terisi maupun kosong.

4.5. Analisa Hasil

4.5.1. Analisa Hasil Pengujian Data Lokasi Matos

Pada pengujian dilakukan dengan menggunakan tiga buah *threshold* yang berbeda nilainya, yaitu 10, 30, dan 70. Pemilihan nilai *threshold* ini didasarkan pada percobaan dengan beberapa nilai, dan didapatkan ketiga nilai tersebut yang memiliki hasil deteksi yang

paling signifikan perbedaannya. Batasan nilai minimum 10 dan maksimum 70 dipilih karena jika nilai *threshold* kurang dari nilai minimum atau lebih dari nilai maksimum, maka hasil pendeteksian akan menghasilkan banyak kesalahan sehingga tingkat akurasi yang dihasilkan menjadi rendah. Sedangkan pengujian dengan *threshold* 20, 40, 50, dan 60 bertujuan untuk mengetahui detail perubahan akurasi. Dari tabel 4.1 didapatkan bahwa *threshold* sebesar 10 dan 20 merupakan *threshold* yang paling bagus untuk mendeteksi tempat parkir yang terisi. *Threshold* sebesar 60 dan 70 merupakan *threshold* yang paling bagus untuk mendeteksi tempat parkir yang kosong di lokasi Matos. Sedangkan *threshold* sebesar 40 dan 50 merupakan *threshold* yang paling bagus untuk mendeteksi tempat parkir secara keseluruhan. Grafik yang menggambarkan pengaruh besar *threshold* terhadap deteksi tempat parkir, baik terisi, kosong, maupun keselurhan, di lokasi matos dapat dilihat pada gambar 4.6.

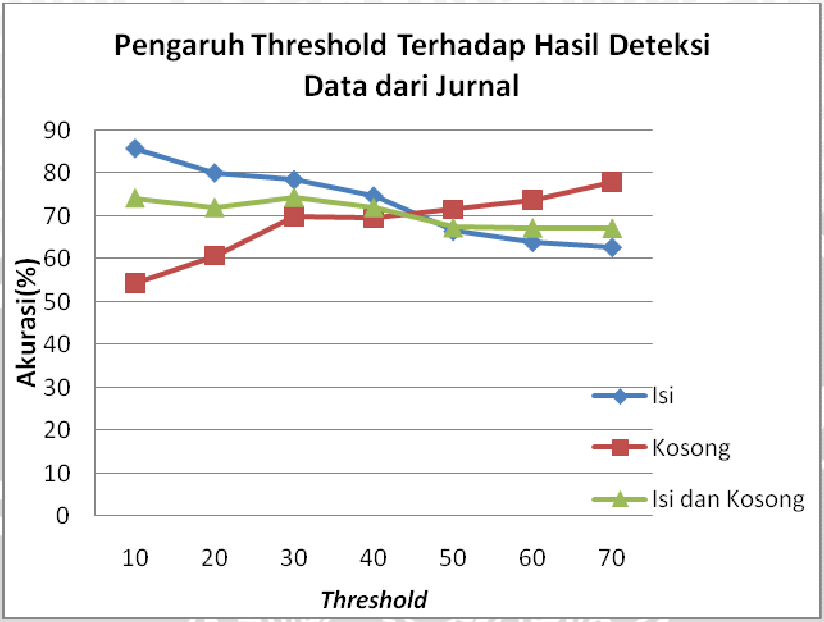


Gambar 4.6. Grafik Pengaruh *Threshold* terhadap Hasil Deteksi Lokasi Matos

Dari grafik tersebut dapat dilihat bahwa *threshold* 10 dan 20 menghasilkan tingkat akurasi yang paling tinggi sebesar 99,4% pada saat mendeteksi tempat parkir yang terisi. Tingkat akurasi 99,4% dapat diartikan bahwa sistem dapat mendeteksi tempat parkir yang terisi dengan benar sebesar 99,4% dan kesalahan deteksi sebesar 0,6% dari keseluruhan jumlah tempat parkir. Hal ini menunjukkan bahwa metode yang diterapkan sudah cukup optimal untuk mendeteksi keberadaan tempat parkir yang terisi. Pada garis biru yang menunjukkan pengaruh *threshold* terhadap tingkat akurasi pendeteksian tempat yang terisi, didapatkan bahwa semakin besar nilai *threshold* maka akurasi yang dihasilkan akan semakin rendah. Hal ini berkebalikan dengan pengaruh *threshold* terhadap tingkat akurasi untuk mendeteksi tempat kosong yang ditunjukkan dengan garis merah. Semakin besar nilai *threshold*, maka akan semakin besar pula tingkat akurasi yang dihasilkan.

