PROYEK AKHIR - VE180626

APLIKASI SONAR UNTUK MENGIDENTIFIKASI *OBSTACLE* MENGGUNAKAN METODE NAIVE BAYES

Salma Puspa Mega
10311600000075

Pembimbing
Imam Arifin, S.T., M.T.
Prof. Lam Siew Kei
Muhammad Fajar Adianto, S.T.

DEPARTEMEN TEKNIK ELEKTRO OTOMASI
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember
Surabaya 2019

FINAL PROJECT – VE180626

SONAR APLICATION TO IDENTIFY OBSTACLE USING THE NAIVE BAYES METHOD

Salma Puspa Mega
10311600000075

Supervisors
Imam Arifin, S.T., M.T.
Prof. Lam Siew Kei
Muhammad Fajar Adianto, S.T.

DEPARTMENT OF ELECTRICAL AUTOMATION ENGINEERING
Vocational Faculty
Institut Teknologi Sepuluh Nopember
Surabaya 2019

PERNYATAAN KEASLIAN PROYEK AKHIR

Dengan ini penulis menyatakan bahwa isi sebagian maupun kesuluruhan Proyek Akhir dengan judul:

"APLIKASI SONAR UNTUK MENGIDENTIFIKASI *OBSTACLE* MENGGUNAKAN METODE NAÏVE BAYES"

Adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang penulis akui sebagai karya sendiri. Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka. Apabila ternyata pernyataan ini tidak benar, penulis bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 17 Juli 2019

Salma Puspa Mega
NRP.10311600000075

Halaman ini sengaja dikosongkan

APLIKASI SONAR UNTUK MENGIDENTIFIKASI *OBSTACLE* MENGGUNAKAN METODE NAIVE BAYES

PROYEK AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Ahli Madya Teknik
Pada
Departemen Teknik Elektro Otomasi
Fakultas Vokasi
Institut Teknologi Sepuluh Nopember

Menyetujui:

Pembimbing 1	Pembimbing 2
Imam Arifin, S.T., M.T.	Prof. Lam Siew Kei
NIP. 19730222 200212 1 001	

KEMENTERIAN RISET, TEKNOLOGI, DAN PENDIDIKAN TINGGI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
FAKULTAS VOKASI
DEPARTEMEN TEKNIK ELEKTRO OTOMASI

SURABAYA
JULI, 2019

Halaman ini sengaja dikosongkan

APLIKASI SONAR UNTUK MENGIDENTIFIKASI *OBSTACLE* MENGGUNAKAN METODE NAIVE BAYES DI PT BHIMASENA RESEARCH AND DEVELOPMENT

PROYEK AKHIR

Disusun oleh:

Salma Puspa Mega NRP. 10311600000075

Menyetujui,

Kepala *Human Resources Department*, Pembimbing Perusahaan,

BHIMASENA Research & Development

Muhammad Nur Cahyo Utomo
NIK. 021042017

M.Fajar Adianto, S.T.
NIK. 020582016

Halaman ini sengaja dikosongkan

APLIKASI SONAR UNTUK MENGIDENTIFIKASI *OBSTACLE* MENGGUNAKAN METODE NAIVE BAYES

Salma Puspa Mega
10311600000075

Pembimbing I : Imam Arifin, S.T.,M.T.
Pembimbing II : Prof. Lam Siew Kei
Pembimbing III : Muhammad Fajar Adianto, S.T.

ABSTRAK

Gambar yang diperoleh menggunakan sensor Sonar di Kapal Tempur Bawah Air (KTBA) belum mampu mengidentifikasi *obstacle*, sehingga sulit bagi *user* untuk membedakan objek di bawah laut. Dalam penelitian ini, dikembangkan aplikasi untuk mengidentifikasi *obstacle* menggunakan metode Naive Bayes.

Tahapan yang dilakukan dalam mengembangkan aplikasi untuk identifikasi *obstacle* terdiri dari 4 tahap. Pertama, *object detection* untuk memisahakan objek dari *backgroud* dengan menggunakan metode *Selective Search*. Kedua, *feature extraction* untuk mengekstraksi ciri dari objek menggukan *metode Oriented FAST and Rotated BRIEF* (ORB). Ketiga, klasifikasi objek menggunakan metode Naive Bayes. Keempat, desain *Graphical User Interface* (GUI) untuk menampilkan hasil klasifikasi menggunakan *framework* Qt.

Hasil pengujian menggunakan metode *histogram of occupancy* memiliki akurasi yang lebih tinggi daripada metode *histogram of distance* normal dan *histogram of distance softweight*. Hal tersebut menunjukkan algoritma menggunakan *histogram of occupancy* mampu melakukan identifikasi *obstacle* dalam bentuk wajah manusia dengan akurasi 88,5%. Dengan hasil akurasi 88,5 % menunjukkan metode yang digunakan dapat digunakan untuk mengidentifikasi *obstacle* pada KTBA.

Kata Kunci: KTBA, Sonar, Naive Bayes, Klasifikasi Objek

Halaman ini sengaja dikosongkan

SONAR APLICATION TO IDENTIFY OBSTACLE USING THE NAIVE BAYES METHOD

Salma Puspa Mega
10311600000075

Supervisor I : Imam Arifin, S.T.,M.T.
Supervisor II : Prof. Lam Siew Kei
Supervisor III: Muhammad Fajar Adianto, S.T.

ABSTRACT

Images obtained using the Sonar sensor on the Underwater Combat Ship (KTBA) have not been able to overcome obstacles, making it difficult for users to determine objects under the sea. In this study, an application was developed to overcome obstacles using the Naif Bayes method.

The steps taken in developing applications to complete obstacles consist of 4 cups. First, object detection to separate objects from backgroud by using the Selective Search method. Second, feature extraction for extracting features from objects uses FAST-Oriented and Rotated BRIEF (ORB) methods. Third, object classification uses the Naive Bayes method. Fourth, the design of Graphical User Interface (GUI) to display the results of classification using the Qt framework.

The test results using the residential histogram method have higher testing than the normal distance histogram and soft distance histogram methods. This shows that the algorithm using a residential histogram is able to perform a barrier in the form of a human face with an accuracy of 88.5%. The evaluation results showed that 88.5% of the methods could be used for constraints at the KTBA.

Keywords: KTBA, Sonar, Naive Bayes, Object Classification

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji syukur Alhamdulilah, penulis panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan berkah, rahmat, dan karunia-Nya sehingga penulis dapat menyelesaikan Proyek Akhir yang berjudul: **"Aplikasi Sonar untuk Mengidentifikasi *Obstacle* Menggunakan Metode Naive Bayes"**. Proyek Akhir ini disusun sebagai salah satu syarat kelulusan yang harus ditempuh oleh setiap mahasiswa di Departemen Teknik Elektro Otomasi Fakultas Vokasi Institut Teknologi Sepuluh Nopember, Surabaya.

Pada kesempatan ini penulis ingin mengucapkan terima kasih kepada berbagai pihak yang telah memberikan bantuan dan dukungan dalam penyelesaian Proyek Akhir, terutama kepada: Bapak, Ibu, dan keluarga yang senantiasa memberikan doa, semangat, dan motivasi yang luar biasa, Bapak Ir. Joko Susila, M.T., selaku Kepala Departemen Teknik Elektro Otomasi yang selalu mendukung kegiatan bermanfaat bagi penulis, Bapak Imam Arifin, S.T., M.T., selaku dosen pembimbing sekaligus Kepala Laboratorium yanng telah banyak memberikan bimbingan, saran, dan arahan yang sangat berarti bagi penulis, Bapak Prof. Lam Siew Kei selaku pembimbing kedua yang senantiasa membimbing dan memberikan arahan yang berarti kepada penulis dalam pengerjaan Proyek Akhir, seluruh dosen Departemen Teknik Elektro Otomasi yang telah banyak memberikan ilmu selama penulis menempuh kuliah., Bapak Dipl,-Ing. Aris Budiyarto selaku CEO PT Bhimasena Research and Development yang telah memberikan kesempatan program magang dan pengerjaan Proyek Akhir, Mas Fajar dan Mas Irfan selaku pembimbing di PT. Bhimasena Research and Development yang selalu memberikan bimbingan, arahan, dan ilmunya, Mas Luqman selaku Kepala Divisi Tim Underwater-KTBA yang selalu memberi masukan dan arahan, Syahmina Nisya, Rachma Aji, Menik Nur Hidayati, Ashfar Afif, Wildan Adition, Arik Sugianto, Andi Octri, Bayu Bagus, dan Wahyu Aditya selaku teman-teman seperjuangan selama magang dan pengerjaan Proyek Akhir di PT.Bhimasena Research and Development, Teman-teman di Departemen Teknik Elektro Otomasi yang tidak dapat penulis sebutkan satu persatu, terutama teman-teman Laboratorium Cyber Physical, Otomasi dan Robot Industri yang selalu memberikan dukungan kepada penulis dalam menyelesaikan Proyek Akhir.

