TUGAS AKHIR – KI141502

PENDETEKSIAN GAWANG MENGGUNAKAN ALGORITMA *RANSAC* PADA *PLATFORMDARWIN-OP* BERBASIS PERATURAN KRSBI 2016

UTI SOLICHAH
NRP 5112100215

Dosen Pembimbing I
Dr. Eng. Nanik Suciati, S.Kom., M.Kom.

Dosen Pembimbing II
Muhtadin, S.T., M.T.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

UNDERGRADUATE THESIS – KI141502

GOALPOST DETECTION USING RANSAC ALGORITHM ON DARWIN-OP PLATFORM BASED ON RULE OF KRSBI 2016

UTI SOLICHAH
NRP 5112100215

Supervisor I
Dr. Eng.Nanik Suciati, S.Kom., M.Kom.

Supervisor II
Muhtadin, S.T.,M.T.

DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA2016

LEMBAR PENGESAHAN

PENDETEKSIAN GAWANG MENGGUNAKAN ALGORITMA *RANSAC* PADA *PLATFORM DARWIN-OP* BERBASIS PERATURAN KRSBI 2016

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh
UTI SOLICHAH
NRP : 5112 100 215

Disetujui oleh Dosen Pembimbing Tugas Akhir

1. Dr. Eng. Nanik Suciati, S.Kom., M.Kom. ..... (Pembimbing 1)
   NIP: 19710428 199412 2 001
2. Muhtadin, S.T., M.T. ..... (Pembimbing 2)
   NIP: 19810609 200912 1 003

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

PENDETEKSIAN GAWANG MENGGUNAKAN ALGORITMA *RANSAC* PADA *PLATFORM DARWIN-OP* BERBASIS PERATURAN KRSBI 2016

Nama Mahasiswa	**: UTI SOLICHAH**
NRP	**: 5112100215**
Jurusan	**: Teknik Informatika FTIF-ITS**
Dosen Pembimbing 1	**: Dr. Eng. Nanik Suciati, S.Kom., M.Kom.**
Dosen Pembimbing 2	**: Muhtadin, S.T., M.T.**

Abstrak

Dalam strategi sepak bola robot, robot dituntut untuk bergerak secara autonomous, untuk itu kemampuan dasar yang harus dimiliki setiap robot adalah dapat mengenali fitur penanda dalam lapangan, seperti bola, gawang dan garis lapangan. Kolaborasi pendeteksian gawang dengan informasi orientasi dapat membantu robot dalam memposisikan dirinya (x, y,θ), dan dapat membantu robot untuk memutuskan arah tendangan bola, apakah mengarah ke gawang lawan atau sebaliknya.

Hough Transform merupakan metode pendeteksian gawang yang sudah dikembangkan oleh tim robot ITS, namunmetode ini belum optimal dalam segi kecepatan dan ketepatan jarak.Algoritma Ransac merupakan Algoritma deteksi garis yang memproses semua point untuk dijadikan sebagai garis.Untuk meningkatkan performa robot dalam hal ketepatan jarak, pada Tugas Akhir ini Algoritma Ransac dipilih sebagai metode lain yang akan diterapkan pada pendeteksian gawang.

Hasil uji coba menunjukkan bahwa robot dapat mendeteksi gawang dengan lingkungan pertandingannya. Rata-rata galat pengujian estimasi jarak robot pada gawang dengan Algoritma Ransac lebih kecil dibandingkan dengan

Metode Hough Transform, dengan nilai rata-rata galat 0,03062meter untuk Algoritma Ransac dan0,1452meter untuk Metode Hough Transform. Sedangkan pengujian estimasi waktu program menunjukkanbahwa Metode Hough Transform lebih cepat dibandingkan dengan Algoritma Ransac dengan mencapai rata-rata waktu eksekusi 0,0333 detik untuk Metode Hough Transform dan 0,0435 detik untuk Algoritma Ransac.

Kata kunci : Autonomous, Gawang, Self Positioning, Hough Transform, Pendeteksian gawang, Algoritma Ransac

GOALPOST DETECTION USING RANSAC ALGORITHM ON DARWIN-OP PLATFORM BASED ON RULE OF KRSBI 2016

Student's Name	**: UTI SOLICHAH**
Student's ID	**: 5112100215**
Department	**: Teknik Informatika FTIF-ITS**
First Advisor	**: Dr.Eng. Nanik Suciati, S.Kom., M.Kom.**
Second Advisor	**: Muhtadin, S.T., M.T.**

Abstract

In the strategy game of soccer robot, robot is required to move autonomously. Therefore, the basic ability that each robot should have is able to identify markers in the field of features, such as a ball, goalpost and field lines.Collaboration goalpost detection and or ientation information can help the robot to posistion itself (x,y,θ), beside that, the robot can decide direction of kick ball, is led into the oppposing goalpost or otherwise.

Hough Transform is a method of detecting the goalpost that has been developed by ITS Robotic Team, but these method are not optimal in terms of speed and accuracy of the distance. Ransac Algorithm is an algorithm which processes all point to become a line. Therefore, to improve perfomance of Robot in terms of accuracy of distanece, this final project using Ransac Algorithm for goalpost detection.

Experimenal results show that the robot can detect goalpost with their environment . Experemintal on distance estimation testing robot show that the average error of Ransac Algorithm more smaller than Hough Transform with average error value0,03062meters for Ransac Algorithm and

0.1452meters for Hough Transform. The other side show on experemintal time execution that Hough Transform more faster than Ransac Algorithm with average time execution 0.0333 seconds for Hough Transform and 0.0435 seconds for Ransac Algorithm.

Keywords: Autonomous, Goalpost, Self Positioning, Hough Transform, Goalpost Detection, Ransac Algorithm

KATA PENGANTAR

Alhamdulillahirabbil'alamin, segala puji bagi Allah SWT,yang telah melimpahkan rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan Tugas Akhir yang berjudul **"PENDETEKSIAN GAWANG MENGGUNAKAN ALGORITMA *RANSAC* PADA *PLATFORM DARWIN-OP* BERBASIS PERATURAN KRSBI 2016".**

Tugas Akhir ini disusun dalam rangka sebagai persyaratan menyelesaikan pendidikan S1 Teknik Informatika, Institut Teknologi Sepuluh Nopember.Selesainya Tugas Akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga, Ibu, Mama, Teteh, Aang, Ponakan-Ponakan tersayang tante ti dan Saudara tercinta yang telah senantiasa memberikan dukungan spritual, moral dan material dalam penyelesaian Tugas Akhir ini.
3. Ibu Dr.Eng. Nanik Suciati, S.Kom., M.Kom.selaku pembimbing Iyang telah membimbing dan membantu penulis serta memberikan motivasi dalam menyelesaikan Tugas Akhir ini.
4. Bapak Muhtadin, S.T., M.T. selaku pembimbing II sekaligus pembimbing robot yang telah sabar memotivasi, membimbing, membagi pengalaman dan ilmu kepada penulis serta membantu penulis dalam menyelesaikanTugas Akhir ini.
5. BapakDarlis Herumurti, S.Kom., M.Kom. selaku Kepala Jurusan Teknik Informatika ITS, Bapak Radityo Anggoro, S.Kom.,M.Sc. selaku koordinator TA,dan segenap Bapak-

Ibu dosen pengajar dan karyawan yang telah memberikan ilmu, motivasi dan bantuan selama masa perkuliahan ini.

6. Bapak Ir. Tasripan, M.T., Bapak Dr. Tri Arief Sardjono, M.T., Bapak Rudy Dikairono, S.T., M.Sc dan segenap Bapak Pembimbing Tim Robotika ITS yang selama ini memberikan ilmu, pengalaman, semangat serta motivasi kepada penulis.
7. Mas-Mbak, Teman-Teman, Adek-Adek Tim Robotika ITS, serta Karyawan Pusat Robotika yang selalu menemani dan selalu berbagi, khususnya tim ICHIRO yang selalu menemani, mengingatkan dan juga memberikan dukungan penuh atas kelancaran Tugas Akhir ini.
8. Rantau, Anak Panti Cirebon, Btari, Karlina, Anies, Egi serta sahabat penulis yang tidak bisa penulis sebutkan satu-satu, terimakasih selalu ada untuk penulis dalam suka maupun duka.
9. Partner setia penulis dalam segala hal, terimakasih atas segalanya, *i'm the lucky one to have you.*
10. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan Tugas Akhir ini.

Penulis menyadari bahwa Tugas Akhir ini masih memiliki banyak kekurangan. Sehingga, penulis mengharapkan kritik dan saran yang membangun dari pembaca untuk perbaikan ke depannya.

Surabaya, Juni 2016

Penulis

DAFTAR ISI

[Halaman ini sengaja dikosongkan]

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam strategi sepak bola robot, robot dituntut untuk bergerak secara *autonomous*, untuk itu kemampuan dasar yang harus dimiliki setiap robot adalah dapat mengenali fitur penanda dalam lapangan, seperti bola, gawang dan garis lapangan. Pengenalan fitur gawang ditambah dengan informasi orientasi dapat membantu robot dalam memposisikan dirinya *(x,y,θ)* [1]. Dengan mengetahui posisi dirinya, robot dapat memutuskan arah tendangan bola, apakah mengarah ke gawang lawan atau menjauhkan bola dari gawang sendiri [2].

Salah satu metode yang sudah dikembangkan dalam pendeteksian gawang adalah dengan menggunakan metode *Hough Transform* [3]. Pendeteksian gawang dengan menggunakan Metode *Hough Transform* telah sukses diaplikasikan pada Tim Robot Sepak Bola ITS, namun Metode ini belum optimal dalam segi kecepatan pendeteksian dan ketepatan jarak gawang dengan robot. Sehingga, dibutuhkan pendekatan metode lain untuk meningkatkan performa kemampuan pendeteksian gawang pada Tim Robot Sepak Bola ITS.

Algoritma *Ransac* merupakan Algoritma deteksi garis yang memproses semua *point* untuk dijadikan sebagai garis. Untuk meningkatkan performa robot dalam hal ketepatan jarak, pada Tugas Akhir ini Algoritma *Ransac* dipilih sebagai metode lain yang akan diterapkan pada pendeteksian gawang, Selain alasan diatas, beberapa penelitian dibawah ini juga mendasarkan Algoritma *Ransac* dipilih sebagai metode pendeteksian gawang pada Tugas Akhir, pertama oleh *Stephen* [4], yang menyebutkan bahwa pendekatan Algoritma *Ransac* lebih efektif daripada pendekatan *Hough Transform*. Kedua, penelitian yang dilakukan oleh *Madison* [5] yang

mengaplikasikan Algoritma *Ransac* pada robot Darwin-OP untuk pendeteksian gawang. Diharapkan dengan pengaplikasian Algoritma *Ransac*, performa pendeteksian gawang yang dilakukan oleh Tim Robot Sepak Bola ITS akan lebih baik, dan lebih efektif dalam segi kecepatan pendeteksian dan akurasi jarak gawang dengan robot.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini dapat dipaparkan sebagai berikut:

1. Bagaimanakah pendeteksian gawang menggunakan Algoritma *Ransac*?
2. Bagaimanakah menguji performa dari Algoritma *Ransac* dalam pendeteksian gawang?

1.3 Tujuan

Tujuan dari Tugas Akhir ini mengaplikasikan Algoritma *Ransac* pada pendeteksian gawang pada *platform* robot *Darwin-OP*.

1.4 Manfaat

Dibuatnya Tugas Akhir ini adalah untuk menemukan metode alternatif pendeteksian gawang pada *platform* robot *Darwin-OP*. Harapannya, dengan pengimplementasian Algoritma *Ransac*, akan mempercepat dan memudahkan *self-localization* dari robot sehingga bisa meningkatkan performa bermain Tim Robot Sepak Bola ITS.

1.5 Batasan Masalah

Permasalahan yang dibahas dalam Tugas Akhir ini memiliki beberapa batasan, yaitu sebagai berikut:

1. Robot yang digunakan adalah robot Darwin-OP

2. *Environment* yang digunakan mengikuti peraturan KRSBI 2016
3. Bahasa Pemograman yang digunakan adalah C++
4. *Library* yang digunakan adalah OpenCV
5. Data Set yang digunakan untuk pengujian adalah data set yang diambil sendiri mengikuti dengan *environment* pengujian
6. Deteksi garis yang digunakan adalah tiang kanan dan tiang kiri gawang

1.6 Metodologi

Tahapan-tahapan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut:

1. Penyusunan proposal Tugas Akhir.

   Tahap awal untuk memulai pengerjaan Tugas Akhir adalah penyusunan proposal Tugas Akhir. Pada proposal Tugas Akhir tersebut berisi rencana Tugas Akhir yang akan dikerjakan sebagai syarat untuk menyelesaikan studi dan meraih gelar Strata-1 Teknik Informatika. Pada proposal tersebut berisi tentang pendahuluan dari Tugas Akhir yang meliputi latar belakang, rumusan masalah, batasan, manfaat dan tujuan dari Tugas Akhir. Selain pendahuluan, dijabarkan pula tinjauan pustaka yang digunakan sebagai referensi pendukung pembuatan Tugas Akhir dan juga sub bab metedologi yang berisi mengenai tahapan penyususnan Tugas Akhir dari proposal hingga penyusunan buku, serta dijabarkan pula sub bab jadwal kegiatan dari jadwal pengerjaan Tugas Akhir.

2. Studi literatur

   Tahap ini merupakan tahap pengumpulan informasi yang diperlukan untuk pengerjaan Tugas Akhir sekaligus mempelajarinya. Beberapa literatur yang perlu dipelajari lebih dalam lagi adalah Pendeteksian Gawang, *Robocup*

Soccer, KRSBI, Tim Robot Sepak Bola ITS, Darwin-OP, Open-CV, Algoritma *Ransac, Hough Transform, Thresholding, Morphology Opening,* dan Algoritma *Canny,* Regresi Polynomial*, Euclidean distance, dan* Jarak Antara Titik dan Garis*.*

3. Analisis dan desain perangkat lunak

   Pada tahap ini akan dilakukan analisis dan desain perancangan aplikasi sesuai dengan tujuan yang dijabarkan. Kemudian disesuaikan dengan metode yang tepat, hal ini dimaksudkan agar nantinya ketika diimplementasikan ke dalam aplikasi dapat berjalan sesuai yang diharapkan.

4. Implementasi perangkat lunak

   Perangkat lunak ini akan dibangun menggunakan bahasa permograman C++ dengan bantuan *library OpenCV*.

5. Pengujian dan evaluasi

   Sistem akan diuji setelah selesai diimplementasikan menggunakan tiga skenario uji. Skenario pertama, yaitu pengujian pada parameter proses *thresholding* dan deteksi garis menggunakan Algoritma *Ransac*, kemudian skenario perbandingan performa antara Algoritma *Ransac* dengan Metode *Hough Transform*, yang meliputi pengujian ketepatan jarak dan kecepatan waktu eksekusi program. Pengujian parameter yang dilakukan pada proses *thresholding* dan parameter variabel deteksi garis adalah dengan mengubah nilai dari tiap parameter yang bersangkutan, sehingga didapatkan parameter terbaik. Pada pengujian perbandingan performa antara Algoritma *Ransac* dengan Metode *Hough Transform* akan dibandingkan jarak yang diteksi robot terhadap gawang dengan jarak robot sebenarnya. Selain itu pada pengujian

estimasi waktu eksekusi sistem akan menunjukkan berapa lama sistem itu berjalan.

6. Penyusunan buku Tugas Akhir.
   Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam Tugas Akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku Tugas Akhir ini merupakan laporan secara lengkap mengenai Tugas Akhir yang telah dikerjakan baik dari sisi teori, rancangan, maupun implementasi sehingga memudahkan bagi pembaca dan juga pihak yang ingin mengembangkan lebih lanjut. Sistematika penulisan buku Tugas Akhir secara garis besar antara lain:

Bab I Pendahuluan
Bab ini berisi penjelasan latar belakang, rumusan masalah, batasan masalah dan tujuan pembuatan Tugas Akhir. Selain itu, metodologi pengerjaan dan sistematika penulisan laporan Tugas Akhir juga terdapat di dalamnya.

Bab II Tinjauan Pustaka
Bab ini berisi penjelasan secara detail mengenai dasar-dasar penunjang dan teori-teori yang digunakan untuk mendukung pembuatan Tugas Akhir ini.

Bab III Desain Perangkat Lunak
Bab ini berisi penjelasan tentang rancangan dari sistem yang akan dibangun. Rancangan ini dituliskan dalam bentuk *flowchart.*

Bab IV Implementasi

Bab ini berisi penjelasan implementasi dari rancangan yang telah dibuat pada bab sebelumnya. Implementasi disajikan dalam bentuk *code* secara keseluruhan disertai dengan penjelasannya.

Bab V Uji Coba Dan Evaluasi

Bab ini berisi penjelasan mengenai data hasil percobaan dan pembahasan mengenai hasil percobaan yang telah dilakukan.

Bab VI Kesimpulan Dan Saran

Bab ini merupakan bab terakhir yang menyampaikan kesimpulan dari hasil uji coba yang dilakukan dan saran untuk pengembangan perangkat lunak ke depannya.

BAB II
TINJAUAN PUSTAKA

Dalam bab ini akan dijelaskan mengenai teori-teori yang merupakan dasar dari pembangunan sistem. Selain itu terdapat penjelasan yang menunjang pengerjaan Tugas Akhir ini sehingga dapat memberikan gambaran secara umum sistem yang akan dibangun.

2.1 Pendeteksian Gawang

Dalam permainan sepak bola robot, robot dituntut bergerak secara *autonomous.* Bergerak secara *autonomous* akan menjadikan robot bermain sepak bola lebih pintar layaknya manusia.

Selain bergerak secara *autonomous,* robot juga dituntut untuk mengenali fitur penanda yang ada di lapangan. Fitur-fitur penanda dalam permainan sepak bola robot meliputi fitur bola, garis dan gawang. Ketiga hal ini menjadi hal yang penting yang harus dikenali robot dengan cepat, karena dengan robot dapat mengenali fitur penanda yang ada di lapangan, robot akan mudah untuk memposisikan dirinya pada lapangan [6].

Pengenalan fitur gawang atau pendeteksian gawang sangat berpengaruh dalam keberhasilan penentuan posisi robot dalam lapangan. Hasil dalam pendeteksian gawang adalah informasi gawang. Pengenalan fitur gawang ditambah dengan informasi orientasi dapat membantu robot dalam memposisikan dirinya (x,y,θ) [1], dan juga dapat membantu robot memutuskan arah tendangan bola, apakah mengarah ke gawang lawan atau menjauhkan bola dari gawang sendiri [2].

2.2 Robocup Soccer

Robocup merupakan kompetisi robot sepak bola Internasional yang diselenggarakan rutin setiap tahunnya. Robocup diadakan sejak tahun 1997, di Nagoya, Jepang. Setiap tahun tuan rumah

Robocup berpindah-pindah dari satu negara ke negara lainnya, dan peraturan yang diberikanpun bertambah kompleks dari tahun sebelumnya. Hal ini dikarenakan sejalan dengan tujuan pada Robocup yang menginginkan robot humanoid pada tahun 2050 dapat bertanding dan mengalahkan juara sepak bola FIFA sesuai dengan peraturan yang diberlakukan pada pertandingan FIFA[7].

