TUGAS AKHIR – TE 141599

RANCANG BANGUN SISTEM PENGENAL KARAKTER BRAILLE DENGAN METODE *LOCAL BINARY PATTERN*

Halum Ghulami
NRP 0711 12 40000 128

Dosen Pembimbing
Dr. Tri Arief Sardjono, ST., MT.
Dr. Ir. Hendra Kusuma, M.Eng., Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

TUGAS AKHIR – TE 141599

RANCANG BANGUN SISTEM PENGENAL KARAKTER BRAILLE DENGAN METODE *LOCAL BINARY PATTERN*

Halum Ghulami
NRP 0711 12 40000 128

Dosen Pembimbing
Dr. Tri Arief Sardjono, ST., MT.
Dr. Ir. Hendra Kusuma, M.Eng., Sc.

DEPARTEMEN TEKNIK ELEKTRO
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2018

FINAL PROJECT – TE 141599

BRAILLE CHARACTER RECOGNITION SYSTEM DESIGN USING LOCAL BINARY PATTERN METHOD

Halum Ghulami
NRP 0711 12 40000 128

Advisor
Dr. Tri Arief Sardjono, ST., MT.
Dr. Ir. Hendra Kusuma, M.Eng., Sc.

DEPARTEMENT OF ELECTRICAL ENGINEERING
Faculty of Electrical Technology
Sepuluh Nopember Institute of Technology
Surabaya 2018

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "Rancang bangun Sistem Pengenal Karakter Braille dengan Metode *Local Binary Pattern*" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak dijinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi peraturan yang berlaku.

Surabaya, 15 Januari 2018

Halum Ghulami
NRP. 07111240000128

RANCANG BANGUN SISTEM PENGENAL KARAKTER BRAILLE DENGAN METODE *LOCAL BINARY PATTERN*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan Untuk Memperoleh
Gelar Sarjana Teknik
Pada
Bidang Studi Elektronika
Departemen Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I,

Dr. Tri Arief Sardjono, ST., MT.
Nip: 197002121995121001

Dosen Pembimbing II,

Dr. Ir. Hendra Kusuma, M.Eng., Sc.
Nip: 196409021989031003

SURABAYA
JANUARI, 2018

Rancang Bangun Sistem Pengenal Karakter Braille Dengan Metode *Local Binary Pattern*

Halum Ghulami
07111240000128

Dosen Pembimbing I : Dr. Tri Arief Sardjono, ST., MT.
Dosen Pembimbing II : Dr. Ir. Hendra Kusuma, M.Eng., Sc.

Abstrak:

Braille adalah media yang digunakan tunanetra untuk kebutuhan media baca tulis. Pola karakter untuk merepresentasikan huruf, angka dan tanda baca lainnya dilambangkan dengan pola titik-titik yang tersusun pada 6 titik posisi untuk setiap karakter yang dinamakan karakter braille. Pada tugas akhir ini dirancang sistem yang dapat mengenali dan menerjemahkan susunan karakter braille pada media kertas dengan metode *local binary pattern*, metode yang digunakan akan diujikan untuk mengetahui seberapa sesuai metode tersebut untuk digunakan pada pengenalan karakter braille. Sistem yang dirancang terdiri dari dua bagian yaitu sistem perangkat keras dan perangkat lunak. Hasil yang didapat melalui metode *local binary pattern* kemudian akan dibandingkan dengan metode lain (evaluasi piksel terprediksi) untuk membandingkan performa setiap metode. Pada sistem perangkat keras digunakan raspberry pi 3 model B, sedangkan pada sistem perangkat lunak digunakan *library* opencv yang memiliki fungsi-fungsi pengolahan data citra digital. Hasil dari tugas akhir ini menunjukkan bahwa dengan menggunakan metode LBP untuk mendapatkan hasil yang baik maka dibutuhkan entri *data learning* yang cukup banyak dan berefek pada tempo kalkulasi yang dibutuhkan menjadi lebih lama, sedangkan hasil dari metode yang tidak menyertakan LBP dan *data learning* menunjukkan hasil yang lebih baik dan tempo kalkulasi yang lebih cepat. Dengan metode LBP 3 *data learning* didapatkan persentase pengenalan karakter benar sebesar 79,855 %, dengan metode LBP 6 *data learning* 82,294 %, sedangkan dengan metode lain sebesar 94,399 %. Hal ini menunjukkan bahwa performa metode lain yang digunakan sebagai pembanding relatif lebih baik dibanding metode LBP.

Kata kunci: *Local Binary Pattern*, Pengenal Karakter Braille.

Halaman ini sengaja dikosongkan

Braille Character Recognition System Design Using Local Binary Pattern Method

Halum Ghulami
07111240000128

Supervisor I : Dr. Tri Arief Sardjono, ST., MT.
Supervisor II : Dr. Ir. Hendra Kusuma, M.Eng., Sc.

Abstract:

Braille is media used by the blind in order to read and write. Characters which are used are different to the common media that is used by normal people, the way it represents a character such as letters, numbers and punctuations is through some bulges occupying on six possible positions, it is called braille character. In this final project a system has been designed and built to has ability to recognizes and translates braille character on a paper-printed media into an ASCII code using local binary pattern method, the method will be tested and compared by another mehod (predicted pixel evaluation) to figures out how convenient the method is to recognizes the character. The system includes two part of subsytems, the firts one is hardware system. It is built to captures some images of the paper-printed media and the second one is software system or the codes, it is coded to processes the images datas, recognizes characters in it and translates them into ASCII codes. After the result has been gotten, it will be compared with another method to see both performaces. The hardware system is equipped with mini computer raspberry pi 3 model B, while the software system uses openCV computer vision library which contains many functions that perform digital image data processing. The result of this final project shows that by using LBP method to get good result, it needs a lot of learning data entries and effects to calculation time needed to be longer, whereas the result from the other method shows better result and faster in calculation time. The quantitative result of right recognition obtained from LBP method with 3 datas learning is 79,855 %, LBP with 6 datas learning is 82,294 %, and another method is 94,399 %. This shows that the another method has better performance than LBP itself.

Keywords: *Local Binary Pattern, Braille Character Recognition*.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Segala puji syukur kepada Allah SWT, atas segala nikmat, berkah dan hidayah-Nya yang tak terkira kepada penulis, hingga penulis mampu menyelesaikan Tugas Akhir dengan judul:

Rancang Bangun Sistem Pengenal Karakter Braille dengan Metode *Local Binary Pattern*

Tujuan utama tugas akhir ini adalah sebagai salah satu persyaratan untuk menyelesaikan jenjang pendidikan pada Bidang Studi Elektronika Teknik Elektro Institut Teknologi Sepuluh Nopember.

Atas selesainya penyusunan tugas akhir ini, penulis ingin mengucapkan terima kasih kepada:

1. Dr. Tri Arief Sardjono, ST., MT. dan Dr. Ir. Hendra Kusuma, M.Eng., Sc. selaku dosen pembimbing Tugas Akhir yang telah memberi bimbingan, penjelasan, nasehat dan kemudahan dalam penyelesaian Tugas Akhir ini.
2. Ir. Tasripan, MT., Ronny Mardiyanto, ST., MT. Ph.D, Dr. Ir. Djoko Purwanto, M.Eng., dan Muhammad Attamimi, B.Eng., M.Eng., Ph.D selaku dosen penguji yang telah mengoreksi Tugas Akhir ini.
3. Dr. Eng. Ardyono Priyadi, ST., M.Eng. selaku ketua Departemen Teknik Elektro ITS.
4. Bapak Hasan Nawawi dan ibu Muhimmah selaku orang tua penulis yang selalu mendukung dan mendoakan penulis.
5. Muhammad Ibrahim, ST., M. Rijal Imadul Bilad, Nurdiansyah, teman-teman akselerasy dan ikarima Surabaya-Malang yang telah memberikan motivasi dan semangat kepada penulis.
6. Fauzan Andi Fadhlullah, Jalaluddin Al-Mursyidi, Fahrezi Alwi Muhammad, ST., Raden Mirzha Chainurfaza Sukmaputra, ST., Zakki Muhammad, ST., Muhammad Fajrul Rahman, ST., Hibban Kaldera dan teman-teman E52 yang tidak dapat disebutkan satu persatu.
7. Serta semua pihak yang telah membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis berharap para pembaca Tugas Akhir ini bersedia memberikan kritik, saran dan masukan yang membangun dan bisa dijadikan referensi bagi Tugas Akhir selanjutnya.

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Huruf braille merupakan sistem aksara yang digunakan oleh tunanetra sebagai alat komunikasi pada media tulis, semua karakter alfanumerik dapat direpresentasikan ke dalam huruf atau karakter braille hanya dengan mengkombinasikan paling banyak eman buah titik pada posisi tertentu, satu karakter pada sistem braille terdiri dari paling banyak eman buah titik yang tersusun pada dua kolom dan tiga baris. Penulisan karakter braille pada media tulis tidak dengan digores atau dicetak seperti media tulis biasa akan tetapi dengan cara menekan media tulis yang digunakan sehingga muncul tonjolan titik-titik penyusun karakter braille yang dapat dirasakan oleh pembaca menggunakan indra peraba.

Sejarah penemuan karakter braille sendiri sudah cukup lama akan tetapi perkembangan teknologi dalam dunia braille saat ini tidak semaju seperti pada sistem aksara biasa yang berakibat pada biaya pengadaan baik media maupun alat tulis dan cetak yang tergolong tinggi. Dalam proses pencetakan naskah braille mesin pencetak memerlukan input berupa kode ASCII yang dapat diterjemahkan komputer sebagai karakter alfanumerik yang selanjutnya oleh komputer diterjemahkan ulang ke dalam karakter braille sehingga output yang dihasilkan berupa naskah yang berisikan tulisan dalam rangkaian huruf braille yang dapat dirasakan menggunakan indera peraba, hal ini akan menjadi masalah jika terdapat naskah lama yang akan dicetak ulang sedangkan *soft-copy* dari pada naskah tidak ditemukan yang mengakibatkan perlunya pembacaan ulang naskah tersebut secara manual dan memasukkan kembali karakter demi karakter ke dalam mesin pencetak dan proses pencetakan ulang dapat dikerjakan. Sehingga adanya alat yang dapat mendigitalisasi naskah-naskah braille tanpa harus membaca secara manual akan sangat membantu kegiatan digitalisasi naskah braille yang telah ada sebelumnya dan mempercepat proses pencetakan ulang naskah braille yang sudah lama. Ide ini lah yang mendasari munculnya penelitian perancangan alat pengenal karakter braille dengan metode *local binary pattern*, selain itu terdapat kebutuhan *braille duplicator* yang dapat mencetak naskah barille dengan inputan naskah braille asli

layaknya mesin *photo-copy* juga menjadi dasar munculnya ide tugas akhir ini.

Metode *local binary pattern* dipilih dalam tugas akhir ini karena dengan metode ini tekstur data citra digital dapat dikeluarkan dengan hasil yang relatif sama walaupun tingkat iluminasi yang didapatkan pada saat pengambilan citra digital mengalami perubahan. Sehingga dari tekstur yang didapatkan tersebut dapat dikenali macam dari karakter braille pada objek.

Alat pengenal karakter braille yang dirancang memiliki dua buah kamera yang dipasang pada bagian atas untuk mendapatkan citra dari naskah braille yang diletakkan dibawah kamera, kedua kamera digerakkan manggunakan motor stepper untuk mangambil eman buah gambar naskah braille pada enam sisi yang berbeda, selanjutnya proses pengolahan citra mulai dilakukan dengan menggabungkan enam buah citra yang telah diambil menjadi satu citra naskah braille utuh, kemudian dilakukan segmentasi karakter braille pada citra dan terakhir mendapatkan nilai *local binary pattern* pada setiap segmentasi citra, dari nilai tersebut mesin pengolah menentukan atau mengenali karakter braille dan menerjemahkan kedalam kode ASCII. Keseluruhan proses pada alat dikendalikan oleh mesin *raspberry pi 3* karena mesin ini sudah memiliki kemampuan untuk mengerjakan semua keseluruhan proses.

Metode yang digunakan untuk mengenali karakter braille pada alat yang dirancang adalah *local binary pattern*, yaitu suatu operasi yang memberikan label pada suatu piksel citra dengan suatu angka desimal yang didapat dari mengevaluasi piksel-piksel disekitarnya, metode ini dipilih karena proses komputasinya yang sederhana dan diharapkan dengan komputasi sederhana ini dihasilkan waktu proses yang relatif singkat dan akurasi yang baik.

Hasil penelitian pengenalan karakter braille pada tugas akhir ini diharapkan dapat menjadi pengembangan varian atau fitur dari alat cetak braille hasil riset tim ITS yaitu Braillits, sehingga diharapkan akan menambah fitur Braillits serta menambah daya saing di pasaran.

1.2 Perumusan Masalah

Pengenalan karakter braille telah dikembangkan dengan menggunakan berbagai metode yang berbeda. Pada penelitian tugas akhir ini dicoba metode *local binary pattern* karena proses penghitungan yang sederhana dan memiliki banyak varian dengan kelebihan yang berbeda-beda sehingga tersedia banyak pilihan varian yang sesuai untuk

penggunaan pengenalan karakter braille. Permasalahan yang harus diselesaikan pada penelitian tugas akhir ini yaitu:

1. Bagaimana menentukan teknik pengambilan gambar naskah braille dengan kamera agar dihasilkan citra karakter braille yang baik.
2. Bagaimana melakukan proses pengolahan citra, agar hasil pengambilan citra dari kamera dapat disempurnakan.
3. Bagaimana menerapkan metode local binary pattern agar dapat mengenali setiap karakter braille dengan benar.
4. Bagaimana mengetahui kelebihan dan kekurangan metode local binary pattern terhadap metode-metode lain yang sudah pernah digunakan peneliti-peneliti lain.

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari penelitian ini adalah:

1. Mengetahui teknik pengambilan citra karakter braille dengan kamera agar didapatkan citra karakter braille yang jelas.
2. Mengetahui proses pengolahan citra, agar citra yang dihasilkan kamera mempunyai kualitas gambar yang lebih jelas dan baik.
3. Mengetahui cara mambaca dan mengartikan citra karakter braille dengan metode *local binary pattern*.

Manfaat dari penelitian ini adalah dapat dihasilkan suatu sistem pengenalan karakter braille dan menerjemahkan menjadi teks bahasa Indonesia. Sehingga dapat memudahkan tunanetra maupun orang awas dalam membaca tulisan huruf braille dan file teks yang dihasilkan dapat dijadikan dokumentasi file master dari buku braille yang belum terdokumentasi secara digital.

1.4 Metodologi Penelitian

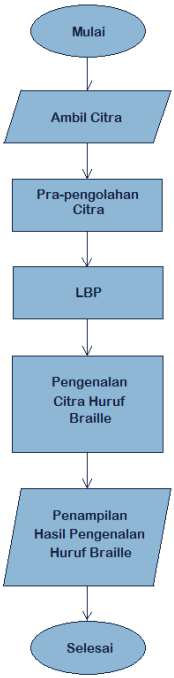
Dalam penyelesaian tugas akhir ini digunakan metodologi sebagai berikut:

1. Studi literatur
   Pada tahap ini dilakukan pengumpulan dasar teori yang menunjang dalam penelitian tugas akhir. Dasar teori ini dapat diambil dari buku, jurnal dan artikel di internet dan forum-forum diskusi yang relevan.
2. Perancangan perangkat keras
   Perancangan perangkat keras terdiri dari perancangan mekanik dan elektronik. Perancangan mekanik dilakukan untuk

perancangan aktuator berupa bagian penggerak kamera seperti motor *stepper*, *timing pulley*, *timing belt*, *linear shaft* dan *camera mounting*. sedangkan perancangan elektronik dilakukan untuk mengendalikan keseluruhan sistem seperti *controller*, *power supply*, *stepper motor driver* dan *switch* untuk inputan sensor posisi kamera. *Controller* yang digunakan adalah mesin komputer berukuran kecil *raspberry pi 3.*

3. Perancangan perangkat lunak

   Perancangan perangkat lunak meliputi perancangan program *preprocessing* citra yang digunakan untuk mengkondisikan citra agar siap digunakan pada tahap pengenalan karakter braille menggunakan metode *local binary pattern*, yang selanjutnya akan dilakukan proses pengenalan karakter dan penampilan hasil pada layar. Secara garis besar perancangan perangkat lunak dapat digambarkan menggunakan diagram alir berikut:



Gambar 1.1 Diagram alir algoritma perangkat lunak.

1. Ambil citra
   Algoritma pengambilan citra dirancang untuk mengendalikan driver motor stepper agar dapat menggerakan motor *stepper* pada tiga posisi yang tepat yaitu bagian atas naskah, bagian tengah naskah dan bagian bawah naskah dan mengendalikan kedua kamera untuk mengambil citra pada enam bagian naskah.
2. Pra-pengolahan citra
   Proses ini dilakukan untuk mengkondisikan citra yang telah diambil melalui kamera agar dapat di proses dan di ketahui *label local binary* pattern piksel yang sesuai. Proses pra-pengolahan citra ini meliputi penggabungan enam buah citra naskah menjadi satu citra naskah utuh dan segmentasi setiap karakter braille pada naskah.
3. LBP (*local binary pattern*)
   Pada proses ini program dirancang untuk mendapatkan nilai-nilai *local binary pattern* piksel-piksel yang terdapat pada area segmentasi karakter barille pada citra.
4. Pengenalan citra huruf braille
   Pengenalan karakter braille ditentukan oleh nilai-nilai *local binary pattern* piksel yang terletak pada setiap segmentasi karakter pada citra.
5. Penampilan hasil pengenalan huruf braille
   Proses terakhir adalah menampilkan hasil dari pengenalan karakter braille pada layar agar hasil dapat dianalisis dan diketahui jika terjadi kesalahan.
4. Pengujian alat
   Pengujian alat dilakukan untuk menentukan akurasi dan keandalan dari dari sistem yang telah dirancang. Pengujian dilakukan untuk melihat apakah program dan algoritma yang digunakan dapat bekerja secara baik. Pengujian dilakukan dengan metode uji terkendali.
5. Analisis
   Analisis dilakukan terhadap hasil pengujian sehingga dapat ditentukan karakteristik dari program dan algoritma yang telah digunakan. Karakteristik yang perlu diuji adalah keakuratan hasil pengenalan karakter dan kecepatan respon program terhadap masukan, apabila hasil pengenalan karakter belum sesuai maka perlu dilakukan perbaikan pada sistem.
6. Penarikan kesimpulan dan saran

Penarikan kesimpulan mengacu pada data pengujian, analisis data dan referensi terkait. Kesimpulan menunjukkan hasil kerja secara garis besar sesuai rumusan masalahyang telah dibuat.

7. Penyusunan laporan
   Proses terakhir adalah membuat dokumentasi pelaksanaan tugas akhir yang meliputi dasar teori, proses perancangan, pembuatan dan pengujian aplikasi.

1.5 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari lima bab dengan sistematika penulisan sebagai berikut:

Bab 1: PENDAHULUAN
Bab ini meliputi latar belakang masalah yang mendasari ide penelitian, perumusan masalah yang berkaitan dengan penelitian tugas akhir ini, tujuan dan manfaat penelitian, metodologi penelitian yang digunakan, sistematika penulisan dan relevansi penelitian.

Bab 2: DASAR TEORI
Bab ini meliputi dasar-dasar teori penunjang yang digunakan dalam pengerjaan tugas akhir ini yang terdiri dari dasar-dasar teori tentang braille, raspberry pi, motor stepper, wiringPi, OpenCV dan dasar teori tentang local binary pattern.

Bab 3: PERANCANGAN SISTEM
Bab ini meliputi penjabaran tentang perancangan sistem yang dibangun seperti perancangan mekanik, perancangan elektrik yang terdiri dari perancangan motor yang digunakan, driver motor, sesor posisi, pencahayaan dan mini komputer yang digunakan yaitu respberry pi, dan yang terakhir perancangan perangkat lunak.

Bab 4: PENGUKURAN DAN ANALISIS SISTEM
Bab ini meliputi pengujian hasil dari rancang bangun dan ide yang telah direalisasikan.

Bab 5: PENUTUP
Bab ini menjelaskan tentang kesimpulan dari hasil penelitian meliputi kekurangan-kekurangan pada kerja alat, sistem, algoritma dan metode

yang digunakan serta dijelaskan pula saran untuk pengembangan penelitian ke depan.

1.6 Relevansi

Hasil dari penelitian tugas akhir ini diharapkan dapat memberikan manfaat sebagai berikut:

1. Dapat digunakan sebagai alat pengenal karakter braille yang memiliki akurasi yang baik.
2. Hasil dari penelitian diharapkan dapat menjadi pengembangan varian atau penambahan fitur dari alat cetak braille hasil riset tim ITS yaitu Braillits.
3. Sebagai dasar penelitian lebih lanjut, agar dapat lebih dikembang.

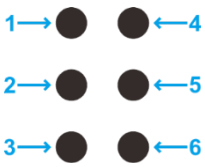
Halaman ini sengaja dikosongkan

BAB II
DASAR TEORI

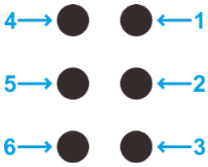
2.1 Braille

Pada tahun 1860 braille telah ditetapkan secara resmi di seluruh Eropa sebagai aksara yang digunakan oleh tunanetra untuk alat berkomunikasi melalui media tulis, awalnya penemu braille yaitu Louis Braille terispirasi dari kode militer Perancis yang digunakan untuk menyampaikan suatu informasi rahasia menggunakan 12 titik timbul yang dapat dibaca melalui indera peraba, kemudian Louis Braille yang juga seorang tunanetra dari umur 3 tahun menyusun kembali kode tersebut menjadi 6 buah titik timbul yang kemudian diterima Eropa sebagai tulisan resmi bagi tunanetra. Braille baru dikenal oleh bangsa Indonesia 41 tahun kemudian tepatnya pada tahun 1901.

Satu karakter braille yang disebut dengan sel braille terdiri dari enam buah titik yang disusun secara matrik dengan susunan 2 kolom dan 3 baris, ke enam posisi titik tersebut diberikan nomor urut 1-2-3, 4-5-6 seperti pada gambar 3.1. Cara baca braille diawali dari kiri ke kanan, sedangkan karena braille merupakan tulisan timbul yang dihasilkan dari proses *embossing* dengan cara ditekan dari sisi belakang kertas maka cara penulisan braille berkebalikan dengan cara pembacaannya yaitu dari kanan ke kiri dan menggunakan posisi titik barille negatif seperti yang ditunjukkan pada gambar 3.2.



Gambar 2.1 Posisi huruf baca.



Gambar 2.2 Posisi huruf tulis.

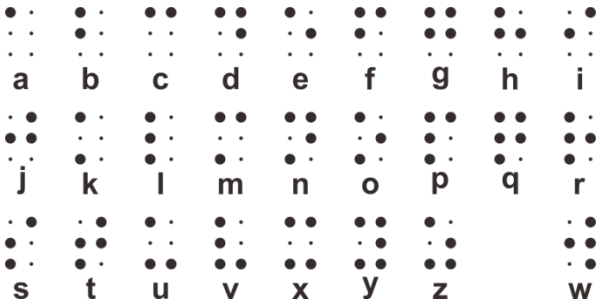
Keenam titik pada setiap sel dapat disusun sedemikian rupa hingga menghasilkan 64 macam kombinasi, sebagian besar karakter alfanumerik dapat di wakilkan dengan sebuah sel braille namun terdapat beberapa karakter yang diwakilkan lebih dari satu sel braille seperti tanda petik tunggal (‘) yang memerlukan dua sel untuk setiap tanda pada petik tunggal buka dan petik tunggal tutup. Layaknya penulisan

rangkaian kata pada sistem penulisan biasa, pada braille juga menggunakan jarak atau spasi untuk memisahkan setiap kata pada kalimat dengan membiarkan kosong satu area sel diantara dua kalimat.

Gambar 2.3 (a)Braille untuk tanda petik tunggal buka. (b) Braille untuk tanda petik tunggal tutup. Keduanya tersusun dari dua sel braille.

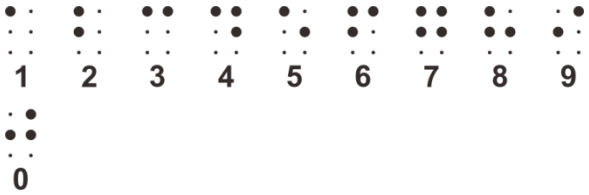
2.1.1 Cara membaca braille

Di Indonesia sistem braille memiliki EYD yang telah disempurnakan pada tahun 2000, ejaan ini mengatur tentang penulisan barille dalam beberapa bidang yaitu: bahasa Indonesia, matematika, fisika, kimia dan musik yang berorientasi pada simbol musik braille internasional. Pada penelitian tugas akhir ini bahasan dibatasi pada bidang bahasa indonesia. Dalam bidang bahasa Indonesia diatur cara penyusunan huruf braille pada kalimat dan penggunaan tanda baca, secara garis besar penyusunan huruf atau karakter braille sama seperti penyususan huruf biasa hanya saja menggunakan simbol huruf yang berbeda, selain itu terdapat sedikit perbedaan pada penggunaan tanda baca antara sistem braille dengan sistem tulisan biasa di antarnya penggunaan tanda angka yang memiliki kombinasi nomor posisi titik 3-4-5-6, tanda angka tidak ada dalam sistem tulisan biasa sedangkan pada sistem braille tanda ini digunakan untuk membedakan karakter apakah karakter tersebut digunakan sebagai angka atau huruf, hal ini disebabkan karena simbol braille digit setiap angka merupakan simbol yang sama seperti beberapa huruf tertentu maka untuk membedakannya digunakan tanda angka yang diletakkan pada awal angka untuk menunjukan simbol yang dimahsud adalah angka bukan huruf. Berukut gambar simbol-simbol braille yang digunakan untuk menuliskan huruf, angka dan tanda baca:



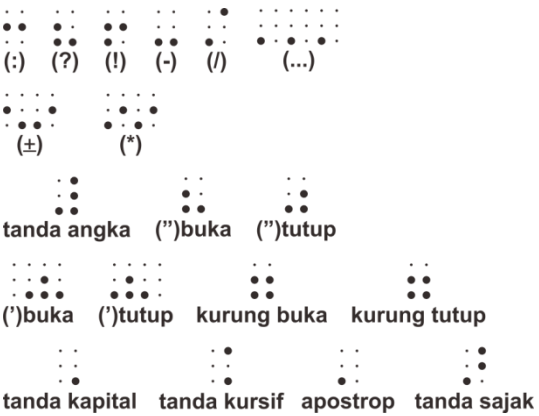
Gambar 2.4 Simbol huruf pada sistem braille.

Penggunaan simbol huruf braille sama seperti penggunaan simbol huruf biasa, jika ingin menuliskan suatu kata maka huruf disusun dari kiri ke kanan sesuai dengan urutan pengejaan huruf penyusun kata tersebut, begitu juga dengan penyusunan kalimat, setiap kata dipisahkan dengan jarak satu karkter atau spasi seperti yang terlihat pada contoh di gambar 3.6.



Gambar 2.5 Simbol angka pada sistem braille.

Untuk simbol angka braille harus diawali dengan tanda angka pada awalnya kemudian baru dituliskan simbol angkanya tanpa ada jarak atau spasi seperti contoh pada gambar 3.6.



Gambar 2.6 Simbol tanda baca pada sistem braille.

Aturan penulisan tanda baca braille sedikit berbeda dengan aturan penulisan tanda baca biasa, aturan penulisan tanda baca braille telah dibakukan dalam Keputusan Menteri Pendidikan Nasional nomor: 052/U/2000, tangal 13 April 2000. Berikut aturan penulisan tanda baca braille sesuai keputusan yang telah disepakati:

1. Tanda titik (.)
   a. Penggunaan tanda titik sesuai dengan EYD.
   b. Tanda titik tidak digunakan dalam menulis bilangan besar untuk memisahkan angka ribuan, jutaan dsb.
   c. Dalam menuliskan tanda waktu, tanda titik tidak dipakai untuk memisahkan angka jam dari angka menit.
   d. Tidak digunakan untuk menuliskan elepsis.
2. Tanda koma (,)
   a. Penggunaan tanda koma sesuai dengan EYD.
   b. Tidak digunakan sebagai tanda desimal.
   c. Digunakan untuk memisahkan angka besar seperti ribuan, jutaan dsb.
3. Tanda titik dua (:)
   a. Penggunaan tanda titik dua sesuai dengan EYD.
   b. Digunakan untuk menuliskan tanda waktu atau memisahkan unit jam dan menit.
4. Tanda tanya (?)
   a. Penggunaan tanda tanya sesuai dengan EYD.

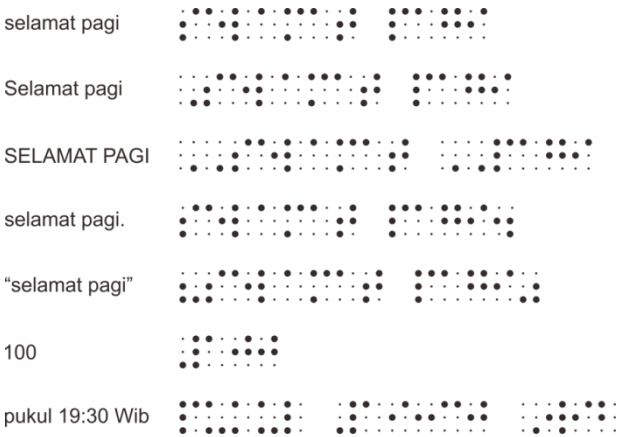
5. Tanda angka (untuk membedakan karakter angka dan huruf)

Tanda angka digunakan untuk membedakan karakter digit angaka terhadap karakter huruf karena karakter digit pada simbol angka braille memiliki simbol yang sama dengan karakter huruf a sampai dengan j, penulisan tanda angka di letakkan di awal kemudian diikuti dengan angka tanpa spasi.

6. Tanda petik (“)
   a. Penggunaan tanda petik sesuai dengan EYD.
7. Tanda petik tunggal (‘)
   a. Penggunaan tanda petik tunggal sesuai dengan EYD.
   b. Digunakan dalam penulisan petikan dalam petikan.
8. Tanda seru (!)
   a. Penggunaan tanda seru sesuai dengan EYD.
9. Tanda kurung ( )
   a. Penggunaan tanda kurung sesuai dengan EYD.
10. Tanda hubung (-)
    a. Penggunaan tanda hubung sesuai dengan EYD.
11. Tanda pisah (--)
    a. Penggunaan tanda pisah sesuai dengan EYD.
12. Tanda garis miring (/)
    a. Penggunaan tanda miring sesuai dengan EYD.
13. Tanda elepsis (...)
    a. Di tengah kalimat menggunakan tiga tanda elepsis.
    b. Penulisan tanda elepsis menggunkan spasi, kecuali jika ditulis dengan tanda baca yang mengikutinya.
    c. Di akhir kalimat menggunakan tiga tanda elepsis dan satu tanda titik.
14. Tanda huruf kapital (untuk penulisan huruf besar)
    a. Ditulis di awal kata tanpa spasi
    b. Untuk satu atau dua kata yang semua huruf ditulis kapital digunakan dua tanda huruf kapital untuk masing-masing kata dan ditulis di depan kata.
    c. Untuk lebih dari dua kata dengan semua huruf besar digunakan tiga tanda huruf kapital di depan kata pertama dan dua tanda kapital sebelum kata terakhir.
    d. Ketentuan c tidak berlaku untuk judul buku, karangan, bab yang semuanya ditulis besar, akan tetapi dengan menggunakan dua tanda huruf kapital yang ditulis di depan masing-masing kata.

15. Tanda kursif (untuk huruf cetak miring)
    a. Ditulis langsung tampa spasi di depan kata.
    b. Tanda kursif digunakan untuk kalimat yang bercetak miring atau tebal atau bergaris bawah.
    c. Satu sampai tiga kata digunakan satu tanda kursif untuk masing-masing kata.
    d. Untuk lebih dari tiga kata digunakan dua tanda kursif di depan kata pertama dan satu tanda kursif sebelum kata terakhir.
16. Tanda kurang lebih (±)
    a. Penulisannya dipisahkan dengan satu satu spasi dari huruf atau tanda baca yang mendahului atau mengikutinya.
    b. Penulisannya tidak dipisahkan dengan spasi dari angka atau singkatan mata uang atau ukuran yang mengikutinya.
17. Tanda bintang (*)
    a. Ditulis langsung tanpa spasi di belakang kata yang diterangkan.
18. Tanda apostrop (‘)
    a. Jika digunakan untuk menyingkat angka tahun ditulis antara tanda angka dan angka tanpa spasi.
19. Tanda sajak

Penulisan puisi tidak ditulis secara berbaris untuk setiap sajaknya, akan tetapi seperti penulisan pada prosa dengan memberikan tanda sajak pada setiap akhir baris, sedang kan untuk perpindahan bait ditandai dengan satu baris kosong.