Penulis menyadari bahwa Proyek Akhir ini masih jauh dari kesempurnaan dengan segala kekurangannya. Untuk itu penulis mengharapkan adanya kritik dan saran dari semua pihak demi kesempurnaan dari Proyek Akhir ini. Besar harapan penulis bahwa Proyek Akhir ini dapat memberikan informasi dan manfaat bagi pembaca pada umumnya mahasiswa Departemen Teknik Elektro Otomasi.

Surabaya, Juni 2019

Penulis
Salma Puspa Mega

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kapal Tempur Bawah Air (KTBA) merupakan wahana benam bawah laut berbentuk kapal tempur yang memiliki misi untuk mengintai, membawa, dan meletakkan objek melalui jalur bawah laut. KTBA perlu mengetahui keadaan sekitar untuk melakukan pergerakan di bawah laut. Masalah utama yang berkaitaan dengan pemetaan dan melokasi kapal yang bergerak adalah sinyal ketidakpastian, dimana kapal tidak tahu posisinya dan tidak tahu keadaan sekitarnya [1]. Permasalahan tersebut mengakibatkan kapal tidak dapat melakukan pergerakan untuk mencapai lokasi lawan yang dituju. Dengan demikian, perlu adanya solusi untuk menanggulangi masalah tersebut. Salah satu solusi untuk menanggulangi masalah tersebut adalah dengan cara pendekatan yang dilakukan dengan berbagai cara deteksi objek yang berbeda di sekitar lingkungan kapal [1]. Deteksi objek dapat dilakukan dengan memanfaatkan alat sensor sonar.

Mengingat jangkauan dan kemampuan yang terbatas pada penerapan visual lingkungan bawah air, sonar telah menjadi solusi pilihan untuk pengamatan dasar laut sejak dimulai pada tahun 1950-an [2]. Sonar merupakan singkatan dari *Sound, Navigation and Ranging* dan dapat didefinisikan sebagai metode dengan memantulkan gelombang suara dari bawah air, untuk menentukan lokasi atau benda alam yang ada di laut [3]. Fungsi utama sensor sonar pada KTBA adalah untuk membaca pemetaan bawah laut dan digunakan untuk memperpanjang jarak pandang. Sensor Sonar memerlukan sebuah aplikasi untuk mengolah data dan menampilkan data berupa gambar objek. Hal ini dikarenakan sensor Sonar juga memiliki keterbatasan dan permasalahan penting dalam menginterpretasikan data hasil pembacaan Sonar. Adapun permasalahan tersebut adalah sensitivitas deteksi dari sensor dan karakteristik objek yang akan dideteksi [4].

Pengenalan pola merupakan salah satu cara untuk menangani permasalahan dalam mendeteksi suatu objek. Pengenalan pola mengelompokan data numerik dan simbolik (termasuk citra) secara otomatis oleh komputer yang memiliki tujuan untuk mengenali suatu objek [5]. Pengenalan pola pada proyek akhir ini diimplementasikan dalam aplikasi untuk mendeteksi *obstacle*. Aplikasi tersebut dibuat menggunakan beberapa tahapan. Tahap pertama yang harus dilakukan

adalah mengubah data Sonar *Intensity* menjadi bentuk gambar. Tahap kedua melakukan pemisahan objek gambar dari *frame* atau bisa disebut dengan *Image Object Detection*. Tahap selanjutnya, melakukan *Feature Extraction* dengan menggunakan metode *Oriented FAST and Rotated BRIEF*(ORB). Serta tahap yang terakhir adalah mengklasifikasikan objek dengan metode Naive Bayes. Pengambilan keputusan menggunakan Metode Bayes dibutuhkan suatu informasi dalam bentuk probabilitas untuk setiap alternatif yang ada pada persoalan yang sedang dihadapi dan nantinya akan menghasilkan nilai harapan sebagai dasar pengambilan keputusan [6].

1.2 Rumusan Masalah

Sensor yang digunakan pada KTBA untuk mengetahui keadaan sekitar dan melakukan pergerakan di bawah laut dengan memperpanjang jarak pandang adalah sensor Sonar. Sensor ini tidak dapat langsung digunakan tanpa mengolah data yang didapatkan terlebih dahulu. Hasil pengolahan data tersebut belum mampu melakukan identifikasi terhadap objek yang dideteksi.

1.3 Batasan Masalah

Penelitian ini hanya mengklasifikasikan satu objek berupa manusia. Pengklasifikasian dilakukan dengan identifikasi gambar wajah manusia. Pengujian pengaplikasian ini menggunakan gambar dari kamera *webcam personal computer.*

1.4 Tujuan

Penggunaan sensor Sonar dapat membantu pergerakan KTBA di bawah laut. Hal ini dikarenakan Sonar mampu memperpanjang jarak pandang dan mengetahui keadaan sekitar. Tujuan dari proyek akhir ini adalah melakukan pengolahan data Sonar supaya dapat mendeteksi *obstacle* di wilayah gerak KTBA. Membuat aplikasi untuk mengidentifikasi *obstacle* yang terdeteksi Sonar. Dengan proyek akhir ini, pengguna dapat mengetahui keadaan sekitar lebih jelas.

1.5 Metodologi

Metodologi yang digunakan pada penelitian ini terdiri dari empat tahap. Tahap yag pertama merupakan studi pustaka. Studi pustaka dilakukan untuk pemahaman materi dan pengumupulan dasar teori. Hal tersebut diakukan untuk menunjang penguasaan materi berupa materi

metode *image object detection*, metode *feature extraction*, dan Metode klasifikasi Naive Bayes. Studi pustaka ini berupa rujukan dalam bentuk buku, jurnal internasional dan nasional, dan artikel di internet.

Tahap kedua dari penelitian ini adalah perancangan dan pembuatan aplikasi yang akan digunakan untuk mengidentifikasi objek. Perancangan aplikasi berupa pembuatan diagram alir dari pemrograman yang digunakan untuk membangun aplikasi. Pembuatan aplikasi dilakukan dengan tahapan sebagai berikut : pengambilan data Sonar *Intensity* dilakukan dengan menggunakan komunikasi *ethernet*, pemisahan gambar objek dengan latar belakang menggunakan *image object detection*, ekstraksi ciri yang mengandung banyak informasi dari suatu objek menggunakan *feature extraction* dengan metode ORB dan K-Means *clustering* , mengklasifikasikan objek dengan menggunakan metode Naive Bayes.

Tahap ketiga merupakan tahapan untuk melakukan pengujian aplikasi yang telah dibuat. Aplikasi diuji untuk mengetahui tingkat keberhasilan dalam mengindentifikasi *obstacle* pada sensor sonar. Pengujian ini dilakukan menggunakan gambar *testing* yang diperoleh dari kamera *webcam personal* computer. Hasil dari pengujian kemudian dianalisis dengan membandingkan hasil dari metode yang digunakan.

Tahap keempat yang merupakan tahap terakhir adalah menyusun laporan yang terdapat penjelasan mengenai alur proses dan hasil dari pelaksanaan penelitian ini.

1.6 Sistematika Penulisan

Pembahasan yang lebih rinci dan jelas terdapat pada Buku Proyek Akhir ini yang memiliki susunan sistematika penulisan yang terdiri dari lima bab. Adapun sistematika tersebut adalah sebagai berikut:

BAB I PENDAHULUAN

Membahas tentang uraian latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, metodologi penelitian, dan sistematika penulisan buku Proyek Akhir yang dilakukan.

BAB II TEORI PENUNJANG APLIKASI SONAR

Menjelaskan mengenai kapal selam, sensor sonar, dan komponen penyusun antarmuka sensor sonar seperti *sonar interface unit*, komunikasi *ethernet*, pemrograman C++, *framework* Qt, *library* OpenCV,

image object detection, feature extraction, dan klasifikasi objek Naive Bayes.