4.5.2. Analisa Hasil Pengujian Data yang Bersumber dari Jurnal

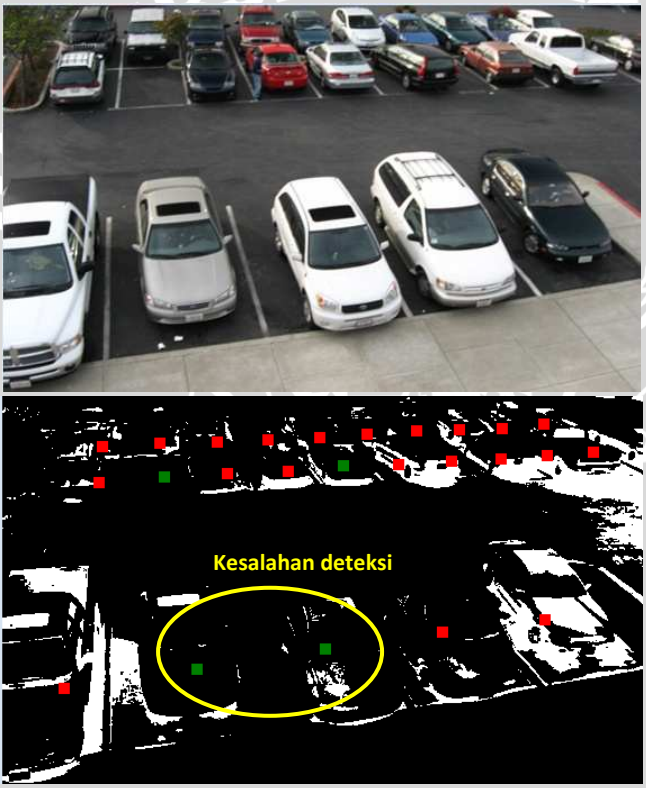
Dari tabel 4.2 didapatkan bahwa *threshold* sebesar 10 merupakan *threshold* yang paling bagus untuk mendeteksi tempat parkir yang terisi. *Threshold* sebesar 70 merupakan *threshold* yang paling bagus untuk mendeteksi tempat parkir yang kosong, dan *threshold* 30 merupakan *threshold* yang paling bagus untuk mendeteksi keseluruhan tempat parkir. Grafik yang menggambarkan pengaruh besar *threshold* terhadap deteksi tempat parkir, baik terisi, kosong, maupun keselurhan, pada data yang bersumber dari jurnal dapat dilihat pada gambar 4.7.



Gambar 4.7. Grafik Pengaruh *Threshold* terhadap Hasil Deteksi yang Bersumber dari Jurnal

Dari grafik dapat dilihat bahwa *threshold* 10 menghasilkan tingkat akurasi yang paling tinggi sebesar 85,77% pada saat mendeteksi tempat parkir yang terisi. Garis biru pada grafik menunjukkan bahwa semakin besar nilai *threshold*, maka tingkat akurasi yang dihasilkan akan semakin rendah. Tingkat akurasi 85,7% dapat diartikan bahwa sistem dapat mendeteksi tempat parkir yang terisi dengan benar sebesar 85,7% dan kesalahan deteksi sebesar 14,3% dari keseluruhan jumlah tempat parkir. Kesalahan deteksi pada tempat parkir yang terisi, dapat disebabkan oleh beberapa hal. Salah satunya adalah warna mobil yang cenderung gelap atau terlalu terang. Ketika dilakukan proses *vehicle detection* dan *edge detection* untuk warna mobil yang gelap seperti hitam, atau yang terlalu terang seperti putih, akan dihasilkan gambar biner dengan dominasi warna

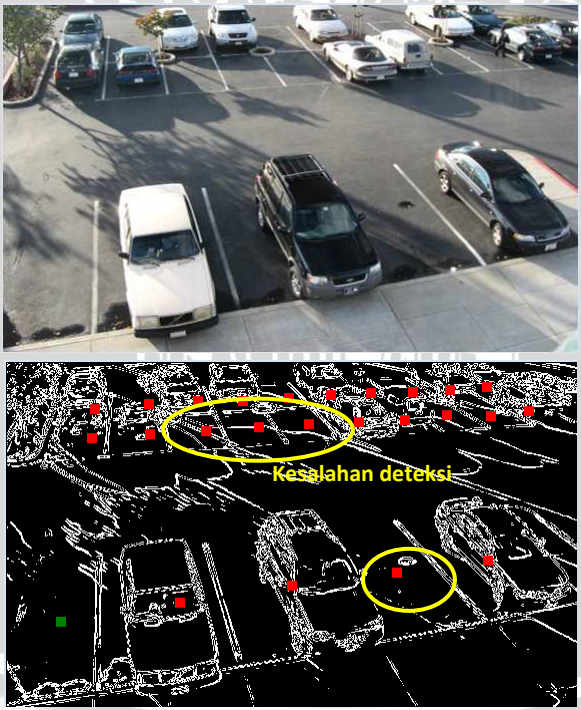
hitam pada tempat parkir yang seharusnya berisi mobil. Hal tersebut menyebabkan jumlah piksel warna putih pada kotak parkir yang berisi mobil berwarna hitam atau putih akan lebih kecil dari *threshold* yang telah ditentukan, sehingga sistem akan menunjukkan bahwa tempat parkir tersebut kosong. Contoh kesalahan pada saat mendeteksi tempat parkir yang terisi oleh mobil berwarna gelap atau terang dapat dilihat pada gambar 4.8.



Gambar 4.8. Contoh Kesalahan Deteksi Tempat Isi

Dari grafik pada gambar 4.7. juga dapat dilihat tingkat akurasi yang paling tinggi pada saat mendeteksi tempat parkir yang kosong adalah 77,8% dengan *threshold* 70. Tingkat akurasi 77,8%