Pada Robocup terdiri dari beberapa tingkatan kelompok robot humanoid yang akan dipertandingkan. Berikut adalah beberapa kelompok tingkatan yang ada pada Robocup : *teen size, kid size*, dan *adult size.*

2.3 Kontes Robot Sepak Bola Indonesia (KRSBI)

Kontes Robot Sepak Bola Indonesia (KRSBI) merupakan salah satu kompetisi yang diadakan oleh Direktorat Jenderal Pendidikan Tinggi Indonesia. KRSBI menjadi salah satu divisi lomba pada Kontes Robot Indonesia (KRI). Kontes Robot Sepak Bola Indonesia dilakukan setiap tahun dan melewati 2 tahap seleksi. Tahap pertama, adalah KRSBI Regional, dimana dalam kontes ini mencari tim-tim terbaik dalam satu regional yang sama. Kemudian tahap kedua yaitu, KRSBI Nasional, dimana peserta yang bergabung adalah tim terbaik yang dinyatakan lolos oleh dewan juri untuk mengikuti Kontes Robot Indonesia.

Peringkat pertama pada KRSBI Nasional akan menjadi delegasi Indonesia untuk tahun berikutnya pada perlombaan sepak bola robot Internasional, Robocup. Dengan begitu, tujuan dan peraturan dari KRSBI ini mengarah pada tujuan dan peraturan Robocup.

Peserta KRSBI diharuskan membuat robot *humanoid,* robot yang mirip manusia dengan tinggi antara 40-90 cm dan dapat bermain sepak bola seperti pertandingan sepak bola yang sangat populer di masyarakat. Aturan main dalam KRSBI biasanya setiap tahun berubah, mengikuti peraturan Robocup divisi *KidSize Humanoid League*. Secara umum, pertandingan sepak bola robot akan mempertandingkan antara dua tim yang

berhadap-hadapan yang dilaksanakan dalam waktu (2x10) menit dengan masa istirahat selama 5 menit pada lapangan sepak bola dengan luas 4x6 meter. Lapangan sepak bola yang digunakan berupa rumput sintesis, dan gawang yang ada berwarna putih. Setiap tim yang bertanding terdiri dari maksimal lima robot *humanoid* yang salah satunya harus diprogram sebagai penjaga gawang dan empat lainnya sebagai pemain penyerang [8].

2.4 Tim Robot Sepak Bola ITS

Tim Robot Sepak Bola ITS merupakan salah satu divisi tim yang ada pada Tim Robotika ITS, dibentuk guna memfasilitasi anggotanya untuk melakukan riset pada permainan sepak bola robot. Tim Robot Sepak Bola ITS dibentuk sejak tahun 2010. Pada tahun 2013, tim robot sepak bola ITS meresmikan nama ICHIRO sebagai nama tim.

ICHIRO memiliki tiga buah robot *humanoid* yang semuanya menggunakan *platform* Darwin-OP, satu robot memilki tugas sebagai *keeper*, dan dua lainnya memiliki tugas sebagai penyerang. Pada Gambar 2.1 memperlihatkan anggota robot dari ICHIRO.

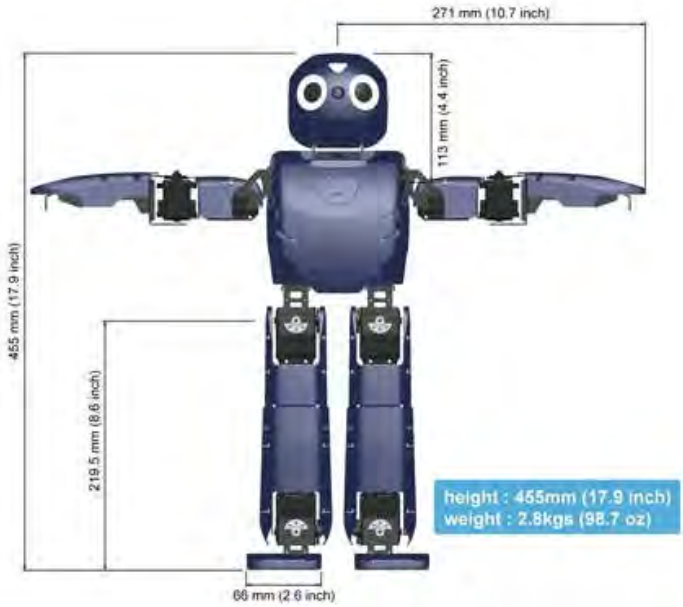
Dalam pengembangangannya, ICHIRO melakukan pendekatan untuk meningkatkan kecerdasan robot [9]: 1. Perbaikan *vision*, sehingga didapatkan citra yang baik sebagai bahan dasar pemrosesan citra selanjutnya; 2. Perbaikan deteksi fitur-fitur lapangan untuk digunakan sebagai penanda/referensi pada proses penentuan lokasi; 3. Kerja sama antar robot. Selain itu, Modifikasi *hardware* pun dilakukan oleh ICHIRO yang mengubah kamera bawaan pada Darwin-OP dengan kamera *logitech C920*. Pengembangan secara *software, hardware* dan algoritma masih dilakukan oleh ICHIRO hingga sekarang, hal ini dilakukan untuk meningkatkan performa bermain robot di lapangan. ICHIRO selama ini mengikuti dua kompetisi, yaitu Kontes Robot Indonesia dan *Singapore Robotic Game.*

Gambar 2.1. Robot ICHIRO

2.5 Darwin-OP

Dynamic Anthropomorphic Robot with Intellegence Open Platform atau yang sering disebut dengan Darwin-OP merupakan platform robot yang dihasilkan dari kolaborasi antara Robotis dengan Universitas Pennsylvania.

Robot *Darwin-OP* memiliki perangkat keras yang terdiri dari sebuah PC dengan tipe *fitpc2i,* modul *Wi-Fi,* kamera *logitech* C905, motor servo MX-28T dan kontroler CM730 [10]. Informasi mengenai *Darwin-OP* dibuka secara publik, baik mengenai mekanik maupun perangkat lunak yang digunakan, oleh karena itu *Darwin-OP* termasuk salah satu *platform* robot yang *open platform.* Gambar *platform* robot *Darwin-OP* ditunjukkan pada Gambar 2.2.



Gambar 2.2. *Platform* Robot *Darwin-OP*

2.6 OpenCV

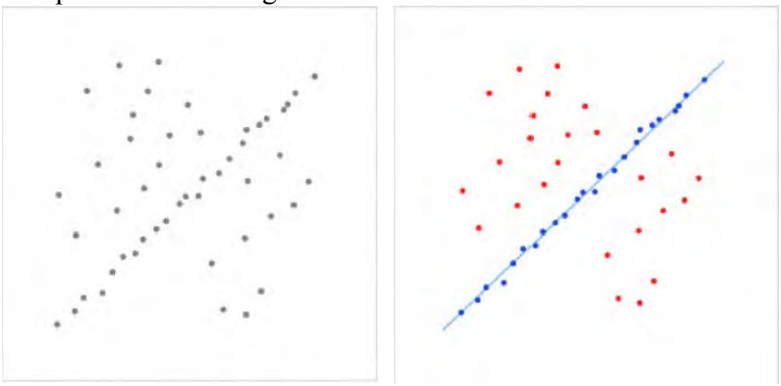
OpenCV (Open Source Vision Library Computer) adalah sebuah *open source library* visi komputer dan perangkat lunak *machine learning. Library OpenCV* memiliki lebih dari 2500 algoritma. Algoritma yang ada dapat digunakan dan diaplikasikan di berbagai macam bidang, contohnya untuk mendeteksi wajah, mengidentifikasi objek, dan mengetahui gerakan kamera.

OpenCV bisa diaplikasikan dalam beberapa bahasa pemograman, seperti C++, C, *Python, Java,* dan bisa dikembangkan pada sistem operasi *Windows, Linux, Mac OS*, maupun *Android* [11].

2.7 Algoritma *Ransac*

Algoritma *Ransac* pertama kali diusulkan oleh Fishler dan Bolles [12] pada tahun 1981. Algoritma *Ransac* merupakan salah satu algoritma estimasi yang paling kuat dan juga digunakan secara luas pada bidang visi komputer[13] . Cara kerja Algoritma *Ransac* adalah dengan melakukan perulangan beberapa kali dan secara acak memilih subset titik dari data input yang akan dijadikan *line,* setelah itu adalah menghitung jumlah *inliers,* metode untuk menghitung jumlah *inliers* adalah melalui jarak antara titik dengan *line* yang dibuat, hal yang selanjutnya dilakukan adalah kemudian menyimpannya sebagai *best inliers.*

Setiap perulangan yang dihasilkan akan menghasilkan *inliers,* dan program akan menyimpan *best inliers,* dengan keadaan demikian, setiap perulangan, akan dibandingkan antara *inliers* dengan *best inliers* nya, jika *inliers* baru yang didapatkan lebih baik daripada *best inliers* yang terakhir kali disimpan, maka, *best inliers* digantikan dengan *inliers* baru. Hal ini dilakukan berulang, sampai dengan batas penentuan akhir untuk mendapatkan model atau titik-titik yang diinginkan [5]. Pada Gambar 2.3 diperlihatkan citra sebelum dan sesudah diimplementasikan Algoritma *Ransac*.



Gambar 2.3. Deteksi garis citra sebelum (kiri) dan sesudah (kanan)

2.8 *Hough Transform*

Hough Transform merupakan teknik ekstraksi fitur yang digunakan dalam analisis citra, visi komputer dan pengolahan citra digial. Metode *hough transform* digunakan pada pendeteksian lingkaran, dan garis lurus. Cara kerja yang dimiliki *hough transform* adalah melalui representasi titik-titik (x,y) ke bentuk *Hough Domain* (ρ,θ) dan memilihnya berdasarkan *voting-point* [14].

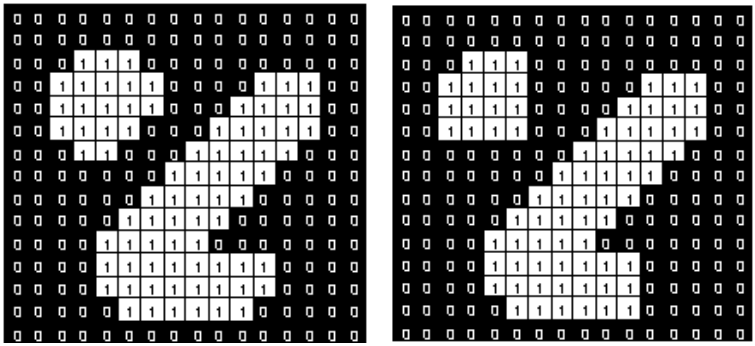
2.9 *Thresholding*

Thresholding merupakan salah satu teknik segmentasi gambar yang paling mudah [15], dimana komputasi *thresholding* lebih simpel dibandingkan dengan teknik segementasi yang lain. Proses *thresholding* digunakan untuk segmentasi warna objek yang kita inginkan. Batas *thresholding* digunakan untuk memisahkan objek yang kita inginkan dengan objek selainnya. Hasil dari proses *thresholding* adalah *binary image* dengan merepresentasikan objek yang terpilih dengan nilai 1, dan nilai 0 sebagai representasi nilai selain objek [1].

2.10 *Morphology Opening*

Morphology Opening merupakan sebuah proses citra digital yang terdiri dari proses erosi kemudian dilanjut dengan proses dilasi [16]. Efek yang dihasilkan dari proses ini adalah menghilangnya objek-objek kecil dan secara umum dapat menghaluskan batas dari objek besar tanpa mengubah area objek. Pada proses *Morphology Opening*, hanya citra hasil *binary* saja yang dapat diproses. Biasanya, *Morphology Opening* dilakukan untuk membantu hasil dari segmentasi warna oleh *thresholding* yang belum baik dikarenakan adanya beberapa *noise* dalam citra. Ilustrasi proses *Morphology Opening dapat* dilihat pada Gambar 2.4. Dimana disana diperlihatkan hasil sebelum dan sesudah

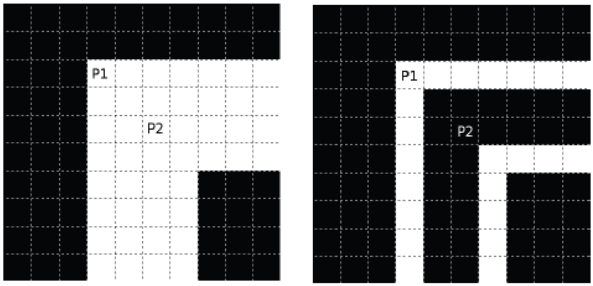
proses *Morphology Opening*. Proses *Morphology Opening* membantu menghilangka*n noise salt* pada citra[17].



Gambar 2.4. Ilustrasi *Morphology Opening*

2.11 Algoritma *Canny*

Algoritma *Canny* merupakan salah satu metode dalam pendeteksian tepi. Pada paper [18] menyebutkan, Algoritma *Canny* adalah metode yang paling baik untuk mendetksi tepi dibandingkan dengan metode *sobel* dan *laplace*. Algoritma *Canny* digunakan untuk mendeteksi tepi citra. Pada Gambar 2.4 bagian kanan, diperlihatkan bahwa P_2 dihapus, , sedangkan area objek pada P_1 dipertahankan karena P_1 merepresentasikan area tepi.



Gambar 2.5. Ilustrasi sebelum dan sesudah deteksi tepi

2.12 Regresi Polynomial

Metode Regresi polynomial adalah sebuah metode statistik untuk memodelkan hubungan antara variabel respon *Y* dengan variabel prediktor *X,* dimana $m(X)$ adalah suatu fungsi regresi yang belum diketahui dan ingin ditaksir dan ε_i adalah suatu variabel acak yang menggambarkan variasi *Y* di sekitar $m(X)$ [19]. Secara umum, hubungan antara *X* dan *Y* dapat ditulis pada persamaan 2.1.

$$Y = m(Xi) + \varepsilon_i \quad (2.1)$$

2.13 *Euclidean Distance*

Euclidean distance adalah metriks yang paling sering digunakan untuk menghitung kesamaan dua vector. Rumus *Euclidean Distance* adalah akar dari kuadrat perbedaan 2 vektor [20]. Rumus dari *Euclidean Distance* dapat dilihat pada Persamaan 2.2 dan Persamaan 2.3. Dimana d_{pq} menyatakan jarak antara vektor p dan *q*. *n* menyatakan jumlah anggota vektor. q_i menyatakan nilai vektor *q* ke-*i* dan p_i menyatakan nilai vektor *p* ke-*i*.

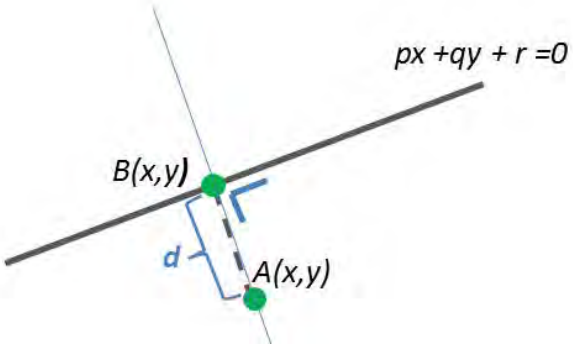
$$d_{pq} = \sqrt{\sum_{k=1}^{n}(q_i - p_i)^2} \quad (2.2)$$

$$d_{pq} = \sqrt{(q1 - p1)^2 + (q2 - p2)^2 + \ldots + (qn - pn)^2} \quad (2.3)$$

2.14 Jarak Antara Titik dan Garis

Penghitungan jarak antara titik dengan garis. Dua hal yang dibutuhkan informasinya adalah adanya persamaan suatu garis tersebut dan Titik(*x,y*) dari titik yang akan dicari jaraknya. Ilustrasi menghitung jarak titik ke garis dapat dilihat pada Gambar 2.4. Dimana, besarnya jarak titik *A*(*x,y*) ke garis *px* +*qy* + *r* = *0* sama saja dengan jarak antara titik *A*(*x,y*) ke titik *B*(*x,y*).

Hal ini didapat karena titik *A* jika dibuat sebuah garis memenuhi persamaan gradien *m1.m2 = -1*, maka akan berpotongan dengan garis *px +qy + r = 0*. Sehingga, *B(x,y)* adalah hasil dari titik potong dari keduanya. Besarnya jarak antara titik *A(x,y)* dengan titik *B(x,y)* menggunakan rumus *euclidean distance* [21] yang dapat dilihat pada Persamaan 2.2 dan Persamaan 2.3.



Gambar 2.6. Ilustrasi jarak antara titik dan garis

BAB III
DESAIN PERANGKAT LUNAK

Pada bab ini akan dijelaskan mengenai perancangan sistem yang akan dibuat. Perancangan yang dijelaskan meliputi data dan proses. Data yang dimaksud adalah data yang akan diolah dalam sistem, sehingga tujuan Tugas Akhir ini bisa tercapai. Proses tahap-tahap sistem meliputi proses *preprocessing*, deteksi garis menggunakan *Algoritma Ransac*, dan *postprocessing.*

3.1 Data

Pada sub bab ini akan dijelaskan mengenai data yang digunakan sebagai masukan perangkat lunak untuk selanjutnya diolah dan dilakukan pengujian sehingga menghasilkan data keluaran yang diharapkan.

3.1.1 Data Masukan

Data masukan adalah data yang digunakan sebagai masukan dari sistem. Data yang digunakan adalah data video, dengan kualitas 320 x 240 piksel dan memiliki shannel R, G, B. Setiap data masukan yang diambil, video yang sedang berjalan *realtime* akan di *capture* pada *frame* tertentu. hasil *capture* tersebut yang akan dijadikan data masukan. Ilustrasi data masukan dapat dilihat pada Gambar 3.1.

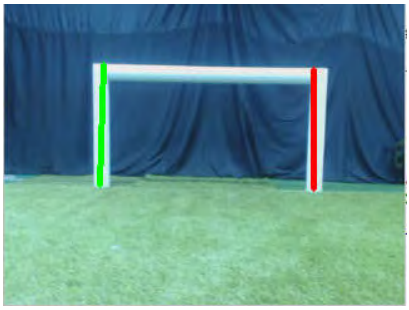
Gambar 3.1. Contoh Citra Masukan Gawang

3.1.2 Data Keluaran

Data masukan akan di proses melalui tiga tahapan. Pertama, tahap *preprocessing* dimana citra masukan akan diubah kedalam citra *greyscale,* kemudian di *thresholding* sehingga hanya mendapatkan objek yang diinginkan yaitu tiang, dan dilanjutkan dengan proses mereduksi *noise* kemudian mendeteksi tepi citra. Setelah mendapatkan tepi citra, hasil tepi citra kemudian di transformasikan ke dalam bentuk *point* sehingga dapat diproses oleh tahap kedua.

Tahap kedua yang dilakukan pada citra masukan adalah mendeteksi garis dari citra tepi yang sudah dalam bentuk *point* tadi menggunakan metode Algoritma *Ransac.*

Garis yang sudah dideteksi oleh Algoritma *Ransac* kemudian dilanjutkan pada tahap *postprocessing,* dimana garis yang ada akan di *cluster* dipilih, apakah termasuk *cluster* tiang kanan, tiang kiri atau tiang atas, setelah di *cluster,* proses selanjutnya adalah menemukan mean tiap *cluster* untuk dijadikan kandidat tiang kanan, dan tiang kiri dari gawang. Setelah mendapatkan, tiang kanan dan kiri gawang, hal selanjutnya adalah menghitung tinggi tiang kanan dan tiang kiri yang terdeteksi, yang kemudian dapat menghitung jarak robot ke posisi gawang yang dideteksi. Ilustrasi citra keluaran dapat dilihat pada Gambar 3.2.