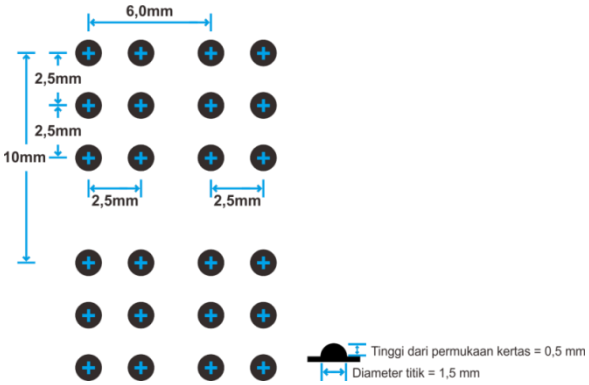


Gambar 2.7 Beberapa contoh penulisan braille.

2.1.2 Dimensi karakter braille

Setiap sel braille pada suatu naskah memiliki dimensi yang konstan, hal ini juga dipengaruhi oleh ukuran diameter titik, jarak antar titik baik secara horizontal maupun vertikal, jarak titik-titik antara sel barille yang satu dengan yang lain, jarak line antar baris dan tinggi tonjolan titik terhadap permukaan kertas. Setiap negara memiliki standar ukuran yang sedikit berbeda, salah satu contoh standar yang dikeluarkan Inggris melalui *United Kingdom Association for Accessible Formats* (UKAAF) yaitu:

1. Jarak antara pusat titik pada satu sel yang sama secara horizontal sebesar 2,50 mm.
2. Jarak antara pusat titik pada satu sel yang sama secara vertikal sebesar 2,50 mm.
3. Jarak pusat dua titik yang memiliki nomor posisi yang sama antar sel yang berdekatan sebesar 6,00 mm.
4. Jarak pusat dua titik yang memiliki nomor posisi yang sama dengan sel pada baris di bawahnya sebesar 10,00 mm.
5. Tinggi tonjolan titik terhadap permukaan kertas sebesar 0,5 mm dengan toleransi 0,1 mm.
6. Diameter pada setiap titik sebesar 1,5 mm dengan toleransi 0,25 mm.



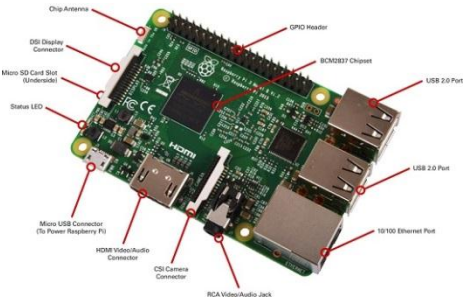
Gambar 2.8 Dimensi braille pada kertas.

2.2 Raspberry Pi 3

Raspberry Pi 3 adalah salah satu *single broad computer* yang memiliki ukuran kecil tetapi memiliki performa yang cukup baik untuk melakukan tugas komputasi layaknya komputer sederhana dan dapat menjalankan beberapa sistem operasi *open source* linux seperti debian, yang paling sering dijalankan oleh mesin ini adalah sistem operasi raspbian yaitu sistem operasi berbasis debian yang dikembangkan untuk mengoperasikan Raspberry Pi. Raspberry Pi 3 sendiri merupakan pengembangan dari versi sebelumnya, pada Raspberry Pi 3 ditanamkan spesifikasi seperti berikut:

- Quard Core 1.2GHz Broadcom BCM2837 64bit CPU.
- 1GB RAM.
- BCM43438 wireless LAN dan Bluetooth Low Energi (BLE) on board.
- 40-pin extended GPIO.
- 4 USB 2 ports.
- 4 Pole stereo output dan composite video port.
- Full size HDMI.
- CSI camera port untuk koneksi Raspberry Pi camera.
- DSI display port untuk koneksi Raspberry Pi touchscreen display.
- Micro SD port untuk tempat sistem operasi dan menyimpan data.
- Upgraded switched Micro USB power source sampai 2.5A.

Dari spesifikasi yang ditanamkan ini Raspberry Pi sangat cocok digunakan untuk aplikasi robotik, *Internet of Thing*, aplikasi lain yang memerlukan proses komputasi *real time* dan *mobile*. Dalam pengoperasiannya Raspberry Pi 3 memerlukan *power supply* 5V 2.5A untuk sumber daya yang diperlukan, Micro SD card direkomendasikan di atas 8GB untuk *loading* sistem operasi dan menyimpan data, *keyboard*, *mouse*, kabel HDMI dan layar yang memiliki HDMI port untuk *user interface*-nya. Untuk *user interface* yang lain terdapat pilihan menggunkan beberapa aplikasi penyambung Raspberry Pi ke PC seperti VNC.

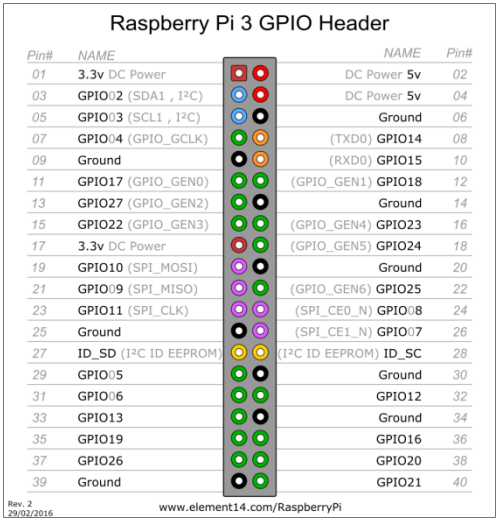


Gambar 2.9 Raspberry Pi 3.

2.2.1 General Purpose Input Output (GPIO)

Raspberry Pi 3 memiliki 40 pin *header* untuk *general porpose input output*, pin-pin ini digunakan untuk *interface* data ke dan dari ic lain, sebagai inputan dari sensor atau sebagai outputan dari respon suatu kondisi tertentu.

Dari 40 pin terdapat dua pin 3,3V DC *power* dengan maksimum *current draw* 50mA untuk masing-masing pin, dua pin 5V DC *power* dengan maksimum *current draw* tergantung dari *power supply* yang digunakan, delapan pin *ground*, 27 pin GPIO yang dapat digunakan sebagai input atau output yang sudah terdapat *internal pull-up* dan *pull-down resistor* sebesar 50k Ohm pada masing-masing pin dan bekerja pada tegangan 3,3V. Selain dapat digunakan sebagai input atau output beberapa pin gpio juga dapat digunakan sebagai *interface* komunikasi data *serial* antar *device* yaitu komunikasi *Inter-Integrated Circuit* (I2C), *Serial Peripheral Interface* (SPI) dan *Universal Asynchronous Receiver/Transmitter* (UART). Sedangkan 2 pin yang lain digunakan untuk ID_SD dan ID_SC yang sangat direkomendasikan untuk tidak menggunakan kedua pin ini sebelum benar-benar mengetahui fungsinya karena dapat menyebabkan kerusakan pada *board*. Ke-40 pin pada Raspberry Pi 3 dapat di lihat pada gambar 3.10.



Gambar 2.10 *Pin out* 40 header Raspberry Pi 3.

Sinyal masukan pada input GPIO dapat berupa sinyal dari sensor atau hanya sekedar sinyal dari *switch on*/*off* (sinyal digital) biasa dilambangkan dengan logika 1 untuk *on* dengan tegangan 3,3V dan logika 0 untuk *off*, untuk sinyal analog diperlukan proses konversi sinyal dari analog ke digital menggunakan ADC sebelum sinyal diterima oleh input pada GPIO, sinyal masukan pada pin GPIO juga dapat diperlakukan sebagai sinyal *interrupt* dengan menuliskan perintah pada program. Sinyal keluaran yang dapat dihasilkan pin GPIO berupa sinyal keluaran digital dengan besaran tegangan 3,3V untuk logika 1 dan 0V untuk logika 0, jika dinginkan keluaran berupa sinyal analog maka diperlukan proses konversi dari sinyal digital ke analog menggunakan DAC. Karena hanya dapat menerima dan menghasilkan 2 kondisi sinyal maka pada pin GPIO diperlukan *initial condition* pada setiap pin untuk menghindari kondisi *float* dimana kondisi tidak terbaca sebagai logika 1 maupun 0.

Pin GPIO yang dapat digunakan sebagai saluran I2C adalah pin nomor 03 untuk SDA (*data*) dan pin nomor 05 untuk SCL (*clock*) dari kedua pin ini Raspberry Pi 3 dapat melakukan komunikasi data

menggunakan protokol I2C dengan beberapa I2C *slave devices* dan Raspberry Pi 3 hanya dapat berfungsi sebagai *master device* pada bus.

Pin GPIO yang dapat digunakan sebagai saluran SPI adalah pin nomor 19 untuk MOSI (*Master Out, Slave In*), pin nomor 21 untuk MISO (*Master In, Slave Out*), pin nomor 23 untuk SCLK (*Serial Clock*), pin nomor 24 untuk CE0 (*Channel Enable* 0) dan pin nomor 26 untuk CE1 (*Channel Enable* 1). Seperti halnya *interface* I2C *interface* SPI juga dapat melakukan komunikasi data ke beberapa SPI *slave device*, dan Raspberry Pi 3 hanya dapat digunakan sebagai *master device* pada bus.

Pin GPIO yang dapat digunakan sebagai saluran UART adalah pin nomor 08 untuk TX (transmit) dan pin nomor 10 untuk RX (receive) karena kedua pin ini merupakan pin GPIO maka keduanya beroperasi pada tegangan 3,3V tidak seperti RS232 yang bekerja pada tegangan 12V.

2.2.2 Raspbian

Raspbian adalah salah satu sistem operasi yang dikembangkan khusus untuk Raspberry Pi dan merupakan pengembangan dari debian. *File image* raspbian dapat diunduh secara gratis melalui *link* https://www.raspberrypi.org/downloads/raspbian/ untuk instalasinya *file image* dituliskan ke *micro SD* dengan menggunakan aplikasi Etcher.

Cara lain untuk instalasi raspbian adalah dengan menggunakan NOOBS yaitu *installer* sistem operasi untuk Raspberry Pi, tidak hanya raspbian akan tetapi terdapat beberapa pilihan sistem operasi lain yang dapat diinstal dan dijalankan. NOOBS dapat diunduh melalui *link* https://www.raspberrypi.org/downloads/noobs/ secara gratis, pada *link* tersebut terdapat dua *file* yaitu NOOBS dan NOOBS LITE, untuk *file* NOOBS sudah terdapat *file image* raspbian yang siap instal tanpa memerlukan koneksi internet saat proses instalasi, akan tetapi jika diinginkan pilihan instalasi sistem operasi lain maka koneksi internet dibutuhkan dalam proses instalasi untuk mengunduh *file* yang dibutuhkan terlebih dahulu. Pada NOOBS LITE proses instalasi baik raspbian atau pilihan sistem operasi yang lain dibutuhkan koneksi internet saat proses instalasi untuk mengunduh terlebih dahulu *file* yang dibutuhkan.

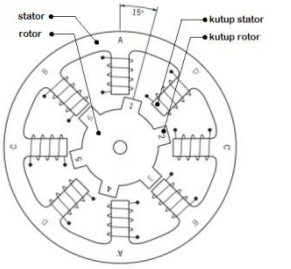
Setelah *file* NOOBS sudah diunduh sambungkan *micro SD card* (direkomenaasikan 8GB ke atas) ke PC, kemudian *micro SD card* diformat menggunakan aplikasi SD Card Formatter yang dapat diunduh di https://www.sdcard.org/downloads/formatter_4/, setelah diformat

maka ekstrak *file* NOOBS ke micro SD card, pasang micro SD card ke Raspberry Pi dan sambungkan *mouse*, *keyboard*, layar (menggunakan kabel HDMI) dan *power supply*, pilih raspbian atau sistem operasi lain yang diinginkan (memerlukan koneksi internet) maka proses instalasi akan berjalan, setalah instalasi selesai maka Raspberry Pi 3 akan *reboot* dan siap digunakan.

2.3 Motor *stepper*

Motor *stepper* adalah jenis motor listrik yang bekerja secara diskrit, putaran motor stepper dilakukan dengan cara pergerakan *step by step* tidak seperti motor listrik lain yang bergerak secara kontinyu. Karena pergerakan yang diskrit inilah motor listrik digunakan untuk mengatur posisi suatu pergerakan.

Cara kerja motor stepper adalah dengan mengubah polaritas kutup pada stator secara beraturan sehingga kutup-kutup pada rotor tertarik oleh gaya magnet yang dihasilkan oleh stator dan menghasilkan putaran *step by step*. Polaritas pada kutub-kutub stator dapat diubah dengan cara mengatur arus yang mengalir pada coil stator. Jumlah phasa yang dimiliki motor stepper ditentukan oleh jumlah kutub pada stator yaitu setengah dari jumlah kutub stator. perbandingan jumlah kutub stator terhadap kutub rotor pada motor stepper beragam, akan tetapi jumalah kutub rotor selalu lebih sedikit dari pada jumlah kutub pada stator.



Gambar 2.11 Diagram motor stepper 4 phasa A,B,C,D dengan 6 kutup rotor, 8 kutup stator dan memiliki sudut per step 15°.

Jumlah step per putaran yang dilakukan motor stepper untuk berputar sebesar 360° disebut dengan resolusi, resolusi yang dimiliki motor stepper beragam mulai dari 4 sampai 400 step per putaran, yang paling umum digunakan adalah motor stepper dengan resolusi 24, 48

dan 200 step per putaran. Resolusi dari motor stepper ditentukan oleh jumlah kutub yang dimiliki oleh stator dan rotor pada motor *stepper*.

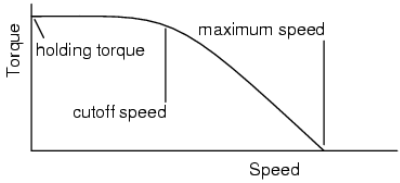
Jika:

Ns = jumlah kutup stator
Nr = jumlah kutup rotor
θs = sudut stator antar kutub yang berdekatan
θr = sudut rotor antar kutub yang berdekatan
θst = sudut per step

maka:

θs = 360°/Ns
θr = 360°/Nr
θst = θr – θs
Resolusi = 360/θst

Torsi yang dihasilkan oleh motor stepper memiliki karakter yang berubah terhadap kecepatan rotasi motor. Semakin rendah kecepatan rotasi maka torsi yang dihasilkan semakin tinggi dan begitu sebaliknya semakin tinggi kecepatan rotasi maka torsi yang di hasilkan semakin rendah. Hubungan antara torsi dan kecepatan motor stepper dapat dilihat pada gambar 3.12.



Gambar 2.12 Hubungan antara torsi (*torque*) dan kecepatan (*speed*) pada motor stepper.

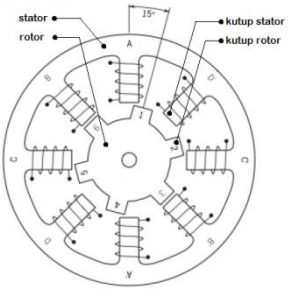
2.3.1 Jenis motor *stepper*

Jenis motor *stepper* dibedakan melalui 2 kategori yaitu konstruksi motor dan rancangan kelistrikan motor. Secara konstruksi motor *stepper* dibagi menjadi 3 jenis, yaitu:

1. Motor *stepper variable reluctance*

Motor *stepper* jenis ini memiliki rotor besi yang bergerigi dan stator yang dapat merubah polarisasi kutubnya untuk menarik gerigi

rotor agar dapat menghasilkan gerakan rotasi. Ini merupakan model yang paling sederhana dari motor *stepper*.

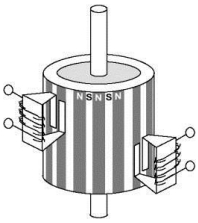


Gambar 2.13 Konstruksi motor *stepper variable reluctace*.

2. Motor *stepper permanent magnet*

Motor *stepper* jenis *permanent magnet* memiliki rotor besi berkutub tidak bergerigi hanya berbentuk silinder dengan permukaan selimut yang rata namun memiliki kutub karena dilapisi magnet permanen yang disusun berselang-seling dengan kutub yang berlawanan. Dengan adanya magnet permanen pada rotor maka intensitas fluks magnet dalam motor meningkat sehingga torsi yang dihasilkan motor *stepper* jenis *permanent magnet* lebih besar dibanding motor *stepper* jenis *variable reluctance*.

Stator pada motor jenis ini disusun secara bertumpuk sehingga coil-coil pada stator dapat diposisikan dengan baik untuk menarik kutub-kutub pada rotor, konstruksi stator yang bertumpuk ini disebut dengan konstruksi *can-stack*. Resolusi yang dimiliki motor stepper jenis *permanent magnet* pada umumnya berkisar antara 7,5° sampai 15° per step atau 48 hingga 24 step setiap rotasi penuh.

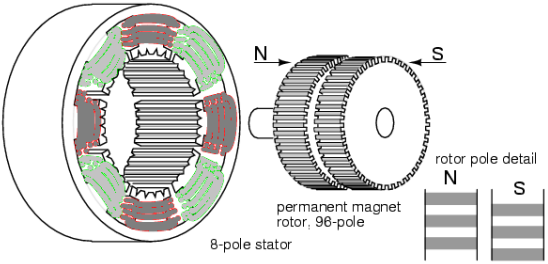


Gambar 2.14 Konstruksi motor *stepper permanent magnet*.

3. Motor *stepper hybrid*

Motor *stepper hybrid* merupakan gabungan dari jenis *variable reluctance* dan *permanet magnet*, rotor pada jenis *hybrid* memiliki gerigi pada permukaannya yang disusun bertumpuk dan setiap tumpukan memiliki polaritas kutub gerigi yang berbeda, pada umumnya kontruksi rotor bertumpuk jenis *hybrid* atau yang disebut konstruksi *can-stack* ini memiliki 2, 4 atau 6 tumpukan rotor tergantung ukuran motor *stepper* dan kekuatan torsi yang diinginkan. Penumpukan rotor disusun seakan akan gerigi kedua rotor tersusun berselang-seling satu sama lain. Stator penggerak pada jenis hybrid juga memiliki gerigi yang memiliki coil untuk mengatur polaritas kutub stator.

Resolusi umum yang dimiliki motor *stepper* jenis *hybrid* berkisar antara 3,6° sampai 0,9° *per step* atau 100-400 *step* setiap rotasi penuh.

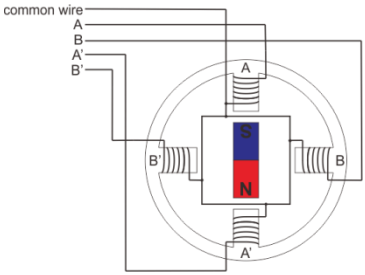


Gambar 2.15 Konstruksi motor *stepper hybrid*.

Secara rancangan kelistrikan motor *stepper* dibagi menjadi 2 jenis, yaitu:

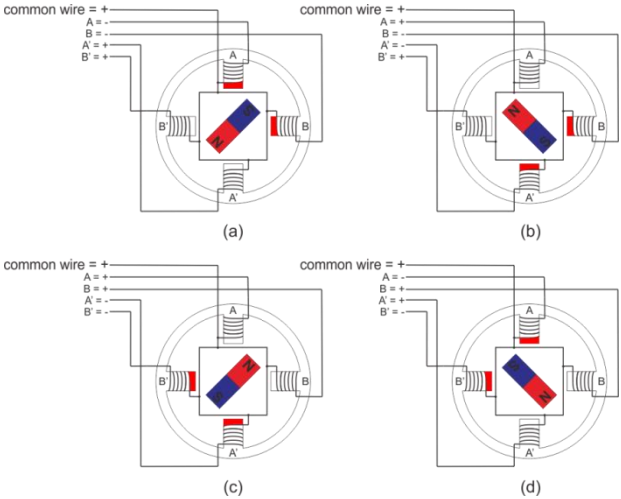
1. *Unipolar*

Pada *unipolar motor* terdapat satu lilitan yang digunakan bersama oleh setiap *phasa*, lilitan ini dinamkan *common wire*. *Common wire* di sambungkan ke tegangan positif begitu juga semua kabel *phasa* pada motor jika motor sedang tidak berotasi, untuk melakukan rotasi hanya cukup mengubah tegangan salah satu kabel pada satu atau dua *phasa* secara berurutan ke 0V atau *ground* sesuai dengan arah rotasi yang diinginkan, perubahan tegangan inilah yang berperan untuk mengubah polaritas magnetik pada salah satu kutub *phasa motor stepper*. karena hanya merubah polaritas magnetik salah satu kutub pada satu *phasa* inilah yang mendasari penamaan *unipolar* pada motor *stepper* jenis ini.

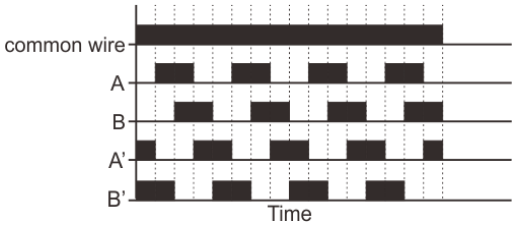


Gambar 2.16 Motor *stepper unipolar* 2 *phasa* A dan B.

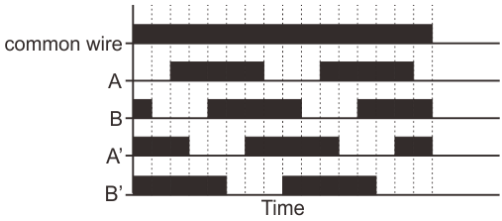
Untuk memutar motor searah dengan jarum jam dilakukan dengan mengubah tegangan pada kabel *phasa* secara berurutan agar polaritas pada *coil* berubah dan dapat menarik rotor *step by step* sehingga terjadi rotasi pada rotor. Pada motor 2 *phasa* perubahan tegangan untuk melakukan putaran penuh searah jarum jam pertama dengan megubah kabel A dan B menjadi 0V, kemudian mengubah A' menjadi 0V dan secara bersamaan kabel A dikembalikan menjadi tegangan positif, selanjutnya tegangan kabel B' diubah menjadi 0V dan secara bersamaan tegangan kabel B dikembalikan menjadi positif, selanjutnya tegangan kabel A diubah menjadi 0V dan kabel A' dikembalikan ke tegangan potif, terahkir tegangan kabel B diubah menjadi 0V dan kabel B' dikembalikan ke positif keadaan ini seperti pada keadaan awal saat motor mulai di putar. Untuk melakukan putaran lebih dari satu kali siklus perubahan tegangan dapat di ulangi lagi. Cara ini disebut dengan *fullstep rotation,* resolusi yang didapatkan sebesar 4 langkah perputaran, untuk menambah resolusi agar posisi yang dihasilkan semakin akurat maka dilakukan dengan cara halfstep rotation. Untuk melakukan satu rotasi penuh pada motor stepper 2 *phasa* dengan cara *halfstep rotation*, setiap *step* hanya merubah 1 buah kabel secara berurutan seperti yang terlihat pada *timing diagram* gambar 3.19, dari halfstep rotation dapat dihasilkan resolusi dua kali dari fullstep rotation yaitu 8 langkah per putaran.



Gambar 2.17 Cara kerja motor *stepper unipolar*.



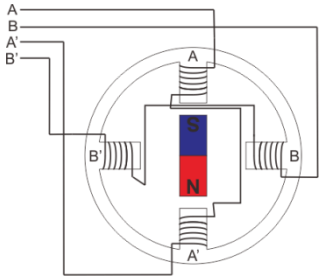
Gambar 2.18 *Timing diagram* dari *unipolar fullstep rotation*.



Gambar 2.19 *Timing diagram* dari *unipolar halfstep rotation*.

2. *Bipolar*

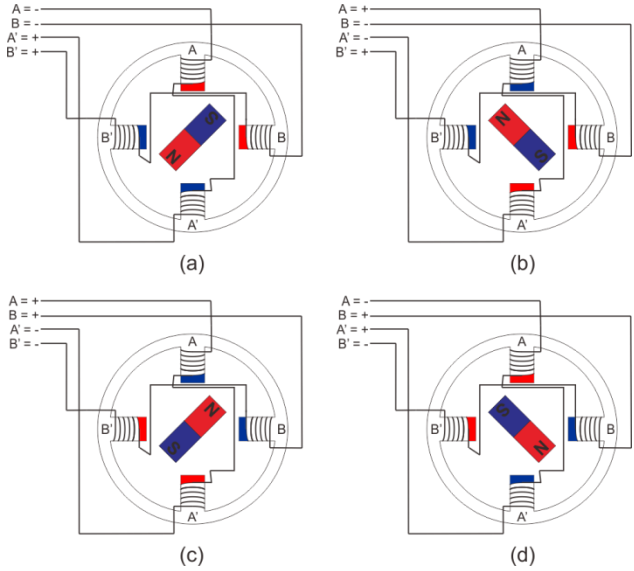
Motor *stepper* jenis *bipolar* memiliki dua buah lilitan pada setiap *phasa*nya dan tidak memiliki *common wire* seperti motor *stepper* jenis *unipolar*, hal ini menyebabkan kedua kutub pada satu *phasa* akan berubah jika tegangan pada pada kabel atau arah arus yang mengalir pada lilitan berubah dan setiap kutub pada satu *phasa* memiliki polaritas magnetik yang berlawanan. Perbedaan polaritas pada semua kutub stator secara beraturan akan menyebabkan rotasi *step by step*. Karena sifat kedua kutub yang selalu berubah dan memiliki polaritas yang berbeda inilah motor stepper jenis ini dinamakan motor *stepper bipolar*.



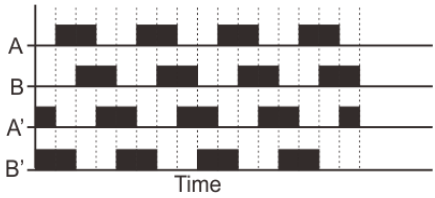
Gambar 2.20 Motor *stepper bipolar* 2 *phasa* A dan B.

Motor *stepper* jenis *bipolar* memiliki torsi yang lebih besar dibanding dengan jenis *unipolar* karena fluks magnet pada motor yang dihasilkan stator lebih besar dari pada yang dihasilkan pada jenis unipolar.

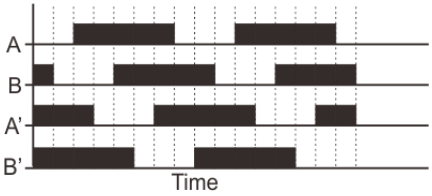
Pada rotasi *fullstep* dan *halfstep* perlakuan pergantian tegangan pada setiap kabel *phasa* sama halnya pada motor jenis *unipolar*, akan tetapi agar dihasilkan torsi yang besar maka kombinasi siklus rotasi harus mengaktifkan semua kutub pada stator, hal ini bisa dicapai hanya jika motor dijalankan dengan metode rotasi *fullstep*, pada metode rotasi *halfstep* torsi yang dihasilkan akan lebih kecil tapi dapat menghasilkan presisi posisi yang lebih baik.



Gambar 2.21 Cara kerja motor *stepper bipolar*.



Gambar 2.22 *Timing diagram* dari *bipolar fullstep rotation*.

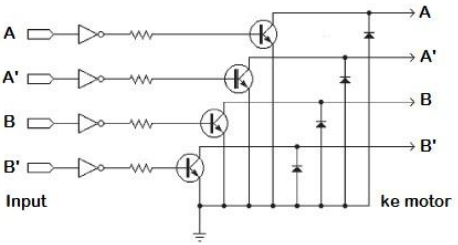


Gambar 2.23 *Timing diagram* dari *bipolar halfstep rotation*.

Selain *fullstep* dan *halfstep rotation* baik *unipolar* maupun *bipolar* dapat dijalankan dengan metode *microstepping* dengan cara mengalirkan arus sinusoidal yang memiliki beda *phasa* 90° pada setiap *phasa*-nya. Dengan metode ini posisi yang dihasilkan akan semakin baik akan tetapi torsi yang dihasilkan semakin kecil.

2.3.2 *Driver* motor *stepper*

Driver atau pengendali yang digunakan pada jenis motor *stepper unipolar* dan *bipolar* memiliki rangkaian yang berbeda hal ini disebabkan oleh susunan lilitan pada stator motor. Pengendali pada motor jenis *unipolar* memiliki rangkaian yang lebih sederhana dibandingkan dengan motor jenis *bipolar*. Rangkaian pengendali yang paling sederhana pada *unipolar* tersusun dari rangkaian transistor yang berperan sebagai saklar pengendali setiap kabel *phasa* yang dapat disambungkan ke *microcontroller* secara langsung atau dengan menambahkan rangkaian logika sebagai translator agar *interface* antara *driver* motor dan *microcontroller* menjadi lebih sederhana dan *user friendly*. *Common wire* pada motor *unipolar* selalu di sambungkan ke tegangan positi yang dibutuhkan motor untuk bergerak.

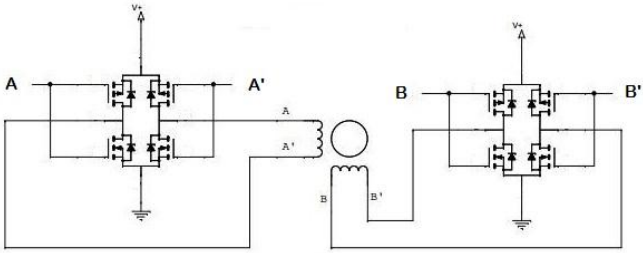


Gambar 2.24 Rangkaian pengendali motor *stepper unipolar* dua *phasa* sederhana.

Karena arus yang dibutuhkan motor relatif besar maka transistor yang digunakan adalah transistor yang dapat menerima arus sesuai dengan kebutuhan motor seperti transistor tipe TIP110, *common wire* pada motor selalu dihubungkan pada tegangan positif sebagai sumber daya yang digunakan motor untuk berotasi.

Rangkaian pengendali motor stepper jenis *bipolar* tersusun dari rangkaian H *bridge* pada setiap *phasa*, rangkaian ini dapat mengalirkan

arus pada lilitan setiap *phasa* dan mengubah polaritas kutub *phasa* dengan cara mengatur arah arus pada lilitan motor. Sama seperti pada pengendali jenis *unipolar*, inputan pengendali *bipolar* dapat disambungkan langsung ke *microcontroller* atau ditambah dengan rangkaian logika tertentu untuk menyederhanakan *interface* ke *microcontoroller* dan lebih *user friendly*.



Gambar 2.25 Rangkaian pengendali motor *stepper bipolar* dua *phasa* sederhana.

Pada rangkaian pengendali *bipolar* di atas digunakan transistor jenis PMOSFET dan NMOSFET untuk mengatur aliran arus pada lilitan motor.

2.4 *WiringPi*

WiringPi adalah *library* yang digunakan untuk mengakses GPIO port pada *Raspberry Pi*. *WiringPi* ditulis menggunakan bahasa C dan dapat digunakan untuk menulis program dalam bahasa C, C++, BASIC dan beberapa bahasa pemrograman lain yang memiliki *wrapper* yang sama. *WiringPi* merupakan *library* tidak berbayar dan bisa diunduh secara gratis. Fungsi-fungsi yang terdapat pada *WiringPi* dapat melakukan tugas untuk melakukan *interfacing* GPIO *Raspberry Pi* dari dan ke *device* lain diantaranya menggunakan GPIO sebagai input atau output, melakukan komunikasi data dengan metode I2C, SPI, UART dan menggunkan salah satu atau dua pin dari 40 pin pada *Raspberry Pi* 3 untuk menghasilkan sinyal PWM.

2.4.1 Cara *install WiringPi*

WiringPi dapat diunduh pada *link* https://git.drogon.net/?p=wiringPi;a=summary secara gratis, file yang berhasil diunduh akan memiliki nama seperti "wiringPi-98bcb20.tar.gz", angka 98bcb20 merupakan

angka yang berbeda pada setiap versi *WiringPi*, pada contoh ini digunakan angka 98bcb20. Setelah berhasil diunduh letakkan *file* pada direktori /home/pi/.