BAB III PERANCANGAN APLIKASI SONAR

Membahas tentang langkah-langkah perancangan aplikasi sonar yang meliputi gambaran umum aplikasi sonar, mengakses sensor sonar menggunakan sonar *intensity*, *image object detection, feature extraction,* klasifikasi objek Naive Bayes.

BAB IV PENGUJIAN DAN ANALISIS

Menjelaskan mengenai hasil pengujian beserta analisis dari hasil identifikasi objek yang dilakukan.

BAB V PENUTUP

Membahas kesimpulan dari proses perancangan dan pengujian aplikasi beserta analisis hasil pengujian serta menjelaskan mengenai saran yang dapat diberikan untuk penelitian berikutnya.

BAB II
TEORI PENUNJANG APLIKASI SONAR

2.1 Kapal Selam

Kapal yang bergerak dibawah permukaan air dan pada umumnya digunakan untuk tujuan kepentingan militer. Sebagian besar Angkatan Laut di seluruh dunia memiliki dan mengoperasikan kapal selam meskipun memiliki populasi yang berbeda-beda di setiap negara. Negara yang terkenal memiliki kapal selam paling menakutkan adalah Negara Rusia. Gambar 2. 1 merupakan kapal selam kelas kilo milik negara Rusia yang memiliki dua generator *diesel* dan *drive* listrik, yang memberikan kekuatan untuk melakukan perjalanan 10 knot di permukaan dan 17 knot di bawah air [7].

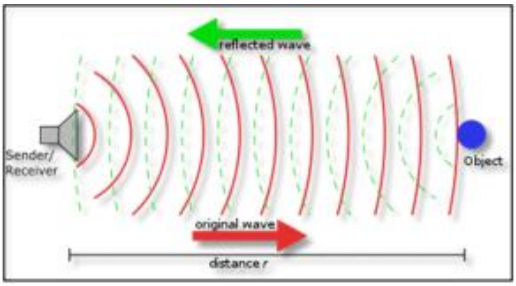
Gambar 2.1 Kapal Selam Kelas Kilo [7]

Negara Indonesia sebagai negara berkembang melalui PT. Bhimasena Research and Development saat ini sedang mengembangkan proyek wahana bawah air yang disebut dengan Kapal Tempur Bawah Air (KTBA). KTBA merupakan kapal selam berjenis *Sheal Driver Vehicle* (SDV) yang memiliki fungsi untuk mengintai, membawa, dan meletakkan objek. Umumnya kapal selam menyelam dan muncul ke permukaan laut dengan cara mengontrol masuk keluarnya air ke dalam tangki-tangki *ballast* [8].

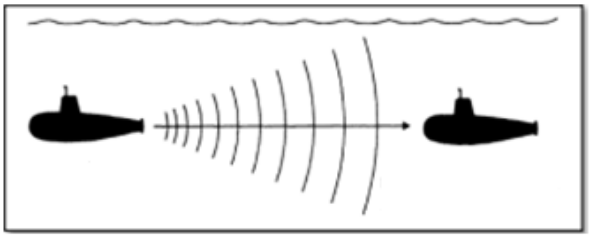
2.2 Sensor Sonar

Sensor yang umum digunakan untuk menentukan jarak sebuah objek. Sonar bekerja berdasarkan prinsip pemantulan gelombang suara, dimana variabel yang diukur adalah waktu pemantuan sejak gelombang tersebut dipancarkan [9]. Dalam penggolongannya sonar dibagi menjadi 2 jenis, yaitu sensor sonar pasif dan sonar aktif. Sonar pasif merupakan sonar yang tidak melakukan pengiriman sinyal gelombang suara keluar, sedangkan sonar aktif merupakan sonar yang mengirim dan menerima kembali sinyal gelombang suara [3].

Gambar 2.2 merupakan skema ketika sonar aktif bekerja. Sonar akan melepas sinyal gelombang suara ke dalam air, jika mengenai suatu objek maka akan terdapat pantulan yang memberikan efek *echo* (gema) dan mengembalikkannya kepada *receiver* pada sonar aktif.



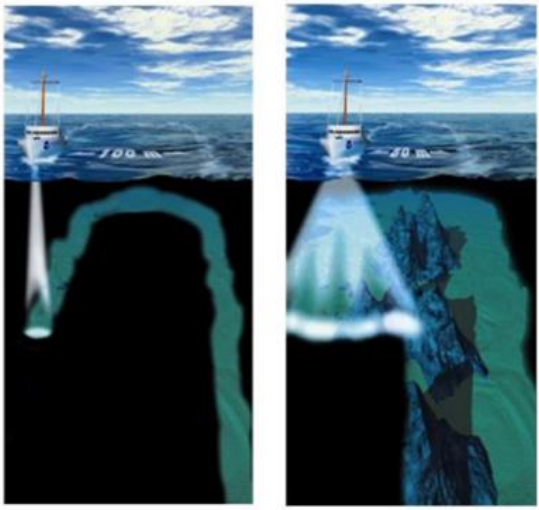
Gambar 2.2 Skema Kerja Sonar Aktif [3]



Gambar 2.3 Skema Kerja Sonar Pasif [3]

Gambar 2.3 menunjukkan skema kerja sonar pasif bekerja, sonar pasif hanya menangkap sinyal gelombang suara yang dipancarkan objek lain seperti getaran pada mesin kapal. Prinsip pemancaran sinyal gelombang suara oleh sonar terbagi menjadi 2 yaitu, *Singlebeam* Sonar

dan *Multibeam* Sonar. Gambar 2.4 menunjukkan perbedaan antara *Singlebeam* Sonar dan *Multibeam* Sonar. *Singlebeam* Sonar hanya memiliki satu pancaran sedangkan *Multibeam* Sonar memiliki lebih dari satu pancaran [10].



a b

Gambar 2.4 *Singlebeam* Sonar (a) dan *Multibeam* Sonar (b) [10]

Forward-Looking Sonar merupakan aplikasi dari sonar aktif dan sonar pasif yang digunakan pada Kapal Tempur Bawah Air (KTBA) yang memiliki misi mengintai dan meletakkan objek.

2.2.1 *Forward-Looking* Sonar (FLS)

Sensor sonar yang memiliki arah pancaran ke depan dan dirancang untuk navigasi kapal bawah air dan menghindari *obstacle* [11]. Bentuk fisik dari FLS terlihat pada Gambar 2.5. FLS dapat dioperasikan selama satu hari penuh dengan satu baterai, memungkinkan pengumpaan data *real-time* jarak jauh, dan memiliki resolusi dengan rentang yang sangat tinggi. FLS beroperasi pada frekuensi sebesar 400kHz dengan jarak jangkauan hingga 250 meter [12].

Gambar 2.5 *Forward-Looking* Sonar [11]

FLS yang digunakan pada KTBA memiliki spesifikasi yang ditunjukkan pada Tabel 2.1.

Tabel 2.1 *Spesification Forward Looking* Sonar

Technical Spesification	
Range	250m
Depth Rating	350m, 1500m, 6000m
Interface	Ethernet 100Mbit/s

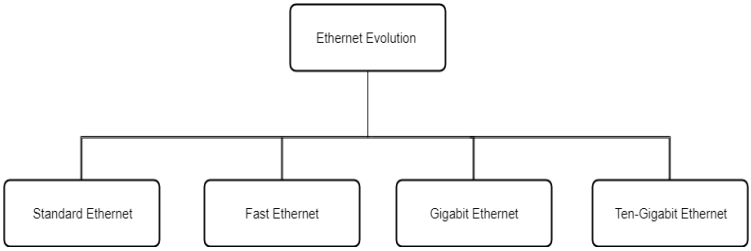
2.2.2 *Sonar Interface Unit* (SIU)

Perangkat untuk mengaktifkan sistem sonar guna berkomunikasi dengan *personal computer*. Bentuk fisik dari SIU dapat dilihat pada Gambar 2.6 .

Gambar 2.6 *Sonar Interface Unit* [12]

2.3 Komunikasi Ethernet

Jenis perkabelan dan pemrosesan sinyal untuk data jaringan komputer yang dikembangkan oleh Robert Metcalfe dan David Boggs di Xerox Palo Alto Research Center (PARC) pada tahun 1972 [13]. Gambar 2.7 menunjukkan evolusi dari *ethernet*. *Ethernet* memiliki 4 generasi yaitu, *Standard Ethernet* yang memiliki kecepatan 10 Mbps, *Fast Ethernet* yang memiliki kecepatan 100 Mbps, *Giga Ethernet* yang memiliki kecepatan 1 Gbps, dan *ten-Gigabit Ethernet* yang memiliki kecepatan 10Gbps [14].