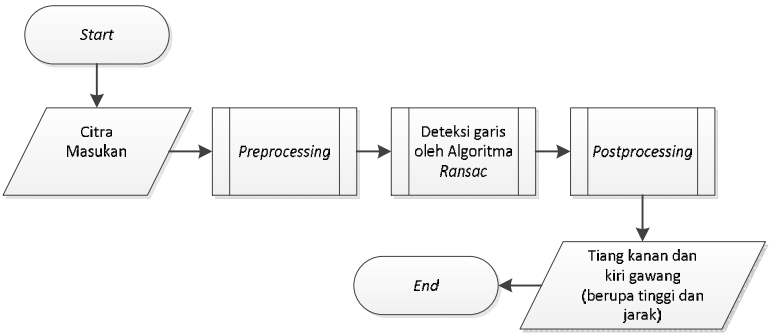


Gambar 3.2. Contoh Citra Keluaran Gawang

3.2 Desain Sistem Secara Umum

Dalam permainan sepak bola robot, robot dituntut untuk bergerak secara *autonomous,* agar dapat bermain sepak bola dengan baik. Dalam menunjang agar robot bergerak *autonomus,* robot perlu memiliki kemampuan dalam mengenali fitur yang ada dalam lapangan, salah satunya kemampuan dalam mengenali fitur gawang. Pengenalan fitur gawang sangat berpengaruh dalam permainan, karena robot dapat dengan mudah memposisikan dirinya dengan gawang.

Sistem dibawah ini akan menjelaskan secara detil mengenai pendeteksian gawang oleh metode ransac pada *platform Darwin-OP* berdasarkan perturan KRSBI 2016. Tahapan sistem dibagi menjadi tiga tahapan besar, yaitu *preprocessing*, deteksi garis kemudian *postprocessing.* Diagram alur desain umum sistem dapat dilihat pada Gambar 3.3.

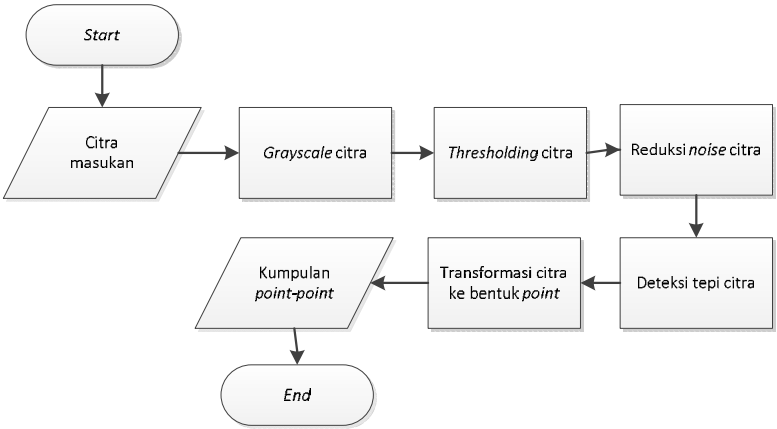


Gambar 3. 3. Diagram alur desain umum sistem deteksi gawang

3.3 *Preprocessing*

Setiap piksel pada citra masukan tidak bisa langsung dilakukan deteksi garis menggunakan Algoritma *Ransac*, ada beberapa tahapan yang harus dilakukan diawal, untuk mempersiapkan citra agar lebih mudah di proses di tahap

selanjutnya, berikut adalah lima tahap yang harus dilakukan, yaitu: mengubah citra ke *grayscale*, *thresholding* citra, reduksi *noise* citra, deteksi tepi citra dan terakhir adalah transformasi citra kedalam bentuk *point*. Diagram alur tahap *preprocessing* ditunjukkan pada Gambar 3.4.



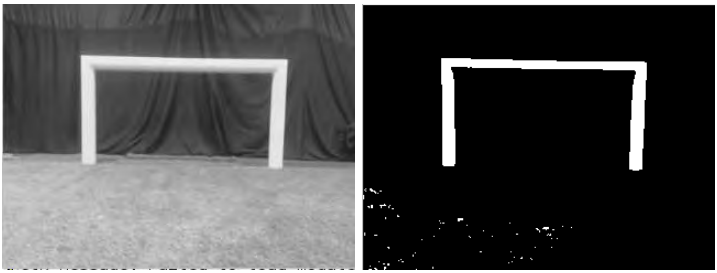
Gambar 3.4. Diagram alur tahap *preprocessing*

3.3.1 *Grayscale* Citra

Citra masukan asli adalah citra masukkan berwarna dengan kualitas 320 x 240 piksel, untuk memudahkan tahapan selanjutnya yang akan dilakukan dalam pendeteksian gawang menggunkan metode Algoritma *Ransac,* citra masukkan yang berwarna ditransformasikan ke dalam bentuk citra *grayscale.* Pengubahan citra *grayscale* disini, memudahkan citra masukan untuk di *thresholding*, karena pada pengenalan fitur gawang, objek yang di deteksi adalah gawang. Berdasarkan peraturan KRSBI 2016 [8], gawang berwarna putih, dengan diubahnya ke bentuk *grayscale* pemilihan citra putih saja yang diambil akan lebih mudah. Ilustrasi citra masukkan yang telah di transformasikan ke dalam bentuk *grayscale* dapat dilihat pada Gambar 3.5.

3.3.2 *Thresholding* Citra

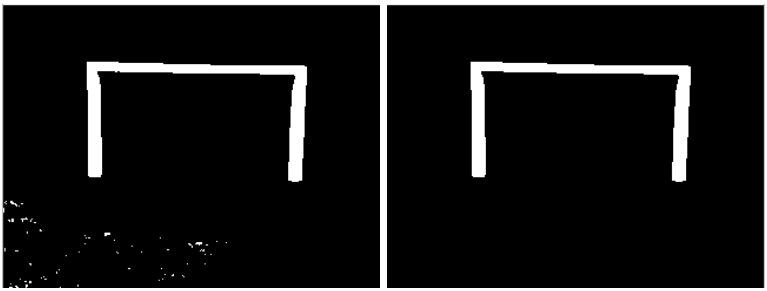
Proses *Thresholding* merupakan sebuah proses segmentasi warna, sehingga citra yang di proses adalah citra yang berwarna gawang saja. *Thresholding* membantu citra masukan untuk memisahkan objek gawang yang dipilih dengan objek selain gawang. *Thresholding* yang baik akan menghasilkan citra yang hanya mendeteksi gawang saja. Berikut adalah ilusrasi hasil *thresholding* citra yang dilakukan dapat dilihat pada Gambar 3.5.



Gambar 3.5. Citra *Grayscale* (kiri) dan Citra *Thresholding* (kanan)

3.3.3 Reduksi *Noise* Citra

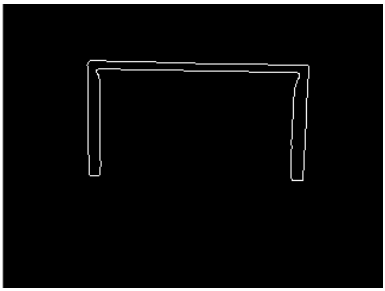
Hasil keluaran citra dari proses *thresholding* terkadang tidak selalu ideal, untuk membantu citra mendapatkan hasil segmentasi yang ideal, citra hasil tadi dilanjutkan pada tahap reduksi *noise. Noise* yang ada pada citra dihilangkan menggunakan pendekatan Metode *Morphology Open* dengan bantuan *library* OpenCV. Pada dasarnya *Morphology Open* adalah sebuah metode pereduksian *noise* yang menggabungkan antara dua sub metode, yaitu pertama adalah proses erosi kemudian proses dilasi. Proses erosi membantu citra menghilangkan titik-titik kecil pada citra kemudian proses dilasi akan membantu citra untuk menebalkan objek gawang yang terdegradasi. Ilustrasi tahapan reduksi *noise* citra ditunjukkan pada Gambar 3.6.



Gambar 3.6. Citra sebelum (kiri) dan sesudah (kanan) hasil reduksi *noise*

3.3.4 Deteksi Tepi Citra

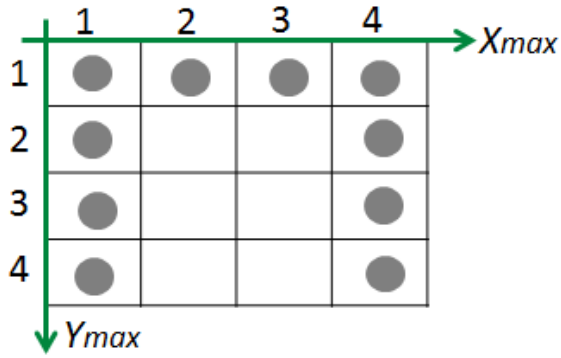
Tahap *preprocessing* selanjutnya setelah reduksi *noise* citra adalah mendeteksi tepi citra gawang. Hal ini dilakukan untuk mengurangi garis yang akan terdeteksi oleh Algoritma *Ransac*. Pada tahapan deteksi tepi citra, Algoritma yang digunakan adalah Algoritma *Canny* dengan menggunakan bantuan *library OpenCV*, dimana pada algoritma ini membantu menghilangkan citra hasil reduksi *noise*. Hasil dari Algoritma *Canny* adalah hanya menampilkan tepi dari citranya gawangnya saja. Hasil deteksi tepi sangat membantu untuk mengurangi jumlah *point* yang di transformasi pada tahap selanjutnya. Ilustrasi hasil deteksi tepi citra gawang yang dihasilkan ditunjukkan pada Gambar 3.7.



Gambar 3.7. Citra hasil deteksi tepi

3.3.5 Transformasi Citra ke *Point*

Deteksi garis menggunakan Algoritma *Ransac* tidak bisa diproses langsung dari hasil tepi citra tadi, melainkan harus melewati satu proses yaitu mentransformasikan citra *binary* hasil tepi ke dalam bentuk *point*. Proses transformasi dari citra ke *point* adalah dengan mengecek nilai piksel koordinat *X* citra dari *X=0* hingga *X= Xmax* piksel citra, dengan ketentuan, setiap nilai *X* yang dikunjungi, dicek juga nilai piksel Y citra dari Y=0 sampai *Y= Ymax* piksel. Ilustrasi cara kerja transformasi citra ke point ditunjukkan pada Gambar 3.8. Pada proses transformasi ini akan menghasilkan *point-point* pada suatu citra yang nantinya disimpan kemudian di proses selanjutnya oleh Algoritma *Ransac* Contoh proses transformasi citra ke *point* diperlihatkan pada LAMPIRAN A, dimana Tabel A.1 memperlihatkan semua point yang diubah dari citra Gambar 3.7.



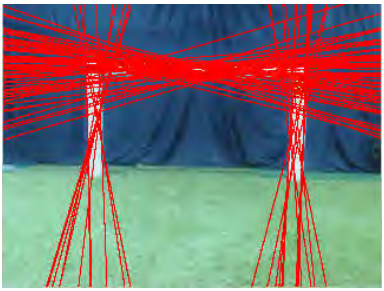
Gambar 3.8. Ilustrasi Transformasi citra ke *Point*

Berdasarkan Ilustrasi dari Gambar 3.8, cara kerja pengecekkan titik dimulai dari koordinat *X* dan mengambil satu nilai *X*, kemudian mengecek nilai *Y* hingga *Ymax,*. Setelah itu, dilanjutkan perulangan mengambil satu nilai *X* hingga *X=Xmax.* Hasil urutan *point* yang disimpan dimulai dari titik (1,1), (1,2), (1,3), (1,4), (2,1), (3,1), (4,1), (4,2), (4,3), (4,4).

3.4 Deteksi Garis Menggunakan Algoritma *Ransac*

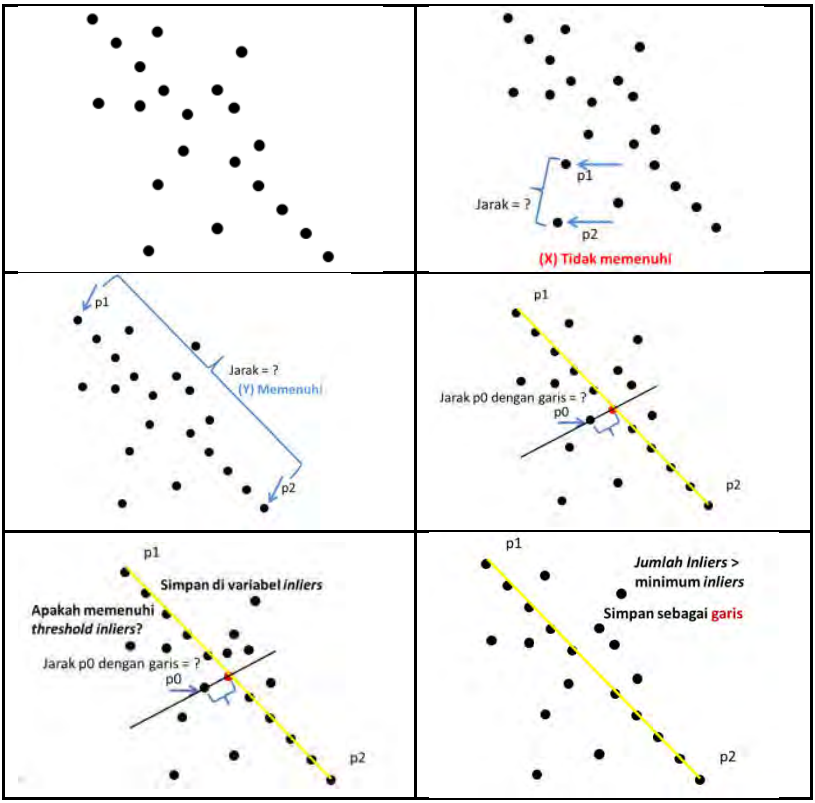
Hasil *binary* tepi citra yang sudah di transformasikan menjadi *point* kemudian di proses menggunakan Algoritma *Ransac* untuk didapatkan garis-garis yang terdeteksi. Himpunan semua *point* pada citra dilanjutkan dengan mengambil dua *point random*, kemudian dilihat, apabila jarak antara satu *point* dengan *point*-nya terlalu dekat, maka dua *point* random tadi tidak dipilih, Sebaliknya, jika pengambilan dua *point* random diatas memenuhi jarak minimum antara dua *point,* maka selanjutnya dilakukan adalah mengambil satu *point* kemudian dihitung jarak antara *point* ke garis tadi.

Apabila jarak point tersebut memenuhi *threshold* jarak *inliers*, maka *point* tadi termasuk *inliers* dari garis tersebut [12]. Pengambilan satu *point* dilakukan berulang hingga semua himpunan *point* sudah dihitung. Setelah semua himpunan *point* sudah di cek, selanjutnya adalah mengecek apakah banyak nya *point inliers* tadi memenuhi minimum *inliers* sebuah garis, apabila tidak memenuhi, maka di hapus, jika memenuhi dimasukkan ke dalam tampungan garis yang terdeteksi. Proses pengambilan dua *point* random hingga berhasil menjadi sebuah garis merupakan satu kesatuan proses, untuk mendapatkan banyak garis, maka dilakukan perulangan, hingga didapatkan banyak garis. Ilustrasi deteksi garis menggunakan Algoritma *Ransac* ditunjukkan pada Gambar 3.9.

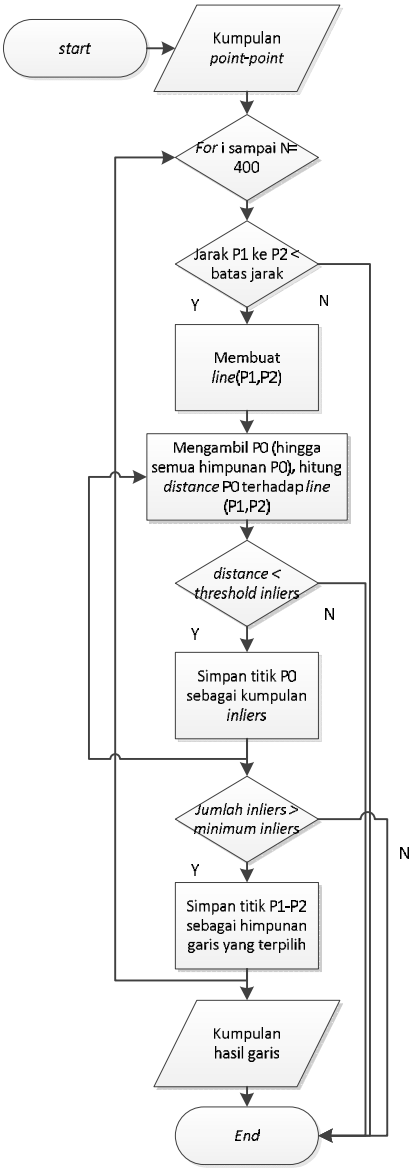


Gambar 3.9. Citra hasil deteksi Garis

Selain menggunakan diagram alur, Algoritma R*ansac* juga diilustrasikan dengan urutan-urutan gambar. Hal tersebut ditunjukkan pada Gambar 3.10. Dimana himpunan *point* yang ada di proses sampai mendapatkan sebuah garis. Untuk mendapatkan banyak garis, maka proses seperti Gambar 3.10 diulang hingga jumlah garis yang ingin didapatkan. Diagram alur dari proses deteksi garis Algoritma *Ransac* ditunjukkan pada Gambar 3.11.



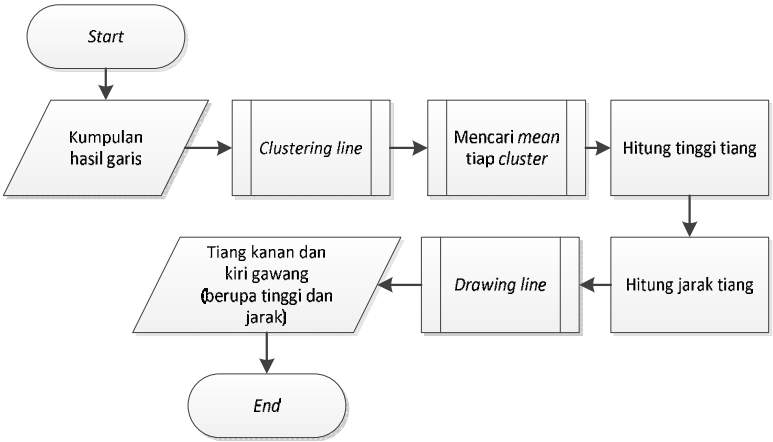
Gambar 3.10. Ilustrasi deteksi garis Algoritma *Ransac*



Gambar 3.11. Diagram alur deteksi garis.

3.5 *Postprocessing*

Deteksi garis oleh Algoritma *Ransac* menghasilkan kumpulan garis-garis. Untuk mendapatkan sebuah informasi mengenai objek gawang atau bukan, informasi tinggi tiang, dan juga informasi jarak dari sebuah garis yang terdeteksi, maka diperlukan satu tahapan lagi untuk mendapatkan itu semua, yaitu tahapan *postprocessing*. Diagram alur *postprocessing* ditunjukkan pada Gambar 3.12.