Langkah-langkah untuk instalasi *WiringPi*:

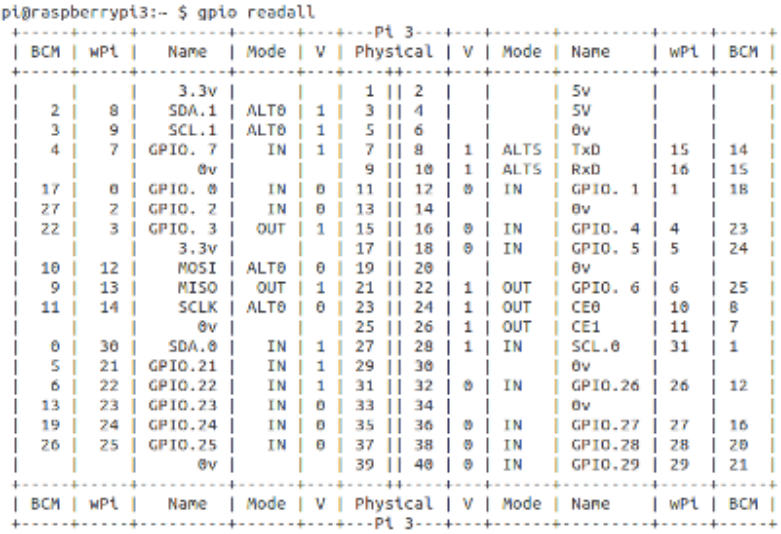
1. Buka terminal pada *Raspberry Pi*.
2. Masuk ke direktori dimana *file WiringPi* diletakkan (/home/pi/) dengan mengetikkan *command* "cd home/pi" tanpa tanda petik.
3. Ekstrak *file* dengan mengetikkan *command* "tar xfz wiringPi-98bcb20.tar.gz" tanpa tanda petik.
4. Kemudian masuk ke *folder* wiringPi-98bcb20 hasil ekstraksi dengan menuliskan *command* "cd wiringPi-98bcb20" tanpa tanda petik.
5. Kemudian ketik "./build" tanpa tanda petik untuk me-*run installer wiringPi*.

Untuk memeriksa apakah *wiringPi* sudah terinstall maka pada terminal tuliskan *command* "gpio –v" tanpa tanda petik, jika keluar keterangan versi gpio dan *copyright* maka *wiringPi* berhasil terinstall, jika tidak keluar apapun pada terminal maka instalasi gagal dan langkah diatas dapat diulang kembali.

2.4.2 Fungsi pada *WiringPi*

Terdapat banyak fungsi yang tersedia pada *library WiringPi*, untuk menggunakan fungsi yang tersedia harus dilakukan inisialisasi terlebih dahulu dengan memilih salah satu setup dari 4 setup yang tersedia yaitu:

- **int wiringPiSetup(void)**: fungsi ini menginisiasi program dan menggunakan penomoran pin *raspberry* dengan skema penomoran pin *wiringPi* seperti pada gambar 3.26.
- **int wiringPiSetupGpio(void)**: fungsi ini menginisiasi program dan menggunakan penomoran pin *raspberry* dengan skema penomoran pin GPIO *Broadcom* seperti pada gambar 3.26.
- **int wiringPiSetupPhys(void)**: fungsi ini menginisiasi program dan menggunakan penomoran pin *raspberry* dengan skema penomoran *physical* pin numbers seperti pada gambar 3.26.
- **int wiringPiSetupSys(void)**: fungsi ini menginisiasi program dan menggunakan penomoran pin *raspberry* dengan skema penomoran GPIO *Broadcom* seperti pada fungsi *setup* wiringPiSetupGpio.



```
pi@raspberrypi3:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALTS | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALTS | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |  OUT | 1 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |  OUT | 1 | 21 || 22 | 1 | OUT  | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

Gambar 2.26 Skema penomoran pada *setup wiringPi*.

Setelah *wiringPi* diinisiasi fungsi-fungsi berikut dapat digunakan pada program:

- **void pinMode(int pin, int mode)**: fungsi ini digunakan untuk menyeting fungsi pin pada *raspberry*, parameter int pin diisikan dengan nomor pin yang ingin digunakan sesuai dengan *setup* yang digunakan dan parameter int mode dapat diisikan dengan salah satu dari mode yang tersedia yaitu: **INPUT** (jika pin difungsikan sebagai *input*), **OUTPUT** (jika pin difungsikan sebagai *output*), **PWM_OUTPUT** (jika pin difungsikan sebagai *output*-an sinyal pwm, hanya pin 12 yang dapat digunakan), atau **GPIO_CLOCK** (jika pin difungsikan sebagai *output*-an sinyal pendetak, hanya pin 7 yang dapat digunakan).
- **void pullUpDnControl(int pin, int pud)**: fungsi ini menyeting pin *raspberry* untuk mengaktifkan *internal pull up resistor* atau *pull down resistor* atau menonaktifkan internal *pull up/down resistor*. Parameter int pi diisikan dengan nomor pin yang diinginkan sedangkan paramaeter int pud diisikan dengan **PUD_OFF** (jika ingin menonaktifkan *internal pull up/down resistor*), **PUD_DOWN** (jika ingin mengaktifkan *internal pull dowm*

resistor ke *ground*) atau **PUD_UP** (jika ingin mengaktifkan *internal pull up resistor* ke 3.3v).

- **void digitalWrite(int pin, int value)**: fungsi ini digunakan untuk menuliskan nilai logika *output* pada pin yang telah diseting sebagai *output*. Parameter int pin diisikan nomor pin yang diinginkan sedangkan parameter int value diisikan **HIGH** (jika ingin menuliskan logika 1) atau **LOW** (jika ingin menuliskan logika 0).
- **void pwmWrite(int pin, int value)**: fungsi yang digunakan untuk menuliskan nilai pwm pada *register* PWM pin yang digunakan sebagai keluaran sinyal pwm, parameter int pin hanya bisa diisikan dengan pin yang dapat digunakan sebagai keluaran sinyal pwm, sedangkan parameter int value diisikan dengan nilai int pwm yang diinginkan dari range 0-1024. Fungsi ini tidak dapat digunakan jika *setup* yang digunkan utnuk inisiasi adalah sys mode.
- **void digitalRead(int pin)**: fungsi ini digunakan untuk membaca logika sinyal dari luar *raspberry* pada pin yang dituliskan di parameter int pin.
- **analogRead(int pin)**: digunakan untuk membaca sinyal analog pada pin *input*, dibutuhkan modul analog tambahan untuk fungsi ini.
- **analogWrite(int pin, int value)**: digunakan untuk menuliskan sinyal analog pada pin *output*, dibutuhkan modul analog tambahan untuk fungsi ini.

Selain fungsi-fungsi di atas terdapat fungsi-fungsi lainnya pada *library wiringPi* yang digunakan untuk *interfacing* dengan *device* lainnya seperti fungsi-fungsi yang digunakan untuk mengakses pin-pin yang digunakan untuk komunikasi serial, spi, I2C dan lainnya. Fungsi-fungsi tersebut dapat di dilihat pada situs wiringpi.com.

2.5 *OpenCV*

OpenCV adalah *library* yang dapat digunakan untuk aplikasi pengindraan visual, *library* ini tidak berbayar. OpenCV dikembangkan dan ditulis menggunakan bahasa C dan C++ dan dapat dijalankan pada sistem operasi Linux, Windows dan OS X. Selain dapat digunakan menggunakan antarmuka C dan C++, OpenCV juga dapat digunakan menggunakan antarmuka lainnya seperti Python, Ruby, Matlab dan beberapa bahasa pemrograman lain walaupun masih dalam tahap pengembangan.

Terdapat lebih dari 500 fungsi yang terdapat pada openCV. Untuk keperluan penelitian tugas akhir ini akan dijelaskan beberapa fungsi yang memiliki relevansi dengan penelitian tugas akhir yang dikerjakan diantaranya fungsi untuk mengambil citra dari *file* atau kamera, konversi citra dari satu jenis ke jenis lain, *thresholding*, morfologi, *image stiching*, deteksi *contour* untuk analisa *blob* dan fungsi-fungsi yang lainnya. Salah satu stuktur citra yang digunakan untuk seluruh operasi yang dilakukan oleh fungsi-fungsi yang dimiliki openCV adalah truktrur citra IplImage, pada stuktur IplImage data citra akan dialokasikan secara otomatis pada memori dan akan me-*return* nilai pada data citra jika suatu *pointer* digunakan untuk mengakses data tersebut. Struktur IplImage suatu citra harus dideklarasikan pada awal program sebelum dapat digunakan untuk menampung data citra, contoh penulisan deklarasi IplImge seperti berikut:

```
IplImage* citra;
```

Di mana citra adalah nama *pointer* yang digunakan untuk menunjuk alamat data citra pada memori. Struktur IplImage pada akhir program harus di-*release* agar memori yang digunakan untuk data citra pada IplImage struktur dapat dialokasikan untuk keperluan lainnya. Untuk mereset struktur IplImage digunakan fungsi:

```
cvReleaseImage(&citra);
```

2.5.1 Akses citra dari *file* dan kamera

Fungsi yang digunakan untuk mengakses citra pada *file* yang telah ada adalah fungsi cvLoadImage() yang memiliki struktur berikut:

```
IplImage* cvLoadImage(
    Const char* filename,
    Int iscolor = CV_LOAD_IMAGE_COLOR
);
```

Terdapat dua *flag* pada fungsi cvLoadImage yaitu *filename* dan *iscolor*, *flag filename* adalah nama *file* citra yang ingin di muat, penulisannya dengan menyertakan ekstensi format citra yang didahului dan diakhiri dengan tanda petik ganda, contohnya seperti (“citra.jpg”). *Flag* yang kedua yaitu *iscolor*, *flag* ini digunakan untuk mengalokasikan

memori untuk jenis citra yang diinginkan, secara bawaan *flag* ini terisi dengan CV_LOAD_IMAGE_COLOR yaitu citra dengan 3 *channe*l (BGR) dan memiliki kedalaman warna 8 bit. Jika digunakan *flag* bawaan pada *iscolor* maka semua jenis citra akan dikonversikan ke dalam format 3 *chanel* dengan kedalaman warna 8 bit. Terdapat pilihan lain pada *flag iscolor* yaitu CV_LOAD_IMAGE_GRAYSCALE, CV_LOAD_IMAGE_ANYCOLOR dan CV_LOAD_IMAGE_UNCHA NGED. Jika digunakan *flag* CV_LOAD_ IMAGE_GRAYSCALE maka memori akan dialokasikan untuk citra dengan format satu *channel* dan kedalaman warna 8 bit, sedangkan jika digunakan *flag* CV_LOAD_IMAGE_ANYCOLOR memori akan dialokasikan sesuai dengan jenis citra yang ingin dimuat dan kedalaman warna 8 bit, dan jika digunakan flag CV_LOAD_IMAGE_UNCHANGED maka memori akan dialokasikan untuk citra dengan channel dan kedalaman warna sesuai dengan format awal citra.

Terdapat *flag* yang bersifat opsional pada fungsi cvLoadImage() yaitu *flag* CV_LOAD_IMAGE_ANYDEPTH, jika *flag* ini digunakan maka kedalaman warna citra yang dimuat pada struktur IplImage akan menyesuaikan dengan kedalaman warna citra awal. *Flag* opsional ini dituliskan setelah *flag iscolor* jika diinginkan.

Kebalikan dari fungsi cvLoadImage adalah fungsi cvSaveImage yaitu fungsi yang digunakan untuk menyimpan citra ke dalam *file* dari struktur citra IplImage. Dengan struktur fungsi sebagai berikut:

```
Int cvSaveImage(
    Cont char* filename,
    Conts CvArr* image
);
```

Di mana *filename* adalah nama citra yang ingin disimpan pada *file* penulisannya seperti pada *flag filename* fungsi cvLoadImage(), dan CvArr* image adalah nama *pointer* struktur IplImage yang ingin disimpan.

Terdapat beberapa fungsi yang dapat digunakan untuk mengakses citra yang berasal dari kamera atau *input*-an citra dari kamera, salah satunya cvCaptureFromCAM(int ID CAM) fungsi tersebut untuk mengaktifkan dan mengambil citra dari kamera, hanya terdapat satu *flag* pada fungsi tersebut yaitu *flag* ID kamera yang digunakan, jika digunakan lebih dari satu kamera maka *flag* ini berfungsi untuk memilih

kamera mana yang aktif. Data yang didapat pada fungsi cvCaptureFrom CAM() harus dilokasikan pada struktur tertentu yaitu struktur cvCapture. Struktur ini tidak hanya digunakan pada fungsi cvCaptureFro mCAM(), akan tetapi juga digunakan untuk mengalokasikan data pada fungsi-fungsi lain seperti fungsi yang digunakan untuk membaca *file* video. Struktur cvCapture harus dideklarasikan pada bagian deklarasi dan di-*release* pada akhir program, penulisan deklarasi dan *release capture* secara berurutan seperti berikut:

```
cvCapture* capture;
cvReleaseCapture(&capture);
```

Di mana *capture* adalah nama *poiter* yang digunakan untuk menunjuk lokasi merori yang digunakan pada strukture cvCapture.

Setelah nilai data dari fungsi cvCaptureFromCAM() dialokasikan pada struktur cvCapture untuk membaca *fram per fram* pada struktur data cvCapture digunakan fungsi cvQueryFrame(capture) yang hanya memiliki satu *flag* yaitu nama *pointer* struktur cvCapture yang ingin dibaca. Berikut contoh program yang digunakan apabila diinginkan mengabil citra dari kamera dan menyimpannya:

```
IplImage* citra;
cvCapture* capture;
capture = cvCaptureFromCAM(0);
citra = cvQuerryFrame(capture)
cvSaveImage(“citra.jpg”,citra);
cvReleaseCapture(&capture);
cvReleaseImage(&citra);
```

Tugas yang dikerjakan contoh program diatas adalah menyiapkan stuktur IplImage dengan nama *pointer* citra, baris kedua menyiapkan struktur cvCapture dengan nama *pointer* capture, baris ke tiga mengaktifkan kamera, mengambil data citra dari kamera dan mengalokasikan pada struktur capture, baris ke empat membaca data citra pada capture kemudian mengirim data tersebut ke struktur IplImage citra, baris ke lima menyimpan data citra pada citra ke dalam *file* dan dua baris selanjutnya bertugas untuk merelease struktur cvCapture dan IplImage secara berurutan.

2.5.2 Konversi citra berwarna ke citra *grayscale*

Citra berwarna adalah citra yang memiliki 3 *channel* yaitu *channel* B (*blue*), G (*green*) dan R (*red*), sedangkan citra *grayscale* adalah citra yang hanya memiliki satu *channel* warna dan memiliki nilai tiap pikselnya dari *range* 0 (putih) – 255 (hitam) untuk 8 bit citra. Secara manual citra *grayscale* didapatkan dari citra berwarna dengan mencari nilai rata-rata setiap piksel dari ketiga *channel*. Contohnya jika terdapat citra berwarna dengan nilai pada piksel (0,0) B = 100, G = 200 dan R = 60, maka nilai pada piksel (0,0) citra *grayscale* jika didapat dari konversi citra berwarna pada contoh adalah (100+200+60) / 3 = 120.

Terdapat beberapa fungsi pada *opencv* yang digunakan untuk mengkonversikan citra berwarna ke citra *grayscale*, diantaranya dengan menggunakan fungsi cvLoadImage() dengan *flag iscolor* diisikan dengan CV_LOAD_IMAGE_GRAYSCALE, dengan fungsi tersebut semua jenis citra akan dikonversikan kedalam citra *grayscale* 8 bit. Selain cvLoadImage() fungsi lain yang digunakan untuk mengkonversi citra berwarna menjadi citra grayscale adalah fungsi cvCvtColor() dengan struktur fungsi:

```
cvCvtColor(
    cons CvArr* src,
    CvArr* dst,
    Int code
);
```

Di mana src adalah pointer Iplimage citra yang ingin dikonversi ke grayscale, dst adalah pointer IplImage citra yang dibuat untuk menyimpan nilai hasil konversi, dan code adalah *code* yang dipilih untuk jenis konversi yang diinginkan, untuk mengkonversi dari citra berwarna ke citra *grayscale* maka *code* yang dituliskan adalah CV_BGR 2GRAY.

Fungsi yang digunakan untuk membuat struktur IplImage baru untuk menampung citra hasil konversi adalah cvCreateImage() yang memiliki struktur fungsi sebagai berikut:

```
cvCreateImage(
    cvSize(x,y),
    int depth,
    int channel
```

);

Di mana cvsize adalah ukuran citra yang diinginkan, depth adalah kedalam warna dan *channel* adalah jumlah *channel* yang diinginkan. Jika digunakan untuk menyimpan hasil conversi dari citra berwarna ke *grayscale* maka depth diisikan menggunakan integer 8 yang melambangkan kedalaman warna 8 bit dan *channel* diisikan dengan integer 1 yang melambangkan 1 *channel*. Berikut contoh program untuk mengkonversi citra berwarna ke *grayscale* menggunkan fungsi cvCvtColor():

```
IplImage* citra;
IplImage* hasil;
Citra = cvLoadImage("citra.jpg",CV_LOAD_IMAGE_COLOR);
Hasil = cvCreateImage(cvGetSize(citra),8,1);
cvCvtColor(citra,hasil,CV_BGR2GRAY);
cvReleaseImage(&citra);
cvReleaseImage(&hasil);
```

Yang dikerjakan contoh program diatas adalah menyiapkan struktur IplImage untuk citra yang ingin dikonversi dan citra hasil konversi, memuat citra yang ingnin dikonversi ke struktur citra, membuat citra baru dengan menggunakan struktur hasil, mengkonversi dan menyimpan hasil pada IplImage hasil, dan me-*release* kedua struktur IplImage.

2.5.3 *Threshold*

Fungsi *threshold* pada *opencv* adalah fungsi yang digunakan untuk mengeliminasi piksel-piksel yang berada pada nilai dibawah atau diatas nilai batas ambang tertentu, fungsi ini sangat berguna untuk membuang piksel-piksel citra yang tidak diingikan atau *background*. Terdapat dua macam fungsi *threshold* pada opencv yaitu cvThreshold() dan cvAdaptiveThreshold(), perbedaan antara keduanya adalah metode penentuan nilai yang menjadi batas ambang atau nilai *threshold*, kedua fungsi *threshold* tersebut hanya dapat diaplikasikan pada citra yang memiliki satu *channel* atau citra *grayscale*.

Fungsi cvThreshold() memberikan nilai ambang batas tertentu dengan memberikan nilai integer pada salah satu *flag* di fungsi cvThreshold() keuntungan dari fungsi ini adalah nilai *threshold* yang

tidak berubah pada seluruh nilai piksel citra dan kelemahannya nilai ambang batas tidak dapat beradaptasi pada tingkat iluminasi citra sehingga proses pembeda antara objek dan *background* pada citra tidak maksimal untuk citra yang memiliki tingkat iluminasi tidak merata. Struktur pada fungsi cvThreshol sebagai berikut:

```
double cvThreshold(
    CvArr* src,
    CvArr* dst,
    double threshold,
    double max_value,
    int threshold_type
);
```

Terdapat 5 flag yang dimiliki fungsi cvThreshold, yaitu *flag* src yang merupakan *flag* berisi nama *pointer* struktur IplImage citra yang akan dilakukan operasi *threshold*, kemudian *flag* dst yang merupakan *flag* yang berisi nama *pointer* struktur IplIlmage citra yang menampung hasil dari operasi *threshold*, kemudian *flag threshold* yang merupakan *flag* berisi nilai ambang batas yang ingin diterapkan pada citra, kemudian *flag* max_value yang merupakan *flag* yang berisi nilai yang diberikan pada citra hasil selain nilai 0 yang tergantung dari tipe *threshold* yang dipilih pada *flag* terakhir, dan yang terakhir adalah *flag* threshold_type yang berisikian nilai integer atau nama tipe *threshold* yang ingin digunakan. Terdapat 5 pilihan tipe *threshold* yang dapat dipilih dan memiliki karakteristik yang berbeda-beda. Jika dst_i dinotasikan untuk hasil pada piksel i, src_i dinotasikan untuk citra pada piksel i, M dinotasikan untuk nilai maksimum dan T dinotasikan untuk nilai ambang batas, maka kelima tipe dan operasi yang dikerjakan sebagai berikut:

Tabel 2.1 tipe threshold dan operasinya

	Tipe threshold	Operasi
0	CV_THRESH_BINARY	$Dst_i = (src_i>T)?M:0$
1	CV_THRESH_BINARY_INV	$Dst_i = (src_i>T)?0:M$
2	CV_THRESH_TRUNC	$Dst_i = (src_i>T)?M:src_i$
3	CV_THRESH_TOZERO_INV	$Dst_i = (src_i>T)?0:src_i$
4	CV_THRESH_TOZERO	$Dst_i = (src_i>T)?src_i:0$

Jika dipilih tipe CV_THRESH_BINARY maka jika nilai piksel citra src_i lebih besar dari nilai ambang batas maka nilai piksel citra dst_i sama dengan nilai maksimum yang telah ditentukan dan jika nilai piksel citra src_i kurang dari nilai ambang batas maka nilai piksel citra dst_i sama dengan 0 atau hitam. Jika dipilih tipe CV_THRESHOLD_BINARY_ INV maka hasil pada citra dst_i berkebalikan dengan tipe CV_ THRESH_BINARY yaitu jika nilai piksel citra src_i lebih besar dari nilai ambang batas maka nilai piksel citra dst_i sama dengan 0 dan jika nilai piksel citra src_i kurang dari nilai ambang batas maka nilai piksel citra dst_i sama dengan nilai maksimum yang telah ditentukan sebelumnya. Jika dipilih tipe CV_THRESH_TRUNC maka jika nilai piksel citra src_i lebih besar dari nilai ambang batas maka nilai piksel citra dst_i sama dengan nilai maksimum yang ditetapkan dan jika nilai piksel citra src_i kurang dari nilai ambang batas maka nilai piksel citra dst_i sama dengan nilai piksel citra src_i atau tidak berubah. Jika dipilih tipe CV_THRESH_ TOZERO_INV maka jika nilai piksel citra src_i lebih besar dari nilai ambang batas maka nilia piksel citra dst_i sama dengan 0 dan jika nilai piksel citra src_i kurang dari nilai ambang batas maka nilai piksel citra dst_i sama dengan nilai piksel citra src_i atau tidak berubah. Jika dipilih tipe CV_THRESH_TOZERO maka jika nilai piksel citra src_i lebih besar dari nilai ambang batas maka nilai piksel citra dst_i sama dengan nilai piksel citra src_i atau tidak berubah dan jika nilai piksel citra src_i kurang dari nilai ambang batas maka nilai piksel citra dst_i sama dengan 0 atau hitam.

Pada fungsi cvAdaptiveThreshold() struktur fungsi yang dimiliki adalah sebagai berikut:

```
void cvAdaptiveThreshold(
    CvArr* src,
    CvArr* dst,
    double max_val,
    int adaptive_method = CV_ADAPTIVE_THRESH_MEAN_C
    int threshold_type = CV_THRESH_BINARY,
    int block_size = 3,
    double param1 = 5
);
```

Fungsi cvAdaptiveThreshold() memiliki 7 *flag* yang mana empat *flag* merupakan *flag* yang sama seperti pada fungsi cvThreshold() dan 3 *flag* yang lain adalah *flag* yang bertanggung jawab atas nilai *threshold*.

Ketiga *flag* tersebut adalah adaptive_method, block_size dan param1. *Flag* adaptive_method adalah *flag* yang berisi pilihan metode yang digunakan untuk mendapatkan nilai *threshold*, tedapat dua pilihan pada *flag* ini yaitu CV_ADAPTIVA_THRESH_MEAN_C yang menjadi pilihan bawaan dan CV_ADAPTIVE_THRESH_GAUSSIAN_C. *Flag* block_size adalah *flag* yang berisikan luas area di sekitar piksel yang ingnin dicari nilai *threshold* atau ambang batas, nilai yang menjadi nilai bawaan pada *flag* block_size adalah nilai 3, dan *flag* param1 adalah *flag* yang berisikan suatu nilai yang digunakan untuk nilai pengurang pada proses pencarian nilai ambang batas, nilai yang menjadi nilai bawaan pada *flag* param1 adalah nilai 5.

Pada metode CV_ADAPTIVE_THRESH_MEAN_C nilai ambang batas didapat dengan cara mencari nilai rata-rata piksel disekitar piksel (x_i,y_i) atau piksel yang sedang dicari nilai ambang batasnya, luas area piksel sekitar ditentukan oleh nilai pada *flag* block_size, kemudian nilai rata-rata tersebut dikurangkan dengan nilai pada *flag* param1, dengan menggunakan metode ini maka nilai ambang batas pada setiap piksel akan berubah-ubah sesuai dengan nilai piksel sekitarnya. Pada metode CV_ADAPTIVE_THRESH_GAUSSIAN_C nilai ambang batas didapatkan dengan cara yang sama seperti pada metode CV_ADAPTIV E_THRESH_MEAN_C hanya saja nilai rata-rata piksel sekitar didapatkan menggunakan fungsi Gaussian sesuai dengan jarak piksel ke titik tengah atau piksel yang dicari ambang batasnya.

2.5.4 Morfologi

Morfologi adalah proses transformasi citra yang memiliki dua operasi dasar yaitu *dilate* dan *erode*.

Operasi *dilate* adalah operasi konvolusi antara suatu citra dengan sebuah *kernel* yang memiliki titik tengah. Operasi *dilate* dilakukan dengan menscan seluruh piksel citra dengan *kernel*, dan menggantikan nilai piksel yang discan oleh *kernel* dengan nilai maksimum yang didapat dari piksel sekitar titik tengah *kernel*, sehingga operasi *dilate* menghasilkan daerah terang lebih luas. Operasi *dilate* dapat diterapkan pada citra untuk menghilangakan atau mereduksi *noise*.

Operasi *erode* adalah operasi kebalikan deri operaso *dilate*, hanya nilai piksel citra yang discan titik tengah *kernel* digantikan dengan nilai minimum piksel disekitar titik tengah *kernel*, sehingga citra yang dihasilkan lebih banyak warna gelap. Jika diterapkan pada citra *binary*

maka operasi *dilate* dan *erode* sangat efektif untuk menghilangkan *noise* yang tidak diinginkan pada citra.

$$dilate(x,y) = [\max(x',y')\, kernel]\, src(x+x', y+y')$$
$$erode(x,y) = [\min(x',y')\, kernel]\, src(x+x', y+y')$$

Struktur fungsi *dilate* dan *erode* sebagai berikut:

```
void cvDilate(
    IplImage* src,
    IplImage* dst,
    IplConvKernel* B = NULL,
    Int iterations = 1
);

void cvErode(
    IplImage* src,
    IplImage* dst,
    IplConvKernel* B = NULL,
    Int iteration = 1
);
```

Baik fungsi cvDelate() maupun cvErode() memiliki 4 *flag*, *flag* pertama adalah *flag* yang berisi nama *pointer* struktur IplImage citra *operand*, flag yang kedua adalah *flag* yang berisi nama *pointer* struktur IplImage citra hasil operasi, *flag* yang ketiga atau *flag* B adalah *flag* yang berisi nama *pointer* struktur *kernel* yang digunakan, jika diisi dengan NULL (0) maka *kernel* yang digunakan adalah *kernel* bawaan yang memiliki luas 3x3 piksel. Dan *flag* yang ke empat adalah *flag* yang berisi nilai integer yang memberikan instruksi berapa kali iterasi yang diingninkan, jika tidak diisi maka iterasi akan diisi dengan nilai bawaan yaitu 1.

2.5.5 *Template matching*

Operasi *template matching* digunakan untuk mencari kecocokan dari dua citra atau lebih, operasi ini berguna untuk membuat citra panoramik. Fungsi yang digunakan untuk operasi *template matching* adalah fungsi cvMatchTemplate() yang memiliki struktur fungsi:

```
void cvMatchTemplate(
    const CvArr* image,
    const CvArr* templ,
    CvArr* result,
    Int method
);
```

Flag yang dimiliki fungsi cvMatchTemplate() ada 4 yaitu image, templ, result dan method. *Flag* image adalah *flag* yang berisi nama *pointer* struktur IplImage citra yang akan dilakukan pencocokan dengan citra yang berada di *pointer* struktur IplImage citra pada *flag* yang kedua yaitu *flag* templ, citra yang menjadi templ harus memiliki ukuran lebih kecil dari pada citra image. *Flag* yang ketiga adalah result yaitu *flag* yang berisi *pointer* struktur IplImage citra hasil pencocokan yang memiliki satu channel dan ukuran (image -> lebar –templ.x + 1, image -> tinggi – templ.y + 1). *Flag* yang terakhir adalah *flag* yang berisi pilihan metode, *flag* ini memiliki 6 pilihan yaitu CV_TM_SQDIFF (*square difference matching methods*), CV_TM_CCORR (*correlation matching methods*), CV_TM_CCOEFF (*correlation coefficient matching methods*), CV_TM_SQDIFF_NORMED (*SQDIFF normalized methods*), CV_TM_CCORR_NORMED (CCORR *normalized methods*) dan CV_TM_CCOEFF_NORMED (CCOEFF *normalized methods*). Hasil dari masing-masing metode dapat dilihat pada tabel 2.2.

Tabel 2.2 Metode *template matching* dan formulasi hasilnya

Metode	Hasil (R=hasil, T=templ, I=image)
CV_TM_ SQDIFF	$R(x,y) = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2$
CV_TM_ CCORR	$R(x,y) = \sum_{x',y'} [T(x',y').I(x+x',y+y')]^2$

CV_TM_ CCOEFF	$R(x,y) = \sum_{x',y'} [T'(x',y').I'(x+x',y+y')]^2$ $T'(x',y') = T(x',y') - \frac{1}{(w.h)\sum_{x',y'} T(x,y)}$ $I'(x+x',y+y') = I(x+x',y+y') - \frac{1}{(w.h)\sum_{x',y'} I(x+x",y+y")}$
CV_TM_ SQDIFF_ NORMED	$R(x,y) = \frac{Rsqdiff(x,y)}{Z(x,y)}$ $Z(x,y) = \sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}$
CV_TM_ CCORR_ NORMED	$R(x,y) = \frac{Rccorr(x,y)}{Z(x,y)}$ $Z(x,y) = \sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}$
CV_TM_ CCOEFF_ NORMED	$R(x,y) = \frac{Rccoeff(x,y)}{Z(x,y)}$ $Z(x,y) = \sqrt{\sum_{x',y'} T(x',y')^2 . \sum_{x',y'} I(x+x',y+y')^2}$

Pada metode CV_TM_SQDIFF piksel yang memiliki hasil kecocokan paling dekat adalah piksel yang memiliki nilai paling kecil pada citra hasil, sehingga nilai 0 merupakan nilai untuk piksel yang memiliki kecocokan sempurna. Pada metode CV_TM_CCORR piksel yang memiliki hasil kecocokan paling dekat adalah piksel yang memiliki nilai paling besar pada citra hasil, sehingga nilai 0 adalah nilai yang dihasilkan oleh piksel yang benar-benar tidak cocok. Pada metode CV_TM_CCOEFF hanya terdapat 3 nilai dari hasil pencocokan yaitu nilai 1, -1 dan 0, nilai 1 merupakan nilai untuk piksel yang memiliki

kecocokan yang sempurna, nilai -1 untuk piksel yang memiliki ketidak cocokan yang sempurna dan nilai 0 merupakan nilai untuk piksel yang sama sekali tidak memiliki kolerasi satu sama lain. Pada ketiga metode yang lain yaitu metode yang merupakan versi normalized dari ketiga metode yang disebutkan sebelumnya nilai yang didapat adalah nilai dari metode yang bukan versi normalised dibagi dengan koofesien Z. Ketiga metode normalized ini memberikan hasil yang lebih akurat dan mereduksi efek dari iluminasi yang tidak merata walaupun membutuhkan tempo komputasi yang lebih lama.

2.5.6 *Contour*

Contour adalah area pada citra yang memiliki nilai piksel yang sama dengan nilai piksel sekitarnya, dalam *opencv* terdapat fungsi yang digunakan untuk menemukan *contour* yang terdapat pada citra, fungsi tersebut adalah cvFindContours() yang akan memberikan nilai integer jumlah *contour* yang ditemukan pada citra, fungsi cvFindContours() memiliki struktur fungsi:

```
int cvFindContours(
    IplImage* img,
    CvMemStorage* storage,
    CvSeq** firstContour,
    int headerSize = sizeof(CvContour),
    CvContourRetrievalMode mode = CV_RETR_LIST,
    CvChainApproxMethod method = CV_CHAIN_APPROX_
                                 SIMPLE
);
```

Fungsi cvFindContours() memiliki 6 *flag* yaitu img, storage, first Contour, headerSize, mode dan method. *Flag* img adalah *flag* yang berisi nama *pointer* struktur IplImage citra yang akan dicari jumlah *contour*-nya, citra tersebut harus memiliki kedalaman warna 8 bit dan satu *channel*. *Flag* yang kedua adalah storage yang merupakan struktur CvMemStorage yang digunakan untuk mengalokasikan rekaman *contour* yang ditemukan pada citra, struktur tersebut harus di alokasikan menggunkan fungsi cvCreateMemStorage(). *Flag* yang ketiga adalah firstContour yang merupakan *flag* yang berisi nama *pointer* struktur CvSeq yang merupakan struktur yang mengalokasikan memori yang digunakan menyimpan *sequence*, struktur CvSeq harus dideklarasikan

terlebih dahulu pada bagian awal program. *Flag* yang ke empat adalah *flag* headerSize yang merupakan *flag* yang berisi informasi ukuran objek yang akan alokasikan, *flag* ini dapat diisi dengan sizeof(CvContour). *Flag* yang ke lima adalah mode yang merupakan *flag* yang berisi pilihan susunan yang digunakan untuk menyusun hasil contour yang ditemukan pada citra, terdapat empat pilihan yaitu CV_RETR-EXTERNAL, CV_RETR_LIST, CV_RETR_CCOMP dan CV_RETR_TREE. *Flag* yang terakhir adalah *flag* method yang berisi pilihan metode yang ingin digunakan, terdapat 5 pilihan metode yang dapat dipilih yaitu CV_CHAIN_CODE, CV_CHAIN_APPROX_NONE, CV_CHAIN_AP PROX_SIMPLE, CV_CHAIN_APPROX_TC89_L1 atau CV_CHAIN_ APPROX_TC89_KCOS dan CV_LINK_RUNS.

2.6 *Local Binnary Pattern*

Pada awalnya metode *Local Binnary Pattern* (LBP) digunakan untuk menganalisa tekstur citra dalam skala yang lebih kecil dari citra yang selanjutnya berkembang menjadi suatu metode yang dapat diterapkan pada banyak aplikasi. Dengan menggunkan metode LBP struktur citra didapatkan dengan cara membandingkan nilai piksel citra dengan beberapa nilai piksel di sekitarnya. Keuntungan yang didapat dari metode ini adalah toleransi terhadap iluminasi monotonik dan komputasi yang sederhana.

Operasi yang dikerjakan oleh LBP adalah operasi yang memberikan label pada suatu piksel citra dengan suatu integer yang disebut dengan *local binnary pattern code*, yang mana label tersebut adalah hasil dari *encode* struktur lokal piksel-piksel yang berada di sekitar piksel citra tersebut. Operasi paling sederhana pada LBP dilakukan dengan cara setiap piksel citra dikomparasikan dengan delapan piksel tetangga disekitarnya dalam jangkauan 3x3 piksel dengan mengurangkan setiap nilai piksel tetangga dengan nilai piksel yang berada di tengah (nilai piksel tengah menjadi nilai *threshold* untuk piksel lokal disekitarnya), piksel yang menunjukkan nilai negatif dikodekan dengan nilai 0 sedangkan piksel yang menunjukkan nilai positif dikodekan dengan nilai 1. Selanjutnya akan didapat 8 buah angka biner dengan cara menyatukan hasil pengkodean pada 8 piksel tetangga mengikuti arah jarum jam dimulai dari piksel kiri atas da nilai desimal dari delapan digit biner ini yang digunakan untuk label LBP pada piksel tengah dari 3x3 piksel citra.

Pada pengembangan metode LBP piksel sekitar dapat lebih dari delapan dan dapat memiliki radius yang bervareasi sesuai dengan kebutuhan dan hasil yang diinginkan, secara umum nilai LBP piksel (Xc,Yc) pada citra dapat dirumuskan sebagai berikut:

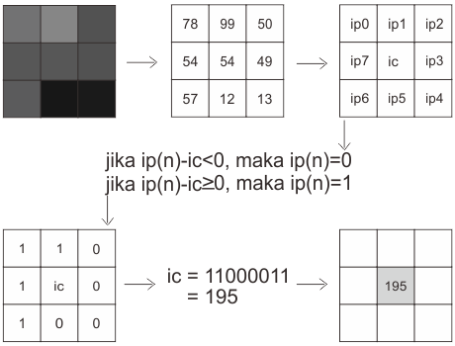
$$LBPp,r\ (Xc,Yc) = \sum_{p=0}^{p-1} s(ip - ic)2^p$$

Dimana,

p = jumlah piksel tetangga
r = jarak radius piksel tetangga
Xc = koordinat (x) piksel tengah
Yc = koordinat (y) piksel tengah
ip = nilai *greyscale* piksel tetangga
ic = nilai *greyscale* piksel tengah, dan fungsi s(x) didefinisikan sebagai:

$$s(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & if\ x\ < 0 \end{cases}$$

Yang menunjukkan bahwa jika hasil pengurangan bernilai negatif maka nilai piksel tetangga dikodekan dengan nilai 0 dan jika hasil pengurangan bernilai 0 atau positif maka nilai piksel tetangga dikodekan dengan nilai 1.



Gambar 2.27 Operasi *Local Binnary Pattern* sederhana.

Limitasi yang dimiliki operasi LBP sederhana adalah ukuran 3x3 piksel tetangga tidak dapat menghasilkan sifat dominan pada struktur berskala besar. Untuk itu pengaturan ukuran radius tetangga dapat diatur sehingga dapat dihasilkan sifat dominan pada struktur berskala besar. Saat ini operasi LBP telah berkembang dan menghasilkan variasi operasi yang lebih *advanced* diantaranya *Improved* LBP (*Mean* LBP), *Hamming* LBP, *Extended* LBP, *Completed* LBP, *Local Temary Pattern*, *Soft* LBP, *Elongated* LBP, *Multi-Block* LBP, *Three/Four Patch* LBP, 3D LBP, *Volume* LBP (LBP-TOP), LBP and *Gabor Wavelet*, LBP and SHIFT, dan LBP *Histogram Fourier* yang masing-masing memiliki karakteristik dan fungsi yang berbeda satu sama lain.

Halaman ini sengaja dikosongkan

BAB III
PERANCANGAN SISTEM

Secara umum terdapat dua bagian utama dalam perancangan sistem pengenal karakter braille yaitu sistem perangkat keras dan sistem perangkat lunak. Pada sistem perangkat keras dibagi menjadi perancangan sistem mekanik dan elektronik, sistem mekanik terdiri dari desain fisik, perakitan komponen-komponen penggerak seperti *pulley*, papan kamera, *linear shaft* dan perangkat gerak lainnya. Sedangkan sistem elektronik terdiri dari sistem rangkaian dan kendali diantaranya motor, *driver* motor, sesor posisi, pencahayaan, dan mesin utama yang digunakan sebagai pusat kendali yaitu komputer berukuran kecil raspberry pi 3.

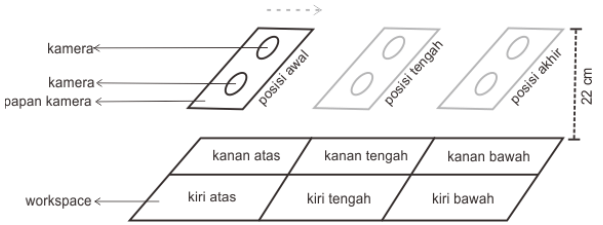
Pada sistem perangkat lunak dirancang program yang menjalankan beberapa tugas pengolahan citra seperti pengambilan gambar naskah, *thresholding, morphology, image stiching* untuk preprocessing citra, segmentasi, penerapan metode *local binnary pattern*, dan proses pengenalan karakter. Keseluruhan sistem ini baik perangkat keras maupun lunak dibangun untuk melaksanakan tugas utama yaitu mengenali karakter braille pada sebuah naskah braille.

3.1 Perancangan sistem mekanik

Disain kerangka alat dirancang untuk dapat mengambil citra objek (naskah braille berukuran 30,5 cm x 25,5 cm) dengan hasil yang baik, beberapa faktor yang mempengaruhi hasil pengambilan citra adalah jarak dan posisi kamera terhadap objek, teknik pengambilan citra, dan teknik pencahayaan.

Pengambilan citra objek dilakukan sebanyak 6 kali pada sisi-sisi objek yang berbeda menggunkan 2 buah kamera. keenam sisi objek tersebut adalah sisi kiri atas, sisi kanan atas, sisi kiri tengah, sisi kanan tengah, sisi kiri bawah dan sisi kanan bawah. Untuk mendapatkan 6 sisi citra yang diinginkan maka kedua kamera dipasang pada sebuah papan secara sejajar di titik tengah setengah masing-masing bagian papan menghadap ke objek yang diletakkan pada bagian bawah (*workspace*). Jarak antara kamera dan objek sekitar 22 cm. Teknik pengambilan gambar dilakukan dengan menggerakkan papan kamera dari posisi awal (bagian atas objek) kemudian ke posisi tengah (bagian tengah objek)

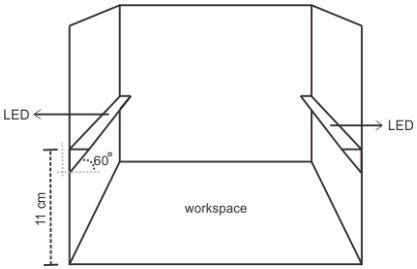
kemudian ke bagian akhir (bagian bawah objek) dan terakhir papan kamera digerakkan ke posisi awal.



Gambar 3.1 Jarak dan posisi kamera terhadap objek.

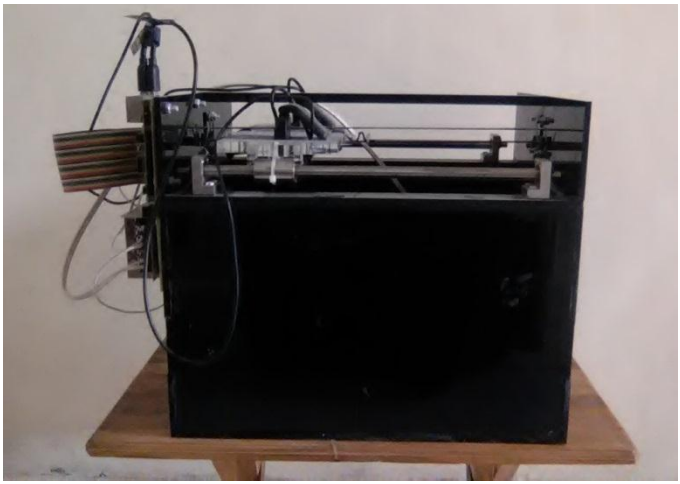
Dua sisi samping papan kamera dipasangkan pada *bushing linear shaft* yang terletak pada dua sisi samping atas kerangkan alat, sedangkan kedua sisi papan kamera yang lain diikat menggunakan *timing belt* berukuran GT2 yang tersambung ke *pulley* pada motor yang terletak pada sisi panel elektrik dan *idler pulley* yang terpasang pada *vertical bracket* di sisi berlawanan. Hal ini memungkinkan motor untuk menggerakkan papan kamera pada posisi yang diinginkan, untuk mengetahui posisi awal dan akhir papan kamera diletakkan sensor posisi berupa dua buah *switch* yang diletakkan pada sisi panel elektrik dan sisi *idler pulley* yang akan memeberikan sinyal pada pusat kendali jika papan kamera berada pada salah satu dari kedua posisi tersebut, sedangkan untuk menentukan posisi tengah dilakukan *tunning manual* untuk mendapatkan berapa kali putaran motor untuk mencapai posisi tengah yang diinginkan.

Pada saat pengambilan citra diperlukan pencahayaan yang tepat agar tekstur tonjolan karakter braille pada objek dapat teridentifikasi dengan baik, hal ini akan memudahkan praproses pada pengolahan citra nantinya. Pencahayaan dilakukan menggunakan led yang dipasangkan pada sisi kanan dan kiri dalam alat yang mengarah ke objek dengan sudut sekitar 60° terhadap permukaan objek dan jarak 11 cm di atas *workspace*. Pada saat kamera mengambil citra sisi kiri objek maka led pada sisi kanan akan menyala dan sebaliknya jika kamera mengambil citra pada sisi kanan objek maka led pada sisi kiri akan menyala. Hal ini memungkinkan tekstur tonjolan pada objek dapat teridentifikasi dengan baik karena bayangan yang dihasilkan dari sorotan cahaya led.



Gambar 3.2 Posisi dan jarak LED terhadap objek.

Panel elektrik memiliki dimensi sebesar 16,5 cm x 11 cm yang terletak pada salah satu sisi yang sama dengan letak motor. Keseluruhan dimensi kerangka alat adalah 34,5 cm x 29 cm x 29,5 cm (panjang, lebar, tinggi) dan berbentuk kotak persegi panjang.

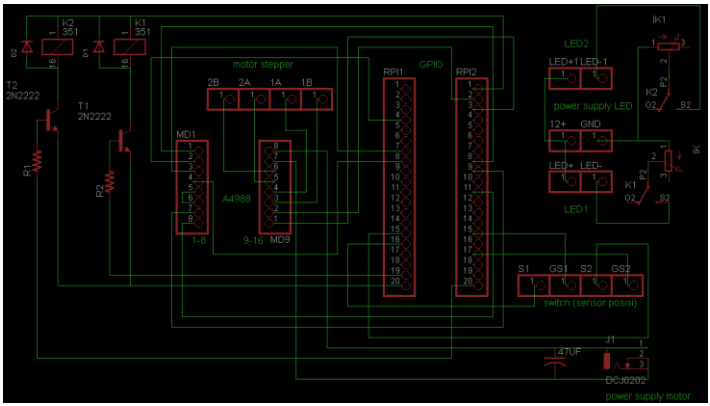


Gambar 3.3 Dimensi dan bagian kerangka alat.

3.2 Perancangan sistem elektronik

Sistem elektronik dirancang sebagai pusat kendali keseluruhan sistem seperti sistem penggerak kamera, pembaca posisi kamera,

pencahayaan, pengambilan citra dan sebagai pusat pemrosesan citra. Raspberrry Pi 3 digunakan sebagai inti dari pusat kendali dengan mempergunakan pin GPIO untuk *interfacing* dengan rangkaian tambahan pada panel elektrik.



Gambar 3.4 Diagram skematik rangkaian pada sistem elektronik.

3.2.1 Motor

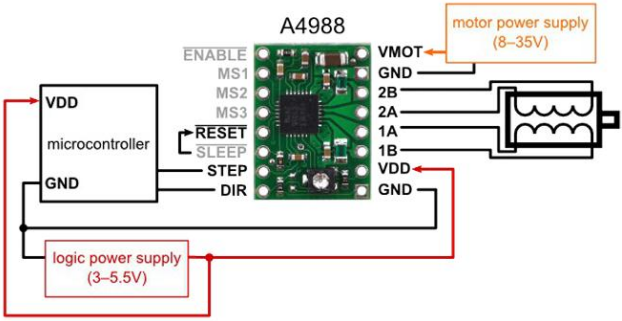
Motor digunakan untuk menggerakkan papan kamera pada posisi yang diinginkan, motor yang digunakan adalah motor jenis *bipolar stepper* karena dapat mengatur sudut rotasi putar dan memiliki torsi yang relatif besar oleh karena itu motor jenis ini sesuai untuk menggerakkan papan kamera ke posisi yang diinginkan. Ukuran motor yang digunakan adalah nema17, memiliki 2 phasa 4 kabel dan *step angle* 1,8°. Torsi yang dihasilkan sebesar 500mN.m *untuk holding torque* dan 15mN.m untuk *detend torque* yang mana sudah cukup untuk menggerakkan papan kamera. Motor ini bekerja pada range tegangan 3,75V - 6V dan arus 1,5A.

Motor *stepper* jenis *bipolar* ini digerakkan dengan cara mengubah polarisasi kutub pada setiap *phasa* secara teratur dan bergantian, langkah atau *step* rotasi motor dapat diatur menjadi *full step*, *half step* atau *micro step* sesuai dengan perubahan polarisasi kutub tiap *phasa*-nya. Rangkaian untuk men-*drive* motor *stepper bipolar* 2 *phasa* adalah 2 buah rangkaian *full* H *bridge* yang dapat dengan mudah mengatur arah aliran arus dengan mengatur tegangan pada setiap transistor. *Driver*

yang digunakan untuk men-*drive* motor *stepper* pada rangkaian adalah modul *driver* motor *stepper bipolar* 2 *phasa* dengan type A4988, inputan tegangan dan arus yang dapat dioperasikan pada *driver* ini sesuai dengan kebutuhkan motor yang digunakan, cara menyambungkan motor dengan modul *driver* hanya dengan memasangkan 4 buah kabel pada motor pada 4 pin output driver A4988 sesuai phasanya. Pada motor stepper yang digunakan memiliki warna yang berbeda pada setiap kabelnya yaitu biru, merah, kuning dan putih. Kabel biru dan merah merupakan 2 kabel yang dimiliki oleh satu phasa sedangkan kabel kuning dan putih adalah 2 kabel yang dimiliki oleh satu *phasa* yang lain. Jika diberikan nama pada setiap kabel maka kabel biru adalah kabel 2B, kabel merah adalah kabel 2A, kabel kuning adalah kabel 1A dan kabel putih adalah kabel 1B. Keempat kabel tersebut di pasangkan ke 4 pin A4988 dengan nama pin yang sama dengan nama yang diberikan pada masing-masing kabel.

3.2.2 *Driver* motor

Driver motor *stepper* Polulu A4988 memiliki diagram minimal *wiring* untuk tersambung ke mikrokontroller dan motor *stepper bipolar* seperti pada gambar 3.5.



Gambar 3.5 Diagram *minimal wiring* A4988.

Polulu A4988 memiliki 16 pin, pin VMOT dan GND disambungkan ke *power supply* motor dengan *range* tegangan 8V - 35V pada rangkaian digunakan *power supply* dengan tegangan 12V dan arus maksimal 1,5A. Pin 2B, 2A, 1A dan 1B disambungkan langsung ke 4 kabel motor sesuai dengan penamaan kabel masing-masing pada subbab

motor. Pin VDD dan GND disambungkan ke pin catu daya pada raspberry pi, VDD ke pin 4 GPIO raspberry yang memiliki tegangan posistif 5V sedangkan GND disambungkan ke pin 6 GPIO raspberry yang juga sebagai GND GPIO raspberry. Pin DIR berfungsi untuk mengatur arah rotasi motor logika yang dapat diberikan adalah 0 (GND) dan 1 (3,3V) pin ini disambungkan langsung ke pin 22 GPIO raspberry yang disetting sebagai outputan untuk mengatur arah rotasi motor. Pin STEP akan menerima pulsa dari raspberry untuk memutar motor, satu pulsa akan memberikan satu *step* rotasi pada motor, pin STEP disambungkan pada pin 18 GPIO raspberry yang telah disetting sebagai output untuk memberikan pulsa pada *driver* motor. Pin RESET dan SLEEP disambungkan satu sama lain sedangkan pin ENABLE disambungkan ke pin 16 GPIO untuk memberikan sinyal untuk mengaktifkan atau menonaktifkan *driver*.

Terdapat 3 pin yang berfungsi sebagai pengatur langkah rotasi motor yaitu pin MS1, MS2 dan MS3. Dari ketiga pin ini langkah rotasi motor dapat diatur menjadi *full step, half step, quarter step, eighth step* dan *sixteenth step,* pengaturan langkah rotasi ini dilakukan *driver* motor dengan cara mengatur besaran arus pada setiap *coil*. Jika disetting langkah rotasi *full step* pada motor *stepper* yang digunakan maka untuk sekali rotasi resolusi setiap putaran sebesar 200 langkah karena motor memili sudut 1,8° setiap langkahnya, jika disetting *half step* maka resolusi menjadi dua kali lipat dari *full step* yaitu 400 langkah setiap rotasi dan begitu seterusnya. Untuk menyeting langkah rotasi ketiga pin di berikan logika yang sesuai pada tabel 3.1.

Tabel 3.1 Logika pengaturan langkah rotasi A4988

MS1	**MS2**	**MS3**	**Langka Rotasi**
Low	*Low*	*Low*	*Full step*
High	*Low*	*Low*	*Half step*
Low	*High*	*Low*	*Quarter step*
High	*High*	*Low*	*Eighth step*
High	*High*	*High*	*Sixteenth step*

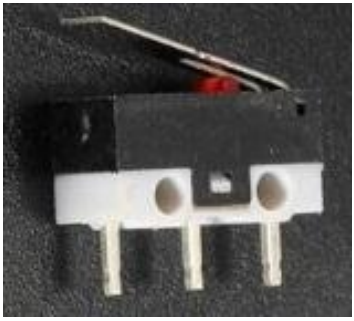
ketiga pin tersebut memiliki *internal pull down* resistor sehingga logika *Low* bisa di berikan hanya dengan membiarkan pin tidak tersambung ke manapun. Pada rangkaian, pin MS1, MS2 dan MS3 di disambungkan pada pin 7, 13 dan 15 GPIO raspberry secara berurutan, rotasi motor disetting untuk memiliki resolusi 400 langkah setiap

putaran atau *half step*. maka pin 7, 13 dan 15 GPIO disetting menjadi output dengan logika High Low Low secara berurutan.

Arus yang mengalir pada *coil* dapat diatur dengan cara memutar *timmer* yang terdapat pada modul A4899, untuk mengukur arus yang diinginkan perlu didapatkan tegangan vref yaitu tegangan pada kepala *timmer* terhadap *ground*, selanjutnya limitasi arus didapat dengan mengalikan vref dengan 2,5. Pengaturan arus yang tepat akan memaksimalkan operasi *driver* dan motor.

3.2.3 Sensor posisi

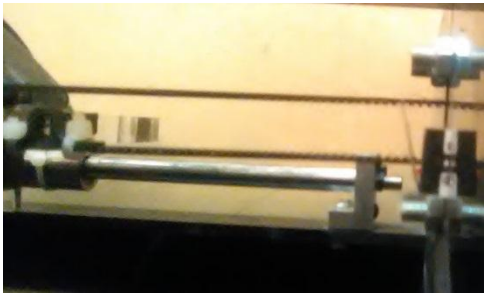
Untuk mengetahui posisi papan kamera digunakan dua buah sensor posisi berupa SPDT *micro switch* pada dua sisi dalam kerangka yang akan memberikan sinyal ke raspberry jika switch tersentuh oleh papan kamera. SPDT *micro switch* memilihi 3 kaki, satu kaki berfungsi untuk kaki *common*, satu kaki untuk kaki *normally open* (NO) dan kaki yang yang lain berfungsi untuk kaki *normally closed* (NC).



Gambar 3.6 SPDT micro switch.

Kedua kaki *common micro switch* disambungkan pada *ground* GPIO raspberry yaitu pin 30 dan 34, kaki NC kedua *switch* tidak disambungkan ke pin lain atau dibiarkan saja, sedangkan kaki NO kedua *switch* disambungkan ke pin GPIO yang telah di *setting* sebagai *input* yang disambungkan ke *internal pull up resistor* agar memiliki *state* awal positif. Pin GPIO yang digunakan adalah pin 31 untuk sensor posisi awal dan pin 29 untuk sensor posisi akhir. Posisi awal papan kamera akan diketahui jika pin 31 berlogika 0 begitu juga posisi akhir papan

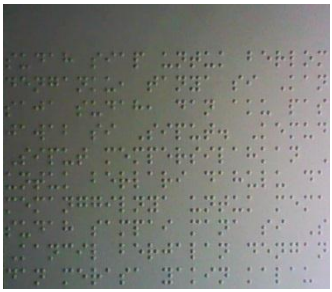
kamera akan diketahui jika pin 29 berlogika 0, hal ini dikarenakan sensor posisi dirancang sebagai sensor *active low*.



Gambar 3.7 Sensor posisi pada alat pengenal karakter braille.

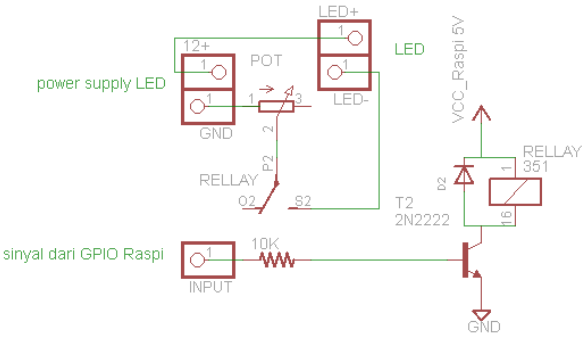
3.2.4 Pencahayaan

Sumber pencahayaan saat pengambilan citra adalah lampu LED dengan tegangan operasi 12V yang dipasang pada dua sisi dalam kanan dan kiri rancangan, hal ini memungkinkan pada saat pengambilan citra sisi kiri naskah sumber pencahayaan didapatkan dari sisi kanan sehingga tekstur bayangan tonjolan naskah dapat ditangkap dengan baik oleh kamera, begitu juga sebaliknya saat pengambilan citra sisi kanan naskah sumber pencahayaan berasal dari sisi kiri untuk mendapatkan tekstur yang baik.



Gambar 3.8 Hasil pengambilan citra sisi kiri menggunakan sumber pencahayaan dari sisi kanan.

Pemilihan sumber pencahayaan pada saat pangambilan citra sisi tertentu dilakukan menggunakan sinyal yang dikeluarkan oleh pin *output* GPIO raspberry yang ditransmisikan ke rangkaian *relay* sebagai *switch* yang dimiliki oleh setiap LED. Rangkaian *rellay* ini berfungsi sebagai *driver* LED.



Gambar 3.9 Diagram skematik rangkaian LED *driver*.

Potensiometer pada rangkaian *driver* LED digunakan untuk mengatur intensitas cahaya yang dihasilkan LED, *power supply driver* disambungkan pada sumber daya dengan tegangan 12V dan arus maksimal 3A. Kaki *rellay* yang digunakan adalah kaki *normaly open* (NO) dan *common*, sedangkan kaki *normaly closed* (NC) dibiarkan tidak tersambung ke *device* lain.

Tegangan *output* yang dihasilkan pin GPIO raspberry sebesar 3,3V untuk logika 1 dan 0V untuk logika 0, sedangkan inputan yang dibutuhkan *rellay* untuk menggerakkan *switch* yang terdapat di dalamnya sebesar 5V. Hal ini menyebabkan pin *output* GPIO tidak dapat secara langsung menjadi masukan ke *rellay* karena tegangan yang dihasilkan saat logika 1 dianggap *rellay* sebagai logika 0, sehingga *rellay* tidak akan merespon walaupun terjadi perubahan *state* pada pin *output* GPIO. Oleh karena itu transistor NPN digunakan untuk men-*driver rellay* menggunakan sinyal dari pin output GPIO raspberry, kaki *collector* NPN disambungkan ke salah satu kaki *coil* pada *rellay*, kaki *coil rellay* yang lain disambungkan ke catu daya 5V pada pin VCC raspberry. Kaki *emitter* NPN disambungkan ke *ground* pada pin *ground*

raspberry, dan kaki *base* NPN digunakan sebagai *input*-an yang menerima sinyal dari pin *output* raspberry.

Pin GPIO raspberry yang difungsikan untuk memberikan sinyal keluaran pada kedua kaki *base* NPN rangkaian *Driver* LED adalah pin 37 dan 40. Pin 37 sebagai penghasil sinyal *input*-an pencahayaan sisi kanan dan pin 40 untuk sisi kiri. Jika salah satu pin tersebut memberikan sinyal 1 pada driver LED maka LED akan menyala.

3.2.5 Raspberry Pi 3

Komputer mini raspberry pi 3 digunakan sebagai pusat kendali dan pemrosesan seluruh sistem, prosessor BCM2837 pada raspberry pi 3 menjadi mesin utama proses komputasi untuk mengendalikan dan memproses seluruh sistem. Sistem operasi raspbian yang dapat dijalankan pada raspberry pi 3 menjadi antarmuka antara mesin dan pengguna, sedangkan antarmuka antara raspberry pi 3 dengan komponen lainnya menggunkan 40 GPIO dan beberapa port yang dimiliki raspberry pi 3.

Sumber tegangan yang dibutuhkan raspberry pi 3 untuk beroperasi sebesar 5V dan arus sebesar 2,5A. Sumber daya di pasangkan pada *port* daya yang terdapat pada raspberry. Dua dari empat USB *port* digunakan untuk antarmuka kamera dengan raspberry, sedangkan *ethernet port* digunakan untuk antarmuka raspberry dengan PC menggunakan *virtual network computing* (VNC) sebagai *graphical desktop sharing system* untuk menyederhanakan penggunaan peripheral *keyboard*, *mouse* dan layar pada raspberry. Dengan *graphical desktop sharing system* ketiga peripheral tersebut dapat digantikan menggunakan peripheral yang terdapat pada PC.

Pin GPIO yang dimiliki raspberry pi 3 digunakan untuk antarmuka beberapa rangkaian elektronik yang dibutuhkan untuk menjalankan alat pengenal karakter braille yaitu rangkaian *driver* LED, driver motor *stepper* dan sensor posisi. Terdapat 16 GPIO yang digunakan untuk antarmuka ketiga rangkaian tersebut, 8 GPIO untuk driver motor, 4 GPIO untuk driver LED dan 4 GPIO untuk sensor posisi.

Modul *driver* motor menggunakan pin 4 raspberry untuk sumber VDD, pin 6 untuk *ground* VDD, pin 18 di-*setting* sebagai *output* untuk masukan DIR *driver* motor, pin 22 di-*setting* sebagai *output* untuk masikan STEP *driver* motor. Pin 7, 13 dan 15 di-*setting* sebagai *output* untuk masukan MS1, MS2 dan MS3 secara berurutan dan pin 16

raspberry di-*setting* sebagai *output* untuk masukan ENABLE *driver* motor.

Pin 2 raspberry yang merupakan VCC 5V digunakan kedua rangkaian driver LED untuk salah satu kaki *coil rellay* setiap rangkaian, sedangkan kedua kaki *coil* lainnya di sambungkan ke kaki *collector* NPN pada setiap rangkaian dan kedua kaki *emitter* NPN tersebut disambungkan ke pin 39 raspberry yang merupakan pin *ground*. Pin 37 dan 40 raspberry di-*setting* sebagai *output* untuk masukan pada rangkaian *driver* LED yang tersambung pada kaki *base* NPN rangkaian *driver* LED kanan dan kaki *base* NPN rangkaian *driver* LED kiri secara berurutan.

Pin *ground* 30 dan 34 rspberry digunakan untuk kaki *common micro switch sensor* posisi awal dan akhir secara berurutan, sedangkan kaki NO *micro switch* sensor posisi awal dan akhir disambungkan ke pin 31 dan 29 secara berurutan yang telah di-*setting* sebagai *input* dengan disambungkan ke *internal pull up resisitor* untuk menerima sinyal dari sensor.

	Pin No.		
3.3V	1	2	5V
GPIO2	3	4	5V
GPIO3	5	6	GND
GPIO4	7	8	GPIO14
GND	9	10	GPIO15
GPIO17	11	12	GPIO18
GPIO27	13	14	GND
GPIO22	15	16	GPIO23
3.3V	17	18	GPIO24
GPIO10	19	20	GND
GPIO9	21	22	GPIO25
GPIO11	23	24	GPIO8
GND	25	26	GPIO7
DNC	27	28	DNC
GPIO5	29	30	GND
GPIO6	31	32	GPIO12
GPIO13	33	34	GND
GPIO19	35	36	GPIO16
GPIO26	37	38	GPIO20
GND	39	40	GPIO21

Gambar 3.10 40 pin GPIO raspberry pi 3.

Raspberry pi 3 diletakkan pada panel elektronik seperti rangkaian dan komponen elektronik lainnya. Pin GPIO disambungkan dengan *device* lain menggunakan kabel ekstensi yang disambungkan ke 40 pin *header* pada panel elektronik.

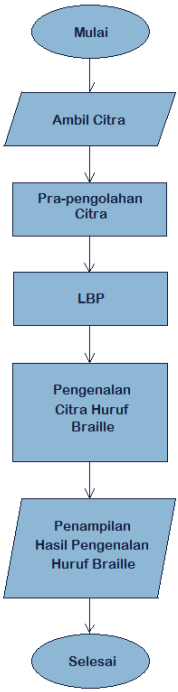
Gambar 3.11 Panel elektronik alat pengenal karakter braille.

3.3 Perancangan perangkat lunak

Sistem operasi yang dijalankan di raspberry pi 3 adalah raspbian yaitu sistem operasi berbasis debian yang dimifikasi untuk raspberry. Bahasa pemrograman yang digunakan untuk menuliskan keseluruhan program adalah C++, untuk melakukan tugas kompilasi program C++ digunakan GNU *Compiler Collection* (GCC).

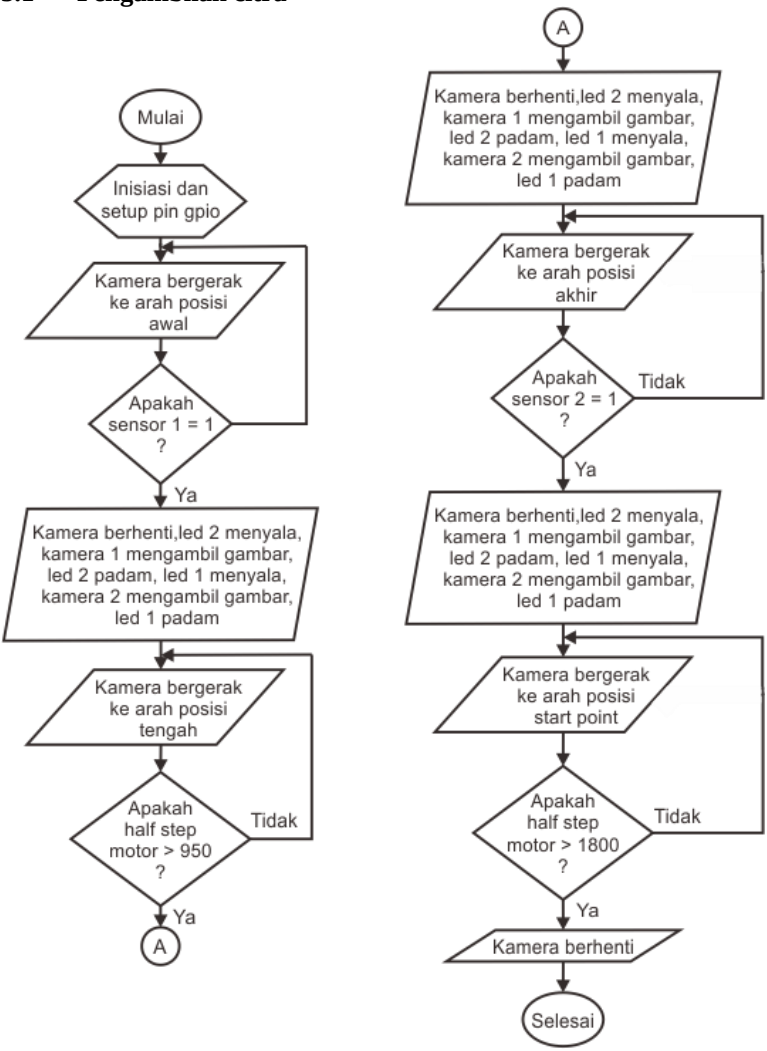
Perancangan perangkat lunak dimulai dari penulisan program yang mengoperasikan motor *stepper* untuk bergerak pada posisi awal, tengah dan akhir untuk mengambil citra naskah braille pada *workspace*, kemudian mengembalikan posisi kamera ke posisi awal. Setelah didapatkan 6 citra, program yang ditulis dirancang untuk melakukan pre-pemrosesan citra agar didapatkan citra yang siap untuk proses segmentasi kemudian penerapan operasi LBP untuk mendapatkan nilai setiap karakter dan yang terakhir pengenalan karakter. Beberapa proses pada pre-pemrosesan citra adalah *cropping* citra, *thresholding*, dilatasi, erosi, penentuan *region of interest* dan penggabungan 6 buah citra menjadi satu naskah utuh yang siap untuk proses segmentasi. Kemudian dilakukan proses segmentasi yang menghasilkan koordinat setiap karakter *braille* pada naskah sehingga program dapat membedakan antara satu karakter dengan karakter yang lainnya, proses segmentasi dilakukan dengan analisa *blob* menggunakan deteksi kontur pada *region*

of interest (ROI) tertentu yang diposisikan pada baris dan kolom pertama setiap karakter braille menggunakan pengukuran manual, setelah didapat kontur pada ROI dihitung titik pusat setiap kontur dan dihitung rata-rata titik kontur keseluruhan, proses segmentasi ini dilakukan mulai dari baris pertama karakter yang berada di *line* pertama sampai baris pertama karakter yang berada di *line* terakhir untuk mendapatkan koordinat y setiap karakter, sedangkan koordinat x setiap karakter didapat dengan proses yang sama hanya saja ROI diposisikan pada kolom pertama karakter pertama pada setiap *line* sampai kolom pertama pada karakter terakhir pada setiap *line*. Setelah proses segmentasi selanjutnya dilakukan proses operasi LBP dengan memeriksa enam koordinat lokasi titik pada setiap karakter, setelah didapatkan nilai LBP kemudian proses pengenalan karakter braille dilakukan dengan mengevaluasi nilai LBP setiap karakter yang didapat.



Gambar 3.12 Diagram alir program pengenalan karakter braille.

3.3.1 Pengambilan citra



Gambar 3.13 Diagram alir program pengambilan citra.

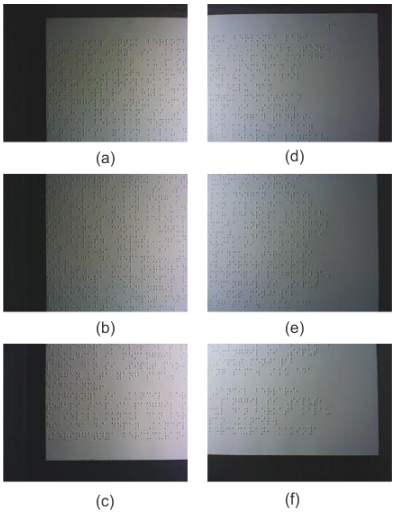
Pengambilan citra dikerjakan dengan menggerakkan kamera ke tiga posisi yang ditentukan yaitu posisi awal, tengah dan akhir. Untuk

menggerakkan kamera beberapa pin GPIO raspberry yang tersambung ke rangkaian driver LED, sensor posisi dan *driver* motor di-*setting* agar dapat mengirim dan menerima sinyal dari dan ke raspberry. *Setup* wiringpi menggunakan fungsi wiringPiSetup() yang menginisiasi fungsi-fungsi wiringpi yang akan digunakan menggunakan penamaan pin GPIO dengan setup wiringpi. Setiap pin yang digunakan di-*setting* satu persatu untuk dijadikan input atau *output* menggunakan fungsi pinMode(int pin, int mode). Pin yang di-*setting* untuk *input* di sambungkan ke *internal pull up* atau *pull down* resistor agar tidak berada pada keadaan *float* menggunakan fungsi pullUpDnControl(int pin, int pud) sedangkan pin yang di-*setting* untuk *output* diberikan keadaan awal menggunakan fungsi digitalWrite(int pin, int value). Pin yang di-*setting* untuk menjadi *input* adalah pin yang tersambung ke sensor posisi awal dan sensor posisi akhir, kedua pin tersebut disambungkan ke *internal pullup resistor* agar jika sinyal 0 dari sensor diterima maka posisi awal atau akhir terbaca oleh raspberry. Pin yang disetting menjadi *output* adalah pin yang tersambung ke masukan driver LED kanan dan kiri, MS1, MS2, MS3, ENABLE, STEP dan DIR pada *driver* motor, semua pin tersebut diberikan keadaan *LOW* kecuali pin MS1 untuk menyetting langkah rotasi motor menjadi *half step*, pin STEP dan DIR yang dibiarkan tidak memiliki kondisi awal, kedua pin ini akan diberikan kondisi untuk menentukan arah dan melakukan langkah pada saat yang tepat nantinya.

Beberapa lokasi memori pada raspberry disiapkan untuk data citra yang diambil dari kamera, digunakan 2 struktur data yang disiapkan untuk menerima data citra setiap *frame* dari kedua kamera menggunakan fungsi [CvCapture *nama] untuk memanggil struktur data yang diperlukan untuk menampung data citra dari kamera, fungsi cvCaptureFromCAM(int kamera) digunakan untuk menunjukkan data yang di-*return* pada CvCapture adalah data citra dari kamera *frame* per *frame* bukan dari *file* video pada lokasi memori yang lain. Agar citra dari kamera di-*return* pada struktur data CvCapture maka fungsi yang dituliskan adalah CvCapture *nama=cvCaptureFromCAM(int kamera). Selain 2 struktur data CvCapture disiapkan juga struktur data untuk mengalokasikan penyimpanan data satu *frame* citra dari kedua kamera pada tiga posisi yang diinginkan menggunkan struktur data IplImage, struktur data ini akan mengalikasikan memori pada raspberry untuk menampung data suatu citra tertentu (citra yang diambil dari kamera). Fungsi yang digunakan untuk menyiapkan struktur data IplImage adalah

fungsi [IplImage* nama]. Enam struktur data IplImage disiapkan untuk menampung data pada setiap citra yang diambil dari dua kamera di ketiga posisi untuk diproses pada tahap selanjutnya.

Setelah semua pin dan struktur data siap selanjutnya program dirancang untuk menggerakkan dan mengambil citra dari 6 sisi naskah braille. Di mulai dari posisi awal citra kiri dan kanan naskah, papan kamera digerakkan ke posisi awal dengan memberikan sinyal 0 ke pin DIR driver motor, selanjutnya pin STEP pada *driver* motor diberikan sinyal 1 dan 0 secara bergantian dengan menggunakan iterasi *for* sampai sensor posisi awal memberikan sinyal 0 ke raspberry, sinyal 0 dari sensor menunjukkan posisi kamera berada pada posisi awal. Pada posisi ini pin masukan *driver* LED kanan diberikan sinyal 1 untuk menghidupkan LED yang terletak pada sisi kanan, kemudian struktur data IplImage untuk citra kiri atas diiskan dengan data citra dari kemera 0 (sisi kiri pada papan kamera) menggunakan fungsi cvQueryFrame(*IplImage). Data citra yang dimasukkan di struktur data IplImage adalah data citra dari *frame* ke 10, hal ini dilakukan agar didapat data citra naskah dengan pencahayaan yang baik. Setelah data citra dimasukkan kemudian data tersebut di simpan menggunakan fungsi cvSaveImage("nama file dan ekstensi",*IplImage). Citra naskah sisi kanan atas didapatkan dengan cara yang sama seperti sisi kiri atas hanya saja sinyal 0 yang dikirimkan ke masukan *driver* LED kanan diubah ke 1 dan sinyal 0 dikirimkan ke masukan *driver* LED kiri untuk memberikan pencahayaan sisi kanan naskah. Setelah kedua citra didapat sinyal yang dikirimkan ke kedua driver LED dijadikan 1 untuk mematikan kedua LED. Selanjutnya papan kamera digerakkan ke posisi tengah dengan mengubah sinyal DIR menjadi 1 dan memeberikan 950 1 dan 0 secara bergantian ke pin STEP pada *driver* motor, kemudian pengambilan citra dilakukan seperti pada posisi awal. Untuk citra pada posisi akhir papan kamera digerakkan dengan memberikan kembali pulsa pada pin STEP sampai pin sensor posisi akhir memberikan sinyal 0, ini menandakan bahwa papan kamera berada pada posisi akhir, kedua citra diambil seperti pada pengabilan citra pada posisi awal.



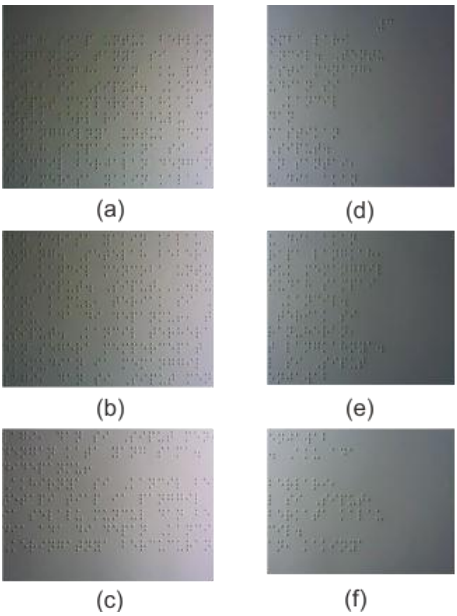
Gambar 3.14 Hasil pengambilan citra (a) kiri atas, (b) kiri tengah, (c) kiri bawah, (d) kanan atas, (e) kanan tengah dan (f) kanan bawah.

Di ambil 6 buah citra pada proses pengambilan gambar dengan tujuan untuk meminimalisir ukuran kerangka perangkat keras, karena jika diambil kurang dari 6 gambar maka dibutuhkan jarak lebih jauh lagi antara kamera dan objek yang berakibat pada ukuran kerangka perangkat keras. Selain itu juga berakibat pada kualitas kamera yang digunakan, artinya jika diinginkan gambar yang baik dengan jarak yang lebih jauh maka dibutuhkan kualitas kamera yang lebih baik pula sehingga pertimbangan biaya juga menjadi salah satu alasan kenapa digunakan 6 buah citra.

Keenam citra yang didapat kemudian di-*crop* ke ukuran tertentu untuk mengkondisikan citra agar sesuai nantinya pada proses *image stitching*. Ukuran *cropping* diukur manual sesuai dengan hasil pengambilan citra yang didapat sebelumnya, masing-masing ukuran citra yang baru untuk citra kiri atas, kiri tengah, kiri bawah, kanan atas, kanan tengah dan kanan bawah adalah 471x410 piksel, 471x350, 471x340, 420x406, 420x345 dan 420x325 piksel secara berurutan. Proses *cropping* dikerjakan menggunakan fungsi cvSetImageROI("citra",cvRect(koordinat x piksel awal, koordinat y piksel awal, lebar, tinggi)).

Tabel 3.2 koordinat awal *cropping* dan ukuran citra baru

Sisi citra	X awal	Y awal	Lebar	Tinggi
Kiri atas	151	55	471	410
Kiri tengah	151	103	471	350
Kiri bawah	151	58	471	340
Kanan atas	163	40	420	406
Kanan tengah	163	88	420	345
Kanan bawah	163	40	420	325

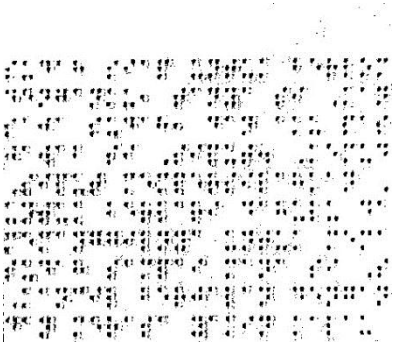


Gambar 3.15 hasil *cropping* citra (a) kiri atas, (b) kiri tengah, (c) kiri bawah, (d) kanan atas, (e) kanan tengah, (f) kanan bawah.

3.3.2 *Thresholding* dan *morphology*

Untuk membedakan antara objek (titik-titik karakter braille) dan *background* citra (area selain titik braille), digunakan operasi *thresholding*, yaitu memberikan nilai tertentu pada setiap piksel citra sesuai dengan nilai *threshold* yang diberikan.

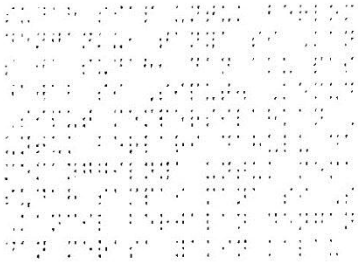
Pada citra dilakukan proses *adaptive threshold* karena iluminasi citra yang didapat dari kamera tidak merata atau terdapat area yang lebih terang dan area yang lebih gelap pada citra akibat pencahayaan yang kurang baik seperti yang terlihat pada gambar 3.8. *adaptive threshold* memberikan nilai atau level *threshold* pada setiap piksel dengan cara menghitung nilai rata-rata area sekitar piksel dan mengurangkannya dengan nilai yang telah ditentukan. Fungsi yang digunakan adalah cvAdaptiveThreshold() yang memiliki 7 parameter yaitu IplImage citra yang ingin di-*threshold*, IplImage hasil *threshold*, nilai maksimal untuk piksel diatas level *threshold*, metode yang digunakan, tipe *threshold*, ukuran luas piksel sekitar dan parameter nilai pengurang. Ketujuh parameter tersebut dituliskan pada fungsi cvAdaptiveThreshold() secara berurutan. Nilai maksimum yang diterapkan pada citra sebesar 255 atau warna hitam, metode yang digunakan adalah CV_ADAPTIVE_THRESH_MEAN_C, tipe *threshold* yang diterapkan adalah CV_THRES_BINARY yang akan memberikan nilai maksimum (255/warna hitam) pada piksel citra yang memiliki nilai di atas level *threshold* yang didapat atau nilai 0 (warna putih) pada piksel yang memiliki nilai di bawah level *threshold* yang didapat. Parameter area dan nilai pengurang didapatkan dengan melakukan beberapa kali percobaan yang akhirnya didapatkan nilai 11dan 6 untuk nilai area dan pengurang secara berurutan.



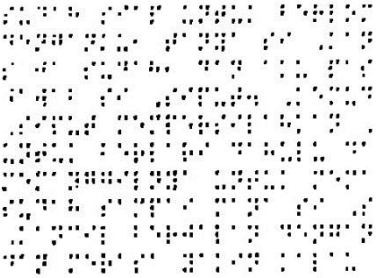
Gambar 3.16 Hasil citra *adaptive threshold*.

Dari hasil proses *adaptive threshold* masih terdapat *noise* pada citra, untuk mereduksi *noise* tersebut dilakukan proses *morphology* yang terdiri dari proses *dilate* dan *erode*. Proses *morphology* diawali

dengan preses *dilate* terlebih dahulu untuk menghilangkan *noise* yang memiliki area lebih kecil dari pada area objek, setelah dilakukan proses *dilate* kemudian dilakukan proses *erode* untuk memperluas citra objek agar memiliki ukuran area semula, sehingga didapatkan citra dengan *noise* yang telah tereduksi. Fungsi yang digunakan untuk melakukan operasi *dilate* adalah cvDelate(IplImage* src, IplImage* dst, IplConvKernel* B = NULL, int iterations = 1) sedangkan untuk fungsi *erode* adalah cvErode(IplImage* src, IplImage* dst, IplConvKernel* B = NULL, int iterations = 1), dimana IplImage * src adalah citra yang akan di-*dilate* atau di-*erode*, IplImage* dst adalah citra hasil *dilate* atau *erode*, IplConvKernel* B adalah *kernel* yang digunakan dengan nilai NULL untuk menggunakan *kernel basic* dari openCV, dan *int iterations* adalah nilai iterasi yang dilakukan saat proses *dilate* atau *erode*.

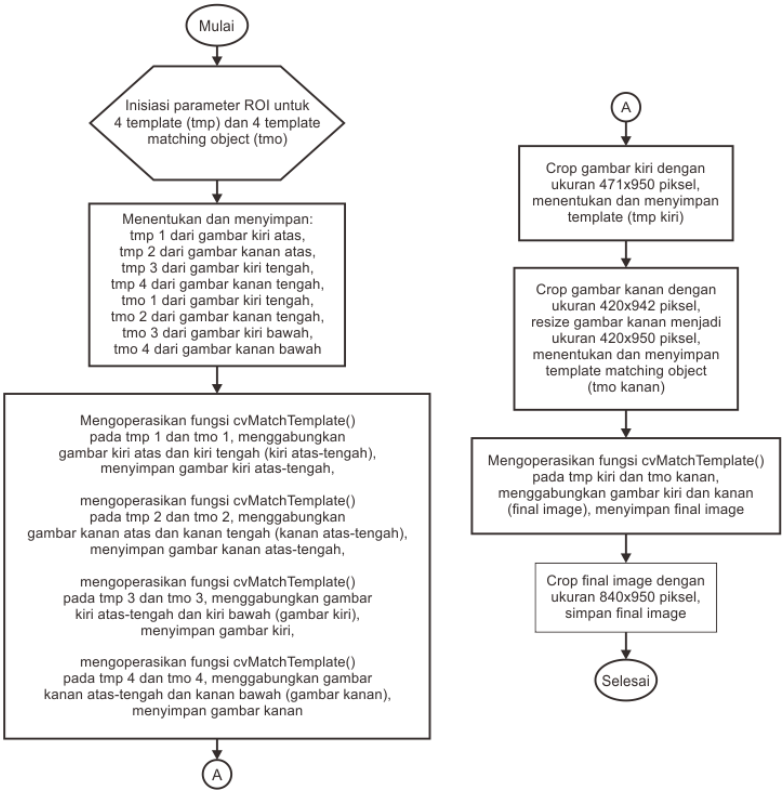


Gambar 3.17 Hasil operasi *dilate* citra yang telah dilakukan operasi *threshold* sebelumnya.



Gambar 3.18 Hasil operasi *erode* citra yang telah dilakukan operasi *dilate* sebelumnya (citra dengan *noise* yang telah direduksi).

3.3.3 *Image stitching*



Gambar 3.19 Diagram alir algoritma *image stitching*.

Proses *image stitching* adalah proses penggabungan 6 buah citra yang telah didapat menjadi satu citra naskah braille utuh. Proses penggabungan dilakukan dari 3 citra bagian kiri terlebih dahulu, kemudian 3 citra bagian kanan dan yang terakhir citra kiri dan kanan digabungkan menjadi satu.

Fungsi utama yang digunakan pada proses ini adalah fungsi cvMatchTemplate(IplImage* citra, IplImage* template, IplImage* hasil, int metode), parameter IplImage* citra adalah citra target yang akan di cocokan dengan citra *template*, parameter IplImage* template adalah citra yang memiliki ukuran lebih kecil dari citra target yang akan

dicocokan ke citra target. Parameter IplImage* hasil adalah citra yang berisi nilai hasil kecocokan antara citra target dan citra *template*, nilai yang terdapat pada citra hasil tergantung dari metode yang digunakan pada parameter int metode, citra hasil harus memiliki ukuran (IplImage* citra -> width – IplImage* template -> width + 1, IplImage* citra -> high – IplImage* template ->high + 1). Parameter int metode adalah parameter yang befungsi untuk memiliki metode pencocokan yang digunakan, terdapat 6 metode yang terdapat pada opencv, untuk proses penggabungan citra braille kali ini digunakan metode CV_TM_SQDIFF, metode ini akan memberikan nilai terkecil pada citra hasil untuk hasil pencocokan yang memiliki kecocokan paling dekat. Nilai yang diberikan pada setiap piksel citra hasil jika menggunakan metode CV_TM_SQDIFF dapat diformulasikan dengan:

$$R(x,y) = \sum_{x',y'} [T(x',y') - I(x+x',y+y')]^2$$

Dimana R = IplImage* hasil, T = IplImage* template dan I = IplImage* citra. Proses pencocokan template ke citra dimulai dari piksel (0,0) kemudian bergeser ke kanan dan ke bawah.

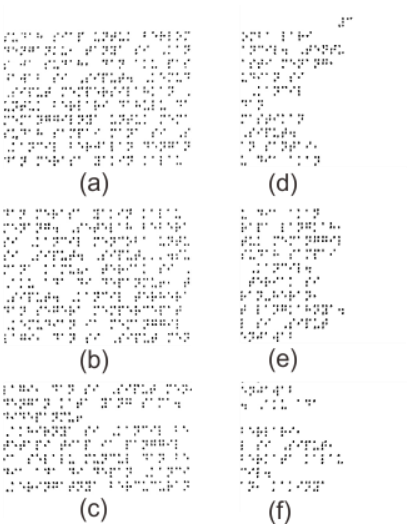
Pada penggabungan 3 citra kiri dan 3 citra kanan IplImage* template dan IplImage* citra merupakan bagian dari pada ke 6 citra dari hasil *morphology*. Untuk menggabungkan citra kiri atas dengan citra kiri tengah IplImage* template merupakan bagian bawah dari citra kiri atas sedangkan IplImage* citra merupakan bagian atas dari citra kiri tengah. Untuk menggabungkan citra hasil kiri atas-tengah dan kiri bawah IplImage* template merupakan bagian bawah citra kiri tengah sedangkan IplImage* citra merupakan bagian atas citra kiri bawah. Untuk menggabungkan citra kanan atas dengan citra kanan tengah IplImage* citra merupakan bagian bawah citra kanan atas sedangkan IplImage* citra merupakan bagian atas citra kanan tengah. Untuk menggabungkan citra hasil kanan atas-tengah dengan citra kanan bawah IplImage* template merupakan bagian bawah citra kanan tengah sedangkan IplImage* citra merupakan bagian atas citra kanan bawah. IplImage* template dan IplImage* citra didapat dengan menggunakan fungsi cvSetImageROI(IplImage* img, cvRect(x, y, lebar, tinggi)).

Tabel 3.3 IplImage* template

IplImage* template	Bagian bawah citra	(x,y) awal	Lebar	Tinggi
1	Kiri atas	(0,375)	471	35
2	Kanan atas	(0,371)	420	35
3	Kiri tengah	(0, 315)	471	35
4	Kanan tengah	(0,310)	420	35

Tabel 3.4 IplImage* citra

IplImage* citra	Bagian atas citra	(x,y) awal	Lebar	Tinggi
1	Kiri tengah	(0,0)	471	40
2	Kanan tengah	(0,0)	420	40
3	Kiri bawah	(0,0)	471	40
4	Kanan bawah	(0,0)	420	40

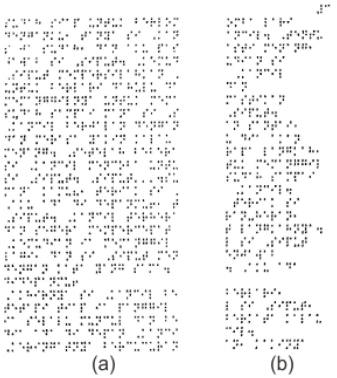


Gambar 3.20 Enam citra hasil morfologi (a) kiri atas, (b) kiri tengah, (c) kiri bawah, (d) kanan atas, (e) kanan tengah dan (f) kanan bawah.

IplImage* hasil dibuat untuk menampung nilai hasil dari proses pencocokan menggunakan fungsi cvCreateImage() dengan channel 1 dan ukuran lebar = lebar IplImage* citra – lebar IplImage* template + 1, dan tinggi = tinggi IplImage* citra – tinggi IplImage* template + 1.

Setelah hasil pencocokan didapatkan kemudian dicari lokasi piksel yang memiliki nilai paling rendah pada IplImage* hasil menggunakan fungsi cvMinMaxLoc(IplImage* img, double min_value, double max_value, point min_location, point max_location). Setalah diketahui lokasi yang paling cocok dari hasil image stitching selanjutnya mengkontruksi citra baru yang merupakan citra gabungan antara dua citra menggunakan fungsi CV_IMAGE_ELEM() untuk mendapatkan nilai piksel dari kedua citra.

Setelah ke tiga citra kiri dan kanan digabungkan hasil citra penggabungan seperti pada gambar 3.19.



Gambar 3.21 Citra hasil penggabungan (a) sisi kiri, (b) sisi kanan.

Dengan metode yang sama selanjutnya citra sisi kiri dan citra sisi kanan digabungkan menjadi satu naskah utuh, proses dimulai dengan melakukan cropping pada kedua citra untuk mengurangi *background* pada kedua citra. Karena citra sisi kanan memiliki posisi baris setiap objek yang berbeda dari citra sisi kiri akibat ketidak sempurnaan posisi kamera maka dilakukan *resizing* pada citra sisi kanan untuk mendapatkan baris objek yang lebih baik pada kedua citra. Fungsi yang digunakan untuk *resizing* adalah cvResize(IplImage* citra, IplImage* citra ukuran baru, int metode), metode yang dipilih adalah metode CV_INTER_LINEAR, ukuran citra sisi kanan baru setelah dilakukan

proses *resize* adalah 420x950 piksel. Kemudian diambil IplImage* template dari bagian kanan citra sisi kiri dan IplImage* citra dari bagian kiri citra sisi kanan, selanjutnya dilakukan pencocokan dan pencarian lokasi piksel yang memiliki nilai terkecil pada IplImage* hasil dan terakhir dilakukan rekonstruksi kedua citra menjadi satu citra utuh.

Tabel 3.5 Ukuran *cropping* dan pengambilan IplImage* template, IplImage* citra pada citra sisi kiri dan citra sisi kanan

ROI	Citra awal	(x,y) awal	Lebar	Tinggi
Cropped citra sisi kiri	Citra sisi kiri	(0,25)	471	950
Cropped citra sisi kanan	Citra sisi kanan	(0,25)	420	942
IplImage* Template	Cropped citra sisi kiri	(448,0)	23	950
IplImage* citra	Cropped citra sisi kanan yang telah di-resize	(19,0)	28	950

Hasil akhir proses *image stitching* merupakan satu citra yang didapat dari penggabungan ke enam citra yang telah melalui proses *thresholding*, reduksi *noise*, beberapa proses *cropping*, *resizing* dan siap untuk diproses ke tahap selanjutnya yaitu segmentasi karakter braille pada citra. Citra hasil *image stitching* dapat dilihat pada gambar 3.22.

Gambar 3.22 Hasil akhir proses image stitching dari enam citra.

3.3.4 Segmentasi