Gambar 2.7 Evolusi *Ethernet* [14]

Ethernet menggunakan metode transmisi *baseband*, yaitu mengirim sinyal serial 1 bit pada satu waktu. *Ethernet* melakukan transmisi data menggunakan kabel jaringan dalam bentuk paket data yang disebut dengan *ethernet frame*. *Frame* tersebut memiliki ukuran minimun 64 *byte* dan ukuran maksimum 1518 *byte* dengan 18 *byte* berisi alamat sumber, tujuan protokol jaringan yang digunakan, dan *header* yang menyimpan mengenai informasi lain. Proses enkapsulasi data pada komunikasi ethernet yang digunakan pada sensor sonar adalah *Ethernet* II dengan protokol TCP/IP.

2.4 Pemrograman C++

C++ adalah bahasa pemrograman yang dibuat oleh Bjarne Stroustrup, yang dikembangkan dari bahasa C. Bahasa C++ merupakan bahasa tingkat tinggi yang bersifat universal dan berbasis *Object Oriented Programming* (OOP). Konsep utama pemrograman berorientasi objek adalah melakukan pemodelan objek dari kehidupan nyata ke dalam tipe data abstrak [15]. Dalam pemrograman OOP terdapat 3 fitur yang dimiliki, yaitu enkapsulasi, inheritansi, dan polymorphism. Enkapsulasi merupakan suatu cara untuk

menyembunyikan implementasi detail dari suatu *class* dalam rangka menghindari akses yang ilegal. Inheritansi merupakan suatu objek yang mempunyai objek turunan. Polymorphism merupakan kemampuan untuk merepresentasikan dua bentuk yang berbeda.

Pemrograman ini digunakan untuk membangun aplikasi sonar dengan berbasis OOP menggunakan bantuan *framework* Qt.5.11 dan *library* OpenCV 3.2.0.

2.4.1 *Framework* Qt

Qt Creator merupakan *cross-platform* C++ *Integrated Development Environment* (IDE) yang merupakan bagian dari Qt *Software Development Kit* (SDK) [16]. Qt Creator mempunyai *debugger* dalam bentuk visual dan *layout Graphical User Interface* (GUI) serta tempat perancangan form. Projek Qt Creator menggunakan format *cross-platform project* (.pro) agar dapat melakukan *share project* kepada *platform* yang berbeda. Dalam pembuatan *project* menggunakan Qt dapat meliputi : file yang dikelompokkan secara bersama, langkah *build* program, *form* dan file *resource*, dan pengaturan untuk menjalankan aplikasi. Pembuatan aplikasi sonar menggunakan *library* yang disediakan oleh Qt untuk tampilan GUI dan beberapa proses dalam menjalankan identifikasi sebuah objek. Adapun *library* Qt yang digunakan dalam pembuatan aplikasi sonar:

a. QWidget

Merupakan kelas dasar dari semua objek *User Interface* (UI). Widget menerima *mouse*,*keyboard*, dan beberapa *event* lainnya dan merepresentasikan tampilan ke layar [17].

b. QString

Kelas yang menyediakan string karakter *Unicode. Unicode* merupakan standar internasional yang mendukung sebagaian besar sistem penulisan yang digunakan saat ini. QString menggunakan berbagai implisit untuk mengurangi penggunaan memori dan untuk menghindari penyalinan data yang tidak perlu [18].

c. QVector

Kelas template yang menyediakan array dinamis. QVector menyimpan semua *item* yang dimiliki dalam bentuk array. Kelas ini menggunakan 0 sebagai index dasar seperti array dalam c++. Untuk mengakses item yang dimiliki oleh QVector dapat menggunakan operator indeks yang diinginkan [19].

d. QLabel

Kelas yang terdapat teks atau gambar *display*. QLabel digunakan untuk menampilkan teks atau gambar dan tidak ada fungsi interaksi pengguna yang disediakan [20].

e. QGridLayout

Kelas yang menjabarkan widget dalam kotak. QGridLayout mengambil ruang yang tersedia, membaginya menjadi baris dan kolom, dan menempatkan setiap widget yang dikelola ke dalam sel yang benar. QGridLayout mencakup dua lebar margin, yaitu margin konten dan spasi(). Margin konten adalah lebar ruang yag dipesan di masing-masing sisi QGridLayout. Spasi() adalah lebar jarak otomatis yang dialokasikan antar kotak tetangga [21].

f. QPushButton

Kelas yang memiliki tombol perintah. QPushButton merupakan widget yang paling umum digunakan dalam antarmuka pengguna grafis. Tekan (klik) tombol untuk memerintahkan komputer guna melakukan beberapa tindakan, atau guna menjawab pertanyaan. Tombol tekan menampilkan label tekstual, dan opsional ikon kecil dan dapat diatur menggunakan konstruktor dan diubah menggunakan *setText()* dan *setIcon()* [22].

g. QThread

Kelas yang menyediakan cara *platform-independent* untuk mengelolah utas. QThread mengelola satu *thread* kontrol dalam program. QThread mulai mengeksekusi dalam fungsi *run()*. Secara *default run()* memulai loop *event* dengan memanggil *exec()* dan menjalankan loop *event* Qt di dalam utas [23].

h. QTimer

Kelas yang menyediakan pengatur waktu yang berulang dan sekali pakai. QTimer menyediakan antarmuka pemrograman tingkat tinggi untuk penghitung waktu. QTimer dapat digunakan dengan menghubungkan sinyal *timeout()* ke slot yang sesuai dan panggil *start()* [24].

i. QMutex

Kelas yang menyediakan serialisai akses di antara utas. Tujuan dari QMutex adalah untuk melindungi objek, struktur data, atau bagian kode sehingga hanya satu utas yang dapat mengaksesnya. Mutex yang paling baik digunakan adalah QMutexLocker karena dapat dipastikan proses mengunci dan membuka kunci dilakukan secara konsisten [25].

j. QImage

Kelas yang menyediakan representasi gambar yang tidak tergantung perangkat keras yang memungkinkan akses langsung ke data piksel, dan dapat digunakan sebagai *paint device.* Kelas QImage mendukung beberapa format gambar dengan format enum. Qimage memiliki kumpulan fungsi yang dapat digunakan untuk memperoleh berbagai informasi tentang gambar dan fungsi untuk melakukan transformasi gambar [26].

k. QWaitCondition

Kelas QWaitCondition menyediakan variabel kondisi untuk menyinkronkan utas. QWaitCondition memungkinkan utas memberi tahu utas lain jika kondisi telah terpenuhi [27].

l. QByteArray

Kelas QByteArray menyediakan byte array yang digunakan untuk menyimpan byte mentah dan string. QByteArray menggunakan berbagai implisit untuk mengurangi penggunaan memori dan menghindari penyalinan data yang tidak perlu. Salah satu cara untuk menginisialisai QByteArray adalah dengan mengirimkannya const char * ke konstruktornya [28].

m. QBitArray

Kelas QBitArraay menyediakan bit array yang memberikan akses ke bit individu dan menyediakan operator (AND, OR, XOR, dan NOT) yang bekerja pada seluruh array bit. QBitArray menggunakan berbagai implisit untuk mengurangi penggunaan memori dan untuk menghindari penyalinan data yang tidak perlu [29].

n. QTime

Kelas QTime menyediakan fungsi waktu yang meliputi jam, menit, detik, milidetik. Kelas ini menyediakan fungsi untuk membandingkan waktu [30].