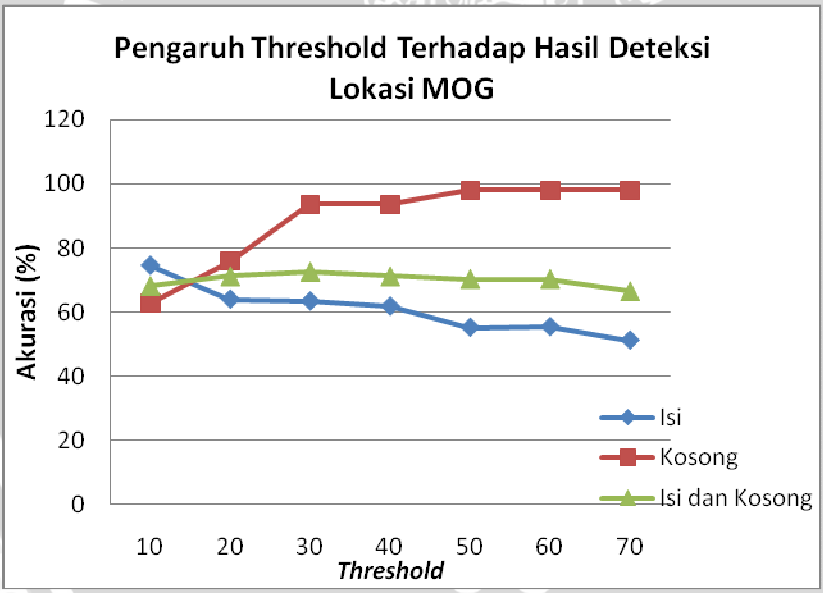
dapat diartikan bahwa sistem dapat mendeteksi tempat parkir yang kosong dengan benar sebesar 77,8% dan kesalahan deteksi sebesar 22,4% dari keseluruhan jumlah tempat parkir. Kesalahan sistem mendeteksi tempat parkir yang seharusnya kosong dapat disebabkan oleh beberapa hal. Salah satunya adalah adanya *noise* berupa bayangan mobil yang mengenai tempat parkir yang kosong, atau adanya genangan air pada tempat tersebut. Jika terdapat *noise* tersebut pada citra masukan, ketika dilakukan proses *edge detection* maka sistem akan menghasilkan gambar biner dengan jumlah piksel warna putih yang banyak pada kotak parkir kosong tersebut. Sehingga ketika dilakukan seleksi menggunakan *threshold* yang telah ditentukan, jumlah piksel warna putih pada kotak tersebut akan lebih besar dibandingkan *threshold*. Jadi sistem akan menunjukkan bahwa tempat parkir yang seharusnya kosong tersebut terisi. Contoh kesalahan pada saat mendeteksi tempat parkir yang kosong dapat dilihat pada gambar 4.9.



Gambar 4.9. Contoh Kesalahan Deteksi Tempat Kosong

4.5.1. Analisa Hasil Pengujian Data Lokasi MOG

Dari tabel 4.3 didapatkan bahwa *threshold* sebesar 10 merupakan *threshold* yang paling bagus untuk mendeteksi tempat parkir yang terisi. *Threshold* sebesar 50, 60, dan 70 merupakan *threshold* yang paling bagus untuk mendeteksi tempat parkir yang kosong, dan *threshold* 30 merupakan *threshold* yang paling bagus untuk mendeteksi keseluruhan tempat parkir. Grafik yang menggambarkan pengaruh besar *threshold* terhadap deteksi tempat parkir, baik terisi, kosong, maupun keselurhan, pada data yang berlokasi di MOG dapat dilihat pada gambar 4.10.

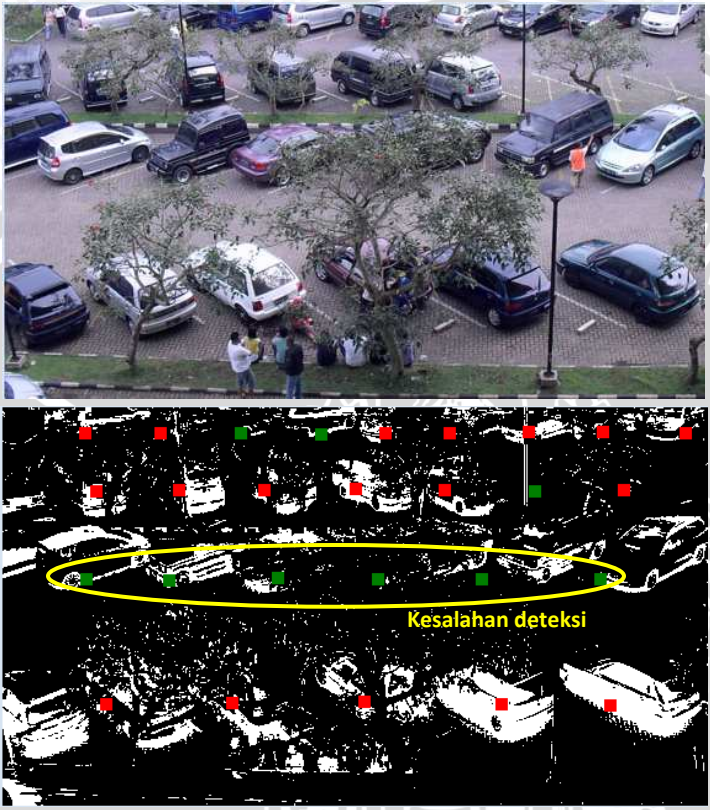


Gambar 4.10. Grafik Pengaruh *Threshold* terhadap Hasil Deteksi Lokasi MOG

Dari grafik dapat dilihat bahwa *threshold* 70 menghasilkan tingkat akurasi yang paling tinggi sebesar 98% pada saat mendeteksi

tempat parkir yang kosong. Tingkat akurasi 98% dapat diartikan bahwa sistem dapat mendeteksi tempat parkir yang kosong dengan benar sebesar 98% dan kesalahan deteksi sebesar 2% dari keseluruhan jumlah tempat parkir. Pengaruh *threshold* pada hasil pendeteksian tempat parkir yang kosong, yaitu semakin besar nilai *threshold*, maka semakin tinggi pula tingkat akurasi yang dihasilkan. Sebaliknya, semakin tinggi nilai *threshold* pada saat mendeteksi tempat parkir yang terisi, maka akan semakin rendah tingkat akurasi yang dihasilkan. Hasil akurasi paling tinggi untuk mendeteksi tempat parkir yang terisi adalah 68, 2%. Hail ini berarti sistem dapat mendeteksi keberadaan tempat parkir yang kosong dengan benar sebesar 68, 2% dan tingkat kesalahannya sebesar 31, 8%.