```
void ()
{
  //Inisiasi parameter area check point 1 dan 2
  Int x_roi = 640; //titik awal x check point 1
  Int y_roi = 0; //titik awal y check point 2
  Int h = 13; //tinggi area check point
  Int w = 200; //lebar area check point
  Int g = 35; //langkah check point
  Int gh = 32; //kalibrasi langkah
  Int a = 5; //parameter hasil jika contour = 1

  Memuat gambar pada IplImage* img;
  Int i = 0;
  Int y[27];
  Menemukan contour check point 1 pertama pada
  cvSetImageROI(img, cvRect(x_roi, y_roi, w, h));
  Menyimpan nilai rata-rata centroid contour pada array y[i];
  i = i + 1;
```

```
x_roi = 400;
y_roi = y_roi + g;
Int x_roi1 = 0; //titik awal x check point 2
Int y_roi1 = y_roi + g; //titik awal y check point 2
For( ; i < 27;  i++)
{
  Menemukan contour check point 1 selanjutnya pada
  cvSetImageROI( img, cvRect( x_roi, y_roi, w, h ));
  if( contour != 1 )
  {
    Int m = nilai rata-rata centroid contour;
  }
  Else
  {
    Int m = y_roi + a;
  }
  Menemukan contour check point 2 selanjutnya pada
  cvSetImageROI( img, cvRect( x_roi1, y_roi1, w, h ));
  if( contour != 1 )
  {
    Int m1 = nilai rata-rata centroid contour;
  }
  Else
  {
    Int m1 = y_roi1 + a;
  }
  y[i] = round( (m+m1)/2 );
  if( i%7 == 0 )
  {
    y_roi = y_roi + gh;
    y_roi1 = y_roi1 + gh;
  }
  Else
  {
    y_roi = y_roi + g;
    y_roi1 = y_roi1 + g;
  }
}
```

```
}

//Catatan : nilai parameter check point seperti h dan a dapat diubah
untuk mendapatkan hasil yang maksimal.
```

Gambar 3.23 *Pseudo code* algoritma segmentasi y.