Gambar 3.12. Diagram alur *postprocessing*

3.5.1 *Clustering Line*

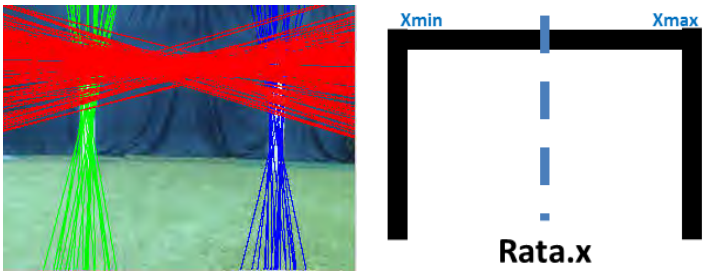
Kumpulan garis yang terdeteksi oleh Algoritma *Ransac* belum bisa langsung di proses, untuk itu diperlukan proses *clustering line* untuk mengambil satu garis terbaik dari tiang kiri, tiang atas dan tiang kanan. Hal pertama yang dilakukan adalah melakukan pemisahan dari tiap-tiap garis, apakah garis itu termasuk *cluster* tiang kanan, tiang atas atupun tiang kiri dengan cara menghitung Y_{min}, Y_{max}, X_{min}, X_{max} dari *binary object* gawang adalah langkah pertama untuk menentukan *cluster* tiang.

Penetuan Y_{min}, Y_{max}, X_{min}, X_{max} dilakukan perulangan dari citra koordinat X dan perulangan dari citra koordinat y. Setelah mendapatkan Y_{min}, Y_{max}, X_{min}, dan X_{max}. Selanjutnya adalah menghitung tengah-tengah koordinat X dari objek gawang dengan perhitungan sesuai dengan persamaan 3.1.

$$\text{Rata X} = X_{min} + \frac{(\text{X}max - \text{X}min)}{2} \qquad (3.1)$$

.

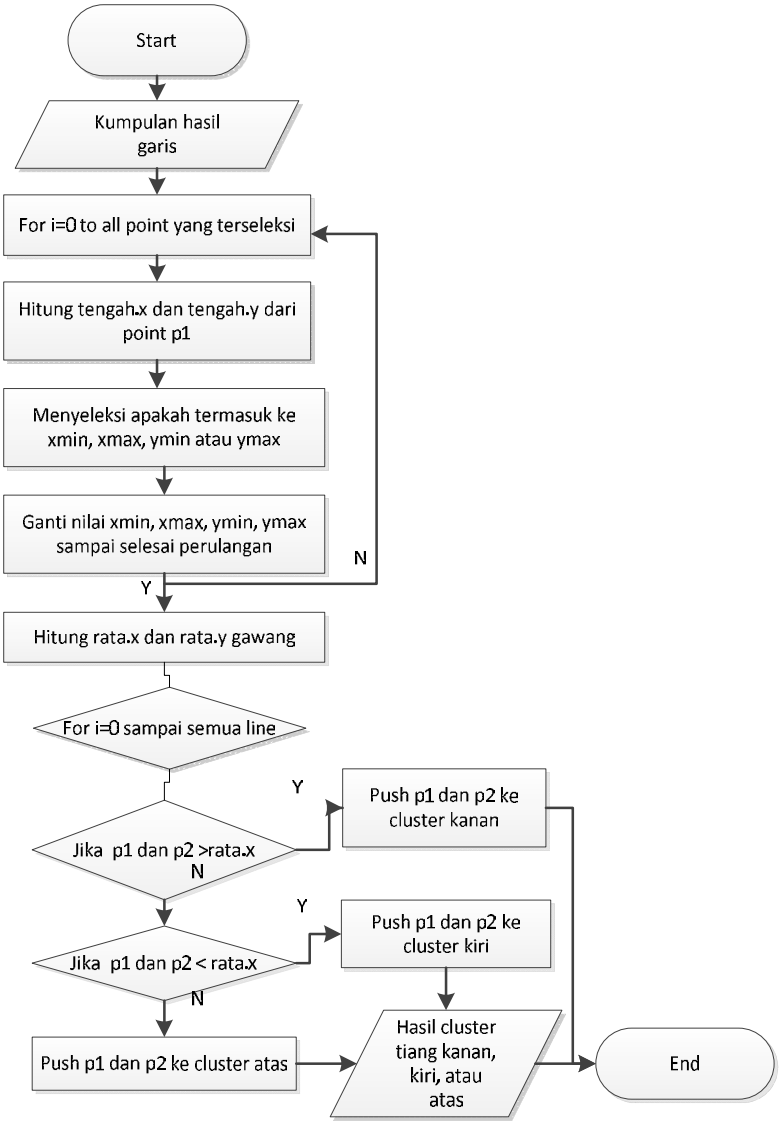
Tengah-tengah dari koordinat X atau Rata X dari objek gawang digunakan sebagai acuan, apakah garis itu termasuk tiang kanan, tiang kiri atau tiang atas. Jika garis itu lebih besar dari tengah koordinat, maka garis tersebut termasuk *cluster* tiang kanan. Jika garis itu lebih kecil dari tengah koordinat X, maka garis itu termasuk *cluster* tiang kiri, sisanya yang tidak termasuk keduanya, termasuk *cluster* tiang atas. Ilustrasi hasil *cluster* dari garis dan ilustrasi *cluster* ditunjukkan pada gambar 3.13. Diagram alur dari proses *clustering line* ditunjukkan pada Gambar 3.14.



Gambar 3.13. Citra hasil *clustering line*(kiri) dan Ilustrasi *cluster*(kanan)

3.5.2 Hitung *Mean* Tiang tiap *Cluster*

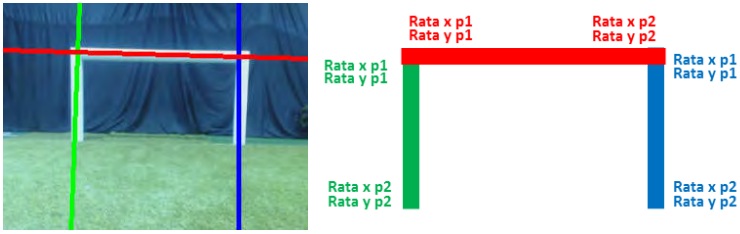
Hasil garis-garis yang sudah ter-*cluster* akan memudahkan untuk menghituang *mean* tiang tiap *cluster*. Dalam menghitung *mean* tiang *cluster* tiang, ada beberapa hal yang harus



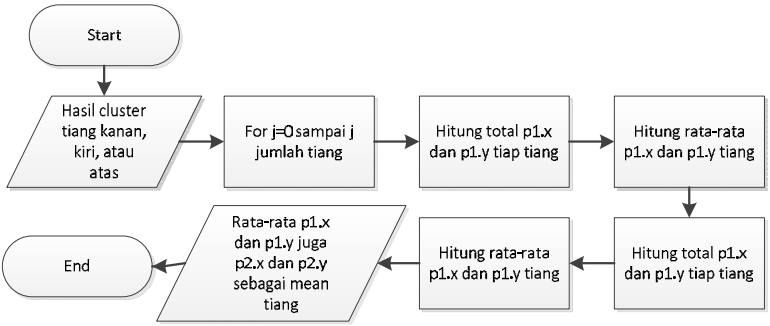
Gambar 3.14. Diagram alur proses *clustering line*

Dilakukan. Untuk lebih jelasnya, kita bisa ambil salah satu contoh bagaimana untuk melakukan *clustering* tiang kanan.

Hal yang pertama dilakukan adalah dengan cara menjumlahkan koordinat *X* dari P_1 tiang kanan kemudian dibagi banyaknya garis, begitu juga dengan menghitung mean koordinat *Y* dari P_1 dari menjumlahkan koordinat *Y* dari P_1 tiang kanan kemudian dibagi banyaknya garis. Setelah itu, menghitung *mean* tiang *cluster* kanan titik P_2 juga sama dengan seperti mencari mean *X* dari P_1, dan mean *Y* dari P_1. Rata-rata *X* dan mencari rata-rata Y dari titik P_2. Pencarian mean koordinat *X* dan *Y* dari P_1 dan P_2 dilakukan di tiap *cluster* kanan, *cluster* atas dan *cluster* kiri. Ilustrasi hasil hitung mean dan ilustrasi mean dapat dilihat pada Gambar 3.15. Diagram alur dari proses menghitung *mean* tiap *cluster* ditunjukkan pada Gambar 3.16.



Gambar 3.15. Citra hasil hitung *mean* (kiri) *dan Ilustrasi hitung mean* (kanan)



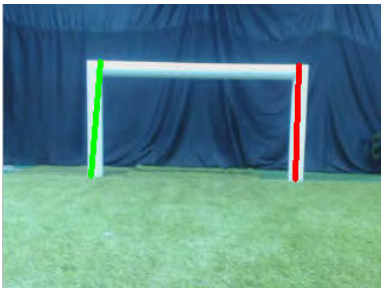
Gambar 3.16. Diagram alur hitung *mean*

3.5.3 *Drawing Line*

Hasil yang didapatkan dari menentukkan *mean* tiang tiap *cluster* tidak dapat mengetahui tinggi piksel dari objek gawang yang terdeteksi, untuk itu diperlukan sebuah proses unuk menggambarkan *line* terbaik dari tiap-tiap *cluster.*Proses *drawing line* terbaik dimulai dari mengetahui *X* dari garis terbaik tersebut. Setelah mendapatkannya, maka di cek pada citra *thresholding* yang sudah ada, apakah garis terbaik itu pada keadaan nilai RGB = 255, jika iya termasuk warna putih kemudian disimpan untuk batas bawah kemudian dilakukan perulangan sehingga sampai citra hasil *thresholding* tidak memiliki warna RGB = 255. Ilustrasi hasil *drawing line* dapat dilihat pada Gambar 3.17. Diagram alur proses *drawing line* juga dapat dilihat pada Gambar 3.18. Pada proses drawing line, line yang digambarkan hanya line dari tiang kiri dan tiang kanan saja. Hal ini dikarenakan, pendekatan pendeteksian gawang menggunakan tiang kanan dan kiri lebih mudah daripada pendekatan yang menggunakan tiang atas.

Alasan menggunakan pendekatan tiang kanan dan kiri, bukan atas adalah, keadaan *environment* pertandingan Kontes Robot Sepak Bola Indonesia sangat beragam, dan sangat tidak bisa ditebak akan ada *noise* seperti apa disana. Untuk meminimalkan kemungkinan *noise* warna putih selain gawang, seperti dinding berwarna putih, spanduk putih, atau baju berwarna putih, maka hal sederhana yang dapat dilakukan adalah dengan tidak menggerakan kamera robot ke atas untuk mencari gawang, melainkan dengan menggerakan kameranya ke kanan dan ke kiri gawang, dimana tiang kanan dan kiri gawang bersentuhan langsung dengan rumput.

Namun, tiang atas nantinya sangat membantu robot dalam menentukan tiang kanan atau kiri, jika pada keadaan robot hanya melihat satu tiang. Pendekatan jika terdeteksi satu tiang menggunakan informasi tiang atas belum di implenetasikan pada Tugas Akhir ini.



Gambar 3.17. Citra Hasil *Drawing Line*

3.5.4 Hitung Tinggi Tiang

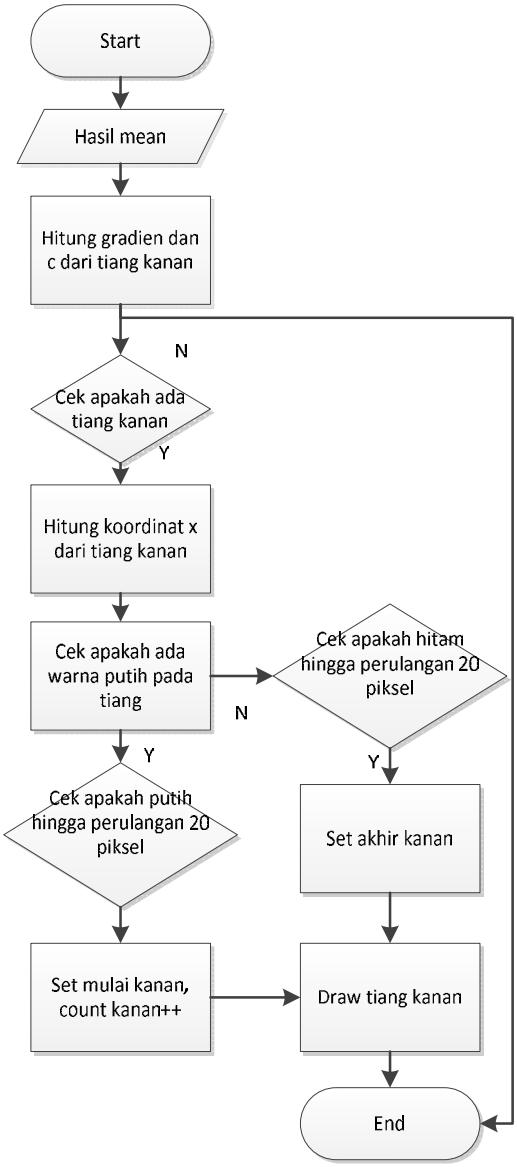
Pada proses *drawing line,* didapatkan juga dua *point*, berupa batas atas dan batas bawah dari tiap tiang kanan dan juga tiang kiri. Dalam menghitung tinggi tiang, dari *best line* tersebut bisa dihitung dengan cara menghitung jarak antara dua titik. Ilustrasi menghitung tinggi tiang dapat dilihat pada Gambar 3.19. Persamaan untuk menghitung jarak antara dua titik tiang kanan menggunakan rumus menghitung menggunakan *eucledian distance* dapat dilihat pada persamaan 3.2.

$$Jarak = \sqrt{(m\_kanan.x - a\_kanan.x)^2 + (m\_kanan.y - a\_kanan.y)^2} \quad (3.2)$$

Berdasarkan pada hasil citra Gambar 3.17. Sistem mendapatkan bahwa tinggi tiang kanan gawang yang didapatkan adalah 95,005 piksel, tinggi tiang kiri gawang yang didapatkan adalah 96,083 piksel.

3.5.5 Hitung Jarak Tiang ke Robot

Proses terakhir dalam tahap *postprocessing* adalah menghitung jarak tiang ke Robot. Penghitungan estimasi jarak robot terhadap tiang gawang diperoleh berdasarkan tinggi tiang



Gambar 3.18. Diagram alur *drawing line*

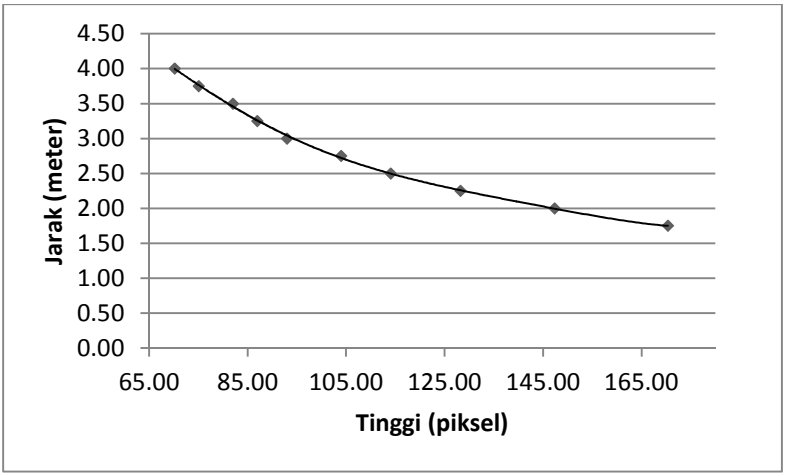
gawang dalam piksel dan jarak sebenarnya. Pada penelitian sebelumnya [22], persamaan ideal yang digunakan untuk mendapatkan jarak sebenarnya digunakan pendekatan regresi polynomial. Pada penelitian ini, sebelum mendapatkan persamaan ideal, dilakukan percobaan untuk mendapatkan data hubungan tinggi piksel dengan jarak sebenarnya. Tabel 3.1. menunjukkan hasil percobaan hubungan tinggi tiang dengan jarak sebenarnya.

Tabel 3.1. Hasil percobaan hubungan tinggi tiang dengan jarak

No	Tinggi (piksel)	Jarak sebenarnya (meter)
1	170,35	1,75
2	147,34	2,00
3	128,25	2,25
4	114,07	2,50
5	104,02	2,75
6	93,05	3,00
7	87,01	3,25
8	82,02	3,50
9	75,06	3,75
10	70,18	4,00

Setelah mendapatkan hasil percobaan antara hubungan tinggi gawang dalam piksel dan jarak sebenarnya, dilakukan perhitungan regresi polynomial dari data tersebut. Regeresi Polynomial orde ke 5 adalah orde yang paling cocok dari data yang diberikan. Sehingga, didapatkan persamaan ideal untuk perhitungan estimasi jarak robot dengan gawang. Hasil persamaan yang didapatkan dapat dilihat pada persamaan 3.3. Ilustrasi hasil regresi polynomial dari hasil percobaan berdasarkan Tabel 3.1 dapat dilihat pada gambar 3.19.

$$\text{Jarak} = (5{,}04\ x\ 10^{-10})X^5 - (2{,}94\ x\ 10^{-7})X^4 + (6{,}53\ x\ 10^{-5})X^3 - (6{,}60\ x\ 10^{-3})X^2 + 0{,}25948X + 2{,}0383 \quad (3.3)$$



Gambar 3.19. Hasil regresi polynomial orde 5

Berdasarkan pada hasil citra Gambar 3.17. dan menggunakan Persamaan 3.3. Hasil jarak yang didapatkan dari tiang kiri ke robot adalah 2,945 meter dan jarak dari tiang kanan ke robot adalah 2,978 meter.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dibahas mengenai implementasi yang dilakukan berdasarkan rancangan yang telah dijabarkan pada bab sebelumnya. Implementasi kode program dilakukan menggunakan bahasa C++.

4.1 Lingkungan Implementasi

Spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam implementasi ini ditampilkan pada Tabel 4.1.

Tabel 4.1. Spesifikasi Perangkat Lunak

Perangkat	Spesifikasi
Perangkat keras	Prosesor : 1.6 GHz Intel Atom Z530 (32 bit) Memori : 1 GB
Perangkat lunak	Sistem Operasi : Linux Perangkat Pembantu : Geany Bahasa Pemograman: C++ Library: OpenCV

4.2 Implementasi

Sub bab implementasi ini menjelaskan tentang implementasi proses yang sudah dijelaskan pada bab desain perangkat lunak.

4.2.1 Implementasi Tahap *Grayscale* Citra

Sub bab ini membahas implementasi tahap grayscale citra. Pada tahap ini data masukan berupa citra video berukuran 320 x 240 piksel dan data keluaran yang dihasilkan pada tahap

ini adalah citra yang telah diubah kedalam bentuk *grayscale.* Implementasi dilakukan dengan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu cvtColor() dan di tunjukkan oleh Kode Sumber 4.1.

```
1.  VideoCapture kamera(0);
2.  Mat src;
3.  gambar = src.clone();
4.  cvtColor(gambar, grayy, CV_BGR2GRAY);
```

Kode Sumber 4.1. *Grayscale* Citra

4.2.2 Implementasi Tahap *Thresholding* Citra

Sub bab ini membahas implementasi tahap *thresholding* citra. Pada tahap ini data masukan berupa citra video yang sudah di ubah dalam bentuk *grayscale* dan data keluaran yang dihasilkan pada tahap ini adalah citra segmentasi warna putih pada gawang yang sudah di *thresholding*. Implementasi dilakukan dengan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu threshold() dan di tunjukkan oleh Kode Sumber 4.2.

```
1.  threshold(grayy, thresh, 140, 255,
    THRESH_BINARY);
```

Kode Sumber 4.2. *Thresholding* Citra

Parameter grayy pada fungsi threshold adalah variabel yang menampung inputan citra. Parameter thresh adalah variabel yang menampung outputan citra. Angka 140 merupakan parameter batas bawah *thresholding,* sedangkan angka 255 merupakan parameter batas atas *thresholding*. Parameter THRESH_BINARY meruapakan parameter yang menunnjukkan bahwa *thresholding* yang dilakukan adalah termasuk *thresholding* binary, dimana objek yang diinginkan akan diberi nilai 1, sedangkan objek yang tidak diinginkan diberi nilai 0.