Kesalahan sistem pada saat mendeteksi kondisi tempat parkir di lokasi MOG ini disebabkan oleh beberapa hal. Diantaranya adalah, tempat parkir tertutup oleh pohon-pohon yang terletak di sekitarnya, sehingga pada saat dilakukan proses vehicle detection akan dihasilkan gambar dengan dominasi piksel warna hitam pada tempat parkir yang seharusnya tertutup pohon. Sehingga sistem akan mendeteksi tempat tersebut sebagai tempat yang kosong. Selain itu kesalahan deteksi pada data dengan lokasi MOG disebabkan karna tempat parkir di baris kedua dari bawah, memiliki kemiringan yang besar. Sehingga pada saat perhitungan batas untuk area kotak parkir pada deret tersebut akan dihasilkan luasan yang sangat kecil. Maka dari itu sistem akan mendeteksi kotak tersbut sebagai kotak yang kosong, padahal terisi. Kesalahan deteksi juga dapat disebabkan pengguna tempat parkir yang tidak memarkir mobilnya sesuai dengan garis pembatas yang telah disediakan. Terdapat beberapa mobil yang parkir melebihi garis pembatas. Hal ini akan mengakibatkan sistem salah mendeteksi kondisi tempat parkir. Contoh kesalahan ini dapat dilihat pada gambar 4.11.



Gambar 4.11. Contoh Kesalahan Deteksi Di Lokasi MOG


UNIVERSITAS BRAWIJAYA

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Dari hasil percobaan dan pembahasan yang telah dilakukan, dapat disimpulkan beberapa hal sebagai berikut:

1. Untuk mendeteksi tempat kosong pada lahan parkir mobil menggunakan metode *vehicle detection* dan *edge detection*, terdapat 3 tahap pada proses *vehicle detection*, yaitu *histogram equalization*, *image subtraction*, dan binerisasi. Sedangkan pada proses *edge detection* terdapat 5 tahap, yaitu konversi citra warna menjadi *grayscale*, *histogram equalization*, *median filtering*, *sobel edge detection*, dan binerisasi. Hasil kedua tahap tersebut dibandingkan menggunakan operator logika AND. *Threshold* yang digunakan sebagai batas penetuan kondisi tempat parkir pada penelitian ini adalah 10, 20, 30, 40, 50, 60, dan 70.
2. Nilai akurasi pendeteksian dengan metode *vehicle detection* dan *sobel edge detection* memiliki nilai yang beragam tergantung dari penentuan nilai *threshold*. Nilai akurasi pendeteksian tertinggi untuk deteksi tempat parkir yang terisi adalah 99,4% dengan *threshold* 10 dengan data yang berlokasi di *Malang Town Square* (Matos). Nilai akurasi pendeteksian tertinggi untuk deteksi tempat kosong adalah 98% dengan *threshold* 50, 60, dan70 dengan data yang berlokasi di *Mall Olympic Garden* (MOG). Sedangkan nilai akurasi pendeteksian tertinggi untuk keseluruhan tempat parkir adalah 96% dengan *threshold* 20 pada data yang berlokasi di Matos

5.2. Saran

Sebagai penelitian lebih lanjut, beberapa saran yang dapat menjadi pertimbangan untuk dilakukan adalah:

1. Pengambilan citra lahan parkir menghindari sudut yang miring (dari samping) atau lahan parkir yang digunakan tidak memiliki kemiringan yang besar pada kotak parkirnya sehingga luasan tempat parkir yang dihitung dapat lebih maksimal.
2. Perlu ditambahkan sebuah metode untuk menghilangkan *noise* pada lahan parkir, sehingga dapat mengurangi tingkat kesalahan pendeteksian.
3. Pencarian titik sudut pada saat proses inisialisasi dapat dikembangkan menjadi otomatis dengan salah satu algoritma titik sudut, sehingga dapat lebih menghemat waktu inisialisasi.
4. Dapat dikembangkan sebuah metode yang lebih efektif dan efisien untuk menghitung luasan setiap tempat parkir.

DAFTAR PUSTAKA

Acharya, Tinku, Ray, Ajoy K. 2005. *Image Processing Principles and Applications*. Canada: John Wiley & Sons, Inc.

Ahmad, Usman. 2009. *10 Langkah Membuat Program Pengolahan Citra Menggunakan Visual C#*. Yogyakarta: Graha Ilmu.

Ajmal, Aisha, M.Hussain, Ibrahim. 2009. *Morphological Process Based Vehicle Detection and Classification*. MASAUM Journal Of Basic and Applied Sciences.

Anonim. 2004. *Techincal Document About FAR, FRR, and EER version 1.0*. SYRIS Technology Corp.

Bong, D.B.L., Ting, K.C., dan Lai, K.C. 2008. *Integrated Approach in the Design of Car Park Occupancy Information System (COINS)*. IAENG International Journal of Computer Science.

Eom Ki-Yeol, Ahn Tae-Ki, dkk. 2009. *Hierarchically Categorized Performance Evaluation Criteria for Intelligent Surveillance System*. Korea: SungKyunKwan Univ./Information and Communication

Fang, Mei, Yue GuangXue, dan Yu QingCang. 2009. *The Study on An Application of Otsu Method in Canny Operator*. China: College of Mathematics and Information Engineering, Jiaxing University.

Gonzalez, Rafael C., Woods, Richard E. 2002. *Digital Image Processing second edition*. New Jersey: Prentice Hall.

Green, Bill. 2002. *Edge Detection Tutorial*. http://www.pages.drexel.edu/. Diakses pada tanggal 16 Februari 2010.

Herdiyeni, Yeni. 2009. *Bahan Ajar Deteksi Tepi (Edge Detection)*. Bogor: Departemen Ilmu Komputer Fakultas MIPA IPB.

Han Sungji, Han Youngjoon, dan Hahn Hernsoo. 2009. *Vehicle Detection Method using Haar-like Feature on Real Time System*. World Academy of Science, Engineering, and Technology.