```
void ()
{
  //Inisiasi parameter area check point 1dan 2
  Int x_roi = 0; //titik awal x check point 1
  Int y_roi = 0; //titik awal y check point 1
  Int x_roi1 = 0; //titik awal x check point 2
  Int y_roi1 = 650; //titik awal y check point 2
  Int h = 300; //tinggi area check point
  Int w = 13; //lebar area check point
  Int g = 20; //langkah check point
  Int gh = 22; //kalibrasi langkah
  Int a = 5; //parameter hasil jika contour = 1

  Memuat gambar pada IplImage* img;
  Int x[41];
  For( int i = 0; i < 41; i++ )
  {
    Menemukan contour check point 1 pada
    cvSetImageROI( img, cvRect( x_roi, y_roi, w, h ));
    if( contour != 1 )
    {
      Int m = nilai rata-rata centroid contour;
    }
    Else
    {
      Int m = x_roi + a;
    }
    Menemukan contour check point 2 pada
    cvSetImageROI( img, cvRect( x_roi1, y_roi1, w, h ));
    if(contour != 1)
    {
```

```
      Int m1 = nilai rata-rata centroid contour;
    }
    Else
    {
      Int m1 = x_roi1 + a;
    }
    x[i] = round( (m+m1)/2 );
    if( i%5 == 0 )
    {
      x_roi = x_roi + gh;
      x_roi1 = x_roi1 + gh;
    }
    Else
    {
      x_roi = x_roi + g;
      x_roi1 = x_roi1 + g;
    }
  }
}

//Catatan : nilai parameter check point seperti h dan a dapat diubah
untuk mendapatkan hasil yang maksimal.
```

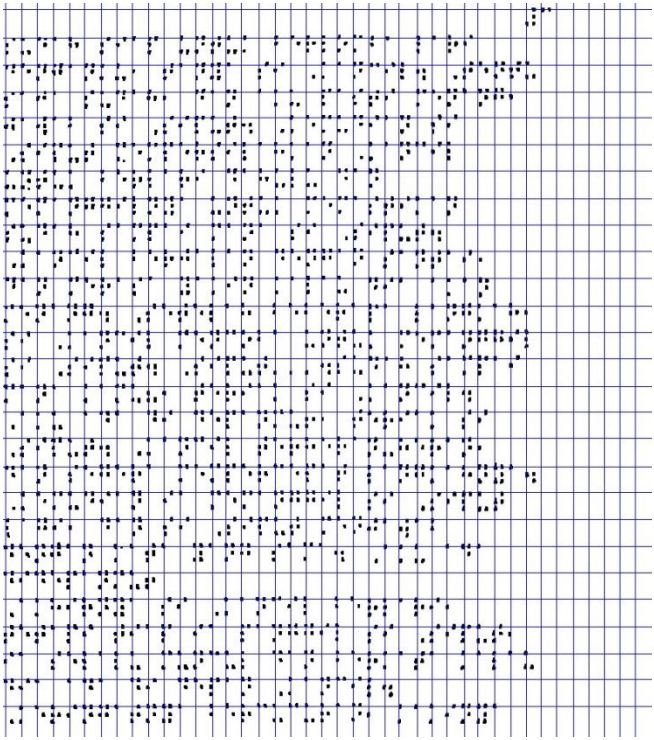
Gambar 3.24 *Pseudo code* algoritma segmentasi x.

Proses segmentasi dilakukan untuk mengetahui koordinat setiap karakter pada citra sehingga dapat dibedakan antara karakter yang satu dengan yang lain. Hasil dari proses segmentasi merupakan dua array yang memiliki nilai koordinat x dan y titik 1 posisi huruf baca setiap karakter braille pada naskah. Proses segmentasi dimulai dari pencarian koordinat y titik 1 dan 4 posisi huruf baca pada setiap baris karakter, keseluran baris karakter pada naskah sebanyak 27 baris. Kemudian proses segmentasi dilakukan untuk pencarian koordinat x titik 1,2 dan 3 posisi huruf baca pada setiap kolom karakter, total keseluruhan kolom karakter pada naskah sebanyak 41 kolom. Dari kedua pencarian koordinat y dan x tersebut dihasilkan koordinat (x,y) titik 1 posisi huruf baca pada setiap karakter yang terdapat pada naskah.

Yang menjadi kunci utama pada proses segmentasi karakter braille adalah penentuan posisi ROI (*region of interest*) atau *chek point* pada naskah untuk menemukan contour baris dan kolom titik-titik karakter. Untuk proses pencarian koordinat y digunakan dua ROI pada setiap baris yang terdapat titik 1 dan 4 posisi huruf baca, ROI pertama diposisikan pada sisi kiri sedangkan ROI kedua dipsisikan pada sisi kanan, hal ini bertujuan untuk mendapatkan nilai rata-rata titik tengah setiap *contour* pada kedua sisi, pencarian nilai rata-rata diperlukan karena ketidak sempurnaan citra naskah yang dihasilkan dari proses sebelumnya, pada baris pertama hanya digunakan satu ROI saja karena hanya terdapat nomor halaman pada baris pertama. Untuk proses pencarian koordinat x juga digunakan dua ROI pada setiap kolom yang terdapat titik 1,2 dan 3 posisi huruf baca, ROI pertama diposisikan pada sisi atas sedangkan ROI kedua diposisikan pada sisi bawah, hal ini juga bertujuan untuk mendapatkan nilai rata-rata titik tengah setiap contour pada kedua sisi.

Setelah ditentukan posisi ROI pada setiap sisi, untuk mendapatkan koordinat y titik 1 posisi huruf baca pada setiap baris terlebih dahulu dicari *contour* yang terdapat pada setiap ROI kedua sisi dengan fungsi cvFindContours() kemudian dicari *boundbox* setiap *contour* menggunakan fungsi cvBoudingRect() kemudian dicari titik tengah koordinat y setiap contour dengan rumus (koordinat y awal ROI + *boundbox* y) + (tinggi *boundbox*/2). Setelah didapat titik tengah koordinat y pada setiap *contour* yang terdeteksi kemudian di cari nilai rata-rata titik tengah pada setiap sisi ROI, kedua hasil rata-rata dari kedua sisi kemudian dijumlahkan dan dibagi dua kemudian dibulatkan agar didapatkan hasil *integer*. Cara yang sama juga digunakan untuk mendapatkan koordinat x titik 1 posisi baca setiap kolom dengan posisi ROI seperti pada tabel diatas. Berikut penggalan program yang digunakan untuk mendapatkan koordinat y titik 1 posisi huruf baca setiap baris pada naskah:

Keakuratan hasil akhir dari proses segmentasi direpresentasikan menggunakan titik potong garis-garis x dan y yang digambarkan pada naskah, koordinat garis-garis tersebut berdasarkan koordinat (x,y) dari hasil segmentasi.

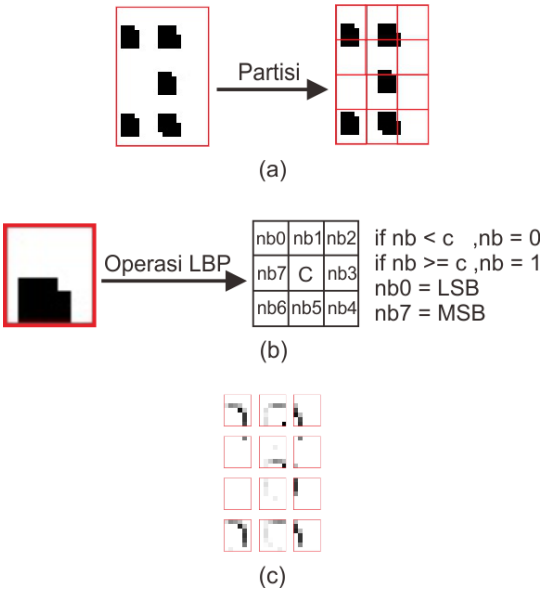


Gambar 3.25 Hasil akhir proses segmentasi.

Karena menggunakan algoritma yang mengikutkan *check point* pada setiap langkahnya, segmentasi tidak bersifat adaptif atau tidak beroperasi dengan baik terhadap objek yang memiliki sifat letak dan ukuran karakter yang berbeda dari objek uji diatas. Untuk mendapatkan hasil segmentasi yang baik harus dilakukan pengaturan ulang nilai parameter *check point* ukuran lebar untuk koordinat x dan tinggi untuk koordinat y.

3.3.5 *Local binary pattern*

Sebelum dilakukan operasi *local binary pattern*, terlebih dahulu dilakukan proses *cropping* untuk mendapatkan citra setiap karakter dengan ukuran 20x30 piksel menggunakan nilai koordinat x dan y hasil segmentasi sebelumnya, hal ini akan memudahkan operasi *local binary pattern* selanjutnya.



Gambar 3.26 Operasi *local binary pattern* pada citra karakter huruf y. (a) Partisi citra karakter menjadi 12 bagian. (b) Operasi LBP pada setiap bagian partisi. (c)Hasil citra LBP ke 12 citra karakter y.

Langkah pertama adalah proses partisi citra karakter menjadi 12 bagian, pembagian ke 12 bagian partisi sesuai dengan data pada tabel dibawa, ukuran setiap bagian partisi adalah 7x8 piksel.

Tabel 3.6 Ukuran dan koordinat bagian partisi citra karakter

Bagian partisi	Koordinat awal		Lebar	Tinggi	Ukuran LBP
	x	y			
1	0	0	8	9	6x7

2	6	0	8	9	6x7
3	12	0	8	9	6x7
4	0	7	8	9	6x7
5	6	7	8	9	6x7
6	12	7	8	9	6x7
7	0	14	8	9	6x7
8	6	14	8	9	6x7
9	12	14	8	9	6x7
10	0	21	7	8	6x7
11	6	21	7	8	6x7
12	12	21	7	8	6x7

Setelah citra karakter dipartisi menjadi 12 bagian, kemudian setiap bagian akan dilakukan operasi *local binary pattern* seperti pada gambar 3.26 (b), dimana piksel tetangga yang memiliki nilai kurang dari nilai piksel tengah di berikan nilai binari 0, sedangkan piksel tetangga yang memiliki nilai lebih besar dari atau sama dengan nilai piksel tengah diberikan nilai binari 1. Kemudian rangkaian nilai binari piksel tetangga dikoversikan ke bilangan desimal dengan nb0 sebagai lest significant bit dan nb7 sebagai most significant bit, nilai desimal tersebut akan menjadi nilai piksel pada citra LBP.

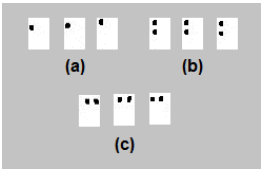
Selanjutnya citra LBP yang didapatkan disimpan dan akan di panggil untuk proses selanjutnya, contoh citra hasil operasi *local binary pattern* seperti pada gambar 3.26 (c).

3.3.6 *Data learning*

Data learning digunakan untuk mendapatkan nilai parameter ukur citra karakter dengan cara membandingkan nilai histogram tiap partisi karakter dengan *data learning* yang ada, dari nilai tersebut dapat ditentukan citra karakter dikenali sebagai karakter tertentu pada tahap selanjutnya.

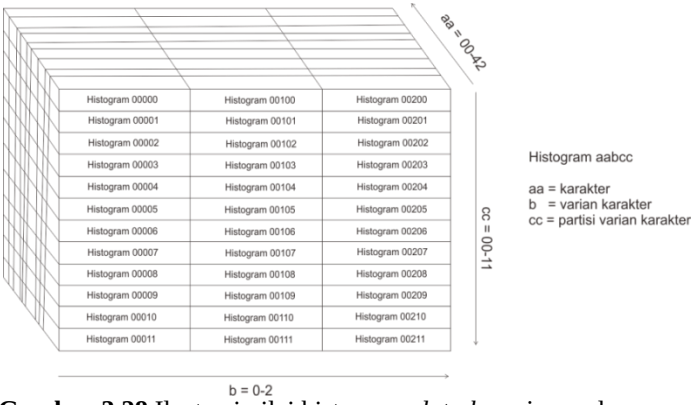
Data learning yang digunakan adalah nilai histogram partisi citra lbp karakter yang telah dipartisi menjadi 12 bagian, untuk setiap karakter huruf braille diberikan 3 citra *data learning* sehingga untuk keseluruhan terdapat 129 citra data learning. Untuk mendapatkan nilai kedekatan karakter, setiap citra karakter di bandingkan dengan 129 citra data learning sehingga menghasilkan nilai terdekat dari karakter tersbut

dengan nilai karakter data learning tertentu yang akan dibahas pada sub bab selanjutnya.



Gambar 3.27 Citra data learning. (a) karakter a, (b) karakter b, (c) karakter c.

Pada proses pengenalan karakter nantinya akan ada bagian inisialisasi *data learning* yaitu proses mendapatkan nilai histogram ke 129 citra *data learning* yang telah melalui operasi lbp dan partisi menjadi 12 bagian, nilai histogram *data learning* disimpan dalam *array* 4 dimensi dengan ukuran array [43][3][12][256]. Dimensi pertama dengan ukuran [256] digunakan untuk menyimpan nilai histogram setiap partisi yang terdiri dari 256 bins, dimensi kedua dengan ukuran [12] digunakan untuk menyimpan nilai histogram setiap partisi karakter, dimensi ketiga dengan ukuran [3] digunakan untuk menyimpan histogram ke 3 varian citra *data learning* setiap karakter yang sama, dan dimensi keempat dengan ukuran [43] digunakan untuk menyimpan nilai histogram keseluruhan jenis karakter yang terdapat 43 jenis, jika digambarkan akan terlihat seperti gambar dibawah ini:



Gambar 3.28 Ilustrasi nilai histogram *data learning* pada *array* 4 dimensi.

Pada gambar ilustrasi nilai *data learning* diatas di ilustrasikan sebagaimana nilai histogram setiap partisi citra karakter di simpan pada *array* 4 dimensi dengan ukuran [43][3][12][256], dimana setiap histogram disimpan pada *array* berukuran 256 untuk menyimpan nilai histogram 256 *bins* dari 0-255. Digunakan pengkodean aabcc untuk memudahkan pembaca pada ilustrasi diatas.

3.3.7 Pengenalan karakter

Proses terakhir adalah pengenalan karakter dengan cara membandingkan nilai histogram setiap partisi citra karakter dengan nilai histogram data learning menggunakan metode *Chi-square*, dengan metode ini nilai terkecil dari hasil perbandingan merupakan indikator citra yang memiliki kesamaan sifat histogram terdekat. Metode *Chi-square* dirumuskan dalam persamaan berikut:

$$chi - square(H1, H2) = \sum_{i} \frac{(H1(i) - H2(i))^2}{H1(i) + H2(i)}$$

Di mana:

- H1(i) = nilai histogram citra 1 bin ke i
- H2(i) = nilai histogram citra 2 bin ke i

Histogram setiap citra karakter dibandingkan dengan semua nilai histogram *data learning*, sehingga didapatkan nilai terkecil dari hasil berbandingan, dan karakter pada citra dapat dikenali dari melihat hasil terkecil tersebut berada pada karakter *data learning* yang mana. Hasil pengenalan karakter berupa nilai *integer* dengan *range* dari 0-42 yang menlambangkan setiap karakter yang terdapat pada sistem braille. Setelah semua karakter dikenali kemudian hasil pengenalan tersebut diterjemah kan ke dalam ASCII menggunakan tabel translasi pada *look up table* yang terdapat pada *code* program.

Halaman ini sengaja dikosongkan

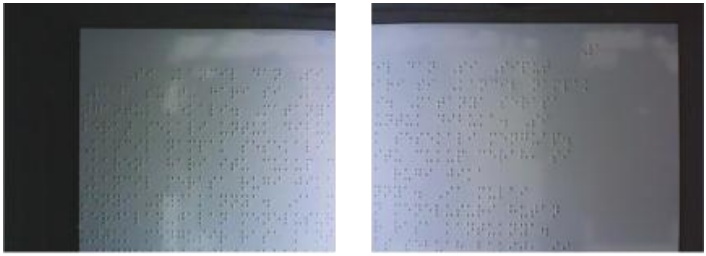
BAB IV
PENGUKURAN DAN ANALISA SISTEM

4.1 Pengujian algoritma *preprocessing*

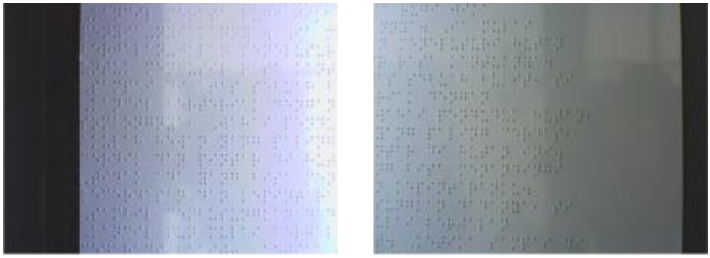
Pada pengujian algoritma *preprocessing* terdiri dari pengujian hasil pengambilan gambar, *thresholding*, morfologi, *image stitching* dan segmentasi.

4.1.1 Pengambilan gambar

Gambar objek direkam menggunakan dua buah kamera yang terpasang pada papan kamera yang digerakkan ke tiga posisi menggunakan motor *stepper*, saat pengambilan gambar salah satu sumber cahaya diaktifkan dari sisi samping objek untuk memperjelas tekstur tonjolan yang terdapat pada objek (naskah braille). Pada posisi awal dua buah gambar direkam dari sisi kanan dan kiri bagian atas objek, pada posisi kedua dua buah gambar direkam dari sisi kanan dan kiri objek bagian tengah, dan pada posisi ketiga dua buah gambar direkam dari sisi kanan dan kiri objek bagian bawah. Keenam gambar tersebut harus sudah mencakup keseluruhan bagian objek agar dapat dijadikan satu buah gambar utuh pada proses selanjutnya.

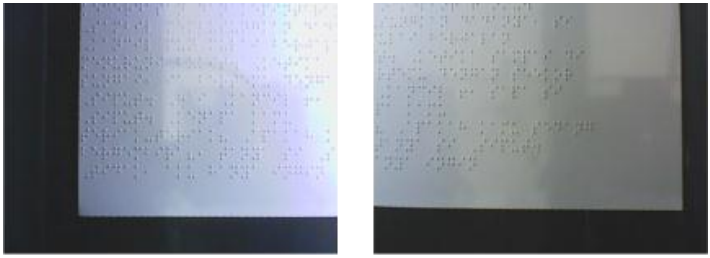


Gambar 4.1 Hasil pengambilan gambar posisi atas. (a) sisi kiri, (b) sisi kanan.



(a) (b)

Gambar 4.2 Hasil pengambilan gambar posisi tengah. (a) sisi kiri, (b) sisi kanan.



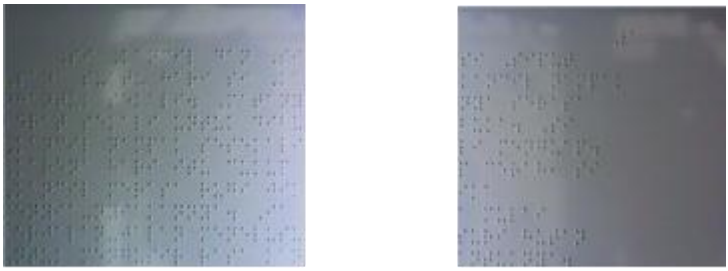
(a) (b)

Gambar 4.3 Hasil pengambilan gambar posisi bawah. (a) sisi kiri, (b) sisi kanan.

Setelah didapatkan gambar objek selanjutnya dilakukan proses *cropping* untuk menghilangkan bagian *background* dari objek. Ukuran *cropping* sesuai dengan tabel dibawah:

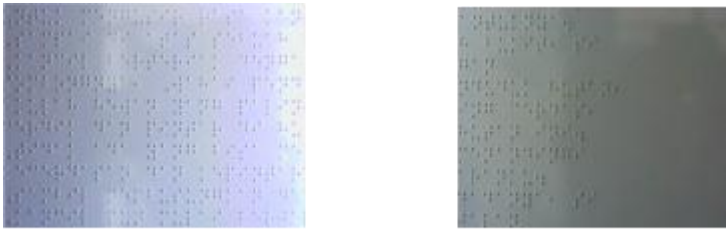
Tabel 4.1 Ukuran *cropping* gambar

Sisi citra	**X awal**	**Y awal**	**Lebar**	**Tinggi**
Kiri atas	151	55	471	410
Kiri tengah	151	103	471	350
Kiri bawah	151	58	471	340
Kanan atas	163	40	420	406
Kanan tengah	163	88	420	345
Kanan bawah	163	40	420	325



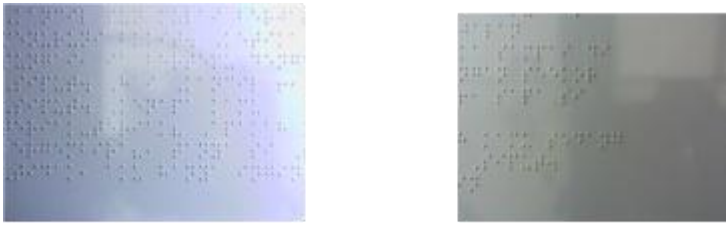
(a) (b)

Gambar 4.4 Hasil *cropping* gambar atas. (a) sisi kiri, (b) sisi kanan.



(a) (b)

Gambar 4.5 Hasil *cropping* tengah. (a) sisi kiri, (b) sisi kanan.

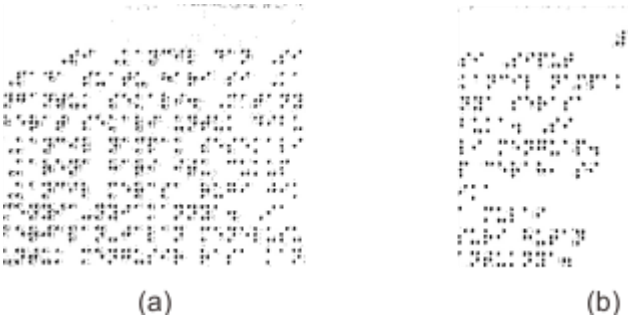


(a) (b)

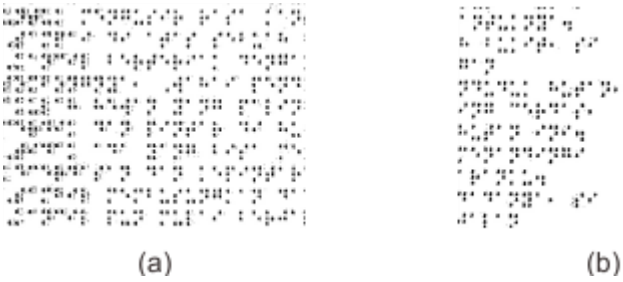
Gambar 4.6 Hasil *cropping* gambar bawah. (a) sisi kiri, (b) sisi kanan.

4.1.2 *Thresholding*

Pada proses *thresholding* digunakan *adaptive threshold* karena iluminasi yang tidak merata. Metode yang digunakan adalah CV_ADAPTIVE_THRESH_MEAN_C, tipe yang dipilih adalah *binary threshold*, dengan nilai parameter *maximum value* 255, *block size* 11 dan param1 (pengurang) 6. Proses ini digunakan untuk memisahkan objek (tonjolan pada naskah) dari *background*.



Gambar 4.7 Hasil *adaptive threshold* gambar atas. (a) sisi kiri, (b) sisi kanan.



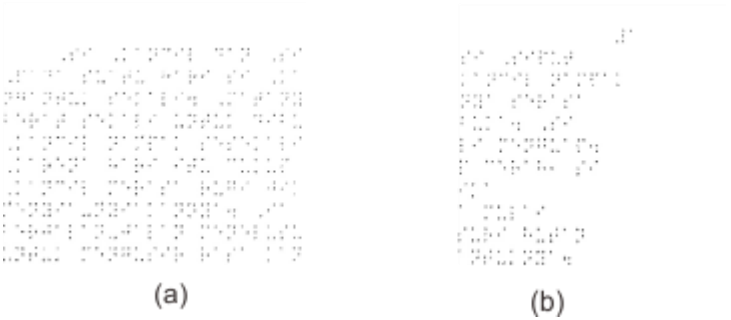
Gambar 4.8 Hasil *adaptive threshold* gambar tengah. (a) sisi kiri, (b) sisi kanan.

Gambar 4.9 Hasil *adaptive threshold* gambar bawah. (a) sisi kiri, (b) sisi kanan.

Masih terdapat *noise* pada gambar hasil *thresholding*, *noise* tersebut dapat dihilangkan pada proses selanjutnya yaitu morfologi.

4.1.3 Morfologi

Untuk menghilangkan *noise* dilakukan proses morfologi yang terdiri dari *dilate* dan *erode*. Proses dilate digunakan untuk menghilangkan *noise* dengan efek ukuran objek menjadi lebih kecil, kemudian untuk mengembalikan ukuran objek ke ukuran semula dilakukan proses *erode*. Baik *dilate* maupun *erode* digunakan kernel berukuran 3x3 piksel.

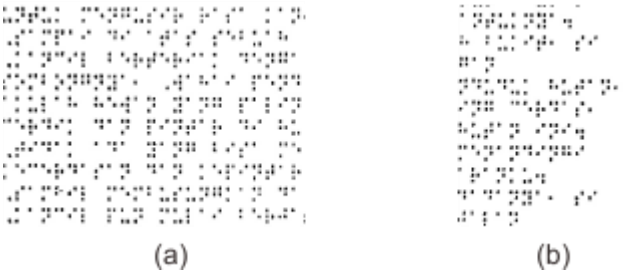


Gambar 4.10 Hasil dilate gambar atas. (a) sisi kiri, (b) sisi kanan.

Gambar 4.11 Hasil dilate gambar tengah. (a) sisi kiri, (b) sisi kanan.

Gambar 4.12 Hasil dilate gambar bawah. (a) sisi kiri, (b) sisi kanan.

Gambar 4.13 Hasil erode gambar atas. (a) sisi kiri, (b) sisi kanan.



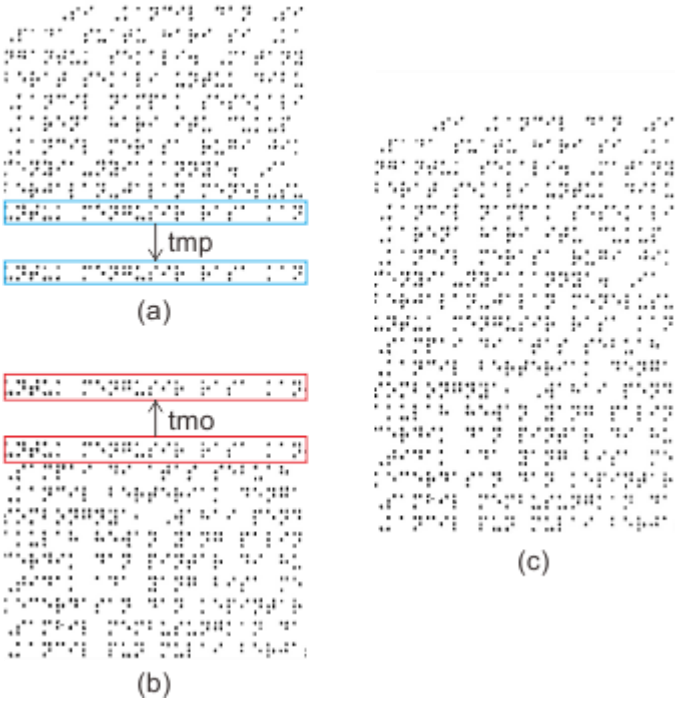
Gambar 4.14 Hasil erode gambar tengah. (a) sisi kiri, (b) sisi kanan.

Gambar 4.15 Hasil erode gambar bawah. (a) sisi kiri, (b) sisi kanan.

Pada hasil akhir morfologi gambar sudah tidak terdapat *noise* atau sudah jauh berkurang.

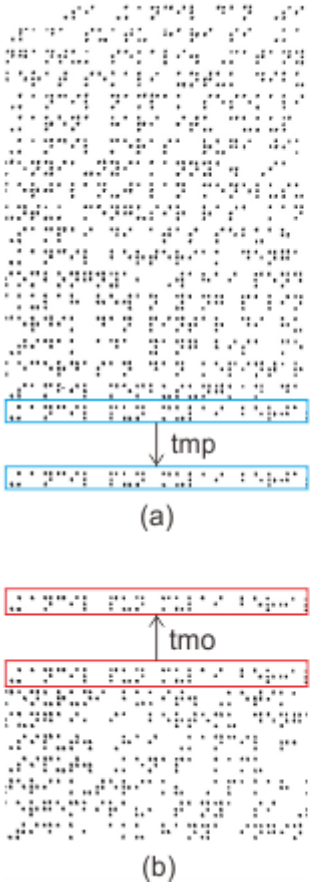
4.1.4 *Image stitching*

Proses *imaage stitching* dimulai dari penggabungan gambar kiri atas dengan gambar kiri tengah, kemudian gambar kanan atas dengan gambar kanan tengah, dilanjutkan gambar kiri atas-tengah dengan gambar kanan bawah, kemudian gambar kanan atas-tengah dengan gambar kanan bawah dan terakhir gambar sisi kiri dengan gambar sisi kanan. Pada setiap penggabungan ditentukan *template* (tmp) dan *template matching object* (tmo).

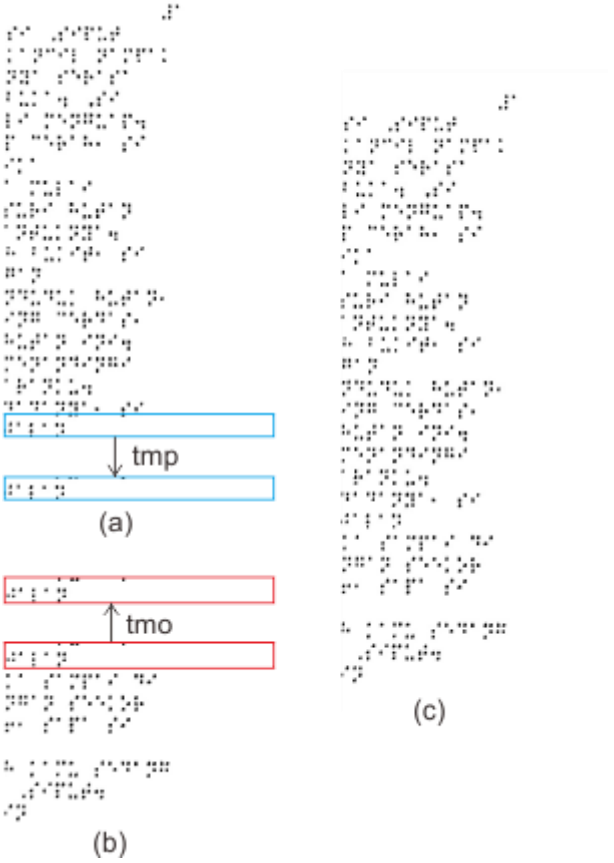


Gambar 4.16 Hasil *image stitching* gambar kiri atas dan kiri tengah. (a) gambar kiri atas dan *template*, (b) gambar kiri tengah dan *template matching object*, (c) hasil penggabungan (gambar kiri atas-tengah).

Gambar 4.17 Hasil *image stitching* gambar kanan atas dan kanan tengah. (a) gambar kanan atas dan *template*, (b) gambar kanan tengah dan *template matching object*, (c) hasil penggabungan (gambar kanan atas-tengah).



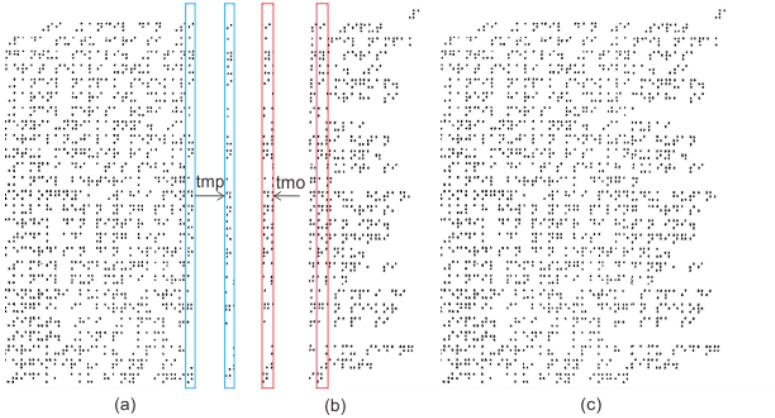
Gambar 4.18 Hasil *image stitching* gambar kiri atas-tengah dan kiri bawah. (a) gambar kiri atas-tengah dan *template*, (b) gambar kiri bawah dan *template matching object*, (c) hasil penggabungan (gambar sisi kiri).



Gambar 4.19 Hasil *image stitching* gambar kanan atas-tengah dan kanan bawah. (a) gambar kanan atas-tengah dan *template*, (b) gambar kanan bawah dan *template matching object*, (c) hasil penggabungan (gambar sisi kanan).

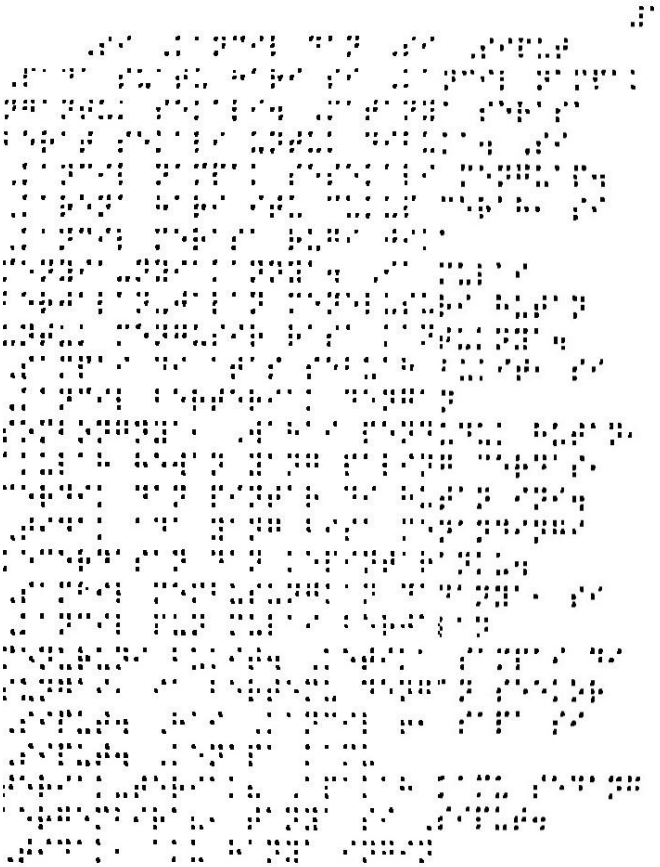
Setelah didapatkan gambar sisi kiri dan gambar sisi kanan, sebelum menggabungkan kedua gambar, pada gambar sisi kanan dan kiri dilakukan *cropping* untuk mendapatkan ukuran yang sesuai, kemudian

dilakukan proses *resize* pada gambar sisi kanan untuk menyeimbangkan kedua gambar.



Gambar 4.20 Hasil *image stitching* gambar sisi kiri dan sisi kanan. (a) gambar sisi kiri dan *template*, (b) gambar sisi kanan dan *template matching object*, (c) hasil penggabungan (gambar utuh).

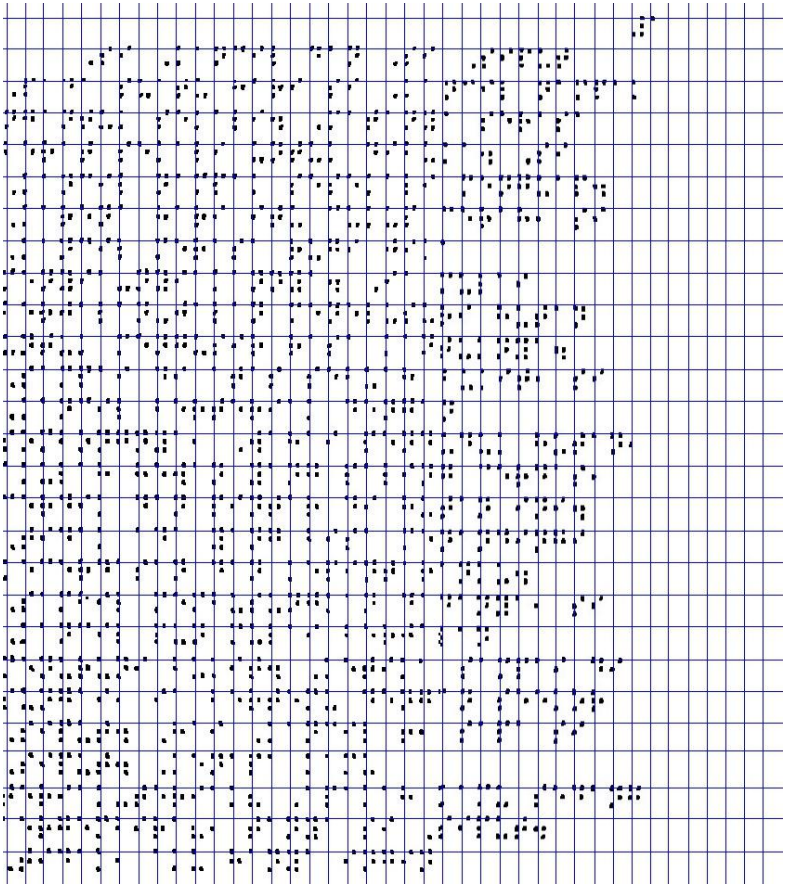
Langkah terakhir adalah melakukan *cropping* hasil penggabungan menjadi ukuran 840x950 piksel.



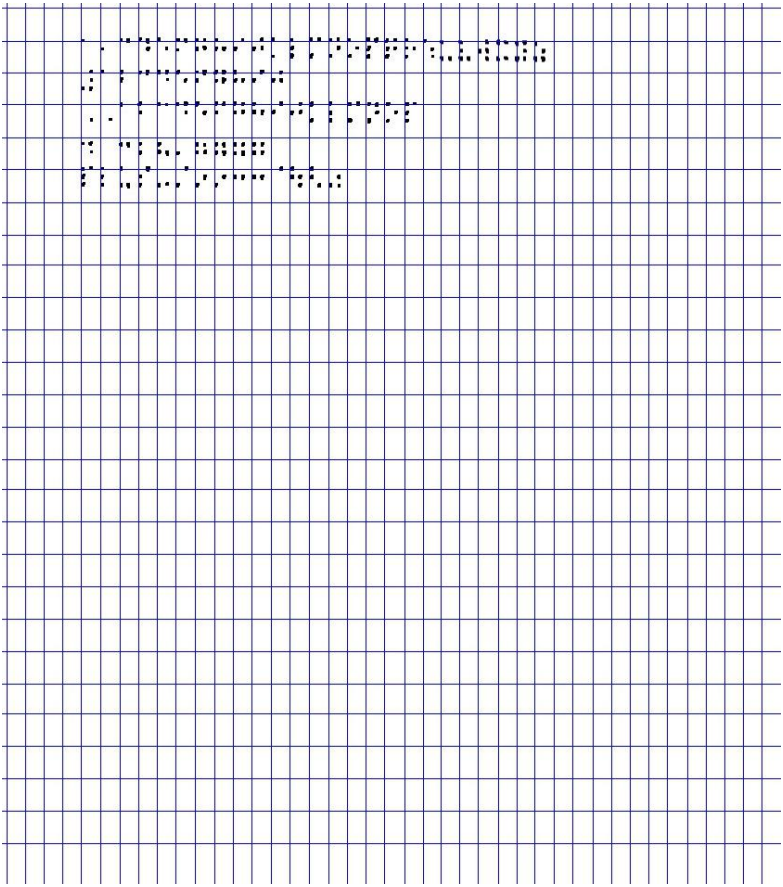
Gambar 4.21 Hasil akhir *image stitching*.

4.1.5 Segmentasi

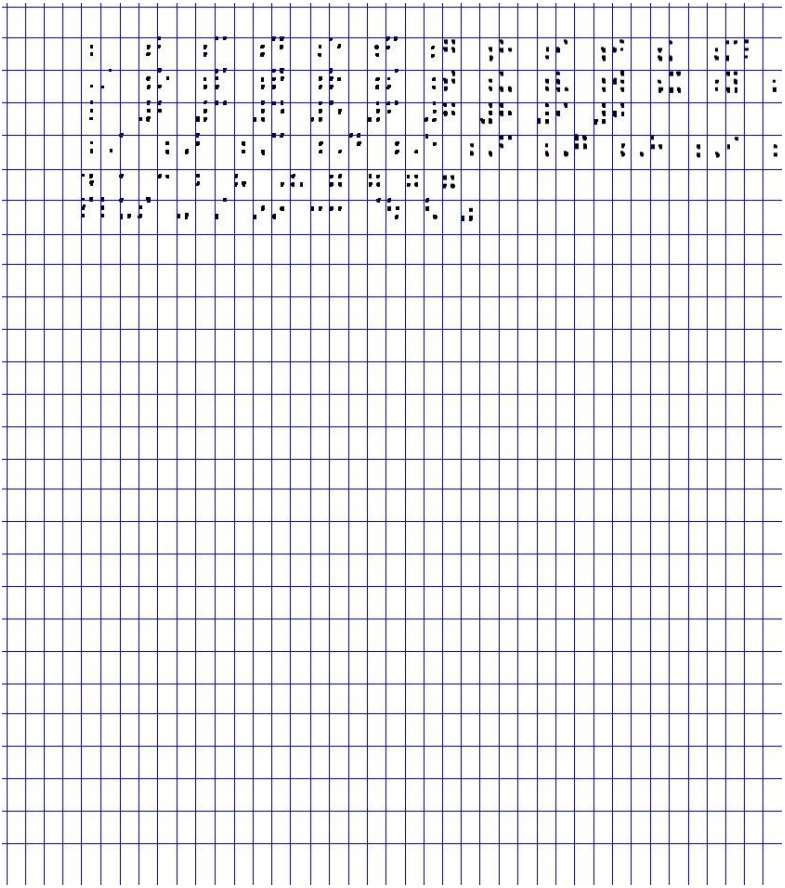
Algoritma yang digunakan pada proses segmentasi dapat memetakan karakter yang terdapat pada naskah dengan total baris sebanyak 27 baris dan total kolom sebanyak 41 kolom, selain itu algoritma pada proses segmentasi tidak memiliki sifat *adaptive* sehingga dibutuhkan pemberian nillai parameter yang tepat untuk mendapatkan hasil segmentasi yang baik.



Gambar 4.22 Hasil segmentasi.



Gambar 4.23 Hasil segmentasi.



Gambar 4.24 Hasil segmentasi.

Tabel 4.2 Nilai parameter *check point* segmentasi kolom

Gambar	Kolom						
	Origin1	*Origin2*	h	w	g	gh	a
4.22	(0,0)	(0,650)	300	13	20	22	5
4.23	(0,0)	(0,650)	300	13	20	22	5
4.24	(0,0)	(0,650)	300	13	20	22	5

Tabel 4.3 Nilai parameter *check point* segmentasi baris

Gambar	Baris							
	Origin	*Origin1*	*Origin2*	h	w	g	gh	a
4.22	(640,0)	(0,g)	(400,g)	20	200	35	32	5
4.23	(640,0)	(0,g)	(400,g)	13	200	35	32	5
4.24	(640,0)	(0,g)	(400,g)	5	200	35	32	4

Di mana:

- *Origin* = titik awal *check point* baris 1
- *Origin1* = titik awal *check point* 1
- *Origin* = titk awal *check point* 2
- h = tinggi area *check point*
- w = lebar area *check point*
- g = *check point step*
- gh = *check point step calibration*, (untuk kolom kalibrasi setiap 5 langkah dan baris setiap 7 langkah)
- a = nilai penambah jika *contour* sama dengan 1

Setelah didapatkan koordinat segmentasi selanjutnya dilakukan proses *cropping* untuk mengekstrak gambar setiap karakter dengan ukuran 20x30 piksel.

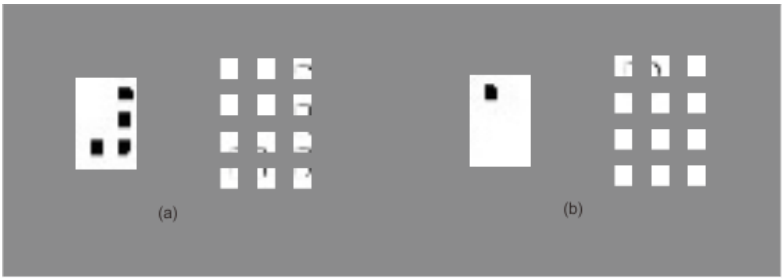
Gambar 4.25 Hasil *cropping* karakter gambar 4.22 baris ke 2.

4.2 Pengujian proses utama

Pada pengujian proses utama terdiri dari pengujian hasil operasi operasi lbp, inisiasi data learning, pengenalan karakter dan penerjemahan karakter.

4.2.1 Operasi local binary pattern

Sebelum dilakukan operasi *local binary pattern* pada gambar hasil *cropping*, dilakukan proses partisi terlebih dahulu pada gambar menjadi 12 bagian dengan ukuran 8x9 piksel, kemudian setiap bagian partisi tersebut dilakukan operasi lbp menjadi 6x7 piksel.



Gambar 4.26 Hasil operasi lbp pada beberapa potongan karakter. (a) karakter tanda angka, (b) karakter a atau atau 1.

4.2.2 Inisiasi data learning

Hasil inisiasi data *learning* merupakan array 4 dimensi yang berisi nilai histogram dari gambar karakter yang digunakan untuk data *learning*. Terdapat 129 gambar karakter yang digunakan untuk data *base* data *learning*, yang terdiri dari 43 jenis karakter yang setiap jenisnya memiliki 3 varian yang berbeda. Berikut 129 gambar data *base* yang digunakan untuk inisiasi data *learning*:

Gambar 4.27 Gambar data *base* data *learning.*

Setiap gambar karakter data *learning* dipartisi menjadi 12 bagian kemudian dilakukan operasi lbp, kemudian setiap partisi dihitung histogramnya dengan *bins range* dari 0-255. Array data *learning* inilah yang digunakan untuk membandingkan histogram pada setiap karakter objek.

4.2.3 Pengenalan karakter

Pada hasil pengenalan karakter, disajikan 3 buah hasil pembacaan naskah yang sama akan tetapi menggunakan 3 cara yang berbeda yaitu cara pertama dengan menggunakan metode *local binary pattern* dengan varian *data learning* 3 gambar setiap karakter, metode *local binary pattern* dengan varian *data learning* 6 gambar setiap karakter dan metode evaluasi piksel terprediksi yang dijelaskan secara singkat pada lampiran.

Hasil pengenalan karakter dengan 3 metode:

- Hasil pengenalan dengan metode LBP 3 *data learning:*

Hasil pengenalan	Pengenalan yang benar
,	1
Si Kanc,l can S, sisuj	Si Kancil dan Si Siput
Saca suatu haoi si Kancil nammak	Pada suatu hari si Kancil nampak
canuk sekal,. Kaanna seoasa	ngantuk sekali. Matanya serasa
bcla! sekali untuk eibkaa: Si	berat sekali untuk dibuka. Si
Kancil nakpak sesekali memguam.	Kancil nampak sesekali menguap.
Ka!ena hao, itu cukup ceoah, si	Karena hari itu cukup cerah, Si
Kanc,l keoasa ouji jika	kancil merasa rugi jika
kenoba-nniakanoya+- ,a ca`ai	menyia-nyiakannya. Ia mulai
beljalan-jalan keoebusuoi :atan	berjalan-jalan menelusuri hutan
untuk keohusio !asa kaoaayya-	untuk mengusir rasa kantuknya.
P kpai ci asas seauae baaij, mi	Sampai diatas sebuah bukit, si
Kancik beoeosak eeogan	Kancil berteriak dengan
sokboogoya, Wahai pemeadaa	sombongnya, Wahai penduduk
hatad`	hutan,
kulah hewao yaof pa,ing calda	akulah hewan yang paling cerdas,
ceod,k dao pbojao d, hutan ini	cerdik dan pintar di hutan ini.
Tidak ada yaog bisa kedandng/	Tidak ada yang bisa menandingi
kecereasam eam `eminta!ana -	kecerdasan dan kepintaranku.
Sakbib kepbusungkan dae nya *	Sambil membusungkan dadanya, si
/aoc,l l.o /ula, barialad	Kancil pun mulai berjalan
kenuus ,xk,. Kejila sammai di	menuruni bukit. Ketika sampai di
suofa,, ia behte/u dejan ieeaao	sungai, ia bertemu dengan seekor
Scp-tw Ha, Aaoc,l , sapa ii	Siput. Hai Kancil !, sapa si
simat. aedama aama	Siput, kenapa kamu
teo,a/-seo,am- Apakah aaca /adadg	teriak-teriak? Apakah kamu sedang
,e!gembira-, jaya s, Iimat+-	bergembira?, tanya si Siput.
J,dak, aku eaya iggim	Tidak, aku hanya ingin

- Hasil pengenalan dengan metode LBP 6 *data learning:*

Hasil pengenalan	Pengenalan yang benar
1	1
Si Kaocil can Si Simu!	Si Kancil dan Si Siput
Q ca su su aali si Kancil nammak ndanuk sckali: Kaanna seoasa belat sekali unuk eibuka: Si Kaoc,l nakpak sesekaki menguam.	Pada suatu hari si Kancil nampak ngantuk sekali. Matanya serasa berat sekali untuk dibuka. Si Kancil nampak sesekali menguap.
/aoena har, ,tu cukul cehah, si laqcil keoasa ouji jika /enyia-oyia/annna. Ia malai	Karena hari itu cukup cerah, Si kancil merasa rugi jika menyia-nyiakannya. Ia mulai
,eojapao-jakao menelusuoi hatan ugtu` menjviio oasa `anaamya-	berjalan-jalan menelusuri hutan untuk mengusir rasa kantuknya.
Sa/pai di atas sefuah bzait, si `aatcil behje!ia/ eengam	Sampai diatas sebuah bukit, si Kancil berteriak dengan
/ok,oohqya, Wahai meodadua hzjan,	sombongnya, Wahai penduduk hutan,
amulah hewao yaj mapigg cerdam,	akulah hewan yang paling cerdas,
ae!d,k dan pintao di hutan ini.	cerdik dan pintar di hutan ini.
S,daa ada yaog ,isa kenandingi /eceodaian dam kemintaranaa.	Tidak ada yang bisa menandingi kecerdasan dan kepintaranku.
/a/ail /embusunglan eadanya, mi /a/a,k puo mulai beojalan	Sambil membusungkan dadanya, si Kancil pun mulai berjalan
/eouruy, ,-k,j. Ketfka mampai di suogai, ia bertemu deqgan see`or `iipuj. Hai Kanfik :, iapa ti	menuruni bukit. Ketika sampai di sungai, ia bertemu dengan seekor Siput. Hai Kancil !, sapa si
/tmat- aayaa ma ter,ak-teoiaa- Apakah /am- iedaog	Siput, kenapa kamu teriak-teriak? Apakah kamu sedang
aeogem,ioa-, anya mi sim`t+-	bergembira?, tanya si Siput.
",,dak, aau haya ihgin	Tidak, aku hanya ingin

- Hasil pengenalan dengan metode evaluasi piksel terprediksi:

Hasil pengenalan	Pengenalan yang benar
1	1
#si Laqcil dan Si Siput	Si Kancil dan Si Siput
#pada s#atz hari si Kancil nampak qgaqtzl sekali. #matanya serasa berat sekali untuk dibuka. Si Kancil nampak sesekali menguap. Karena hari itu cukup cerah, si kancil merasa rugi jika menyia-nyiakannya. Ia mulai berjalan-jalan menelusuri hutan untuk mengusir rasa kant-lq#b(Sampai di atas sebuah bukit, si `kancil berteriak dengan lombongnya, Wahai penduduk hutan, akulah hewan yang paling cerdas, aerdik dan pintar di hutan ini. Tidak ada yang bisa menandingi kecerdasan dan kepintaranku. Sapbil pepbusungkan dadanya, ti Kaqcil pun mulai berjalan menuruni bukit. Ket,ka sampai di #ungai, ia bertemu daoban seekor `siput. Hai Kancil !, sapa si `!tqv#? Lb)b!, l,rv bbriak-seriak? Apakah kamu sedang brbemb,la?, tanya si Sipkt# -bicak, aku hanya ingin	Pada suatu hari si Kancil nampak ngantuk sekali. Matanya serasa berat sekali untuk dibuka. Si Kancil nampak sesekali menguap. Karena hari itu cukup cerah, Si kancil merasa rugi jika menyia-nyiakannya. Ia mulai berjalan-jalan menelusuri hutan untuk mengusir rasa kantuknya. Sampai diatas sebuah bukit, si Kancil berteriak dengan sombongnya, Wahai penduduk hutan, akulah hewan yang paling cerdas, cerdik dan pintar di hutan ini. Tidak ada yang bisa menandingi kecerdasan dan kepintaranku. Sambil membusungkan dadanya, si Kancil pun mulai berjalan menuruni bukit. Ketika sampai di sungai, ia bertemu dengan seekor Siput. Hai Kancil !, sapa si Siput, kenapa kamu teriak-teriak? Apakah kamu sedang bergembira?, tanya si Siput. Tidak, aku hanya ingin

Tabel 4.4 Hasil pengenalan karakter

Metode	Total karakter	Benar	Salah	Persentase kebenaran
LBP 3 *data learning*	1107	884	223	79,855 %
LBP 6 *data learning*	1107	911	196	82,294 %
Evaluasi piksel terprediksi	1107	1045	62	94,399 %

Dari hasil yang didapat terlihat bahwa semakin banyak *data learning* yang ditanamkan persentasi kebenaran semakin besar akan tetapi memrlukan waktu komputasi yang lebih lama. Jika dibandingkan dengan metode pembanding evaluasi piksel terprediksi waktu komputasi pada metode evaluasi piksel terprediksi jauh lebih singkat dengan persentase kebenaran paling tinggi.

Faktor *error* yang paling besar adalah hasil segmentasi gambar perkarakter yang kurang baik, yaitu posisi karakter pada gambar yang tidak tepat berada pada titik tengah sehingga gambar dikenali sebagai karakter yang lain, salah satu error yang terjadi pada hasil pengenalan menggunakan LBP dengan 3 *data learning* karakter baris pertama yang berisi nomor halaman, karakter yang seharusnya dikenalai sebagai angka 1 dikenali sebagai tanda baca koma (,) yang mana kedua karakter ini memiliki hanya satu titik akan tetapi dengan posisi yang berbeda.

Tabel 4.5 Analisa *error* hasil pengenalan

Hasil	Karakter	Data learning dikenal			Data learning benar		
Segmentasi							
LBP							
Nilai pengenalan	2	2	2	2	1	1	1
Pengenalan	,	,	,	,	a;1	a;1	a;1

Pada tabel diatas menunjukkan bahwa karakter angka 1 dikenali sebagai (,) karena posisi karakter yang sedikit ke bawah pada hasil

segmentasi yang menghasilkan karakter seperti karakter (,), sehingga dikenali sebagai (,) bukan angka 1.

BAB V
PENUTUP

5.1 Kesimpulan

Pada tugas akhir ini dirancang sebuah purnarupa alat pengenal karakter braille yang secara garis bersar dapat dibedakan menjadi dua sistem utama yaitu sistem perengkat keras dan perangkat lunak, pada sistem perangkat keras dirancang suatu model kerangka alat untuk mengabil gambar dari objek atau naskah braille, kerangka alat dirancang dari dua komponen utama yaitu komponen mekanik dan elektronik. Pada sistem perangkat lunak dirancang dua program utama yaitu program pengolah gambar pada tahap *preprocessing* dan tahap proses utama. Tahap *preprocessing* terdiri dari proses pengambilan gambar, *thresholding, morphology* dan *image stitching.* Tahap proses utama terdiri dari proses segmetasi, ekstraksi gambar karakter, operasi *local binary pattern*, inisiasi *data learning* dan pengenalan karakter dengan *chi-square.* Hasil dari tugas akhir ini menunjukkan bahwa dengan menggunakan metode LBP untuk mendapatkan hasil yang baik maka dibutuhkan entri *data learning* yang cukup banyak dan berefek pada tempo kalkulasi yang dibutuhkan menjadi lebih lama, sedangkan hasil dari metode yang tidak menyertakan LBP dan *data leraning* menunjukkan hasil yang lebih baik dan tempo kalkulasi yang lebih cepat. Dengan metode LBP menggunakan 3 varian *data learning* untuk setiap karakternya didapatkan persentase pengenalan karakter benar sebesar 79,855 %, dengan metode LBP menggunkanan 6 *data learning* untuk setiap karakternya didapat 82,294 %, sedangkan dengan metode lain atau metode evaluasi piksel terprediksi didapatkan hasil sebesar 94,399 %. Hal ini menunjukkan bahwa performa metode lain yang digunakan sebagai pembanding relatif lebih baik dibanding metode LBP. Dan dapat disimpulkan bahwa untuk pengenalan karakter braille tidak perlu menggunakan metode yang tergolong kompleks, cukup dengan metode sederhana karena tekstur karakter braille sendiri yang mudah untuk dibedakan.

5.2 Saran

1. Gunakan kamera yang memiliki permukaan datar, agar posisi kamera menjadi lebih baik dan gambar yang dihasilkan memiliki kesejajaran yang lebih baik.

2. Perbaikan algoritma pada proses segmentasi, agar hasil segmentasi perkarakter tepat berada di tengah.
3. Peningkatan performa algoritma pada tahap *preprocessing*, agar gambar yang akan dikenali meiliki kualitas yang baik.

DAFTAR PUSTAKA

[1] 2012. *The Raspberry Pi Education Manual*.

[2] Bradski, Gary. Adrian Kaehler. 2008. *Learning OpenCV Computer Vision with the OpenCV Library*, California: O'Reilly Media, Inc.

[3] Pietikanen, Matti. Abdenour Hadid. Guoying Zhao. Timo Ahonen. 2011. *Computer Vision Using Local Binary Patterns.* Chester: Spinger.

[4] Silva, Chamalee. Chandana Srilal. Harsha Athapaththu. Ranathynga. 2016. *Novel Segmentation Method for Optical Braille Character Identification.* University of Moratuwa: EEECOS paper.

[5] Subagya. *Menulis-Membaca Huruf Braille Tingkat Dasar.*

[6] Szeliski, Richard. 2011. *Computer Vision: Algorithms and Applications.* Chester: Spinger.

[7] Upton, Eben. Gareth Halfacree. 2012. *Raspberry Pi User Guide*. Chichester: A Jhon Wiley and Sons Publication.

Halaman ini sengaja dikosongkan

LAMPIRAN

- PROGRAM *PREPROCESSING*:

```
#include <wiringPi.h>
#include <softPwm.h>
#include <highgui.h>
#include <iostream>
using namespace std;

const int led1 = 25;
const int led2 = 29;
const int switch_1 = 22;
const int switch_2 = 21;
const int ms1 = 7;
const int ms2 = 2;
const int ms3 = 3;
const int enable = 4;
const int step = 5;
const int dir = 6;

int main()
{
//====(CAPTURE IMAGES OF BRAILLE TEXT)====
CvCapture *jepret_1 = cvCaptureFromCAM(0);
CvCapture *jepret_2 = cvCaptureFromCAM(1);
IplImage* img;
int i;

wiringPiSetup();
pinMode(led1,OUTPUT);
pinMode(led2,OUTPUT);
pinMode(switch_1,INPUT);
pinMode(switch_2,INPUT);
pinMode(ms1,OUTPUT);
pinMode(ms2,OUTPUT);
pinMode(ms3,OUTPUT);
pinMode(enable,OUTPUT);
```

```
pinMode(step,OUTPUT);
pinMode(dir,OUTPUT);
pullUpDnControl(switch_1,PUD_UP);
pullUpDnControl(switch_2,PUD_UP);
digitalWrite(ms1,HIGH);
digitalWrite(ms2,LOW);
digitalWrite(ms3,LOW);
digitalWrite(enable,LOW);
digitalWrite(led1,LOW);
digitalWrite(led2,LOW);

digitalWrite(dir,LOW);
for(;digitalRead(switch_1)==HIGH;)
{
digitalWrite(step,HIGH);delay(1);
digitalWrite(step,LOW); delay(1);
}
digitalWrite(led2,HIGH);
for(i=0;i<10;i++)
{
img = cvQueryFrame(jepret_1);
}
cvSaveImage("1_hasil_capture/1.jpg",img);
digitalWrite(led2,LOW);
digitalWrite(led1,HIGH);
for(i=0;i<10;i++)
{
img = cvQueryFrame(jepret_2);
}
cvSaveImage("1_hasil_capture/2.jpg",img);
digitalWrite(led1,LOW);

digitalWrite(dir,HIGH);
for(i=0;i<=950;i++)
{
digitalWrite(step,HIGH);delay(1);
digitalWrite(step,LOW); delay(1);
}
digitalWrite(led2,HIGH);
```

```
for(i=0;i<10;i++)
{
img = cvQueryFrame(jepret_1);
}
cvSaveImage("1_hasil_capture/3.jpg",img);
digitalWrite(led2,LOW);
digitalWrite(led1,HIGH);
for(i=0;i<10;i++)
{
img = cvQueryFrame(jepret_2);
}
cvSaveImage("1_hasil_capture/4.jpg",img);
digitalWrite(led1,LOW);
digitalWrite(led1,LOW);

for(;digitalRead(switch_2)==HIGH;)
{
digitalWrite(step,HIGH);delay(1);
digitalWrite(step,LOW); delay(1);
}
digitalWrite(led2,HIGH);
for(i=0;i<10;i++)
{
img = cvQueryFrame(jepret_1);
}
cvSaveImage("1_hasil_capture/5.jpg",img);
digitalWrite(led2,LOW);
digitalWrite(led1,HIGH);
for(i=0;i<10;i++)
{
img = cvQueryFrame(jepret_2);
}
cvSaveImage("1_hasil_capture/6.jpg",img);
digitalWrite(led1,LOW);

digitalWrite(dir,LOW);
for(i=0;i<=1800;i++)
{
digitalWrite(step,HIGH);delay(1);
```

```
digitalWrite(step,LOW); delay(1);
}
digitalWrite(enable,HIGH);
cvReleaseCapture(&jepret_1);
cvReleaseCapture(&jepret_2);
cout <<"--images capturing has completed--"<<endl;

//==(CROPPING ROI, THRESHOLDING, MORPHOLOGY)==
char hasil_capture[512];
char hasil_cropping[512];
char hasil_thresholding[512];
char hasil_morphologyDLT[512];
char hasil_morphologyERD[512];
int x_roi[6]={151,163,151,163,151,163};
int y_roi[6]={55,40,103,88,58,40};
int w_roi[6]={471,420,471,420,471,420};
int h_roi[6]={410,406,350,345,340,325};
IplImage* img_grey;
IplImage* img_thres;
IplImage* img_dilate;
IplImage* img_erode;
for(i=0;i<6;i++)
{
snprintf(hasil_capture,512,"1_hasil_capture/%01d.jpg",i+1);
snprintf(hasil_cropping,512,"2_hasil_cropping/%01d.jpg",i+1);
snprintf(hasil_thresholding,512,"3_hasil_thresholding/%01d.jpg",i+1);
snprintf(hasil_morphologyDLT,512,"4_hasil_morphology/dilate%01d.j
pg",i+1);
snprintf(hasil_morphologyERD,512,"4_hasil_morphology/erode%01d.j
pg",i+1);
img_grey=cvCreateImage(cvSize(w_roi[i],h_roi[i]),8,1);
img_thres=cvCreateImage(cvSize(w_roi[i],h_roi[i]),8,1);
img_dilate= cvCreateImage(cvSize(w_roi[i],h_roi[i]),8,1);
img_erode = cvCreateImage(cvSize(w_roi[i],h_roi[i]),8,1);
img=cvLoadImage(hasil_capture,CV_LOAD_IMAGE_ COLOR);
cvSetImageROI(img,cvRect(x_roi[i],y_roi[i],w_roi[i],h_roi[i]));
cvSaveImage(hasil_cropping,img);
cvCvtColor(img,img_grey,CV_BGR2GRAY);
```

```
cvAdaptiveThreshold(img_grey,img_thres,255,CV_ADAPTIVE_THRE
SH_MEAN_C,CV_THRESH_BINARY,11,6);
cvSaveImage(hasil_thresholding,img_thres);
cvDilate(img_thres,img_dilate,NULL,1);
cvErode(img_dilate,img_erode,NULL,1);
cvSaveImage(hasil_morphologyDLT,img_dilate);
cvSaveImage(hasil_morphologyERD,img_erode);
}
cvReleaseImage(&img_grey);
cvReleaseImage(&img_thres);
cvReleaseImage(&img_dilate);
cvReleaseImage(&img_erode);
cout <<"--cropping, thresholding and morphology have completed--
"<<endl;

//===(MAKING TEMPLATE AND ROI IMAGES)===
char hasil_tmpROI[512];
int x_tmp[8]={0,0,0,0,0,0,0,0};
int y_tmp[8]={375,371,315,310,0,0,0,0};
int w_tmp[8]={471,420,471,420,471,420,471,420};
int h_tmp[8]={35,35,35,35,40,40,40,40};
int src_tmp[8]={1,2,3,4,3,4,5,6};
int res_tmp[8]={1,2,3,4,1,2,3,4};
for(i=0;i<8;i++)
{
snprintf(hasil_morphologyERD,512,"4_hasil_morphology/erode%01d.j
pg",src_tmp[i]);
if(i<4)
{
snprintf(hasil_tmpROI,512,"5_hasil_tmpROI/tmp%01d.jpg",res_tmp[i])
;
}
if(i>3)
{
snprintf(hasil_tmpROI,512,"5_hasil_tmpROI/roi%01d.jpg",res_tmp[i]);
}
img=cvLoadImage(hasil_morphologyERD,CV_LOAD_IMAGE_COLO
R);
cvSetImageROI(img,cvRect(x_tmp[i],y_tmp[i],w_tmp[i],h_tmp[i]));
```