4.2.3 Implementasi Tahap Reduksi Noise Citra

Sub bab ini membahas implementasi tahap reduksi *noise* citra. Pada tahap ini data masukan berupa citra video yang sudah di *threholding* dan data keluaran yang dihasilkan pada tahap ini adalah citra yang sudah direduksi *noise*-nya. Implementasi dilakukan dengan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu morphologyEx() dan di tunjukkan oleh Kode Sumber 4.3.

```
1.  Mat element = getStructuringElement(MORPH_RECT,
    cv::Size(3,3));
2.  morphologyEx(thresh, noisee, MORPH_OPEN ,
    element);
```

Kode Sumber 4.3. Reduksi *Noise* Citra

4.2.4 Implementasi Tahap Deteksi Tepi Citra

Sub bab ini membahas implementasi tahap deteksi tepi citra. Pada tahap ini data masukan berupa citra video yang sudah di reduksi *noise*-nya dan data keluaran yang dihasilkan pada tahap ini adalah tepi citra dari gawang. Implementasi dilakukan dengan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu Canny() dan di tunjukkan oleh Kode Sumber 4.4.

```
1.  Canny(noisee, tepi, 10,3);
```

Kode Sumber 4.4. Deteksi Tepi Citra

4.2.5 Implementasi Tahap Transformasi Citra ke *Point*

Sub bab ini membahas implementasi tahap transformasi citra ke point. Pada tahap ini data masukan berupa citra video yang sudah di reduksi *noise*-nya dan data keluaran yang dihasilkan pada tahap ini adalah citra yang sudah di transformasikan ke dalam bentuk *point-point*. Implementasi dilakukan secara *from scratch* dan di tunjukkan oleh Kode Sumber 4.5.

```
1.  for(int x=0; x<tepi.cols;x++) {
2.        for(int y = 0; y<tepi.rows; y++){
3.              if(tepi.at<uchar>(y,x) == 255){
4.                    points.push_back(Point(x,y));
5.              }
6.        }
7.  }
```

Kode Sumber 4.5. Transformasi Citra ke *Point*

4.2.6 Implementasi Tahap Deteksi Garis

Sub bab ini membahas implementasi tahap deteksi garis. Pada tahap ini data masukan berupa citra video yang sudah di transformasikan ke dalam bentuk *point* dan data keluaran yang dihasilkan pada tahap ini adalah garis-garis yang terdeteksi menggunakan metode Algoritma *Ransac*. Implementasi dilakukan secara *from scratch* mengikuti Algoritma *Ransac* yang sudah ada [5], kemudian dikembangkan pada program C++ dan di tunjukkan oleh Kode Sumber 4.6.

```
1.  for(i=0; i < N ; i++)
2.  {
3.        int inliers = 0;
4.        int index1 = rand() % points.size();
5.        int index2 = rand() % points.size();
6.        Point p1 = points[index1];
7.        Point p2 = points[index2];
8.        double a = p1.y - p2.y;
9.        double b = p1.x - p2.x;
10.       double jarak = pow(a * a + b * b, 0.5 );

11.       if (jarak < batas_jarak)
12.             continue;
13.       double gradien1 = 0;
14.       double c1 = 0;
15.       bool m_takhingga = false;
16.       if (p2.x == p1.x)
17.             m_takhingga = true;
18.       else{
19.             gradien1 = double(p2.y - p1.y) /
    double(p2.x - p1.x) * 1.0;
```

```
20.                 c1 = p1.y - gradien1 * p1.x;
21.         }

22.         if(gradien1 == 0)
23.                 continue;
24.         for(int j=0 ; j < points.size(); j++)
25.         {
26.                 Point p0 = points[j];
27.                 double x = p0.x;
28.                 double y = p1.y;
29.                 if (m_takhingga)
30.                 {
31.                         red = 0;
32.                         green = 255;
33.                         blue = 0;
34.                         x = p1.x;
35.                 }
36.                 else if (gradien1 == 0)
37.                 {
38.                         red = 0;
39.                         green = 0;
40.                         blue = 255;
41.                         y = p1.y;
42.                 }
43.                 else
44.                 {
45.                         red = 255;
46.                         green = 0;
47.                         blue = 0;
48.                         double   c2   =   p0.y   +
    p0.x/gradien1;
49.                         x = (c2 -c1) / (gradien1 +
    1.0/gradien1);
50.                         y = gradien1 * x + c1;
51.                 }

52.                 double c = y - p0.y;
53.                 double d = x - p0.x;
54.                 double distance = pow(c * c + d *
    d , 0.5);
55.                 if (distance < threshold_inliers)
56.                         inliers++;
57.         }
58.         int yatas = 0;
```

```
59.         int ybawah = tepi.rows;
60.         int xatas = (c1 / gradien1) * -1;
61.         int xbawah= (tepi.rows - c1) / gradien1;

62.         if(inliers > min_inliers)
63.         {
64.               if (xatas < 0)
65.               {
66.                     xatas = 0;
67.                     yatas = c1;
68.               }
69.               if (xatas > tepi.cols)
70.               {
71.                     xatas = tepi.cols;
72.                     yatas    =    tepi.cols    *
    gradien1 + c1;
73.               }
74.               if (xbawah < 0)
75.               {
76.                     xbawah = 0;
77.                     ybawah = c1;
78.               }
79.               if (xbawah > tepi.cols)
80.               {
81.                     xbawah = tepi.cols;
82.                     ybawah    =    tepi.cols    *
    gradien1 + c1;
83.               }
84.               if (yatas > tepi.rows)
85.               {
86.                     yatas = tepi.rows;
87.                     xatas = (tepi.rows - c1) /
    gradien1;
88.               }
89.               if (yatas < 0)
90.               {
91.                     yatas = 0;
92.                     xatas = (c1 / gradien1) * -
93.   1;
94.               }
95.               if (ybawah < 0)
96.               {
97.                     ybawah = 0;
                      xbawah = (c1 / gradien1) *
```

```
98.  -1;
99.                  }
100.                 if (ybawah > tepi.rows)
101.                 {
102.                        ybawah = tepi.rows;
103                         xbawah = (tepi.rows - c1) /
     gradien1;
104.                 }

105.                 best_p1.push_back(Point(xatas,
     yatas));
106.                 best_p2.push_back(Point(xbawah,
     ybawah));
107.          }
108. }
```

Kode Sumber 4.6. Deteksi Garis

4.2.7 Implementasi Tahap *Clustering Line*

Sub bab ini membahas implementasi tahap *clustering* line. Pada tahap ini data masukan berupa citra video yang sudah mendeteksi garis objek gawang dan data keluaran yang dihasilkan pada tahap ini adalah garis-garis tadi sudah ter-*cluster*. Implementasi dilakukan secara *from scratch* dan di tunjukkan oleh Kode Sumber 4.7.

```
1.  for(int j=0; j < best_p1.size(); j++)
2.  {
3.  if(best_p1[j].x  >  rata_x  &&  best_p2[j].x  >
    rata_x)
4.  {
5.        if(best_p1[j].y > best_p2[j].y)
6.        {
7.              tkanan_p1.push_back(best_p1[j]);
8.              tkanan_p2.push_back(best_p2[j]);
9.        }
10.       else if (best_p1[j].y < best_p2[j].y)
11.       {
12.             tkanan_p1.push_back(best_p2[j]);
13.             tkanan_p2.push_back(best_p1[j]);
14.       }
```

```
15.  }
16.  else if (best_p1[j].x < rata_x && best_p2[j].x <
     rata_x)
17.  {
18.         if(best_p1[j].y > best_p2[j].y)
19.         {
20.                tkiri_p1.push_back(best_p1[j]);
21.                tkiri_p2.push_back(best_p2[j]);
22.         }
23.         else if(best_p1[j].y < best_p2[j].y)
24.         {
25.                tkiri_p1.push_back(best_p2[j]);
26.                tkiri_p2.push_back(best_p1[j]);
27.         }
28.  }
29.  else (best_p1[j].y < rata_y && best_p2[j].y <
     rata_y)
30.  {
31.         if(best_p1[j].x < best_p2[j].x)
32.         {
33.                tatas_p1.push_back(best_p1[j]);
34.                tatas_p2.push_back(best_p2[j]);
35.         }
36.         else if(best_p1[j].x > best_p2[j].x)
37.         {
38.                tatas_p1.push_back(best_p2[j]);
39.                tatas_p2.push_back(best_p1[j]);
40.         }
41.  }
42.  }
```

Kode Sumber 4.7. *Clustering Line*

4.2.8 Implementasi Tahap Hitung Mean Tiap Cluster

Sub bab ini membahas implementasi tahap hitung mean tiap *cluster*. Pada tahap ini data masukan berupa citra video dengan garis-garis yang sudah ter-*cluster* dan data keluaran yang dihasilkan pada tahap ini adalah citra dengan garis terbaik di tiap *cluster* tiang kanan, tiang kiri dan tiang atas. Implementasi dilakukan secara *from scratch* dan di tunjukkan oleh Kode Sumber 4.8.

```
1.  for(int j=0; j < best_p1.size(); j++)
2.  {
3.  int tengahp_x = best_p1[j].x + (best_p2[j].x -
    best_p1[j].x) / 2;
4.  int tengahp_y = best_p1[j].y + (best_p2[j].y -
    best_p1[j].y) / 2;

5.          if(x_min == -1 || x_min > tengahp_x)
6.          x_min = tengahp_x;

7.          if(x_max == -1 || x_max < tengahp_x)
8.          x_max = tengahp_x;

9.          if(y_min == -1 || y_min > tengahp_y)
10.         y_min = tengahp_y;

11.         if(y_max == -1 || y_max < tengahp_y)
12.         y_max = tengahp_y;
13. }
14. rata_x = (x_max - x_min) / 2 + x_min;
15. rata_y = (y_max - y_min) / 2 + y_min;

16. for(int j=0; j < tkiri_p1.size(); j++)
    {
17. sum_x_p1_kiri = sum_x_p1_kiri + tkiri_p1[j].x;
18. sum_y_p1_kiri = sum_y_p1_kiri + tkiri_p1[j].y;
    }
19. sum_x_p1_kiri = sum_x_p1_kiri / tkiri_p1.size();
20. sum_y_p1_kiri = sum_y_p1_kiri / tkiri_p1.size();

21. for    (int j=0; j < tkiri_p2.size(); j++)
22. {
23. sum_x_p2_kiri = sum_x_p2_kiri + tkiri_p2[j].x;
24. sum_y_p2_kiri = sum_y_p2_kiri + tkiri_p2[j].y;
25. }
26. sum_x_p2_kiri = sum_x_p2_kiri / tkiri_p2.size();
27. sum_y_p2_kiri = sum_y_p2_kiri / tkiri_p2.size();

28. for(int j= 0; j< tkanan_p1.size(); j++)
29. {
30. sum_x_p1_kanan      =      sum_x_p1_kanan      +
    tkanan_p1[j].x;
31. sum_y_p1_kanan      =      sum_y_p1_kanan      +
    tkanan_p1[j].y;
```

```
32.  }
33.  sum_x_p1_kanan      =      sum_x_p1_kanan      /
     tkanan_p1.size();
34.  sum_y_p1_kanan      =      sum_y_p1_kanan      /
     tkanan_p1.size();

35.  for( int j =0; j<tkanan_p2.size(); j++)
36.  {
37.  sum_x_p2_kanan      =      sum_x_p2_kanan      +
     tkanan_p2[j].x;
38.  sum_y_p2_kanan      =      sum_y_p2_kanan      +
     tkanan_p2[j].y;
39.  }

40.  sum_x_p2_kanan          =          sum_x_p2_kanan
     /tkanan_p2.size();
41.  sum_y_p2_kanan          =          sum_y_p2_kanan
     /tkanan_p2.size();
42.
43.  for(int j =0; j <tatas_p1.size(); j++)
     {
44.  sum_x_p1_atas = sum_x_p1_atas + tatas_p1[j].x;
45.  sum_y_p1_atas = sum_y_p1_atas + tatas_p1[j].y;
46.  }
47.  sum_x_p1_atas = sum_x_p1_atas / tatas_p1.size();
48.  sum_y_p1_atas = sum_y_p1_atas / tatas_p1.size();

49.  for(int j = 0; j < tatas_p2.size(); j++)
50.  {
51.  sum_x_p2_atas = sum_x_p2_atas + tatas_p2[j].x;
52.  sum_y_p2_atas = sum_y_p2_atas + tatas_p2[j].y;
53.  }
54.  sum_x_p2_atas = sum_x_p2_atas / tatas_p2.size();
55.  sum_y_p2_atas = sum_y_p2_atas / tatas_p2.size();

56.  line(tiang_kanan,Point(sum_x_p1_kanan,
     sum_y_p1_kanan),Point(sum_x_p2_kanan,
     sum_y_p2_kanan), Scalar(255, 0, 0), 3, 1, 0);

57.  line(tiang_kanan,Point(sum_x_p1_kiri,
     sum_y_p1_kiri),Point(sum_x_p2_kiri,
     sum_y_p2_kiri), Scalar(0, 255, 0), 3, 1, 0);
```

```
58.  line(tiang_kanan,Point(sum_x_p1_atas,
     sum_y_p1_atas),Point(sum_x_p2_atas,
     sum_y_p2_atas), Scalar(0, 0, 255), 3, 1, 0);
```

Kode Sumber 4.8. Hitung *Mean* Tiap *Cluster*

4.2.9 Implementasi Tahap *Drawing Line*

Sub bab ini membahas implementasi tahap *drawing line*. Pada tahap ini data masukan berupa citra video dengan garis terbaik di tiap *cluster* dan data keluaran yang dihasilkan pada tahap ini adalah citra dengan garis terbaik sesuai dengan tinggi piksel gawang. Implementasi dilakukan dengan menggunakan fungsi yang sudah disediakan oleh OpenCV yaitu line() dan di tunjukkan oleh Kode Sumber 4.9.

```
1.   for(j=tepi.rows -1; j >= 0; j--)
2.   {
3.   if(tkanan_p1.size() > 0 && !kanan_ok)
4.   {
5.         int x_tiang_kanan = (j - c_tiang_kanan)
     / gradien_t_kanan;
6.         if(thresh.at<uchar>(j,x_tiang_kanan)   ==
     255)
7.         {
8.         if(!lagi_putih_kanan)
9.         {
10.              lagi_putih_kanan = true;
11.        }
12.        else
13.        {
14.        count_putih_kanan++;
15.        if(count_putih_kanan > limitpixel)
16.        {
17.              if(!tg_kanan)
18.              {
19.              tg_kanan = true;
20.              mulai_kanan = Point(x_tiang_kanan,
     j + 20);
21.              }
22.        }
23.        }
```

```
24.         fixkanan = 0;
25.         dapet_lastj_kanan = false;
26.         }
27.         else
28.         {
29.               count_putih_kanan = 0;
30.               lagi_putih_kanan = false;
31.               if(tg_kanan && !dapet_lastj_kanan)
32.               {
33.                     dapet_lastj_kanan = true;
34.                     lastj_kanan = j + 1;
35.               }
36.               if(lastj_kanan >= 0)
37.               {
38.                     if(fixkanan > limitpixel)
39.                     {
40.                           akhir_kanan         =
      Point(x_tiang_kanan, lastj_kanan);
                            kanan_ok = true;
41.                     }
42.                     else
43.                     {
44.                           fixkanan++;
45.                           if(j == 0)
46.                           akhir_kanan         =
      Point(x_tiang_kanan, lastj_kanan);
47.               }
48.                     }
49.         }
50.   }
51.         if(tkiri_p1.size() > 0 && !kiri_ok)
52.         {
53.         int x_tiang_kiri = (j - c_tiang_kiri) /
      gradien_t_kiri;
54.         if(thresh.at<uchar>(j,x_tiang_kiri)    ==
      255)
55.         {
56.         if(!lagi_putih_kiri)
57.         {
58.               lagi_putih_kiri = true;
59.         }
60.         else
61.         {
62.               count_putih_kiri++;
```

```
63.                 if(count_putih_kiri > limitpixel)
64.                 {
65.                 if(!tg_kiri)
66.                 {       //printf("j\n");
67.                         tg_kiri = true;
68.                         mulai_kiri                  =
69.   Point(x_tiang_kiri, j + 20);
70.                 }
71.                 }
72.         }
73.         fixkiri = 0;
74.         dapet_lastj_kiri = false;
75.         }
76.         else
77.         {
78.                 count_putih_kiri = true;
79.                 lagi_putih_kiri = false;
80.                 if(tg_kiri && !dapet_lastj_kiri)
81.                 {
82.                         dapet_lastj_kiri = true;
83.                         lastj_kiri = j + 1;
84.                 }
85.                 if(lastj_kiri >= 0)
86.                 {
87.                         if(fixkiri > limitpixel)
88.                         {
89.                                 akhir_kiri          =
      Point(x_tiang_kiri, lastj_kiri);
                                  kiri_ok = true;
90.                         }
91                          else
92.                         {
93.                                 fixkiri++;
94.                                 if(j == 0)
95.                                 akhir_kiri          =
      Point(x_tiang_kiri, lastj_kiri);
96.                         }
97.                 }
98.         }
99.   }
100.  }
```

Kode Sumber 4.9. *Drawing Line*

4.2.10 Implementasi Tahap Hitung Tinggi Tiang

Sub bab ini membahas implementasi tahap hitung tinggi tiang. Pada tahap ini data masukan berupa citra video dengan garis terbaik di tiap *cluster* dan data keluaran yang dihasilkan pada tahap ini adalah tinggi tiang dari gawang yang terdeteksi. Implementasi dilakukan secara *from scratch* dan di tunjukkan oleh Kode Sumber 4.9.

```
1.  double ab = mulai_kanan.y - akhir_kanan.y ;
2.  double bc = mulai_kanan.x- akhir_kanan.x;
3.  double tinggi_kanan = pow(ab * ab + bc * bc,
4.  0.5);
    printf("tinggi tiang kanan : %f pixel\n",
    tinggi_kanan);
5.
6.  double cd = mulai_kiri.y - akhir_kiri.y ;
7.  double de = mulai_kiri.x- akhir_kiri.x;
8.  double tinggi_kiri = pow(cd * cd + de * de, 0.5);
    printf("tinggi tiang kiri : %f pixel\n",
    tinggi_kiri);
```

Kode Sumber 4.10. Hitung Tinggi Tiang

4.2.11 Implementasi Tahap Hitung Jarak Tiang ke Robot

Sub bab ini membahas implementasi tahap hitung jarak tiang ke robot. Pada tahap ini data masukan berupa citra video dengan tinggi tiang yang sudah terdeteksi dan data keluaran yang dihasilkan pada tahap ini adalah citra yang sudah dihitung jarak dari tinggi tiang yang terdeteksi. Implementasi dilakukan dengan cara *from scratch* dan di tunjukkan oleh Kode Sumber 4.10.

```
1.  #define apoly 5.0410362879098070e-010
2.  #define bpoly -2.9457583806258708e-007
3.  #define cpoly 6.5349044300403393e-005
4.  #define dpoly -6.6090443712097250e-003
5.  #define epoly 2.5948650626820319e-001
6.  #define fpoly 2.0383272015271494e+000
```

```
7.  double distance = apoly*pow(tinggi_kanan, 5) +
    bpoly*pow(tinggi_kanan,          4)          +
    cpoly*pow(tinggi_kanan,3)                    +
    dpoly*pow(tinggi_kanan, 2) + epoly*tinggi_kanan
    + fpoly;
8.  printf("jarak tiang kanan : %f\n", distance);

9.  double distance_kiri = apoly*pow(tinggi_kiri, 5)
    +       bpoly*pow(tinggi_kiri,       4)       +
    cpoly*pow(tinggi_kiri,3)                     +
    dpoly*pow(tinggi_kiri, 2) + epoly*tinggi_kiri +
10. fpoly;
    printf("jarak     tiang     kiri     :     %f\n",
    distance_kiri);
```

Kode Sumber 4.11. Hitung Jarak Tiang ke Robot

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Padabab ini akan dijelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi

5.1 Lingkungan Uji Coba

Sebelumnya, perlu diketahui lingkungan uji coba, baik perangkat keras maupun perangkat lunak, yang digunakan pada uji coba Tugas Akhir ini. Lingkungan tersebut ditunjukkan pada Tabel 5.1 berikut ini.