Hsu Chihping, Tanaka Toshimitsu,dkk. 2002. *Locating Vehicle in a Parking Lot by Image Processing*. Jepang: Department of Electrical and Electronic Engineering, Meijo University.

Idris, M.Y.I, Y.Y. Leng, dkk. 2009. *Car Park System: A Review Of Smart Parking System And Its Technology*. Malaysia: Faculty Of Computer Science And Information Technology, University Of Malaya.

Idris, M.Y.I, E.M. Tamil, dkk. 2009. *Smart Parking System Using Image Processing Techniques in Wireless Sensor Network Environment*. Malaysia: Faculty Of Computer Science And Information Technology, University Of Malaya.

Indra. *Image Processing-part 1&2*. http://ai.indra-ehm.net/. Diakses tanggal 15 Juni 2010.

Irawan, Dwi Ardi. 2006. *Pengolahan Citra Digital*. http://www.dwiardiirawan.com/. Diakses tanggal 22 Oktober 2009.

Litman, Todd. 2008. *Parking Management – Strategies, Evaluation, and Planning*. Canada: Victoria Transport Policy Institute.

Munir, Renaldi. 2004. *Pengolahan Citra Digital dengan Pendekatan Algoritmik. Bandung: Penerbit Informatika*.

Phillips, Dwayne. 2000. *Image Processing In C Second Edition*. Kansas: R & D Publications.

Sutoyo, T., Mulyanto, Edi, dkk. 2009. *Teori Pengolahan Citra Digital*. Yogyakarta: Penerbit Andi.

Winarno, Edy. 2009. *Pengolahan Citra*. Semarang: FTI Unisbank.

LAMPIRAN

Lampiran 1 Data Citra Uji

 matos_1.bmp	 matos_2.bmp
 matos_3.bmp	 matos_4.bmp
 matos_5.bmp	 matos_6.bmp
 matos_7.bmp	 matos_8.bmp

 matos_9.bmp	 matos_10.bmp
 matos_11.bmp	 matos_12.bmp
 nick_1.bmp	 nick_2.bmp
 nick_3.bmp	 nick_4.bmp

nick_5.bmp

nick_6.bmp

nick_7.bmp

nick_8.bmp

nick_9.bmp

nick_10.bmp

nick_11.bmp

nick_12.bmp

 nick_13.bmp	 nick_14.bmp
 nick_15.bmp	 nick_16.bmp
 nick_17.bmp	 nick_18.bmp
 _mog_1.bmp	 _mog_2.bmp

_mog_3.bmp

_mog_4.bmp

_mog_5.bmp

Lampiran 2 Pendeteksian Data Uji Lokasi Matos dengan *Threshold* = 10

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	matos_1	15	2	13	2	0	100	12	1	93	14	1	93
2	matos_2		13	2	13	0	100	0	2	0	13	2	87
3	matos_3		13	2	13	0	100	1	1	50	14	1	93
4	matos_4		14	1	14	0	100	0	1	0	14	1	93
5	matos_5		14	1	14	0	100	1	0	100	15	0	100
6	matos_6		14	1	14	0	100	0	1	0	14	1	93
7	matos_7		13	2	13	0	100	1	1	50	14	1	93
8	matos_8		15	0	15	0	100	0	0	100	15	0	100
9	matos_9		4	11	4	0	100	9	2	81	13	2	87
10	matos_10		15	0	14	1	93	0	0	100	14	1	93
11	matos_11		13	2	13	0	100	1	1	50	14	1	93
12	matos_12		15	0	15	0	100	0	0	100	15	15	100

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 3 Pendeteksian Data Uji Lokasi Matos dengan *Threshold* = 20

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	matos_1	15	2	13	2	0	100	13	0	100	15	0	100
2	matos_2		13	2	13	0	100	0	2	0	13	2	87
3	matos_3		13	2	13	0	100	1	1	50	14	1	93
4	matos_4		14	1	14	0	100	0	1	0	14	1	93
5	matos_5		14	1	14	0	100	1	0	100	15	0	100
6	matos_6		14	1	14	0	100	0	1	0	14	1	93
7	matos_7		13	2	13	0	100	2	0	100	15	0	100
8	matos_8		15	0	15	0	100	0	0	100	15	0	100
9	matos_9		4	11	4	0	100	10	1	91	14	1	93
10	matos_10		15	0	14	1	93	0	0	100	14	1	93
11	matos_11		13	2	13	0	100	2	0	100	15	0	100
12	matos_12		15	0	15	0	100	0	0	100	15	15	100

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 4 Pendeteksian Data Uji Lokasi Matos dengan *Threshold* = 30

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	matos_1	15	2	13	2	0	100	13	0	100	15	0	100
2	matos_2		13	2	13	0	100	0	2	0	13	2	87
3	matos_3		13	2	13	0	100	1	1	50	14	1	93
4	matos_4		14	1	14	0	100	0	1	0	14	1	93
5	matos_5		14	1	14	0	100	1	0	100	15	0	100
6	matos_6		14	1	14	0	100	0	1	0	14	1	93
7	matos_7		13	2	13	0	100	2	0	100	15	0	100
8	matos_8		15	0	14	1	93	0	0	100	14	1	93
9	matos_9		4	11	4	11	100	11	0	100	15	0	100
10	matos_10		15	0	14	1	93	0	0	100	14	1	93
11	matos_11		13	2	13	0	100	2	0	100	15	0	100
12	matos_12		15	0	15	0	100	0	0	100	15	0	100

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 5 Pendeteksian Data Uji Lokasi Matos dengan *Threshold* = 40