```
cvSaveImage(hasil_tmpROI,img);
}
cout <<"--template and ROI have completed--"<<endl;
//====(BUILD HALF-WHOLE IMAGES)====
char hasil_halfwhole[512];
IplImage* tmp;
IplImage* roi;
IplImage* res;
IplImage* img1;
IplImage* img2;
CvPoint minloc, maxloc;
double minval, maxval;
for(i=0;i<4;i++)
{
snprintf(hasil_tmpROI,512,"5_hasil_tmpROI/tmp%01d.jpg",i+1);
tmp=cvLoadImage(hasil_tmpROI,CV_LOAD_IMAGE_GRAYSCALE)
;
snprintf(hasil_tmpROI,512,"5_hasil_tmpROI/roi%01d.jpg",i+1);
roi=cvLoadImage(hasil_tmpROI,CV_LOAD_IMAGE_GRAYSCALE);
if(i<2)
{
snprintf(hasil_morphologyERD,512,"4_hasil_morphology/erode%01d.j
pg",i+1);
}
if(i>1)
{
snprintf(hasil_morphologyERD,512,"6_hasil_halfwhole/%01d.jpg",i-1);
}
img1=cvLoadImage(hasil_morphologyERD,CV_LOAD_IMAGE_GRA
YSCALE);
snprintf(hasil_morphologyERD,512,"4_hasil_morphology/erode%01d.j
pg",i+3);
img2=cvLoadImage(hasil_morphologyERD,CV_LOAD_IMAGE_GRA
YSCALE);
res=cvCreateImage(cvSize(roi->width-tmp->width+1,roi->height-tmp->
height+1),IPL_DEPTH_32F,1);
cvMatchTemplate(roi,tmp,res,CV_TM_SQDIFF);
cvMinMaxLoc(res,&minval,&maxval,&minloc,&maxloc,0);
```

```
img=cvCreateImage(cvSize(img1->width,(img1->height+img2->height)
-(minloc.y+tmp->height)),8,1);
for(int y=0;y<img->height;y++)
{
for(int x=0;x<img->width;x++)
if(y<img1->height)
{
CV_IMAGE_ELEM(img,uchar,y,x)=CV_IMAGE_ELEM(img1,uchar,y
,x);
}
else
{
CV_IMAGE_ELEM(img,uchar,y,x)=CV_IMAGE_ELEM(img2,uchar,(
y-img1->height)+(minloc.y+tmp->height),x);
}
}
snprintf(hasil_halfwhole,512,"6_hasil_halfwhole/%01d.jpg",i+1);
cvSaveImage(hasil_halfwhole,img);
}
cout<<"--half-whole image has completed--"<<endl;

//====(BUILD WHOLE-IMAGE)====
img=cvLoadImage("6_hasil_halfwhole/3.jpg",CV_LOAD_IMAGE_CO
LOR);
cvSetImageROI(img,cvRect(0,25,471,950));
cvSaveImage("7_hasil_wholeimage/left_halfwhole.jpg",img);
cvSetImageROI(img,cvRect(448,0,23,950));
cvSaveImage("7_hasil_wholeimage/tmp.jpg",img);
img1=cvLoadImage("6_hasil_halfwhole/4.jpg",CV_LOAD_IMAGE_C
OLOR);
cvSetImageROI(img1,cvRect(0,25,420,942));
cvSaveImage("7_hasil_wholeimage/right_halfwhole_unresized.jpg",img
1);
img=cvCreateImage(cvSize(420,950),8,1);
img1=cvLoadImage("7_hasil_wholeimage/right_halfwhole_unresized.jp
g",CV_LOAD_IMAGE_GRAYSCALE);
cvResize(img1,img,CV_INTER_LINEAR);
cvSaveImage("7_hasil_wholeimage/right_halfwhole.jpg",img);
```

```
img=cvLoadImage("7_hasil_wholeimage/right_halfwhole.jpg",CV_LO
AD_IMAGE_COLOR);
cvSetImageROI(img,cvRect(19,0,28,950));
cvSaveImage("7_hasil_wholeimage/roi.jpg",img);
tmp=cvLoadImage("7_hasil_wholeimage/tmp.jpg",CV_LOAD_IMAGE
_GRAYSCALE);
roi=cvLoadImage("7_hasil_wholeimage/roi.jpg",CV_LOAD_IMAGE_
GRAYSCALE);
img1=cvLoadImage("7_hasil_wholeimage/left_halfwhole.jpg",CV_LO
AD_IMAGE_GRAYSCALE);
img2=cvLoadImage("7_hasil_wholeimage/right_halfwhole.jpg",CV_L
OAD_IMAGE_GRAYSCALE);
res=cvCreateImage(cvSize(roi->width-tmp->width+1,roi->height-tmp-
>height+1),IPL_DEPTH_32F,1);
cvMatchTemplate(roi,tmp,res,CV_TM_SQDIFF);
cvMinMaxLoc(res,&minval,&maxval,&minloc,&maxloc,0);
img=cvCreateImage(cvSize((img1->width+img2->width)-(minloc.x+
tmp->width)-19,img1->height),8,1);
for(int x=0;x<img->width;x++)
{
for(int y=0;y<img->height;y++)
if(x<img1->width)
{
CV_IMAGE_ELEM(img,uchar,y,x)=CV_IMAGE_ELEM(img1,uchar,y
,x);
}
else
{
CV_IMAGE_ELEM(img,uchar,y,x)=CV_IMAGE_ELEM(img2,uchar,y
,(x-img1->width)+(minloc.x+tmp->width+19));
}
}
cvSaveImage("7_hasil_wholeimage/whole_image.jpg",img);
img=cvLoadImage("7_hasil_wholeimage/whole_image.jpg",CV_LOAD
_IMAGE_COLOR);
cvSetImageROI(img,cvRect(0,0,840,950));
cvSaveImage("final image.jpg",img);
cvReleaseImage(&img1);
cvReleaseImage(&img2);
```

```
cvReleaseImage(&tmp);
cvReleaseImage(&roi);
cvReleaseImage(&res);
cout<<"--final image.jpg has been built--"<<endl;

//====(DISPLAY FINAL IMAGE.JPG)====
cout<<"--displaying final image.jpg--"<<endl;
cvNamedWindow("final image.jpg",CV_WINDOW_AUTOSIZE);
cvShowImage("final image.jpg",img);
cvWaitKey(0);
cvReleaseImage(&img);
cvDestroyWindow("final image.jpg");
cout<<"--preprocessing has completed--"<<endl;

return 0;
}
```

- PROGRAM PREPARASI *DATA LEARNING*:

```
#include <highgui.h>
#include <cv.h>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc.hpp>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <math.h>
using namespace std;

int main()
{
IplImage* img;
IplImage* img_part;
IplImage* img_lbp;
img_lbp=cvCreateImage(cvSize(6,7),8,1);
int roi_x[12]={0,6,12,0,6,12,0,6,12,0,6,12};
int roi_y[12]={0,0,0,7,7,7,14,14,14,21,21,21};
int lbp_nb[8];
```

```
int aa=0;
int b=0;
int cc;
char character_images[512];
char lbp_images[512];
for(int i=1;i<130;i++)
{
snprintf(character_images,512,"data_learning_images/%03d.jpg",i);
img_part=cvLoadImage(character_images,CV_LOAD_IMAGE_GRAY
SCALE);
cvThreshold(img_part,img_part,10,255,CV_THRESH_BINARY);
for(int ii=0;ii<12;ii++)
{
cvSetImageROI(img_part,cvRect(roi_x[ii],roi_y[ii],8,9));
for(int y=1;y<8;y++)
{
for(int x=1;x<7;x++)
{
int center=CV_IMAGE_ELEM(img_part,uchar,y+roi_y[ii],x+roi_x[ii]);
lbp_nb[0]=CV_IMAGE_ELEM(img_part,uchar,y-1+roi_y[ii],x-1+
roi_x[ii]);
lbp_nb[1]=CV_IMAGE_ELEM(img_part,uchar,y-1+roi_y[ii],x+
roi_x[ii]);
lbp_nb[2]=CV_IMAGE_ELEM(img_part,uchar,y-1+roi_y[ii],x+1+
roi_x[ii]);
lbp_nb[3]=CV_IMAGE_ELEM(img_part,uchar,y+roi_y[ii],x+1+
roi_x[ii]);
lbp_nb[4]=CV_IMAGE_ELEM(img_part,uchar,y+1+roi_y[ii],x+1+
roi_x[ii]);
lbp_nb[5]=CV_IMAGE_ELEM(img_part,uchar,y+1+roi_y[ii],x+
roi_x[ii]);
lbp_nb[6]=CV_IMAGE_ELEM(img_part,uchar,y+1+roi_y[ii],x-1+
roi_x[ii]);
lbp_nb[7]=CV_IMAGE_ELEM(img_part,uchar,y+roi_y[ii],x-1+
roi_x[ii]);
for(int iii=0;iii<8;iii++)
{
if (lbp_nb[iii]<center)
{
```

```
lbp_nb[iii]=0;
}
else
{
lbp_nb[iii]=1;
}
}
center=lbp_nb[7]*128+lbp_nb[6]*64+lbp_nb[5]*32+lbp_nb[4]*16+lbp
_nb[3]*8+lbp_nb[2]*4+lbp_nb[1]*2+lbp_nb[0]*1;
CV_IMAGE_ELEM(img_lbp,uchar,y-1,x-1)=center;
}
}
cc=ii;
snprintf(lbp_images,512,"lbp data learning images/%02d%01d%02d.jpg
",aa,b,cc);
cvSaveImage(lbp_images,img_lbp);
}
b++;
if(i%3==0)
{
aa++;
b=0;
}
}
cout <<"--data learning has completed"<<endl;

return 0;
}
```

- PROGRAM UTAMA (SEGMENTASI, LBP, PENGENALAN KARAKTER DAN TRANSLASI):

```
#include <highgui.h>
#include <cv.h>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/core/mat.hpp>
#include <opencv2/core/types_c.h>
```

```
#include <opencv2/core/core_c.h>
#include <opencv2/highgui.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include <math.h>
using namespace std;
using namespace cv;

class FindCharacter{
IplImage* img_char=cvLoadImage("final image.jpg",CV_LOAD_IMA
GE_GRAYSCALE);
char path_file_name[512];

public:
void PathFileName(int counter)
{
snprintf(path_file_name,512,"proses_2_charactercropping/%04d.jpg",co
unter);
}
void FindBraille(int x,int y)
{
if(x<5)
{
cvSetImageROI(img_char,cvRect(0,y-5,20,30));
}
if(x>=5)
{
cvSetImageROI(img_char,cvRect(x-5,y-5,20,30));
}
cvSaveImage(path_file_name,img_char);
}
void releaseIplImage(){cvReleaseImage(&img_char);
}};

class table{
string translation_table[6][43]=
{
```

```
{" ","a",",","b","`","k","l","c","i","f","/","m","s","p","e",":","h",
 "","o","!","r","d","j","g","","n","t","q","","+-","-","u","","v",
 "","x",".","","z","","w","","y"},
{" ","A",",","B","#","K","L","C","I","F","/","M","S","P","E",":","H",
 "","O","!","R","D","J","G","","N","T","Q","","+-","-","U","#","V",
 "","X",".","","Z","","W","","Y"},
{" ","A",",","B","`","K","L","C","I","F","/","M","S","P","E",":","H",
 "","O","!","R","D","J","G","","N","T","Q","","+-","-","U","?","V",
 "","X",".","","Z","","W","","Y"},
{" ","A",",","B","`","K","L","C","I","F","/","M","S","P","E",":","H",
 "","O","!","R","D","J","G","","N","T","Q","","+-","-","U","","V",
 "","X",".","","Z","","W","","Y"},
{" ","1",",","2","`","#","#","3","9","6","/","#","#","#","5",":","8",
 "","#","!","#","4","0","7","","#","#","#","","+-","-","#","?","#",
 "","#",".","","#","","#","","#"},
 {"#","A","#","B","#","K","L","C","I","F","#","M","S","P","E","#","H",
 "","O","#","R","D","J","G","#","N","T","Q","#","#","#","U","","V",
 "#","X","#","#","Z","#","W","#","Y"}
};

int state_generator[6][43]={
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
0,0,0,4,0},
{0,5,0,5,0,5,5,5,5,5,0,5,5,5,5,0,5,0,5,0,5,5,5,5,0,5,5,5,2,0,0,5,5,5,0,5,0,0,
5,0,5,4,5},
{0,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,3,2,2,2,2,2,2,2,2,2,
2,2,2,4,2},
{3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,1,3,3,3,3,3,3,3,3,3,
3,3,3,4,3},
{0,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,4,
4,4,4,4,4},
{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,
0,0,0,4,0}
};

public:
int state=0;
int p_state=0;
string translate(int karakter) {
```

```
                        state=state_generator[p_state][karakter];
                        p_state=state;
                        return translation_table[state][karakter];
                       }
};

int main()
{
//====(SEGMENTATION)===
int x[41],y[27],x_roi,y_roi,x_roi1,y_roi1,nc,ncc,k,i,h,g,gh,a;
float cy,cx,sum_cy,sum_cx,m,m1;
IplImage* img;
IplImage* sample;
IplImage* grey;
IplImage* bin;
CvMemStorage* storage;
storage=cvCreateMemStorage();
CvSeq* first_contour;
CvSeq* c;
CvRect boundbox;

h=13;
g=35;
gh=32;
a=5;

img=cvLoadImage("final image.jpg",CV_LOAD_IMAGE_COLOR);
sample=cvLoadImage("final  image.jpg",CV_LOAD_IMAGE_COLOR)
;
grey=cvCreateImage(cvSize(200,h),8,1);
bin=cvCreateImage(cvGetSize(grey),8,1);

//===--mulai y--===
i=0;
x_roi=640;
y_roi=0;
//===--kanan atas (halaman)--===
cvSetImageROI(sample,cvRect(x_roi,y_roi,200,h));
cvCvtColor(sample,grey,CV_BGR2GRAY);
```

```
cvThreshold(grey,bin,10,255,CV_THRESH_BINARY);
nc=cvFindContours(bin,storage,&first_contour,sizeof(CvContour),CV_
RETR_LIST);
cvResetImageROI(sample);
if(nc!=1)
{
c=first_contour;
sum_cy=0;
for(k=0;k<(nc-1);k++)
{
boundbox=cvBoundingRect(c);
cy=(y_roi+boundbox.y) + (boundbox.height/2);
c=c->h_next;
sum_cy=sum_cy+cy;
}
m=sum_cy/(nc-1);
y[i]=round(m);
i=i+1;
y_roi=y_roi+g;
}
else
{
m=y_roi+a;
y[i]=m;
i=i+1;
y_roi=y_roi+g;
}

//===--baris kedua--===
//===--kiri--===
x_roi=400;
x_roi1=0;
y_roi1=y_roi;
for(;i<27;i++)
{
cvSetImageROI(sample,cvRect(x_roi1,y_roi1,200,h));
cvCvtColor(sample,grey,CV_BGR2GRAY);
cvThreshold(grey,bin,10,255,CV_THRESH_BINARY);
```

```
nc=cvFindContours(bin,storage,&first_contour,sizeof(CvContour),CV_
RETR_LIST);
cvResetImageROI(sample);
if(nc!=1)
{
c=first_contour;
sum_cy=0;
for(k=0;k<(nc-1);k++)
{
boundbox=cvBoundingRect(c);
cy=(y_roi1+boundbox.y)+(boundbox.height/2);
c=c->h_next;
sum_cy=sum_cy+cy;
}
m1=sum_cy/(nc-1);
if(i%7==0)
{
y_roi1=y_roi1+gh;
}
else
{
y_roi1=y_roi1+g;
}
}
else
{
m1=y_roi1+a;
if(i%7==0)
{
y_roi1=y_roi1+gh;
}
else
{
y_roi1=y_roi1+g;
}
}

//===--kanan--===
cvSetImageROI(sample,cvRect(x_roi,y_roi,200,h));
```

```
cvCvtColor(sample,grey,CV_BGR2GRAY);
cvThreshold(grey,bin,10,255,CV_THRESH_BINARY);
nc=cvFindContours(bin,storage,&first_contour,sizeof(CvContour),CV_
RETR_LIST);
cvResetImageROI(sample);
if(nc!=1)
{
c=first_contour;
sum_cy=0;
for(k=0;k<(nc-1);k++)
{
boundbox=cvBoundingRect(c);
cy=(y_roi+boundbox.y)+(boundbox.height/2);
c=c->h_next;
sum_cy=sum_cy+cy;
}
m=sum_cy/(nc-1);
if(i%7==0)
{
y_roi=y_roi+gh;
}
else
{
y_roi=y_roi+g;
}
}
else
{
m=y_roi+a;
if(i%7==0)
{
y_roi=y_roi+gh;
}
else
{
y_roi=y_roi+g;
}
}
m=(m+m1)/2;
```

```
y[i]=round(m);
}

//===--mulai x--===
h=13;
g=20;
gh=22;
a=5;
grey=cvCreateImage(cvSize(h,300),8,1);
bin=cvCreateImage(cvGetSize(grey),8,1);

x_roi=0;
y_roi=650;
x_roi1=0;
y_roi1=0;
for(i=0;i<41;i++)
{
//===--atas--===
cvSetImageROI(sample,cvRect(x_roi1,y_roi1,h,300));
cvCvtColor(sample,grey,CV_BGR2GRAY);
cvThreshold(grey,bin,10,255,CV_THRESH_BINARY);
nc=cvFindContours(bin,storage,&first_contour,sizeof(CvContour),CV_
RETR_LIST);
cvResetImageROI(sample);
if(nc!=1)
{
c=first_contour;
sum_cx=0;
for(k=0;k<(nc-1);k++)
{
boundbox=cvBoundingRect(c);
cx=(x_roi1+boundbox.x)+(boundbox.width/2);
c=c->h_next;
sum_cx=sum_cx+cx;
}
m1=sum_cx/(nc-1);
if(i%5==0)
{
x_roi1=x_roi1+gh;
```

```
}
else
{
x_roi1=x_roi1+g;
}
}
else
{
m1=x_roi1+a;
if(i%5==0)
{
x_roi1=x_roi1+gh;
}
else
{
x_roi1=x_roi1+g;
}
}

//===--bawah--===
if(i==0)
{
grey=cvCreateImage(cvSize(8,300),8,1);
bin=cvCreateImage(cvGetSize(grey),8,1);
cvSetImageROI(sample,cvRect(x_roi,y_roi,8,300));
cvCvtColor(sample,grey,CV_BGR2GRAY);
cvThreshold(grey,bin,10,255,CV_THRESH_BINARY);
nc=cvFindContours(bin,storage,&first_contour,sizeof(CvContour),CV_
RETR_LIST);
cvResetImageROI(sample);
if(nc!=1)
{
c=first_contour;
sum_cx=0;
for(k=0;k<(nc-1);k++)
{
boundbox=cvBoundingRect(c);
cx=(x_roi+boundbox.x)+(boundbox.width/2);
c=c->h_next;
```

```
sum_cx=sum_cx+cx;
}
m=sum_cx/(nc-1);
x_roi=x_roi+18;
}
else
{
m=x_roi+4;
x_roi=x_roi+18;
}
grey=cvCreateImage(cvSize(h,300),8,1);
bin=cvCreateImage(cvGetSize(grey),8,1);
}
else
{
cvSetImageROI(sample,cvRect(x_roi,y_roi,h,300));
cvCvtColor(sample,grey,CV_BGR2GRAY);
cvThreshold(grey,bin,10,255,CV_THRESH_BINARY);
nc=cvFindContours(bin,storage,&first_contour,sizeof(CvContour),CV_
RETR_LIST);
cvResetImageROI(sample);
if(nc!=1)
{
c=first_contour;
sum_cx=0;
for(k=0;k<(nc-1);k++)
{
boundbox=cvBoundingRect(c);
cx=(x_roi+boundbox.x)+(boundbox.width/2);
c=c->h_next;
sum_cx=sum_cx+cx;
}
m=sum_cx/(nc-1);
if(i%5==0)
{
x_roi=x_roi+gh;
}
else
{
```

```
x_roi=x_roi+g;
}
}
else
{
m=x_roi+a;
if(i%5==0)
{
x_roi=x_roi+gh;
}
else
{
x_roi=x_roi+g;
}
}
}
m=(m+m1)/2;
x[i]=round(m);
}

//===--PRINT DAN GAMBAR LINE GRID X[41],Y[27]--===
for(i=0;i<27;i++)
{
cvLine(img,cvPoint(0,y[i]),cvPoint(840,y[i]),(0,0,255),1,4);
}
for(i=0;i<41;i++)
{
cvLine(img,cvPoint(x[i],0),cvPoint(x[i],950),(0,0,255),1,4);
}
cvSaveImage("proses_1_segmentation/segmentasi.jpg",img);

ofstream segfile;
segfile.open("proses_1_segmentation/segmentasi.txt");
segfile << "y:\n";
for(i=0;i<27;i++)
{
if(i%10==0)
{
segfile << "\n";
```

```
}
segfile << "[" << y[i] << "] ";
}
segfile << "\n" << "x:\n";
for(i=0;i<41;i++)
{
if(i%10==0)
{
segfile << "\n";
}
segfile << "[" << x[i] << "] ";
}
segfile.close();

cout<<"--segmentation has completed--"<<endl;
cout<<"--displaying segmentation.jpg--"<<endl;
cvNamedWindow("segmentasi.jpg",CV_WINDOW_AUTOSIZE);
cvShowImage("segmentasi.jpg",img);
cvWaitKey(0);
cvDestroyWindow("segmentasi.jpg");
cvReleaseImage(&sample);
cvReleaseImage(&grey);
cvReleaseImage(&bin);

//===(CHARACTER CROPPING)===
FindCharacter braille;
int counter=0;
for(i=0;i<27;i++)
{
for(k=0;k<41;k++)
{
braille.PathFileName(counter);
braille.FindBraille(x[k],y[i]);
counter++;
}
}
braille.releaseIplImage();
cout<<"--character cropping has completed--"<<endl;
```

```
//====(CHARACTER RECOGNITION)====
IplImage* img_part;
IplImage* img_lbp;
img_lbp=cvCreateImage(cvSize(6,7),8,1);
int roi_x[12]={0,6,12,0,6,12,0,6,12,0,6,12};
int roi_y[12]={0,0,0,7,7,7,14,14,14,21,21,21};
int lbp_nb[8];
char character_images[512];
char lbp_imagestemp[512];
char data_learning[512];
int char_recog[1107];

int i,k;
ofstream segfile;
IplImage* img;

//===--partition and lbp operations--===
cout <<"--Generating lbp images...--"<<endl;
k=0;
for(i=1;i<130;i++)
{
snprintf(character_images,512,"samples/%03d.jpg",i);
img_part=cvLoadImage(character_images,CV_LOAD_IMAGE_GRAY
SCALE);
cvThreshold(img_part,img_part,10,255,CV_THRESH_BINARY);
for(int ii=0;ii<12;ii++)
{
for(int y=1;y<8;y++)
{
for(int x=1;x<7;x++)
{
int center=CV_IMAGE_ELEM(img_part,uchar,y+roi_y[ii],x+roi_x[ii]);
lbp_nb[0]=CV_IMAGE_ELEM(img_part,uchar,y-1+roi_y[ii],x-1+
roi_x[ii]);
lbp_nb[1]=CV_IMAGE_ELEM(img_part,uchar,y-1+roi_y[ii],x+
roi_x[ii]);
lbp_nb[2]=CV_IMAGE_ELEM(img_part,uchar,y-1+roi_y[ii],x+1+
roi_x[ii]);
```

```
lbp_nb[3]=CV_IMAGE_ELEM(img_part,uchar,y+roi_y[ii],x+1+
roi_x[ii]);
lbp_nb[4]=CV_IMAGE_ELEM(img_part,uchar,y+1+roi_y[ii],x+1+
roi_x[ii]);
lbp_nb[5]=CV_IMAGE_ELEM(img_part,uchar,y+1+roi_y[ii],x+
roi_x[ii]);
lbp_nb[6]=CV_IMAGE_ELEM(img_part,uchar,y+1+roi_y[ii],x-1+
roi_x[ii]);
lbp_nb[7]=CV_IMAGE_ELEM(img_part,uchar,y+roi_y[ii],x-1+
roi_x[ii]);
for(int iii=0;iii<8;iii++)
{
if (lbp_nb[iii]<center)
{
lbp_nb[iii]=0;
}
else
{
lbp_nb[iii]=1;
}
}
center=lbp_nb[7]*128+lbp_nb[6]*64+lbp_nb[5]*32+lbp_nb[4]*16+lbp
_nb[3]*8+lbp_nb[2]*4+lbp_nb[1]*2+lbp_nb[0]*1;

CV_IMAGE_ELEM(img_lbp,uchar,y-1,x-1)=center;
}
}
snprintf(lbp_imagestemp,512,"proses_3_lbpparttemp/%04d.jpg",k);
cvSaveImage(lbp_imagestemp,img_lbp);
k++;
}
}
cout <<"--lbp images have completed"<<endl;

//data learning initialization
Mat img_mat;
Mat hist_mat;
const int channels=0;
const int numBins=256;
```

```
const float rangevals[2]={0.f,256.f};
const float* ranges=rangevals;
float datalearning[43][6][12][256];

cout <<"--Data learning initialization...--"<<endl;
for(int aa=0;aa<43;aa++)
{
for(int b=0;b<3;b++)
{
for(int cc=0;cc<12;cc++)
{
snprintf(data_learning,512,"data learning/lbp data learning
images/%02d%01d%02d.jpg",aa,b,cc);
img_mat=imread(data_learning,CV_LOAD_IMAGE_ANYCOLOR);
calcHist(&img_mat,1,&channels,noArray(),hist_mat,1,&numBins,&ran
ges);
for(size_t i=0;i<numBins;++i)
{
float val=hist_mat.at<float>(i);
datalearning[aa][b][cc][i]=val;
}
}
}
}

segfile.open("data learning/histogramDL.txt");
for(int aa=0;aa<43;aa++)
{
for(int b=0;b<3;b++)
{
for(int cc=0;cc<12;cc++)
{
snprintf(data_learning,512,"Histogram data learning %02d%01d%02d:",
aa,b,cc);
segfile <<data_learning<<endl;
for(size_t i=0;i<numBins;++i)
{
segfile << datalearning[aa][b][cc][i] << "|";
}
```

```
segfile << "\n";
}
segfile << "\n";
}
segfile << "\n";
}
segfile.close();
cout <<"--Data learning initialization has completed.--"<<endl;

//recognizing
cout <<"--Recognizing characters...--"<<endl;
float hist_char[12][256];

for(int qr=0;qr<129;qr++)
{
for(k=0;k<12;k++)
{
int qrk=qr*12+k;
snprintf(lbp_imagestemp,512,"proses_3_lbpparttemp/%04d.jpg",qrk);
img_mat=imread(lbp_imagestemp,CV_LOAD_IMAGE_ANYCOLOR)
;
calcHist(&img_mat,1,&channels,noArray(),hist_mat,1,&numBins,&ran
ges);
for(size_t i=0;i<numBins;++i)
{
float val=hist_mat.at<float>(i);
hist_char[k][i]=val;
}
}

float comp=3.40282347E+38;
for(int aa=0;aa<43;aa++)
{
for(int b=0;b<3;b++)
{
float comp_val=0;
for(int cc=0;cc<12;cc++)
{
float chi_sqr=0;
```

```
for(size_t i=0;i<numBins;++i)
{
float pembilang=datalearning[aa][b][cc][i]-hist_char[cc][i];
float penyebut=datalearning[aa][b][cc][i]+hist_char[cc][i];
if(penyebut!=0)
{
chi_sqr=chi_sqr+(pow(pembilang,2)/(penyebut));
}
}
comp_val=comp_val+chi_sqr;
}
if(comp_val<comp)
{
comp=comp_val;
char_recog[qr]=aa;
}
}
}
}
cout <<"--character recognizing has completed--"<<endl;
//cvReleaseImage(&img);
cvReleaseImage(&img_part);
cvReleaseImage(&img_lbp);

//===--print recognizing result in char_recog.txt--===
cout<<"--recognizing result documenting...--"<<endl;
segfile.open("proses_4_recognizing/char_recog.txt");
for(i=0;i<129;i++)
{
if(i%12==0)
{
segfile << "\n";
}
segfile << "[" << char_recog[i] << "] ";
}
segfile.close();
cout <<"--recognizing result documenting has completed--"<<endl;

//===(TRANSLATING)===
```

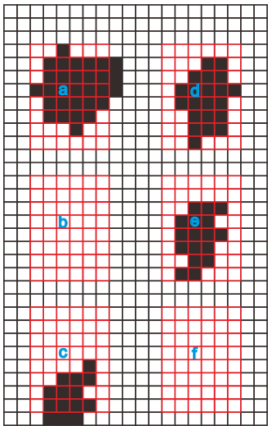
```
table lut;
string baca[129];
 int state;
 int kur=0;
 for(i=0;i<129;i++)
 {
baca[i]=lut.translate(char_recog[i]);
state=lut.state;
if(char_recog[i]==4)
{
 if(state==0 || state==2 || state==3)
{
 if(char_recog[i-1]==37)
{
baca[i]="";
}
if(char_recog[i-1==4] || char_recog[i+1==4])
{
baca[i]=".";
}
}
}
if(char_recog[i]==17)
{
if(char_recog[i-1]==17)
{
 baca[i]="*";
 }
 else
{
 baca[i]="";
}
}
if(char_recog[i]==32)
{
 if(state==0 || state==3)
{
 if(char_recog[i-1]==0)
{
```

```
 baca[i]="";
}
else
{
baca[i]='?';
}
}
}
if(char_recog[i]==37)
{
if(char_recog[i+1]==4)
{
baca[i]="'";
}
else
{
baca[i]="";
}
}
if(char_recog[i]==39)
{
if(kur==0)
{
baca[i]="(";
kur=1;
}
else
{
baca[i]=")";
kur=0;
}
}
}

//===--print character in baca.txt--===
segfile.open("baca.txt");
for(i=0;i<129;i++)
{
if(i%12==0)
```

```
{
segfile << "\n";
}
segfile << baca[i];
}
segfile.close();
cout <<"--translation has completed--"<<endl;

return 0;
}
```

- METODE EVALUASI PIKSEL TERPREDIKSI

Pada metode evaluasi piksel terprediksi digunakan *template* untuk mendeteksi keenam titik posisi karakter braille apakah terdapat piksel yang bernilai 0 (hitam) pada area setiap titik tersebut atau tidak, selanjutnya hasil deteksi di labelkan menggunakan angka binari 6 digit dengan nilai 1 untuk area titik yang memiliki nilai 0, selanjutnya angka binari tersebut dikonversi ke desimal dan angka desimal itulah yang selanjutnya diterjemahkan ke kode ASCII.



Gambar 0.1 Gambar ilustrasi metode evaluasi piksel terprediksi.

Terlihat pada gambar terdapat enam area berbeda (kotak berwarna merah) untuk mengevaluasi piksel-piksel yang diprediksikan menjadi lokasi posisi titik karakter.

Halaman ini sengaja dikosongkan

BIODATA PENULIS

Halum Ghulami, lahir di Lamongan 31 Agustus 1993. Anak keempat dari lima bersaudara. Penulis memulai jenjang pendidikan di jenjang sekolah dasar di MI Muhammadiyah 09 Kranji, Lamongan 1999. Setelah lulus Sekolah Dasar tahun 2005 penulis melanjutkan ke jenjang Sekolah Menegah Pertama di MTs Islam Al-Mukmin, Sukoharjo 2005. Kemudian penulis melanjutkan jenjang pendidikan sekolah menengah atas di MATIQ Isykarima, Karanganyar 2008. Setelah menyelesaikan pendidikan di jenjang Sekolah Menengah Atas penulis melanjutkan jenjang pendidikannya di Institut Teknologi Sepuluh Nopember departemen Teknik Elektro dengan konsentrasi di bidang studi Elektronika.

Email: halumghulami@gmail.com

Halaman ini sengaja dikosongkan