2.4.2 *Open Source Computer Vision Library*(OpenCV)

Library open-source berlisensi *Berkeley Software Distribution* (BSD) yang mencakup beberapa ratusan algoritma visi komputer [31]. OpenCV memiliki antarmuka yang mendukung bahasa pemrograman C++, C, Phyton, dan Java. Secara teori OpenCV digunakan seperti meniru cara kerja sistem visual manusia yaitu dengan melihat objek melalui "penglihatan/mata" dan citra pada objek tersebut diteruskan ke otak untuk memproses sehingga mengerti objek apa yang tampak pada pandangan mata manusia [32]. Dalam pembuatan aplikasi sonar

menggunakan *library* OpenCV untuk memproses data dan *user interface* (UI). Adapun *library* OpenCV yang digunakan adalah sebagai berikut:

a. Core

Modul ringkas yang mendefinisikan struktur data dasar, termasuk array multi dimensi yang padat (Mat) dan fungsi dasar yang digunakan oleh semua modul lainnya [33].

b. HighGui

Antarmuka yang mudah digunakan untuk merekam video, gambar dan video codec, serta kemampuan UI sederhana [33].

c. ImgCodecs

Modul yang memberikan akses membaca dan menulis gambar ke / dari disk atau memori [34].

d. ImgProc

Modul pemrosesan gambar yang mencakup penyaringan gambar linier dan non-linier, transformasi gambar geometris, konversi ruang warna, histogram, dan sebagainya [33].

e. Fatures2d

Pendeteksi fitur yang menonjol, deskriptor, dan pencocokan deskriptor [33].

f. Calib3d

Algoritma geometri tampilan dasar, kalibrasi kamera tunggal dan stero, estimasi pose objek, algoritma korespondensi stereo, dan elemen rekonstruksi 3D [33].

g. Video

Modul analisi video yang mencakup estimasi gerakan, pengurangan latar belakang, dan algoritma pelacakan objek [33].

h. XimageProc

Modul tambahan *library* OpenCV yang memiliki kepanjangan *extended image processing* [31] dan mencakup tambahan modul untuk proses *image processing*.

2.5 *Image Object Detection*

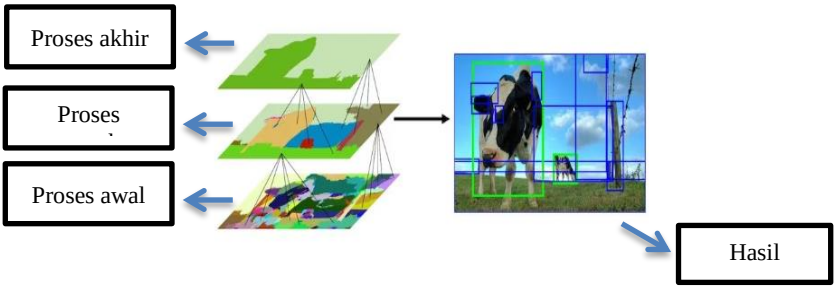
Metode yang memiliki tujuan untuk memisahkan objek dengan *background* yang ada dalam gambar. Proses pendeteksian objek tersebut akan memudahkan manusia mengenali bentuk dari objek tertentu seperti pemetaan geografis, penggunaan pada sensor benda untuk sistem keamanan, pembacaan citra hasil scan medis, dan lain-lain [35]. Dalam melakukan proses *object detection* dapat dilakukan

menggunakan cara, yaitu: segmentasi citra. Segmentasi adalah proses pemisahan objek yang satu dengan objek yang lain dalam suatu gambar (citra) menjadi objek-objek berdasarkan karakteristik tertentu [36].

Algoritma segmentasi citra umumnya didasarkan pada satu dari dua properti nilai intensitas yaitu diskontinuitas dan similaritas. Proses segmentasi dikontinuitas merupakan pemilihan citra berdasarkan perubahan citra secara tiba-tiba atau cukup besar, proses tersebut antara lain: deteksi titik, deteksi garis, dan deteksi tepi. Proses similaritas merupakan pemilihan citra berdasarkan wilayah yang sama menurut beberapa kriteria, antara lain : *thresholding, region growing,* dan *region splitting and merging.* Metode yang digunakan untuk membangun aplikasi sonar adalah metode segmentasi similaritas yaitu *selective search.*

2.5.1 *Selective Search*

Algoritma *region proposal* yang digunakan dalam deteksi objek dengan mensegmentasi gambar berdasarkan intensitas piksel menggunakan metode segmentasi berbasis grafik oleh Felzenszwalb dan Huttenlocher [37]. Gambar 2.8 menunjukan proses *selective search* dengan *input* dari gambar hasil segementasi berbasis grafik oleh Felzenszwalb dan Huttenlocher. Proses tersebut terdiri dari proses awal, tengah, dan akhir yang dikelompokkan berdasarkan kesamaan yang dimiliki gambar.



Gambar 2.8 Proses Segmentasi [37]

Selective Search menggunakan empat hal untuk menentukan kesamaan, yaitu berdasarkan warna, tekstur, ukuran dan kompatibilitas bentuk.

a. Warna

Histogram warna dari 25-*bins* dihitung untuk setiap warna ‘R’ ‘G’ ‘B’ dan digabungkan untuk mendapatkan deskriptor warna 25 x 3 = 75- dimensi. Kesamaan warna dari dua daerah didasarkan pada persimpangan histogram dapat dihitung dengan rumus sebagai berikut:

$$\boldsymbol{s_{color}\left(r_i, r_j\right) = \sum_{k=1}^{n} \min\left(c_i^k, c_j^k\right)} \quad \textbf{(2.1)}$$

Dengan,

c_i^k = nilai histogram untuk k^{th} bin dalam deskriptor warna.

b. Tekstur

Fitur tekstur dihitung dengan mengekstraksi turunan Gaussian pada 8 orientasi untuk setiap warna ‘R’ ‘G ‘B’. Setiap orientasi dan setiap warna dihitung dengan histogram tekstur 10-*bins*, sehingga menghasilkan 10x8x3 = 240- dimensi deskripsi fitur. Kesamaan tekstur dari dua daerah berdasarkan persimpangan histogram dapat dihitung dengan rumus berikut:

$$\boldsymbol{s_{texture}\left(r_i, r_j\right) = \sum_{k=1}^{n} \min\left(t_i^k, t_j^K\right)} \quad \textbf{(2.2)}$$

Dengan,

t_i^k = nilai histogram untuk k^{th} bin dalam deskriptor tekstur.

c. Ukuran

Kesamaan ukuran membawa pengaruh yang besar terhadap segementasi, jika kesamaan ukuran tidak dipertimbangan wilayah yang lebih kecil akan tergabung dengan wilayah yang lebih besar. Sehingga hanya akan terdapat satu wilayah yang tidak menandakan suatu objek. Oleh Karena itu perlu adanya batasan untuk proses tersebut. Adapun hal tersebut dapat ditentukan dalam rumus berikut:

$$\boldsymbol{s_{size}\left(r_i, r_j\right) = 1 - \frac{size(r_i) + size\left(r_j\right)}{size(im)}} \quad (\mathbf{2.3})$$

Dengan,

$size(im)$ = ukuran gambar dalam piksel.

d. Bentuk

Kesamaan bentuk mengukur kecocokan antara dua wilayah (r_i dan r_j) berdasarkan bentuk. Jika kedua wilayah cocok maka akan

digabungkan, sedangkan jika tidak cocok maka tidak akan digabungkan. Penggabungan kedua wilayah tersebut dapat dilakukan dengan rumus dibawah ini:

$$\boldsymbol{s_{fill}\left(r_i, r_j\right) = 1 - \frac{size\left(BB_{IJ}\right) - size\left(r_i\right) - size\left(r_j\right)}{size(im)}} \qquad (2.4)$$

Dengan,
$size(BB_{ij})$ = kotak pembatas di sekitar r_i dan r_j.

Berdasarkan ke-empat kesamaan tersebut *selective search* kemudian melakukan kombinasi liner dengan rumus sebagai berikut:

$$\boldsymbol{s\left(r_i, r_j\right) = a + b + c + d} \qquad (2.5)$$

Dengan,
$a = a_1 s_{color}(r_i, r_j)$
$b = a_2 s_{texture}(r_i, r_j)$
$c = a_3 s_{size}(r_i, r_j)$
$d = a_4 s_{fill}(r_i, r_j)$
r_i, r_j = dua wilayah atau segmen dalam gambar
$a_i \in 0,1$ = menunjukkan ukuran kesamaan digunakan atau tidak.