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	matos_1	15	2	13	2	0	100	13	0	100	15	0	100
2	matos_2		13	2	13	0	100	0	2	0	13	2	87
3	matos_3		13	2	13	0	100	1	1	50	14	1	93
4	matos_4		14	1	14	0	100	0	1	0	14	1	93
5	matos_5		14	1	14	0	100	1	0	100	15	0	100
6	matos_6		14	1	14	0	100	0	1	0	14	1	93
7	matos_7		13	2	13	0	100	2	0	100	15	0	100
8	matos_8		15	0	14	1	93	0	0	100	14	1	93
9	matos_9		4	11	3	1	75	11	0	100	14	1	93
10	matos_10		15	0	14	1	93	0	0	100	14	1	93
11	matos_11		13	2	13	0	100	2	0	100	15	0	100
12	matos_12		15	0	15	0	100	0	0	100	15	0	100

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 6 Pendeteksian Data Uji Lokasi Matos dengan *Threshold* = 50

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	matos_1	15	2	13	2	0	100	13	0	100	15	0	100
2	matos_2		13	2	13	0	100	0	2	0	13	2	87
3	matos_3		13	2	13	0	100	1	1	50	14	1	93
4	matos_4		14	1	14	0	100	0	1	0	14	1	93
5	matos_5		14	1	14	0	100	1	0	100	15	0	100
6	matos_6		14	1	14	0	100	0	1	0	14	1	93
7	matos_7		13	2	13	0	100	2	0	100	15	0	100
8	matos_8		15	0	14	1	93	0	0	100	14	1	93
9	matos_9		4	11	3	1	75	11	0	100	14	1	93
10	matos_10		15	0	14	1	93	0	0	100	14	1	93
11	matos_11		13	2	13	0	100	2	0	100	15	0	100
12	matos_12		15	0	15	0	100	0	0	100	15	0	100

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 7 Pendeteksian Data Uji Lokasi Matos dengan *Threshold* = 60

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	matos_1	15	2	13	2	0	100	13	0	100	15	0	100
2	matos_2		13	2	13	0	100	1	1	50	14	1	93
3	matos_3		13	2	12	1	92	1	1	50	13	2	87
4	matos_4		14	1	14	0	100	1	0	100	15	0	100
5	matos_5		14	1	14	0	100	1	0	100	15	0	100
6	matos_6		14	1	14	0	100	0	1	0	14	1	93
7	matos_7		13	2	13	0	100	2	0	100	15	0	100
8	matos_8		15	0	14	1	93	0	0	100	14	1	93
9	matos_9		4	11	3	1	75	11	0	100	14	1	93
10	matos_10		15	0	13	2	87	0	0	100	13	2	87
11	matos_11		13	2	13	0	100	2	0	100	15	0	100
12	matos_12		15	0	14	1	93	0	0	100	14	1	93

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 8 Pendeteksian Data Uji Lokasi Matos dengan *Threshold* = 70

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	matos_1	15	2	13	2	0	100	13	0	100	15	0	100
2	matos_2		13	2	12	1	92	1	1	50	13	2	87
3	matos_3		13	2	11	2	85	1	1	50	12	3	80
4	matos_4		14	1	14	0	100	1	0	100	15	0	100
5	matos_5		14	1	14	0	100	1	0	100	15	0	100
6	matos_6		14	1	13	1	92	0	1	0	13	2	87
7	matos_7		13	2	13	0	100	2	0	100	15	0	100
8	matos_8		15	0	13	2	87	0	0	100	13	2	87
9	matos_9		4	11	3	1	75	11	0	100	14	1	93
10	matos_10		15	0	13	2	87	0	0	100	13	2	87
11	matos_11		13	2	13	0	100	2	0	100	15	0	100
12	matos_12		15	0	14	1	93	0	0	100	14	1	93

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 9 Pendeteksian Data Uji yang Bersumber dari Jurnal dengan *Threshold* = 10

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	nick_1	25	16	9	16	0	100	6	3	67	22	3	88
2	nick_2		16	9	9	7	56	4	5	44	13	12	52
3	nick_3		17	8	14	3	82	5	3	63	19	6	76
4	nick_4		17	8	16	1	94	4	4	50	20	5	80
5	nick_5		16	9	16	0	100	4	5	44	20	5	80
6	nick_6		21	4	15	6	71	1	3	25	16	9	64
7	nick_7		5	20	4	1	80	11	9	55	15	10	60
8	nick_8		6	19	6	0	100	10	9	53	16	9	64
9	nick_9		15	10	14	1	93	4	6	40	18	7	72
10	nick_10		14	11	13	1	93	5	6	45	18	7	72
11	nick_11		17	8	15	2	88	4	4	50	19	6	76
12	nick_12		14	11	14	0	100	5	6	45	19	6	76

13	nick_13		12	13	9	3	75	11	2	85	20	5	80
14	nick_14		12	13	10	2	83	10	3	77	20	5	80
15	nick_15		18	7	17	1	94	5	2	71	22	3	88
16	nick_16		20	5	18	2	90	2	3	40	21	4	84
17	nick_17		22	3	20	2	91	1	2	33	21	4	84
18	nick_18		15	8	8	7	53	7	1	88	15	10	60

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 10 Pendeteksian Data Uji yang Bersumber dari Jurnal dengan *Threshold* = 20

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	nick_1	25	16	9	16	0	100	6	3	67	22	3	88
2	nick_2		16	9	9	7	56	5	4	56	14	11	56
3	nick_3		17	8	13	4	76	5	3	63	18	7	72
4	nick_4		17	8	17	0	100	5	3	63	22	3	88
5	nick_5		16	9	15	1	94	5	4	56	20	5	80
6	nick_6		21	4	14	7	67	2	2	50	16	9	64
7	nick_7		5	20	4	1	80	11	9	55	15	10	60
8	nick_8		6	19	5	1	83	11	8	58	16	9	64
9	nick_9		15	10	14	1	93	4	6	40	18	7	72
10	nick_10		14	11	13	1	93	5	6	45	18	7	72