Tabel 5.1. Spesifikasi lingkungan uji coba

Perangkat	Spesifikasi
Perangkat keras	Prosesor : 1.6 GHz Intel Atom Z530 (32 bit) Memori : 1 GB
Perangkat lunak	Sistem Operasi : Linux Perangkat Pembantu : Geany Bahasa Pemograman : C++ Library : OpenCV

5.2 Data Uji Coba

Data uji coba yang digunakan untuk pendeteksian gawang berdasarkan Algoritma *Ransac* adalah potongan gambar gawang dengan *environment* lapangan sepak bola robot. Kualitas gambar yang digunakan adalah gambar dengan *size* 320 x 240 piksel dan memiliki *channel* R, G, B. Data gambar diambil dari berbagai posisi jarak yang berbeda dari gawang sesuai dengan lingkungan pertandingan Kontes Robot Sepak Bola Indonesia 2016. Jumlah gambar yang diambil adalah 96 gambar berbeda. Dimana 15 buah gambar digunakan sebagai data uji algoritma pengenalan gawang dengan pergantian parameter *thresholding* dan parameter

variabel pada Algoritma *Ransac*. 40 buah sebagai data uji tinggi dan jarak gawang. Dan sisanya adalah 40 buah sebagai data uji waktu eksekusi algoritma pendeteksian gawang.

5.3 Skenario Uji Coba

Pada sub bab ini akan dijelaskan mengenai skenario uji coba yang telah dilakukan. Telah dilakukan beberapa skenario uji coba, diantaranya yaitu :

1. Perbandingan hasil uji coba parameter yang dilakukan pada proses *thresholding* dan parameter variabel deteksi garis menggunakan Algoritma *Ransac* dalam pendeteksian gawang.
2. Perbandingan performa antara Algoritma *Ransac* dengan Metode *Hough Transform* dalam pendeteksian gawang. Perbandingan performa meliputi pengujian ketepatan jarak dan kecepatan waktu eksekusi program.

5.3.1 Skenario Uji Coba 1

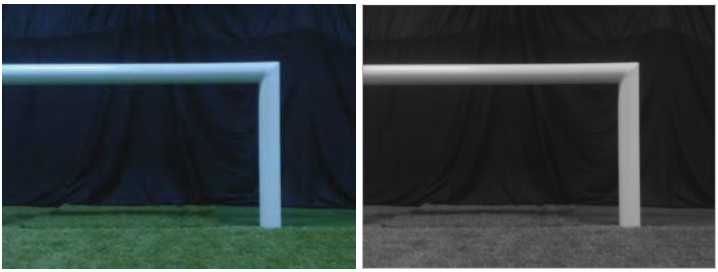
Skenario uji coba 1 adalah perbandingan hasil uji coba parameter yang dilakukan pada proses *thresholding* dan parameter yang dilakukan pada deteksi garis menggunakan Algoritma *Ransac.* Pada tahap *thresholding,* parameter yang diubah adalah parameter batas bawah, dan batas atas citra.

Sebelum masuk pada hasil uji coba *thresholding.* Citra yang dipersiapkan adalah citra yang sudah diproses terlebih dahulu dalam proses pengubahan *grayscale* citra. Ilustrasi citra masukan dan citra *grayscale Image-1* dapat dilihat pada Gambar. 5.1

5.3.1.1 Parameter *thresholding*

Dari hasil uji yang dilakukan, pada tahap *thresholding,* parameter yang paling baik adalah dengan batas bawah 140 dan batas atas 255. Parameter yang baik pada setiap citra akan

berubah, sangat bergantung dengan intensitas citra inputan. Untuk itu, jika metode ini diterapkan pada robot, diperlukan data kalibrasi warna objek untuk membantu segmentasi warna *thresholding* citra [1]. Pada Tabel 5.2. diperlihatkan beberapa perbandingan parameter batas atas dan batas bawah pada citra gambar *Image-1*. Pada Tabel 5.3. diperlihatkan hasil uji coba parameter batas bawah *thresholding* sebesar 140 di berbagai citra gambar. Pada Lampiran B, pada Tabel B.1 diperlihatkan hasil *thresholding* di berbagai citra dengan batas *threshold* yang bermacam-macam sampai dengan hasil keluaran terkahir berupa garis yang dideteksi oleh Algoritma *Ransac.*



Gambar 5.1. Citra masukan (kiri) Citra *grayscale* (kanan)

Tabel 5. 2. Parameter *thresholding* citra pada *Image-1*

No	Nama Gambar	Batas Bawah	Batas Atas	Keterangan
1	*Image-1*	20	255	Terlalu banyak *noise* pada citra
2	*Image-1*	40	255	Banyak *noise* pada citra
3	*Image-1*	60	255	Banyak *noise* pada citra
4	*Image-1*	80	255	Banyak *noise* pada citra
7	*Image-1*	100	255	Banyak *noise* pada citra
8	*Image-1*	120	255	Masih ada *noise* pada citra
9	*Image-1*	130	255	*Noise* Citra tinggal sedikit
10	*Image-1*	140	255	Tidak ada *noise*
11	*Image-1*	160	255	Citra gawang terdegradasi
12	*Image-1*	170	255	Hanya ada tiang atas gawang
13	*Image-1*	200	255	Hasil *thresholding* sedikit

12	*Image-1*	210	255	Hasil *thresholding* sedikit
13	*Image-1*	230	255	Tidak ada hasil *thresholding*

Ilustrasi hasil parameter *thresholding* dengan batas bawah 100 dengan batas bawah terbaik dengan nilai 140 ditunjukkan pada Gambar 5.2. Dimana dengan parameter *thresholding* 100, citra masih memiliki banyak *noise*, sedangkan pada parameter *thresholding* 140, citra hasil sudah ideal.

Gambar 5.2. Hasil *thresholding* batas bawah 100 (kiri) dan batas bawah 140 (kanan)

Tabel 5.3. Parameter *thresholding* beberapa citra

No	Nama Gambar	Batas Bawah	Batas Atas	Keterangan
1	*Image-1*	140	255	Tidak ada *noise*
2	*Image-2*	140	255	Tidak ada *noise*
3	*Image-3*	140	255	Tidak ada *noise*
4	*Image-4*	140	255	Tidak ada *noise*
7	*Image-5*	140	255	Tidak ada *noise*
8	*Image-6*	140	255	Tidak ada *noise*
9	*Image-7*	140	255	Tidak ada *noise*
10	*Image-8*	140	255	Tidak ada *noise*
11	*Image-9*	140	255	Tidak ada *noise*
12	*Image-10*	140	255	Tidak ada *noise*
13	*Image-11*	140	255	Citra gawang terdegradasi
12	*Image-12*	140	255	Hanya ada tiang atas gawang
13	*Image-13*	140	255	Hasil *thresholding* sedikit

14	*Image-14*	140	255	Hasil *thresholding* sedikit
15	*Image-15*	140	255	Tidak ada hasil *thresholding*

5.3.1.2 Parameter Deteksi Garis Algortima *Ransac*

Setelah mendapatkan parameter *thresholding* terbaik sebesar 140. Kemudian citra dilakukan beberapa tahap *preprocessing* lainnya, sebelum dilanjutkan dengan proses deteksi garis oleh Algoritma *Ransac*. Tahap *preprocessing* yang harus dilakukan adalah mereduksi *noise* hasil citra *thresholding*, kemudian deteksi tepi citra, dan selanjutnya adalah transformasi citra ke *point.* Ilustrasi hasil reduksi *noise*, dan deteksi tepi citra dari Gambar *Image-1* ditunjukkan pada Gambar 5.3.

Gambar 5.3. Citra hasil reduksi *noise* (kiri) dan Citra hasil deteksi tepi (kanan)

Dari hasil uji yang dilakukan, pada tahap deteksi garis menggunakan Algoritma *Ransac,* ada beberapa parameter yang di set, diantaranya paramater N yang merepresentasikan banyak perulangan, parameter batas jarak yang merepresentasikan jarak antara P_1 dan P_2, *threshold inliers* yang merepresentasikan jarak P_0 ke garis P_1, P_2 dan *minimum inliers* yang merepresentasikan jumlah minimum *inliers* suatu garis. Parameter yang paling baik adalah dengan *N* sebesar 400, *batas jarak* sebesar 40, *threshold inliers* sebesar 5 dan parameter *minimum inliers* sebesar 85.

Parameter *N* digunakan untuk merepresentasikan jumlah perulangan untuk mendapatkan garis yang akan dibentuk sesuai

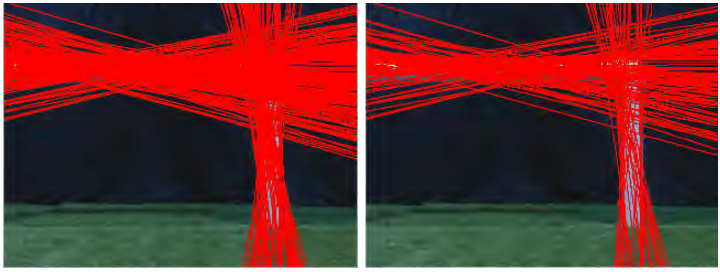
dengan Algoritma *Ransac*. Pada Tabel 5.4 diperlihatkan beberapa perbandingan parameter *N*.

Tabel 5.4. Parameter Perulangan *(N)* minimum garis

No	Besarnya N	Keterangan
1	100	Terlalu sedikit garis yang terbuat
2	200	Garis yang terbuat pada tiang kanan dan tiang kiri terlalu sedikit
3	300	Garis yang terbuat belum ideal
4	400	Garis yang terbuat sudah ideal
5	1000	Terlalu banyak garis yang terbuat

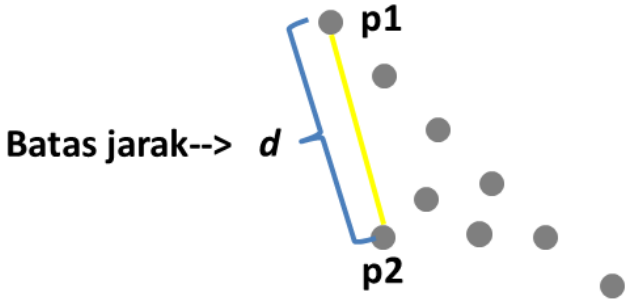
Ilustrasi hasil parameter nilai *N* sebesar 1000 dengan nilai *N* terbaik sebesar 400 ditunjukkan pada Gambar 5.4. Dimana dengan parameter *N* sebesar 1000, garis yang dihasilkan terlalu sedikit, sedangkan dengan parameter 400, garis yang terbentuk cukup ideal. Pemilihan parameter *N* sebesar 400 didasari oleh hasil keluaran tinggi tiang dan hasil drawing line yang tidak terlalu jauh, artinya dengan parameter *N* sebesar 400 sudah mewakili himpunan garis untuk proses deteksi garisnya.

Hal lain yang mendasari adalah pada kecepatan proses deteksi garis, apabila menggunakan parameter *N* sebesar 100, program deteksi garis akan lebih lama, sedangkan ketika menggunakan parameter *N* sebesar 400, program deteksi garis yang dilakukan lebih cepat dibandingkan dengan parameter *N* diatas 400.



Gambar 5.4. Hasil Parameter *N*=1000 (kiri) dan *N*= 400 (kanan)

Setelah mendapatkan parameter *N* terbaik sebesar 400, parameter selanjutnya yang di set adalah parameter *batas jarak*. *Batas jarak* pada Algoritma *Ransac* merepresentasikan jarak minimum yang digunakan sebagai persyaratan jarak antara *point* P_1 dan *point* P_2. Ilustrasi parameter batas jarak dapat dilihat pada Gambar 5.5.



Gambar 5.5. Ilustrasi parameter batas jarak

Parameter batas jarak terbaik adalah 40, ketika jarak *point* P_1 dengan *point* P_2 lebih dari 40, maka tidak akan dibuat *line* P_1, P_2. Sedangkan, jika jaraknya memenuhi, maka *point* P_1 dan P_2 akan di proses menjadi *line*. Pada Tabel 5.5. diperlihatkan beberapa perbandingan parameter batas jarak yang sudah dilakukan.

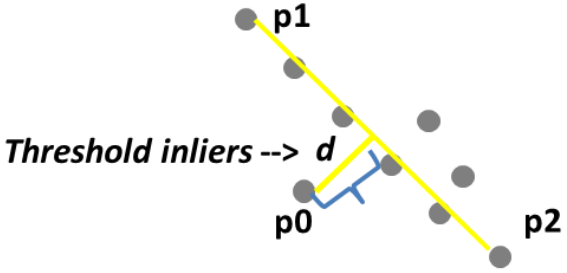
Tabel 5.5. Parameter minimum batas jarak P_1 dengan P_2

No	Batas jarak	Keterangan
1	10	Terlalu banyak garis yang dibuat
2	20	Terlalu banyak garis yang dibuat
3	40	Garis yang dibuat ideal
4	70	Sedikit garis yang dibuat
5	100	Terlalu sedikit garis yang dibuat

Ilustrasi hasil nilai parameter dengan *batas jarak* 100 dengan *batas jarak* terbaik dengan nilai 40 ditunjukkan pada Gambar 5.6. Dimana dengan parameter *batas jarak* 100, garis yang dihasilkan tidak mewakili semua tiang, sedangkan dengan batas jarak 40, garis yang terbentuk mewakili.

Gambar 5.6. Hasil Parameter batas jarak = 100 (kiri) dan batas jarak = 40 (kanan)

Setelah parameter *N* terbaik sebesar 400 dan batas jarak terbaik sebesar 40, selanjutnya parameter pada deteksi garis yang di set adalah parameter *threshold inliers*, dimana parameter ini merepresentasikan syarat jarak *point* P_0 yang dipilih dengan *line* P_1, P_2 yang sudah dibuat. Apabila jarak p_0 ke *line* memenuhi, maka *point* P_0 termasuk *inliers*, sedangkan jika tidak memenuhi, *point* P_0 tidak dimasukkan kedalam *inliers*. Ilustrasi parameter threshold inliers dapat dilihat pada Gambar 5.7. Pada Tabel 5.6 diperlihatkan beberapa perbandingan parameter *threshold inliers.*

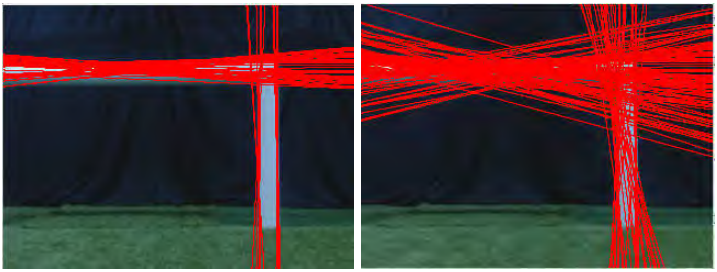


Gambar 5.7. Ilustrasi parameter *threshold inliers*

Tabel 5.6. Parameter minimum *threshold inliers*

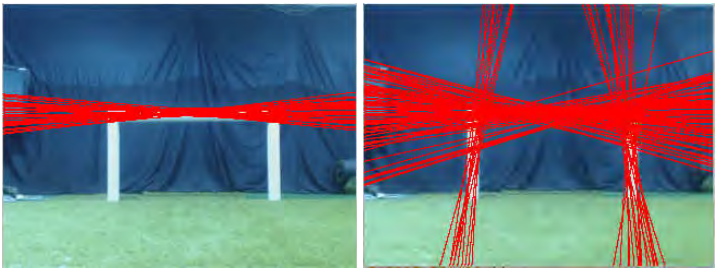
No	*Threshold inliers*	Keterangan
1	2	Sedikit garis yang terbentuk
2	5	Garis yang terbentuk ideal
3	10	Terlalu banyak garis yang terbentuk

Ilustrasi hasil nilai parameter *threshold inliers* 2 dengan *threshold inliers* terbaik dengan nilai 5 ditunjukkan pada Gambar 5.8. Dimana, dengan parameter *threshold inliers* 2, garis yang dihasilkan masih terlalu sedikit, sedangkan dengan parameter *threshold inliers* sebesar 5, garis yang dihasilkan ideal. Mengapa *threshold inliers* 2 tidak diambil, dikarenakan, tidak semua gambar dengan *threshold inliers* 2 mewakili minimum garis untuk dicari meannya, ada beberapa garis yang dihasilkan sangat sedikit dan bahkan tidak terwakilkan, Ilustrasi tersebut dapat dilihat pada Gambar 5.9. Untuk itu, *threshold inliers* terbaik yang diambil adalah dengan nilai 5.