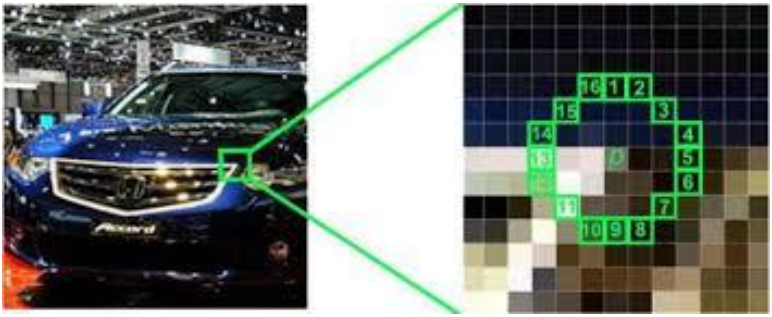
2.6 *Feature Extraction*

Proses pengambilan ciri-ciri yang terdapat pada objek di dalam citra [38]. *Feature extraction* memiliki tujuan untuk memperkecil jumlah data, mengambil informasi yang terpenting dari data yang diolah, dan mempertinggi presisi pengolahan [39]. Metode yang digunakan untuk melakukan *feature extraction* sangatlah beragam. Pembuatan aplikasi sonar ini menggunakan metode *Oriented FAST and Rotated BRIEF* (ORB) dengan bantuan K-Means *clustering* untuk melakukan pengelompokan data ciri objek.

2.6.1 *Oriented* FAST *and Rotated* BRIEF

Metode yang dikembangkan dari metode *Features for Accelered Segment Test* (FAST) *keypoint detector* dan juga *Binary Robust Independent Elementary Features* (BRIEF) *descriptor*. Metode *oriented*-FAST (o-FAST) merupakan metode yang digunakan untuk mencari titik sudut yang ada pada gambar. Pencarian titik sudut

dilakukan dengan mengecek salah satu titik secara acak sebagai nilai tengah lalu membuat lingkaran dengan jumlah 16 pixel disekitar titik tersebut untuk mengecek titik tersebut adalah sebuah sudut [40]. Metode tersebut dapat ditunjukkan dalam Gambar 2.9.



Gambar 2.9 Ilustrasi Metode FAST [40]

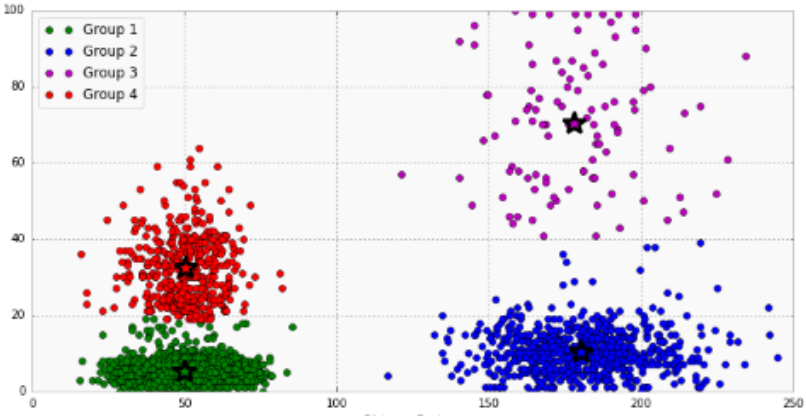
Metode *rotated*-BRIEF (r-BRIEF) melakukan *binary test* pada potongan gambar yang sudah di*smoothing*. Metode ini digunakan untuk mendapatkan vektor ciri dari *keypoint* yang sudah didapat pada tahap o-FAST [40].

2.6.2 K-Means *Clustering*

Metode *unsupervised learning* yang digunakan ketika memiliki data yang tidak berlabel (yaitu, data tanpa kategori atau kelompok yang ditentukan) [41]. Tujuan dari algoritma ini adalah menemukan kelompok dalam data, dengan jumlah kelompok yang diwakili oleh variabel *K* atau disebut *cluster*. Pengelompokan ke dalam variabel *K* didasari pada kesamaan fitur dari objek.

Algoritma dalam proses K-Means *clustering* menggunakan perbaikan iteratif untuk menghasikan hasil akhir. Jumlah variabel *K* menentukan jumlah centroid yang digunakan. *Centroid* adalah lokasi imajiner atau nyata yang mewakili pusat *cluster*. Dengan kata lain, algoritma K-Means mengidentifikasi *K* jumlah *centroid*, kemudian mengalokasikan setiap titik data ke *cluster* terdekat [42]. Proses K-Means *clustering* dimulai dengan pemilihan centroid secara acak yang digunakan sebagai titik awal untuk setiap *cluster*, kemudian melakukan perhitungan berulang untuk mengoptimalkan posisi *centroid*. *Centroid* dikatakan optimal jika *centroid* telah stabil, tidak ada perubahan dalam

nilai *centroid* dan jumlah iterasi yang ditentukan telah tercapai. Gambar 2.10 menunjukkan hasil dari proses K-Means *clustering.*



Gambar 2.10 Ilustrasi K-Means *Clustering* [42]

Penentuan titik data ke centroid terdekat dapat menggunakan perhitungan Euclidean Distance yang memiliki rumus dibawah ini:

$$\boldsymbol{d} = \sqrt{\sum_{i=1}^{n} \left(\boldsymbol{X_i} - \sum_{j=1}^{m} \boldsymbol{Y_j}\right)^2} \qquad \textbf{(2.6)}$$

Dengan,
d = Jarak Euclidean Distance
X = Centroid
Y = Data
n = Jumlah Centroid
m = Jumlah Data

2.7 Klasifikasi Naive Bayes

Salah satu metode *machine learning* yang menggunakan perhitungan probabilitas. Algoritma ini memanfaatkan metode probabilitas dan statistik sederhana dengan asumsi bahwa antar satu kelas dengan kelas yang lain tidak saling tergantung (*independen*) [43]. Pengklasifikasian Bayes didasari oleh teorema bayes yang ditemukan oleh Thomas Bayes pada abad ke-18. Dalam studi pembandingan

algoritma klasifikasi telah ditemukan simple bayesian atau yang biasa dikenal Naive Bayes *Classifier*[44]. Naive Bayes *Classifier* bekerja secara cepat, mudah diimplemantasikan, memiliki struktur yang sederhana, dan efektif [45].

Teorema Bayes merupakan dasar aturan dari Naive Bayes *Classifier* [44]. Berikut adalah rumus dari teorema Bayes :

$$\boldsymbol{P(H|X) = \frac{P(X|H) * P(H)}{P(X)}} \qquad (\mathbf{2.7})$$

Dengan,
$P(H|X)$ = Posterior
$P(X|H)$ = Likelihood
$P(H)$ = Prior
$P(X)$ = Evidence

Dari persamaan 2.7 tersebut yang dimaksud posterior adalah probabilitas akhir bersyarat (*conditional probability*) suatu hipotesis H terjadi jika diberikan bukti X terjadi. Likelihood adalah probabilitas sebuah bukti X terjadi akan mempengaruhi hipotesis H. Prior adalah probabilitas awal hipotesis H terjadi tanpa memandang bukti apapun. Evidence merupakan probabilitas awal bukti X tanpa memandang hipotesis atau bukti lain [46]. Kaitan antara Naive Bayes dengan klasifikasi, korelasi hipotesis dan bukti klasifikasi adalah bahwa hipotesis dalam teorema Bayes merupakan label kelas yang menjadi target pemetaan dalam klasifikasi, sedangkan bukti merupakan fitur yang menjadi masukkan dalam model klasifikasi.

Berdasarkan korelasi tersebut terdapat perbedaan rumus dalam melakukan klasifikasi menggunakan Naive Bayes dengan terorema Bayes saja. Adapun rumus Naive Bayes untuk klasifikasi ditunjukkan pada persamaan 2.8 berikut:

$$\boldsymbol{P(H|X) = \frac{\prod_{i=1}^{q} P(X|H) * P(H)}{P(X)}} \qquad (\mathbf{2.8})$$

Perbedaan rumus antara teorema Bayes dan klasifikasi Naive Bayes terletak pada *evidence* yang terjadi secara berulang sesuai dengan jumlah fitur yang dimiliki. Nilai *evidence* selalu tetap untuk

setiap kelas pada satu sample. Untuk melakukan klasifikasi data kontinyu digunakan Distribusi Gaussian :

$$\boldsymbol{Y = \frac{1}{\sqrt{2\pi\sigma}} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}} \qquad (\mathbf{2.9})$$

Dengan,
Y : Peluang
μ : Rata-rata
σ : Standar Deviasi
x : Nilai atribut

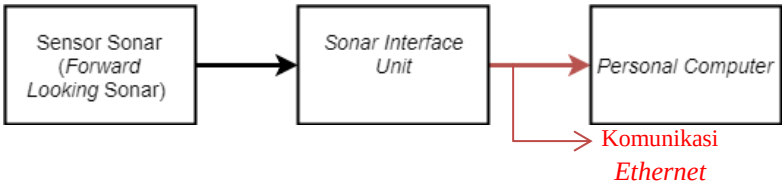
2.8 *Non- Maxima Supression* (NMS)

Bagian yang sangat penting dalam deteksi objek adalah NMS. Saat mencari objek di dalam gambar beberapa titik biasanya ditemukan sebagai objek tetapi beberapa sebenarnya bukan objek [47]. NMS merupakan filter maksimum yang menghitung nilai maksimum dari pixel tetangga [48]. NMS diintegrasikan ke dalam algoritma deteksi objek untuk disaring kotak deteksi. NMS membuat pilihan berdasarkan *Intersection over Union* (IoU) antara kotak deteksi. NMS bekerja dalam dua langkah, yaitu : untuk objek yang diberikan kategori, semua kotak pembatas yang terdeteksi diurutkan berdasarkan nilai dari tinggi ke rendah dan NMS memilih kotak yang mana memiliki nilai tertinggi sebagai hasil deteksi, dan kemudian membuang kotak kandidat lain yang memiliki nilai IoU dengan kotak yang dipilih berada diluar ambang atas [49].