11	nick_11		17	8	14	3	82	4	4	50	18	7	72
12	nick_12		14	11	13	1	93	5	6	45	18	7	72
13	nick_13		12	13	9	3	75	11	2	85	20	5	80
14	nick_14		12	13	7	5	58	11	2	85	18	7	72
15	nick_15		18	7	16	2	89	4	3	57	20	5	80
16	nick_16		20	5	15	5	75	3	2	60	18	7	72
17	nick_17		22	3	16	6	73	2	1	67	18	7	72
18	nick_18		15	8	8	7	53	7	1	88	15	10	60

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 11 Pendeteksian Data Uji yang Bersumber dari Jurnal dengan *Threshold* = 30

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	nick_1	25	16	9	16	0	100	7	2	78	23	2	92
2	nick_2		16	9	11	5	69	6	3	67	17	8	68
3	nick_3		17	8	17	0	100	6	2	75	23	2	92
4	nick_4		17	8	15	2	88	6	2	75	21	4	84
5	nick_5		16	9	15	1	94	6	3	67	21	4	84
6	nick_6		21	4	15	6	71	3	1	75	18	7	72
7	nick_7		5	20	4	1	80	13	7	65	17	8	68
8	nick_8		6	19	5	1	83	12	7	63	17	8	68
9	nick_9		15	10	13	2	87	5	5	50	18	7	72
10	nick_10		14	11	10	4	71	5	6	45	15	10	60

11	nick_11		17	8	13	4	76	6	2	75	19	6	76
12	nick_12		14	11	11	3	79	7	4	64	18	7	72
13	nick_13		12	13	8	4	67	11	2	85	19	6	76
14	nick_14		12	13	7	5	58	11	2	85	18	7	72
15	nick_15		18	7	14	4	78	5	2	71	19	6	76
16	nick_16		20	5	15	5	75	3	2	60	18	7	72
17	nick_17		22	3	15	7	68	2	1	67	17	8	68
18	nick_18		15	8	10	5	67	7	1	88	17	8	68

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 12 Pendeteksian Data Uji yang Bersumber dari Jurnal dengan *Threshold* = 40

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	nick_1	25	16	9	16	0	100	7	2	78	23	2	92
2	nick_2		16	9	11	5	69	6	3	67	17	8	68
3	nick_3		17	8	17	0	100	6	2	75	23	2	92
4	nick_4		17	8	13	4	76	5	3	63	18	7	72
5	nick_5		16	9	13	3	81	6	3	67	19	6	76
6	nick_6		21	4	12	9	57	3	1	75	15	10	60
7	nick_7		5	20	4	1	80	13	7	65	17	8	68
8	nick_8		6	19	5	1	83	13	6	68	18	7	72
9	nick_9		15	10	11	4	73	5	5	50	16	9	64
10	nick_10		14	11	10	4	71	5	6	45	15	10	60
11	nick_11		17	8	13	4	76	6	2	75	19	6	76
12	nick_12		14	11	12	2	86	7	4	64	19	6	76

13	nick_13		12	13	7	5	58	11	2	85	18	7	72
14	nick_14		12	13	7	5	58	11	2	85	18	7	72
15	nick_15		18	7	14	4	78	5	2	71	19	6	76
16	nick_16		20	5	15	5	75	3	2	60	18	7	72
17	nick_17		22	3	15	7	68	2	1	67	17	8	68
18	nick_18		15	8	8	7	53	7	1	88	15	10	60

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 13 Pendeteksian Data Uji yang Bersumber dari Jurnal dengan *Threshold* = 50

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	nick_1	25	16	9	16	0	100	7	2	78	23	2	92
2	nick_2		16	9	7	9	44	5	4	56	12	13	48
3	nick_3		17	8	10	7	65	6	2	75	16	9	64
4	nick_4		17	8	11	6	65	6	2	75	17	8	68
5	nick_5		16	9	12	4	75	6	3	67	18	7	72
6	nick_6		21	4	10	11	48	3	1	75	13	12	52
7	nick_7		5	20	4	1	80	13	7	65	17	8	68
8	nick_8		6	19	4	2	67	13	6	68	17	8	68
9	nick_9		15	10	11	4	73	5	5	50	16	9	64
10	nick_10		14	11	8	6	57	6	5	55	14	11	56

11	nick_11		17	8	11	6	65	6	2	75	17	8	68
12	nick_12		14	11	11	3	79	7	4	64	18	7	72
13	nick_13		12	13	6	6	50	11	2	85	17	8	68
14	nick_14		12	13	6	6	50	11	2	85	17	8	68
15	nick_15		18	7	12	6	67	6	1	86	18	7	72
16	nick_16		20	5	15	5	75	3	2	60	18	7	72
17	nick_17		22	3	15	7	68	2	1	67	17	8	68
18	nick_18		15	8	10	5	67	8	0	100	18	7	72

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 14 Pendeteksian Data Uji yang Bersumber dari Jurnal dengan *Threshold* = 60

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	nick_1	25	16	9	15	1	94	7	2	78	23	2	92
2	nick_2		16	9	7	9	44	5	4	56	12	13	48
3	nick_3		17	8	10	7	65	6	2	75	16	9	64
4	nick_4		17	8	11	6	65	6	2	75	17	8	68
5	nick_5		16	9	11	5	69	6	3	67	17	8	68
6	nick_6		21	4	10	11	48	3	1	75	13	12	52
7	nick_7		5	20	4	1	80	14	6	70	18	7	72
8	nick_8		6	19	3	3	50	14	5	74	17	8	68
9	nick_9		15	10	11	4	73	5	5	50	16	9	64
10	nick_10		14	11	8	6	57	7	4	64	15	10	60