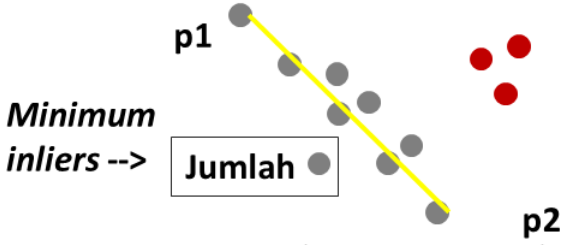
Gambar 5. 8.Hasil Parameter *threshold inliers* = 2 (kiri) dan *threshold inliers* = 5 (kanan)

Dengan parameter *N* sebesar 400, parameter batas jarak sebesar 40, parameter *threshold inliers* sebesar 5, parameter terakhir yang diatur pada Algoritma *Ransac* adalah *minimum inliers*. *Minimum inliers* merepresentasikan jumlah *minimum inliers point* yang terbentuk untuk menjadi satu garis. Apabila jumlah *inliers* yang terkumpul untuk membuat garis kurang,



Gambar 5. 9. Hasil Citra Berbeda Parameter *threshold inliers* = 2 (kiri) dan *threshld inliers* = 5 (kanan)

maka garis dari *point* P_1 dan P_2 tidak akan dibuat dan ditampung pada variabel *best line*. Sebaliknya, jika sebuah garis memenuhi *minimum inliers*-nya, maka garis dari *point* P_1 dan P_2 dimasukkan pada tampungan *best line*. Ilustrasi parameter *minimum inliers* ditunjukkan pada Gambar 5.10. Pada Tabel 5.7 diperlihatkan beberapa perbandingan parameter *minimum inliers.*

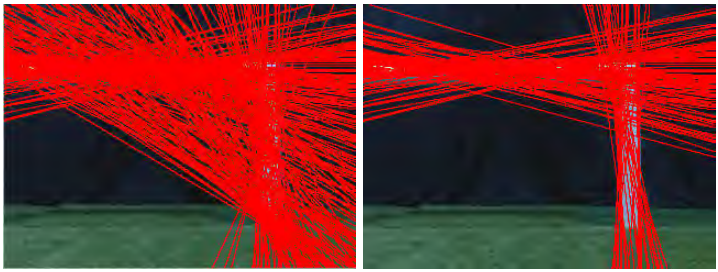


Gambar 5.10. Ilustrasi parameter *minimum inliers*

Tabel 5.7. Parameter *minimum inliers* garis

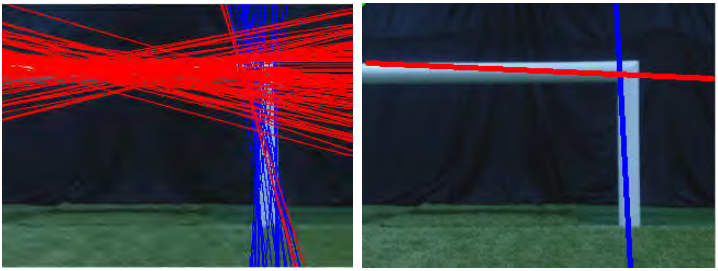
No	*Minimum inliers*	Keterangan
1	20	Terlalu banyak garis yang terbuat
2	30	Banyak garis yang terbuat
3	40	Banyak garis yang terbuat
4	60	Cukup banyak garis yang terbuat
5	85	Garis yang terbuat ideal
6	100	Garis yang terbuat sedikit
7	200	Garis yang terbuat hanya di tiang atas saja
8	1000	Tidak ada garis yang terbuat

Ilustrasi hasil nilai parameter *minimum inliers* 10 dengan *minimum inliers* terbaik 85 ditunjukkan pada Gambar 5.11. Dimana, dengan parameter *minimum inliers* = 10, garis yang dibuat terlalu banyak, sedangkan dengan *minimum inliers* sebesar 85, garis yang terbuat ideal. Pengambilan *minimum inliers* dengan nilai 85 adalah didasari dengan jumlah garis yang terdeteksi, jika nilai *minimum inliers* di set kecil, maka akan berpengaruh pada jumlah garis yang terdeteksi sangat banyak, padahal dia tidak mewakili garis pada tiang, sama seperti yang ada pada hasil parameter *minimum inliers* 10, di Gambar 5.11 bagian kiri.

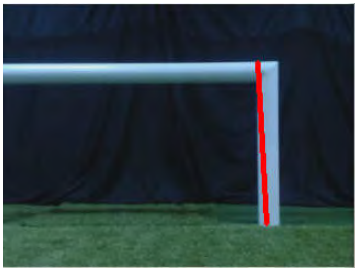


Gambar 5.11. Hasil Parameter *minimum inliers* = 10 (kiri) dan *minimum inliers* = 85 (kanan)

Dari semua parameter Algoritma *Ransac*, didapatkan bahwa parameter tebaik untuk *N* adalah sebesar 400, parameter terbaik untuk *batas jarak* adalah sebesar 40, parameter terbaik untuk *threshold inliers* adalah sebesar 5 dan terakhir parameter terbaik untnuk *minimum inliers* adalah sebesar 85. Selanjutnya, bisa dilihat tahap dari *postprocessing* sampai dengan mendapatkan *output*-an akhir berupa tinggi gawang dan jarak gawang. Tinggi gawang yang didapatkan pada *Image-1* adalah 147,217 piksel. Jarak gawang yang didapatkan pada *Image-1* adalah 1,9984 meter. Ilustrasi hasil *clustering line* dan mean tiang dapat dilihat pada Gambar 5.12. Ilustrasi hasil *drawing line* dapat dilihat pada Gambar 5.13.



Gambar 5.12. Hasil *clustering line* (kiri) dan hasil mean tiang (kanan)



Gambar 5.13 . Hasil *drawing line*

5.3.2 Skenario Uji Coba 2

Skenario uji coba 2 adalah Perbandingan performa antara Algoritma *Ransac* dengan Metode *Hough Transform* dalam pendeteksian gawang. Perbandingan performa meliputi pengujian ketepatan jarak dan kecepatan waktu eksekusi program

5.3.2.1 Pengujian Ketepatan Jarak

Dari hasil uji yang dilakukan, pada Tabel 5.8 diperlihatkan ada 40 dataset citra yang masing-masing diproses yang satu menggunakan Algoritma *Ransac* yang satu menggunakan metode *Hough Transform*. Hal yang bisa dianalisa adalah tinggi dan jarak yang didapat dari tiap-tiap metode. Uji coba 3, robot ditempatkan di posisi yang sama, dengan keadaan

yang sama, kemudian sama-sama di proses dengan metode masing-masing. Dengan skenario uji yang sama, bisa dibandingkan galat jarak gawang yang terdeteksi antar metode. Jarak gawang didapat dari tinggi piksel yang sudah dijelaskan di bab tiga buku ini. Berdasarkan rata-rata galat yang didapat. Algoritma *Ransac* memiliki keakuratan jarak robot dengan gawang yang lebih baik daripada Metode *Hough Transform*, dengan rata-rata galat jarak 0,03062 meter untuk Algoritma *Ransac* dan rata-rata galat jarak sebesar 0,1452 meter untuk Metode *Hough Transform*.

Tabel 5.8. Pengujian ketepatan jarak gawang

No	Jarak Sebenarnya (meter)	Metode *Ransac* (meter)	Galat *Ransac*	Metode *Hough* (meter)	Galat *Hough*
1	1,70	1,7569	0,03345	1,677	0,0135
2	1,75	1,7719	0,01250	2,2727	0,2987
3	1,80	1,8505	0,02808	1,7445	0,0308
4	1,90	1,9860	0,04528	1,8231	0,0405
5	2,00	2,0660	0,03300	1,8954	0,0523
6	1,95	1,9103	0,02035	2,57141	0,3187
7	2,06	1,9235	0,06627	2,4309	0,1800
8	2,10	2,1483	0,02301	1,9475	0,0726
9	2,15	2,1215	0,01324	2,6671	0,2405
10	2,20	2,1352	0,02946	2,7671	0,2578
11	2,25	2,2895	0,01754	2,0593	0,0848
12	2,35	2,2336	0,04955	2,8748	0,2233
13	2,45	2,4781	0,01149	3,1114	0,2700
14	2,50	2,4781	0,00874	2,2765	0,0894
15	2,60	2,3830	0,08345	3,2965	0,2679
16	2,65	2,6728	0,00862	3,2410	0,2230
17	2,75	2,6493	0,03663	3,0039	0,0923
18	2,80	2,8240	0,00857	3,4313	0,2255
19	2,87	2,6011	0,09368	3,5996	0,2542
20	3,00	2,9167	0,02778	2,7932	0,0689
21	3,02	2,8255	0,06440	3,4072	0,1282
22	3,10	3,1471	0,01518	3,9676	0,2799

23	3,25	3,2963	0,01425	3,0039	0,0757
24	3,28	3,4226	0,04346	3,8695	0,1797
25	3,30	3,3727	0,02203	3,9676	0,2023
26	3,40	3,5001	0,02944	4,0711	0,1974
27	3,45	3,2157	0,06792	4,0187	0,1648
28	3,50	3,5344	0,00982	3,2885	0,0604
29	3,53	3,3679	0,04592	4,0249	0,1402
30	3,55	3,6275	0,02183	4,1155	0,1593
31	3,60	3,4891	0,03080	4,2255	0,1738
32	3,66	3,5742	0,02345	4,2499	0,1612
33	3,70	3,8142	0,03087	3,7763	0,0206
34	3,75	3,8104	0,01611	3,1611	0,1570
35	3,80	3,9075	0,02829	3,8864	0,0227
36	3,82	3,7685	0,01349	3,9431	0,0322
37	3,90	4,0041	0,02670	3,5198	0,0975
38	3,92	3,8598	0,01535	3,61939	0,0767
39	3,98	3,8614	0,02979	3,6193	0,0906
40	4,00	4,0997	0,02493	3,6706	0,0824
Rata – Rata Galat			**0,03062**		**0,1452**

5.3.2.2 Pengujian Kecepatan Waktu Eksekusi Program

Dari hasil uji yang dilakukan, pada Tabel 5.9. diperlihatkan perbedaan waktu eksekusi yang dilakukan pada sekian dataset, dimana masing-masing dataset diproses dengan Algoritma *Ransac* dan Metode *Hough Transform*. Batasan program perbandingan hanya meliputi *preprocessing* sampai deteksi garis saja. Berdasarkan hasil rata-rata waktu eksekusi yang didapat pada setiap metode. *Hough Transform* lebih cepat dalam memproses algoritmanya untuk pendeteksian garis gawang dibandingkan dengan metode Algoritma *Ransac*, dengan rata-rata waktu eksekusi program 0,0333 detik untuk *Hough Transform* dan rata-rata waktu eksekusi program 0,0435 detik untuk Algoritma *Ransac.*

Tabel 5.9. Pengujian perbandingan waktu eksekusi program deteksi garis gawang

No	Nama Dataset	Waktu *Ransac* (detik)	Waktu *Hough (detik)*
1	*Image*-1	0,0606	0,0328
2	*Image*-2	0,0456	0,0347
3	*Image*-3	0,0463	0,0377
4	*Image*-4	0,0401	0,0316
5	*Image*-5	0,0542	0,0330
6	*Image*-6	0,0668	0,0334
7	*Image*-7	0,0458	0,0393
8	*Image*-8	0,0656	0,0350
9	*Image*-9	0,0394	0,0359
10	*Image*-10	0,0420	0,0359
11	*Image*-11	0,0383	0,0317
12	*Image*-12	0,0561	0,0363
13	*Image*-13	0,0493	0,0362
14	*Image*-14	0,0406	0,0270
15	*Image*-15	0,0501	0,0322
16	*Image*-16	0,0483	0,0315
17	*Image*-17	0,0377	0,0225
18	*Image*-18	0,0436	0,0329
19	*Image*-19	0,0340	0,0301
20	*Image*-20	0,0461	0,0346
21	*Image*-21	0,0436	0,0262
22	*Image*-22	0,0482	0,0346
23	*Image*-23	0,0548	0,0284
24	*Image*-24	0,0397	0,0374
25	*Image*-25	0,0346	0,0310
26	*Image*-26	0,0436	0,0347
27	*Image*-27	0,0436	0,0332
28	*Image*-28	0,0373	0,0352
29	*Image*-29	0,0451	0,0358

30	*Image*-30	0,0432	0,0358
31	*Image*-31	0,0372	0,0341
32	*Image*-32	0,0409	0,0318
33	*Image*-33	0,0388	0,0355
34	*Image*-34	0,0314	0,0375
35	*Image*-35	0,0357	0,0313
36	*Image*-36	0,0320	0,0375
37	*Image*-37	0,0314	0,0352
38	*Image*-38	0,0449	0,0262
39	*Image*-39	0,0345	0,0281
40	*Image*-40	0,0306	0,0374
Rata-Rata Waktu		**0,0435**	**0,0333**

LAMPIRAN A

Tabel A.1. Hasil transformasi citra deteksi tepi Gambar 3.7 ke bentuk *point*

No	x	y
1	70	51
2	71	49
3	71	50
4	71	52
5	71	53
6	71	54
7	71	55
8	71	56
9	71	57
10	71	58
11	71	59
12	71	60
13	71	61
14	71	62
15	71	63
16	71	64
17	71	65
18	71	66
19	71	67
20	71	68
21	71	69
22	71	70
23	71	71
24	71	72
25	71	73
26	71	74
27	71	75
28	71	76
29	71	77
30	71	78
31	71	79

No	x	y
32	71	80
33	71	81
33	71	81
34	71	82
35	71	83
36	71	84
37	71	85
38	71	86
39	71	87
40	71	88
41	71	89
42	71	90
43	71	91
44	71	92
45	71	93
46	71	94
47	71	95
48	71	96
49	71	97
50	71	98
51	71	99
52	71	100
53	71	101
54	71	102
55	71	103
56	71	104
57	71	105
58	71	106
59	71	107
60	71	108
61	71	109

No	x	y
62	71	110
63	71	111
64	71	112
65	71	113
66	71	114
67	71	115
68	71	116
69	71	117
70	71	118
71	71	119
72	71	120
73	71	121
74	71	122
75	71	123
76	71	124
77	71	125
78	71	126
79	72	48
80	72	127
81	72	128
82	72	129
83	72	130
84	72	131
85	72	132
86	72	133
87	72	134
88	72	135
89	72	136
90	72	137
91	72	138
92	72	139

No	x	y
93	72	140
94	72	141
95	72	142
96	72	143
97	73	47
98	73	144
99	74	47
100	74	144
101	75	47
102	75	144
103	76	47
104	76	144
105	77	47
106	77	144
107	78	47
108	78	55
109	78	56
110	78	57
111	78	144
112	79	47
113	79	55
114	79	58
115	79	59
116	79	144
117	80	47
118	80	54
119	80	59
120	80	60
121	80	61
122	80	62
123	80	144
124	81	47
125	81	54
126	81	63
127	81	64
128	81	65

No	x	y
129	81	66
130	81	67
131	81	68
132	81	69
133	81	70
134	81	71
135	81	72
136	81	73
137	81	74
138	81	75
139	81	76
140	81	77
141	81	78
142	81	79
143	81	80
144	81	81
145	81	82
146	81	83
147	81	84
148	81	85
149	81	86
150	81	87
151	81	88
152	81	89
153	81	90
154	81	91
155	81	92
156	81	93
157	81	94
158	81	95
159	81	96
160	81	97
161	81	98
162	81	99
163	81	100
164	81	101

No	x	y
165	81	102
166	81	103
167	81	104
168	81	105
169	81	106
170	81	107
171	81	108
172	81	109
173	81	110
174	81	111
175	81	112
176	81	113
177	81	114
178	81	115
179	81	116
180	81	117
181	81	118
182	81	119
183	81	120
184	81	121
185	81	122
186	81	123
187	81	124
188	81	125
189	81	126
190	81	127
191	81	128
192	81	129
193	81	130
194	81	131
195	81	132
196	81	133
197	81	134
198	81	135
199	81	136
200	81	137

No	x	y
201	81	138
202	81	139
203	81	140
204	81	141
205	81	142
206	81	143
207	82	47
208	82	54
209	82	132
210	82	133
211	82	134
212	82	135
213	82	136
214	83	47
215	83	54
216	84	47
217	84	54
218	85	47
219	85	54
220	86	47
221	86	54
222	87	48
223	87	54
224	88	48
225	88	54
226	89	48
227	89	54
228	90	48
229	90	54
230	91	48
231	91	54
232	92	48
233	92	54
234	93	48
235	93	54
236	94	48

No	x	y
237	94	54
238	95	48
239	95	54
240	96	48
241	96	54
242	97	48
243	97	54
244	98	48
245	98	54
246	99	48
247	99	55
248	100	48
249	100	55
250	101	48
251	101	55
252	102	48
253	102	55
254	103	48
255	103	55
256	104	48
257	104	55
258	105	48
259	105	55
260	106	48
261	106	55
262	107	48
263	107	55
264	108	48
265	108	55
266	109	48
267	109	55
268	110	48
269	110	55
270	111	48
271	111	55
272	112	48

No	x	y
273	112	55
274	113	48
275	113	55
276	114	48
277	114	55
278	115	48
279	115	55
280	116	48
281	116	55
282	117	48
283	117	55
284	118	48
285	118	55
286	119	48
287	119	55
288	120	48
289	120	55
290	121	48
291	121	55
292	122	48
293	122	55
294	123	48
295	123	55
296	124	48
297	124	55
298	125	48
299	125	55
300	126	48
301	126	55
302	127	48
303	127	55
304	128	48
305	128	55
306	129	49
307	129	55
308	130	49

No	x	y
309	130	55
310	131	49
311	131	55
312	132	49
313	132	55
314	133	49
315	133	55
316	134	49
317	134	55
318	135	49
319	135	55
320	136	49
321	136	55
322	137	49
323	137	55
324	138	49
325	138	55
326	139	49
327	139	55
328	140	49
329	140	55
330	141	49
331	141	55
332	142	49
333	142	55
334	143	49
335	143	55
336	144	49
337	144	55
338	145	49
339	145	55
340	146	49
341	146	55
342	147	49
343	147	55
344	148	49

No	X	y
345	148	55
346	149	49
347	149	55
348	150	49
349	150	55
350	151	49
351	151	55
352	152	49
353	152	56
354	153	49
355	153	56
356	154	49
357	154	56
358	155	49
359	155	56
360	156	49
361	156	56
362	157	49
363	157	56
364	158	49
365	158	56
366	159	49
367	159	56
368	160	49
369	160	56
370	161	49
371	161	56
372	162	50
373	162	56
374	163	50
375	163	56
376	164	50
377	164	56
378	165	50
379	165	56
380	166	50

No	x	y
381	166	56
382	177	50
383	177	56
384	178	50
385	178	56
386	179	50
387	179	56
388	180	50
389	180	56
390	181	50
391	181	56
392	182	50
393	182	56
394	183	50
395	183	56
396	184	50
397	184	56
398	185	50
399	185	56
400	186	50
401	186	56
402	187	50
403	187	56
404	188	50
405	188	56
406	189	50
407	189	56
408	190	50
409	190	56
410	191	50
411	191	56
412	192	50
413	192	56
414	193	50
415	193	56
416	194	50

No	x	y
417	194	56
418	195	50
419	195	56
420	196	50
421	196	56
422	197	50
423	197	56
424	198	50
425	198	56
426	199	50
427	199	56
428	200	50
429	200	56
430	201	50
431	201	56
432	202	50
433	202	56
434	203	50
435	203	56
436	204	50
437	204	56
438	205	50
439	205	56
440	206	50
441	206	56
442	207	50
443	207	57
444	208	50
445	208	57
446	209	50
447	209	57
448	210	50
449	210	57
450	211	50
451	211	57
452	212	50

No	x	y
453	212	57
454	213	50
455	213	57
456	214	50
457	214	57
458	215	50
459	215	57
460	216	50
461	216	57
462	217	50
463	217	57
464	218	50
465	218	57
466	219	50
467	219	57
468	220	50
469	220	57
470	221	50
471	221	57
472	222	50
473	222	57
474	223	51
475	223	57
476	224	51
477	224	57
478	225	51
479	225	57
480	226	51
481	226	57
482	227	51
483	227	57
484	228	51
485	228	57
486	229	51
487	229	57
488	230	51

No	x	y
489	230	57
490	231	51
491	231	57
492	232	51
493	232	57
494	234	51
495	234	57
496	235	51
497	235	57
498	236	51
499	236	57
500	237	51
501	237	57
502	238	51
503	238	57
504	239	51
505	239	57
506	240	51
507	240	57
508	241	51
509	241	57
510	242	51
511	242	57
512	242	114
513	242	115
514	242	116
515	242	117
516	242	118
517	242	122
518	242	123
519	242	124
520	242	125
521	242	126
522	242	127
523	242	128
524	242	129

No	x	y
525	242	130
526	242	131
527	242	132
528	242	133
529	242	134
530	242	135
531	242	136
532	242	137
533	242	138
534	242	139
535	242	140
536	242	141
537	242	142
538	242	143
539	242	144
540	242	145
541	242	146
542	243	51
543	243	57
544	243	97
545	243	98
546	243	99
547	243	100
548	243	101
549	243	102
550	243	103
551	243	104
552	243	105
553	243	106
554	243	107
555	243	108
556	243	109
557	243	110
558	243	111
559	243	112
560	243	113