NMS akan menghilangkan gambar semua titik yang terdeteksi karena memiliki probabiliti yang tinggi namun sebenarnya bukan objek atau banyak deteksi yang sama[47].

BAB III
PERANCANGAN APLIKASI SONAR

Pada bab ini akan dijelaskan mengenai pengambilan data onar dan perancangan aplikasi sonar dalam Kapal Tempur Bawah Air (KTBA). Sonar yang digunakan pada KTBA adalah sonar berjenis *Forward-Looking Sonar* (FLS) yang berarti memiliki arah pancaran ke depan. FLS menggunakan sistem *multibeam* yang memiliki arah lingkup pancaran yang lebih luas. Pengambilan data sonar dapat dilihat pada Gambar 3.1. Pengambilan data sonar dimulai dari sensor sonar yang dihubungkan ke *Sonar Interface Unit* (SIU) menggunakan kabel khusus. Sensor sonar akan mengirimkan data yang didapatkan kepada SIU. SIU akan menampung data tersebut dan akan dikirimkan ke *personal computer* menggunakan komunikasi *ethernet.*

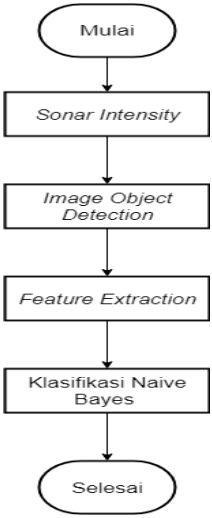


Gambar 3.1 Diagram Blok Pengambilan Data Sonar

Sensor sonar akan memancarkan sinyal berupa gelombang suara ke lingkungan sekitar dengan jarak pancaran 250 m. Pancaran sensor sonar yang mengenai suatu objek akan dipantulkan oleh objek dan diterima *receiver* sensor sonar. Namun, jika pancaran sinyal gelombang suara tidak mengenai objek akan diteruskan. Pancaran yang diterima kembali oleh *receiver* sonar akan dikirimkan ke SIU melalui kabel khusus multiple konektor. Data yang dikirimkan sensor sonar diterima SIU berupa paket data. SIU akan mengirimkan paket data tersebut ke *personal computer* untuk diolah menjadi bentuk gambar menggunakan protokol TCP/IP melalui komunikasi *ethernet* 100 mbps.

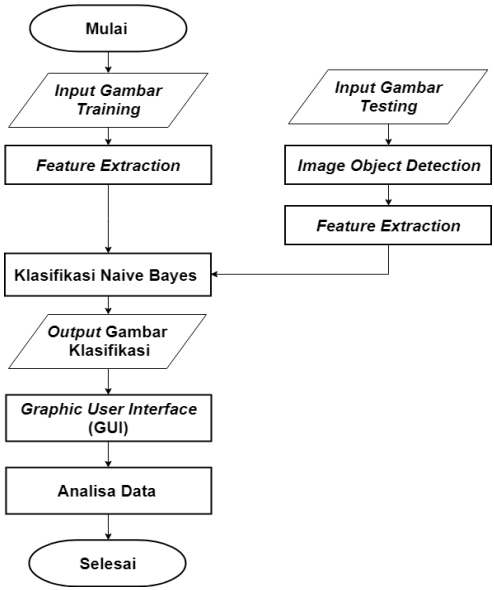
Personal computer memproses paket data dengan dua TCP *port*, yaitu *Command Port* dan *Water Column Data Port. Command Port* berada pada *port* 2209 yang memiliki fungsi untuk memberi perintah ke sistem sonar. *Water Column Data Port* berada pada *port* 2211 dan digunakan sebagai jalur data yang berkaitan dengan pembacaan sonar. Aplikasi sonar ini diharapkan mampu membantu *user* dalam

mengoperasikan KTBA untuk mencapai misinya. Dalam perancangan aplikasi sonar memiliki diagram alir seperti yang ditunjukkan pada Gambar 3.2. Menurut Gambar 3.2 untuk membuat aplikasi sonar memiliki tahapan *sonar intensity, object detection, feature extraction,* dan Klasifikasi Naive Bayes.



Gambar 3.2 Proses Pembuatan Aplikasi Sonar

Tahap *sonar intensity* memiliki tujuan untuk mengubah data dari sonar menjadi bentuk gambar. Tahap *object detection* memiliki tujuan untuk memisahkan objek dengan *background* gambar. Tahap *feature extraction* memiliki tujuan untuk mendeteksi dan mendeskripsikan ciri dari gambar. Tahap klasifikasi memiliki tujuan untuk mengidentifikasi dan mengklasifikasikan gambar. Pada penelitian ini penulis mengerjakan proses pengolahan gambar agar dapat diidentifikasi dan diklasifikasikan, mendesain *Graphical User Interface* (GUI), dan menampilkan hasil gambar identifikasi dan klasifikasi dalam GUI. Dalam memuat aplikasi ini penulis menggunakan gambar dataset wajah manusia dan gambar dari kamera *personal computer*. Proses yang meliputi pengolahan gambar yaitu: *object detection, feature extraction,* dan klasikasi Naive Bayes. Rancangan pengerjaan dapat dilihat dalam diagram alir pada Gambar 3.3.



Gambar 3.3 Rancangan Pengerjaan

Penelitian ini memiliki 2 proses pengerjaan meliputi *training* dan *testing.* Hal tersebut dikarenakan penulis menggunakan metode Naive Bayes, dimana metode tersebut merupakan *supervised learning* untuk melakukan identifikasi dan klasifikasi objek. Proses *training* dimulai dengan adanya *input* gambar *training* dari dataset wajah manusia. Gambar tersebut digunakan untuk masukan pada proses *feature extraction training*. Proses *feature extraction training* mengolah gambar *training* dengan mendeteksi ciri gambar dan mengekstraksi ciri gambar. Hasil ekstraksi kemudian dikelompokan menggunakan K-Means *clustering.* Hasil pengelompokan ekstraksi disebut dengan *Histogram of Occupancy.* Proses *testing* dimulai dengan adanya masukan berupa gambar *testing* dari kamera *personal computer* secara *realtime*. Gambar tersebut kemudian diolah pada proses *object detection* yang memisahkan bagian gambar dianggap objek dengan *background* gambar. Hasil dari proses *object detection* berupa kumpulan potongan gambar yang dianggap objek. Kumpulan potongan objek tersebut kemudian diolah pada proses *feature extraction testing.* Hasil *feature extraction testing* kemudian dikelompokkan dan

menghasilkan *histogram of occupancy testing. Histogram of occupancy training* dan *histogram of occupancy testing* digunakan untuk masukan pada proses klasifikasi Naive Bayes dengan pendekatan Distribusi Gaussian. Hasil akhir dari pengerjaan ini kemudian ditampilkan ke dalam GUI. Proses akhir dari pembuatan aplikasi ini berupa pengujian dan analisis data. Pengujian dilakukan dengan membandingkan metode untuk memperoleh data masukan pada proses klasifikasi Naive Bayes. Adapun dalam penelitian ini menggunakan metode *histogram of occupancy*, sedangkan metode yang digunakan sebagai pembanding adalah metode *histogram of distance* normal dan metode *histogram of distance softweight.*