11	nick_11		17	8	12	5	71	8	0	100	20	5	80
12	nick_12		14	11	11	3	79	8	3	73	19	6	76
13	nick_13		12	13	6	6	50	11	2	85	17	8	68
14	nick_14		12	13	7	5	58	11	2	85	18	7	72
15	nick_15		18	7	11	7	61	5	2	71	16	9	64
16	nick_16		20	5	10	10	50	3	2	60	13	12	52
17	nick_17		22	3	15	7	68	2	1	67	17	8	68
18	nick_18		15	8	10	5	67	8	0	100	18	7	72

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 15 Pendeteksian Data Uji yang Bersumber dari Jurnal dengan *Threshold* = 70

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	nick_1	25	16	9	14	2	88	7	2	78	21	4	84
2	nick_2		16	9	10	6	63	7	2	78	17	8	68
3	nick_3		17	8	10	7	59	6	2	75	16	9	64
4	nick_4		17	8	11	6	76	7	1	88	18	7	72
5	nick_5		16	9	11	5	69	7	2	78	18	7	72
6	nick_6		21	4	10	11	48	3	1	75	13	12	52
7	nick_7		5	20	4	1	80	15	5	75	19	6	76
8	nick_8		6	19	4	2	67	15	4	79	19	6	76
9	nick_9		15	10	9	6	60	6	4	60	15	10	60
10	nick_10		14	11	9	5	64	8	3	73	17	8	68

11	nick_11		17	8	7	10	41	8	0	100	15	10	60
12	nick_12		14	11	10	4	71	8	3	73	18	7	72
13	nick_13		12	13	6	6	50	11	2	85	17	8	68
14	nick_14		12	13	7	5	58	11	2	85	18	7	72
15	nick_15		18	7	11	7	61	5	2	71	16	9	64
16	nick_16		20	5	10	10	50	3	2	60	13	12	52
17	nick_17		22	3	12	10	55	2	1	67	14	11	56
18	nick_18		15	8	10	5	67	8	0	100	18	7	72

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 16 Pendeteksian Data Uji yang Bersumber dari Mall Olympic Garden (MOG) dengan *Threshold* = 10

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	_mog_1	**27**	24	3	17	7	71	2	1	67	19	8	70
2	_mog_2		21	6	13	8	62	4	2	67	17	10	63
3	_mog_3		25	2	17	8	68	1	1	50	18	9	67
4	_mog_4		7	20	6	1	86	14	6	70	20	7	74
5	_mog_5		7	20	6	1	86	12	8	60	18	9	67

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 17 Pendeteksian Data Uji yang Bersumber dari Mall Olympic Garden (MOG) dengan *Threshold* = 20

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	_mog_1	27	24	3	16	8	67	2	1	67	18	9	67
2	_mog_2		21	6	12	9	57	5	1	83	17	10	63
3	_mog_3		25	2	17	8	68	1	1	50	18	9	67
4	_mog_4		7	20	5	2	71	19	1	95	24	3	74
5	_mog_5		7	20	4	3	57	17	3	85	21	4	85

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 18 Pendeteksian Data Uji yang Bersumber dari Mall Olympic Garden (MOG) dengan *Threshold* = 30

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	_mog_1	27	24	3	14	10	58	3	0	100	17	10	63
2	_mog_2		21	6	12	9	57	5	1	83	17	10	63
3	_mog_3		25	2	15	10	60	2	0	100	17	10	63
4	_mog_4		7	20	5	2	71	19	1	95	24	3	89
5	_mog_5		7	20	5	2	71	18	2	90	23	4	85

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 19 Pendeteksian Data Uji yang Bersumber dari Mall Olympic Garden (MOG) dengan *Threshold* = 40

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	_mog_1	27	24	3	14	10	58	3	0	100	17	10	63
2	_mog_2		21	6	12	9	57	5	1	83	17	10	63
3	_mog_3		25	2	13	12	52	2	0	100	15	12	56
4	_mog_4		7	20	5	2	71	19	1	95	24	3	89
5	_mog_5		7	20	5	2	71	18	2	90	23	4	85

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 20 Pendeteksian Data Uji yang Bersumber dari Mall Olympic Garden (MOG) dengan *Threshold* = 50

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	_mog_1	27	24	3	14	10	58	3	0	100	17	10	63
2	_mog_2		21	6	11	10	52	6	0	100	17	10	63
3	_mog_3		25	2	13	12	52	2	0	100	15	12	56
4	_mog_4		7	20	4	3	57	19	1	95	23	4	85
5	_mog_5		7	20	4	3	57	19	1	95	23	4	85

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 21 Pendeteksian Data Uji yang Bersumber dari Mall Olympic Garden (MOG) dengan Threshold = 60

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	_mog_1	27	24	3	15	9	63	3	0	100	18	9	67
2	_mog_2		21	6	10	11	48	6	0	100	16	11	59
3	_mog_3		25	2	13	12	52	2	0	100	15	12	56
4	_mog_4		7	20	4	3	57	19	1	95	23	4	85
5	_mog_5		7	20	4	3	57	19	1	95	23	4	85

Keterangan: % adalah prosentase akurasi hasil deteksi

Lampiran 22 Pendeteksian Data Uji yang Bersumber dari Mall Olympic Garden (MOG) dengan Threshold = 70

No	Input (.bmp)	Real			Hasil Deteksi								
		Jmlh. Tempat	Isi	Kosong	Isi			Kosong			Isi dan Kosong		
					T	F	%	T	F	%	T	F	%
1	_mog_1	27	24	3	10	14	42	3	0	100	13	14	48
2	_mog_2		21	6	10	11	48	6	0	100	16	11	59
3	_mog_3		25	2	13	12	52	2	0	100	15	12	56
4	_mog_4		7	20	4	3	57	19	1	95	23	4	85
5	_mog_5		7	20	4	3	57	19	1	95	23	4	85

Keterangan: % adalah prosentase akurasi hasil deteksi