No	X	y
561	243	114
562	243	115
563	243	116
564	243	117
565	243	118
566	243	119
567	243	120
568	243	121
569	243	147
570	243	148
571	244	51
572	244	57
573	244	78
574	244	79
575	244	80
576	244	81
577	244	82
578	244	83
579	244	84
580	244	85
581	244	86
582	244	87
583	244	88
584	244	89
585	244	90
586	244	91
587	244	92
588	244	93
589	244	94
590	244	95
591	244	96
592	245	51
593	245	57
594	245	70
595	245	71
596	245	72

No	x	y
597	245	73
598	245	74
599	245	75
600	245	76
601	245	77
602	245	148
603	246	51
604	246	57
605	246	67
607	246	68
608	246	69
609	246	148
610	247	51
611	247	57
612	247	64
613	247	65
614	247	66
615	247	148
616	248	51
617	248	58
618	248	62
619	248	63
620	248	148
621	249	51
622	249	58
623	249	59
624	249	60
625	249	61
626	249	62
627	250	51
628	250	148
629	251	51
630	251	148
631	252	51
632	252	140
633	252	141

No	x	y
634	252	142
635	252	143
636	252	144
637	252	145
638	252	146
639	252	147
640	252	148
641	253	51
642	253	123
643	253	124
644	253	125
645	253	126
647	253	127
646	253	128
648	253	129
649	253	130
650	253	131
651	253	132
652	253	133
653	253	134
654	253	135
655	253	136
656	253	137
657	253	138
658	253	139
659	254	51
660	254	105
661	254	106
662	254	107
663	254	108
664	254	109
665	254	110
666	254	111
667	254	112
668	254	113
669	254	114

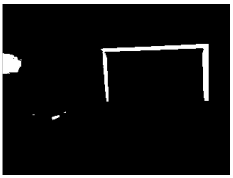
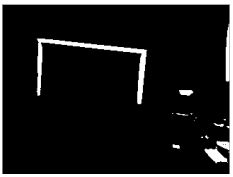
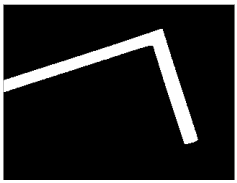
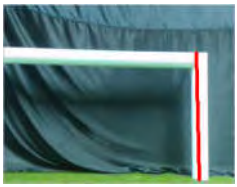
No	x	y
670	254	115
671	254	116
672	254	117
673	254	118
674	254	119
675	254	120
676	254	121
677	254	122
678	255	51
679	255	79
680	255	80
681	255	81
682	255	82
683	255	83
684	255	84
685	255	85
686	255	86
687	255	87
688	255	88
689	255	89
690	255	90
691	255	91
692	255	92
693	255	93
694	255	94
695	255	95
696	255	96
697	255	97
698	255	98
699	255	99
700	255	100
701	255	101
702	255	102
703	255	103
704	255	104
705	256	51

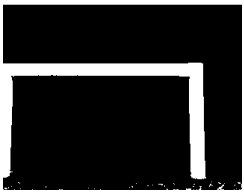
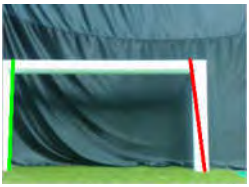
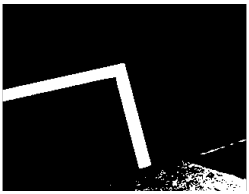
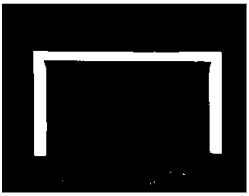
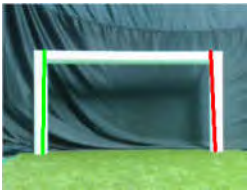
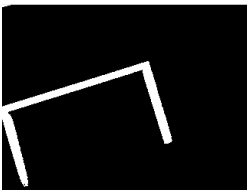
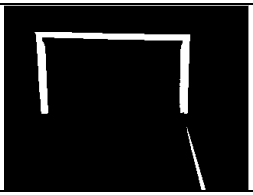
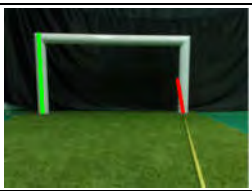
No	x	y
706	256	63
707	256	64
708	256	65
709	256	66
710	256	67
711	256	68
712	256	69
713	256	70
714	256	71
715	256	72
716	256	73
717	256	74
718	256	75
719	256	76
720	256	77
721	256	78
722	257	52
723	257	53
724	257	54
725	257	55
726	257	56
727	257	57
728	257	58
729	257	59
730	257	60
731	257	61
732	257	62

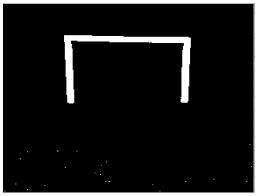
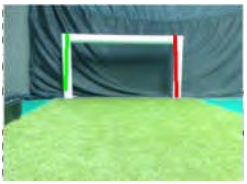
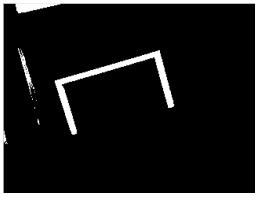
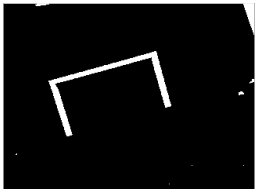
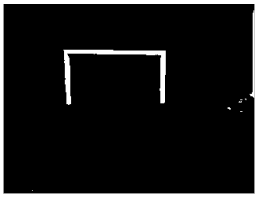
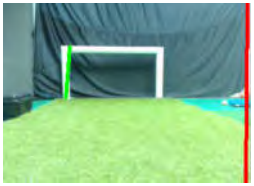
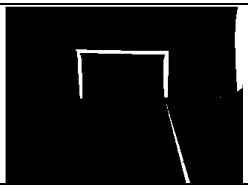
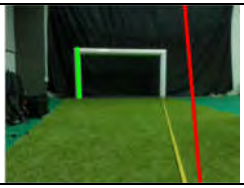
[Halaman ini sengaja dikosongkan]

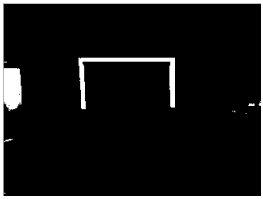
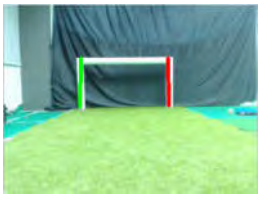
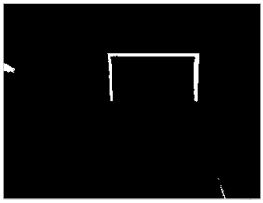
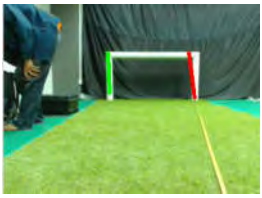
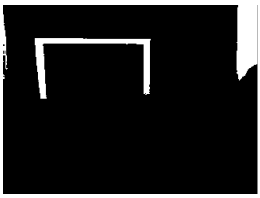
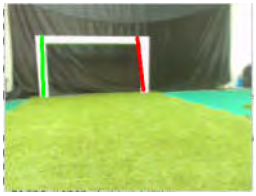
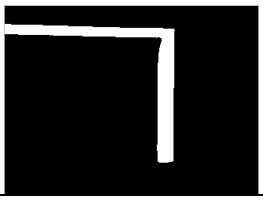
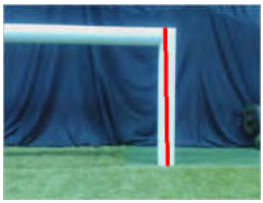
LAMPIRAN B

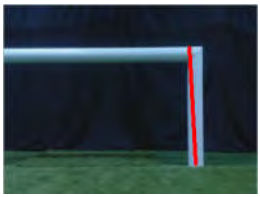
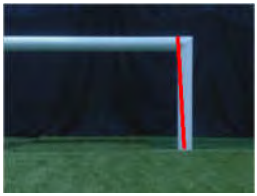
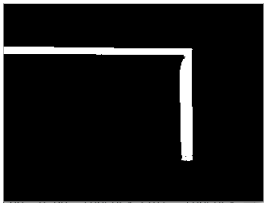
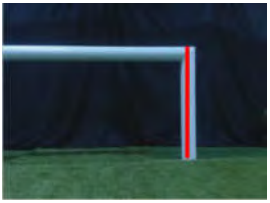
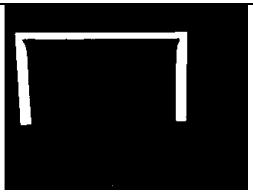
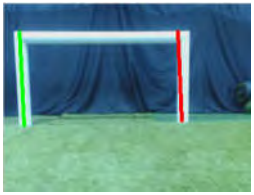
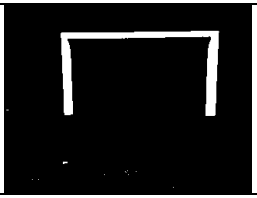
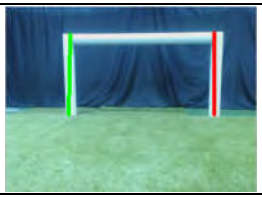
Tabel B.1. Hasil *thresholding* pada berbagai citra gambar

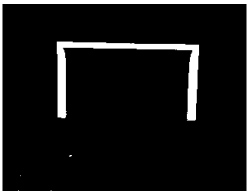
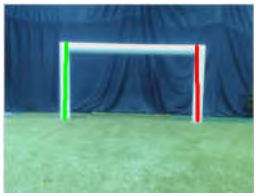
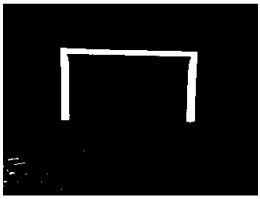
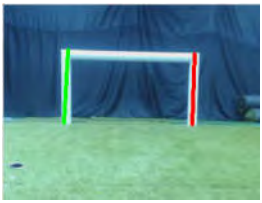
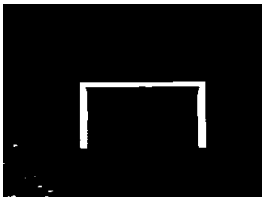
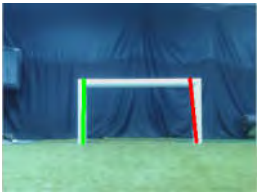
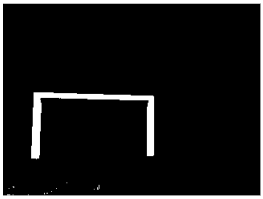
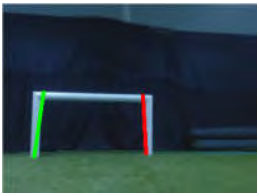
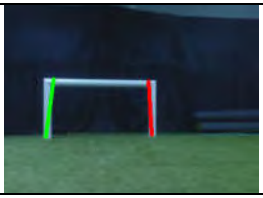
No	batas bawah	Hasil *thresholding*	Hasil deteksi garis
1	130		
2	130		
3	130		
4	170		

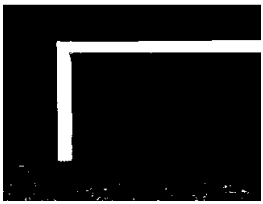
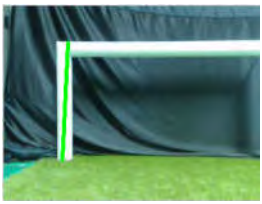
5	180		
6	190		
7	210		
8	170		
9	140		

10	230		
11	160		
12	235		
13	246		
14	150		

15	245		
16	245		
17	248		
18	248		
19	200		

20	140		
21	140		
22	140		
23	210		
24	210		

25	210		
26	210		
27	210		
28	120		
29	140		

30	200		

BAB VI
KESIMPULAN DAN SARAN

Bab VI ini membahas tentang kesimpulan yang didasari oleh hasil uji coba pada bab sebelumnya. Kesimpulan tersebut nantinya menjawab rumusan masalah yang telah ada pada pendahuluan. Selain itu, juga terdapat saran sebagai acuan untuk mengembangkan topik Tugas Akhir ini lebih lanjut di masa depan.

5.4 Kesimpulan

Dari hasil uji coba yang telah dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Tahap *thresholding* mengambil peran penting dalam pendeteksian gawang menggunakan Algoritma*Ransac.* Parameter batas bawah dan batas atas *thresholding*terbaik ada pada nilai 140. Namun, parameter *thresholding* sangat bergantung pada kondisi citra masukan. *Thresholding* yang tidak baik akan menghasilkan deteksi garis yang tidak baik pula.
2. Berdasarkan Tabel 5.4. hasil parameter *N* terbaik pada skenario uji coba 1 adalah 400, dengan nilai 400 hasil deteksi garis yang didapat sama bagusnya dengan nilai *N* yang di set lebih dari 400, dan dengan nilai *N* 400, sistem berjalan lebih cepat dibandingkan dengan nilai *N* lebih dari 400.
3. Berdasarkan Tabel 5.5. hasil parameter *batas jarak*terbaik pada skenario uji coba 1 adalah 40. Jika nilai *batas jarak* di set lebih kecil, hasil garis yang terbuat terlalu banyak dan, sedangkan jika nilai *batas jarak* di set terlalu tinggi, maka garis yang dihasilkan terlalu sedikit.
4. Berdasarkan Tabel 5.6. hasil parameter *threshold inliers*terbaik pada skenario uji coba 1 adalah 5. Jika nilai batas jarak di set lebih kecil, hasil garis yang terbuat

sedikit dan tidak mewakili, sedangkan jika nilai *threshold inliers* di set terlalu tinggi, maka garis yang dihasilkan terlalu banyak..

5. Berdasarkan Tabel 5.7. hasil parameter *minimum inliers*terbaik pada skenario uji coba 1 adalah 85. Jika nilai batas jarak di set lebih kecil, hasil garis yang terbuat terlalu banyak, sedangkan jika nilai batas jarak di set terlalu tinggi, maka garis yang dihasilkan terlalu sedikit.
6. Berdasarkan Tabel 5.8. pada skenario uji coba 2. Algoritma *Ransac* memiliki keakuratan jarak robot dengan gawang yang lebih baik daripada Metode *Hough Transform*, dengan rata-rata galat jarak 0,03062 meter untuk Algoritma Ransac dan rata-rata galat jarak sebesar 0,1452meter untuk Metode *Hough Transform.*
7. Berdasarkan Tabel 5.9. pada skenario uji coba 2. *Hough Transform* lebih cepat dalam memproses algoritmanya untuk pendeteksian garis gawang dibandingkan dengan metode Algoritma *Ransac*, dengan rata-rata waktu eksekusi program 0,0333 detik untuk Metode*Hough Transform* dan rata-rata waktu eksekusi program 0,0435 detik untuk Algoritma *Ransac.*

5.5 Saran

Saran yang diberikan untuk pengembangan perangkat lunak ini adalah:

1. Menerapkan *automatic thresholding* dalam proses segmentasi warna putih gawang
2. Menerapkan pendeteksian *frame* satu tiang terdeteksi dengan informasi tiang atas
3. Menemukan filter yang baik dalam mendeteksi warna putih gawang pada tahap *preprocessing*
4. Penerapan pendeteksian gawang pada *platform* Robot *Darwin-OP*dalam permainan sepak bola robot.

Daftar Pustaka

[1] Budiono, I. (2015). Kerjasama dalam permainan sepak bola robot pada platform Darwin-OP berbasis peraturan KRSBI 2015. Surabaya.

[2] Jose M. Canas, D. P. (2009). Visual Goal Detection for the RoboCup Standard Platform League. X Workshop De Agentes Fiscos.

[3] Pallares, A. M. (2009). Goal Detection for Soccer-playing Robots Based on Hough Transform. Universitat Rovira I Virgili.

[4] Stephen Se, D. L. (2002). Global Localization using Distinctive Visual Feature. IEEE Intl. Conference on Intellegent Robots and System.

[5] Madison Flannery, S. F. (2014). RANSAC: Identification of Higher-Order Geomteric Features and Applications in Humanoid Robot Soccer. Proceedings of Australian Conference on Robotic and Automation.

[6] Luca Iochi, D. N. (1999). Self-Localization in the RoboCup Environtment. Proc. of 3rd International Workshop on RoboCup .

[7] About Robocup. (2015). Dipetik November 8, 2015, dari http://www.robocup.org/about-robocup/

[8] Peraturan dan Persiapan Kontes Robot Sepak Bola Indonesia 2016

[9] I. Budiono, "Tim ichiro dalam kontes robot sepak bola Indonesia 2014," in The 2nd Symposium On Robot Soccer Competition, (Yogyakarta), 2014.

[10] Product Information Darwin-OP. (2010). Dipetik November 9, 2015, dari http://support.robotis.com/en/

[11] About OpenCV.(2015). Dipetik November 9, 2015, dari http://opencv.org/about.html

[12] Martin A. Fischler, R. C. (1981). Random Sample Consensus: A Paradigm for Model Fitting with

Applications to Image Analysis and Automated Cartography. SRI International.

[13] Jinnian Guo, X. W. (2009). An Intelligent Surveillance System Based on RANSAC Algorithm. IEEE.

[14] A.H.Dahlan, "Deteksi fitur dan penentuan lokasi robot pemain sepak bola berbasis penanda yang tidak unik." In The 2nd Symposium On Robot Soccer Competition, (Yogyakarta), 2014. (Dikutip pada halaman 5)

[15] Yogamangalam, R and Karthikeyan B. Segmentation Techniques Compariso In Image Processing. International Journal of Engineering and Technology (IJET)., 2013.

[16] Gonzalez, R., Woods: Digital Image Processing. Addison-Wesley Publishing Company (1992)

[17] Opening.(2003). Dipetik Juni 26, 2016, dari http://homepages.inf.ed.ac.uk/rbf/HIPR2/open.htm

[18] Martins, Daniel A. (2006). Real-time generic ball recognition in RoboCup domain

[19] Montgomery, Douglas. 1991. Introduction To Linear Regression Analysis. New York: John Wiley & Sons, Inc.

[20] Putra, Darma. (2010). Pengolahan Citra Digital. Yogyakarta: Penerbit Andi

[21] Jarak Titik dan Garis(2013). Dipetik Juni 26, 2016, dari https://yos3prens.wordpress.com/2013/09/20/jarak-titik-dan-garis/

[22] Muntaha, M. A. (2013). Penentuan posisi robot humanoid dalam lapangan sepak bola menggunakan metode triangulasi dan particle filter. Surabaya: Institut Teknologi Sepuluh Nopember.

BIODATA PENULIS

Uti Solichah, lahir di Indramayu pada tanggal 30 April 1994. Penulis menempuh pendidikan mulai dari TK Zahrotul Ulum, SDN Karangampel Kidul 1, SMPN 1 Karangampel, SMAN 2 Kota Cirebon, dan sekarang sedang menjalani pendidikan S1 Teknik Informatika di ITS. Selama masa perkuliahan penulis aktif telibat dalam organisasi mahasiswa, diantaranya sebagai staff Hubungan Luar Himpuanan Mahasiswa Teknik Computer periode 2013/2014, sebagai asisten sekjen Badan Eksekutif Mahasiswa ITS periode 2013/2014, sebagai staff Riset dan Teknologi Unit Kegiatan Mahasiswa Robotika. Selain itu, penulis juga aktif menjadi anggota Tim Robotika ITS sejak tahun 2013 sampai tahun 2016. Penulis sangat tertarik dalam bidang Robotika, menulis dan berkeinginan untuk mendalami keduanya di kemudian hari. Komunikasi dengan penulis dapat melalui email: **utisolichah07@gmail.com**