3.1 *Design Requirement*

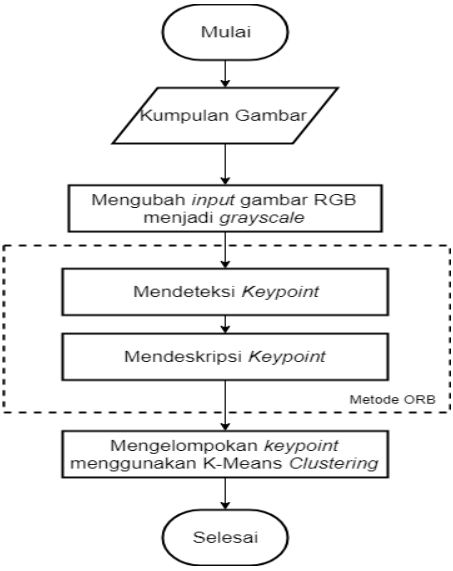
Pada penilitian ini, penulis mengerjakan bagian proses perancangan fungsi identifikasi objek menggunakan masukan berupa gambar dari kamera *webcam personal computer*. Hasil dari penelitian ini berupa *Graphical User Interface* (GUI) yang menampilkan hasil identifikasi objek. GUI berupa video asli secara *realtime* dan menampilkan hasil identifikasi objek yang terdapat dalam video tersebut. Penulis berfokus pada pengerjaan proses *training data, testing data,* identifikasi objek menggunakan klasifikasi Naive Bayes, dan menampilkan hasil identifikasi dan klasifikasi ke dalam GUI. Target yang ingin dicapai penulis meliputi :

- Dapat melakukan identifikasi objek berupa manusia meggunakan pengenalan wajah.
- Dapat menampilkan hasil identifikasi ke dalam GUI dengan memberikan tanda berupa kotak pembatas pada bagian yang dianggap objek.
- Dapat digunakan untuk mengidentifikasi *obstacle* dari data Sonar.

3.2 *Training Data*

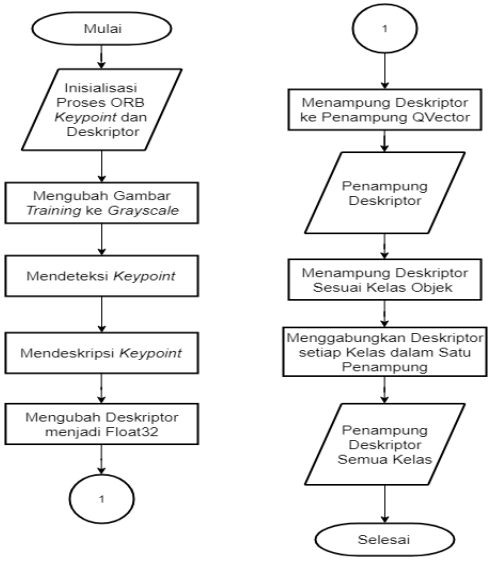
Proses untuk memberikan pembelajaran pada sistem menggunakan data latih. Proses *training data* yang dilakukan pada penelitian ini meliputi proses *feature extraction* dengan metode *Oriented FAST and Rotated BRIEF* (ORB) dan K-Means *Clustering*. Data masukan gambar *training* berupa gambar wajah dari manusia. Hal tersebut dikarenakan penelitian ini memiliki tujuan untuk mengidentifikasi dan mengklasifikasikan objek manusia berdasarkan wajah manusia. Proses *training data* meliputi tahapan yang dapat

dilihat pada Gambar 3.4. Menurut Gambar 3.4 masukan pada tahap *training data* berupa kumpulan gambar. Dari kumpulan gambar tersebut, kemudian diubah menjadi bentuk *grayscale* atau keabu-abuan. Setelah itu mendeteksi *keypoint* dan mendeskripsikan *keypoint* menggunakan metode ORB pada setiap gambar yang telah diubah menjadi bentuk *grayscale. Keypoint* merupakan titik yang mengandung banyak informasi atau ciri dari gambar tersebut. Setelah *keypoint* terdeteksi tahap berikutnya adalah mendeskripsikan *keypoint*. Hasil pendeskripsian *keypoint* disebut dengan deskriptor. Deskriptor yang dihasilkan kemudian dikelompokan berdasarkan struktur yang dimiliki. Proses pengelompokan ini bertujuan untuk meminimalisirkan ciri atau *feature* dari objek yang memiliki struktur mirip. Proses pengelompokan dilakukan menggunakan metode K-Means *Clustering*. Penggunaan metode ini dikarenakan K-Means merupakan metode *unsupervised learning* yang tidak membutuhkan proses pembelajaran. Metode ini digunakan ketika akan mengelompokan data yang belum berlabel. Deskriptor merupakan data yang belum berlabel. Proses K-Means *clustering* menghasilkan data berupa histogram.



Gambar 3.4 Proses *Training Data*

Program *feature extraction training* menggunakan metode ORB dapat ditunjukkan pada Gambar 3.5. Program *feature exraction* menggunakan metode ORB pada penelitian ini menggunakan bantuan *library* OpenCV.



Gambar 3.5 Diagram Alir Proses ORB

Menginisialisasi proses *feature extraction* menggunakan metode ORB, menginisialisai *keypoint*, dan menginisialisasi deskriptor merupakan langkah awal untuk memulai program. Penginisialisasian ORB dilakukan dengan menuliskan program "Ptr<ORB> DetectORB = ORB::create();". Menurut program tersebut proses *feature extraction* ORB diinisialisasi dengan nama DetectORB. Setelah melakukan proses inisialisasi selanjutnya adalah mengubah gambar *training* dari *Red-Green-Blue* (RGB) menjadi bentuk *grayscale* atau keabu-abuan menggunakan *library* OpenCV yaitu "cvtColor". Mengubah gambar RGB menjadi *grayscale* dilakukan karena metode ORB menggunakan *library* OpenCV membutuhkan masukan berupa gambar *grayscale.* Setelah mengubah ke bentuk *grayscale* selanjutnya adalah melakukan deteksi *keypoint* dengan bantuan *library* OpenCV. Deteksi *keypoint* dilakukan dengan menuliskan program "DetectORB->*detect*".

Pemanggilan fungsi deteksi *keypoint* dengan bantuan *library* OpenCV dilakukan dengan nama yang mewakili proses *feature extraction training* menggunakan metode ORB yaitu DetectORB.

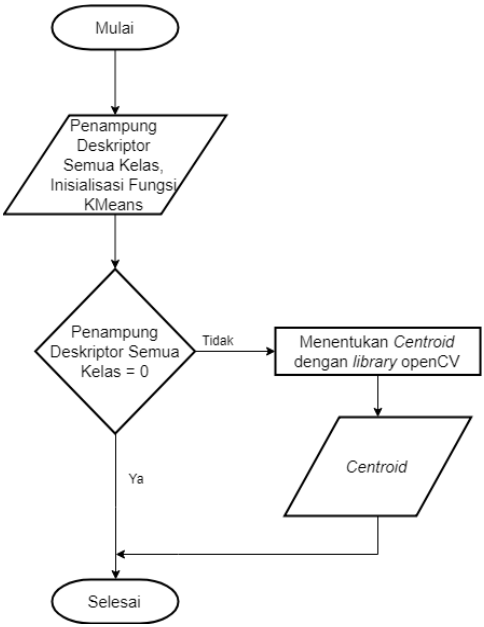
Ketika pendeteksian *keypoint* selesai, dilanjutkan dengan mendeskripsikan *keypoint* menggunakan fungsi *compute* yang tersedia pada *library* OpenCV. Hasil deskripsi atau deskriptor tersebut berupa data *binary* 256 bit yang dituliskan dalam data 32 byte. Deskriptor yang telah didapatkan kemudian diubah ke dalam bentuk tipe data *float* 32. Hal ini dilakukan karena pada proses selanjutnya yaitu K-Means *clustering* membutuhkan masukan bentuk data bertipe *float*32. Deskriptor dari semua gambar *training* kemudian disimpan kedalam penampung bertipe QVector yang diberi nama "Collection Of Descriptor Training". Semua gambar *training* yang telah memiliki deskriptor ditampung pada penampung yang disebut "Member Of Class" berdasarkan nama kelas setiap gambar *traning.* Kemudian, semua deskriptor *training* dari setiap kelas digabung dan ditampung dalam penampung deskriptor *training* semua kelas yang disebut "Collection Data Training All Class". Proses *feature extraction* ORB tersebut dapat dilihat dalam *listing* program yang ditunjukkan pada Gambar 3.6 dan untuk program keseluruhan terdapat pada Lampiran A-9 dan A-10.

```
Mat GrayImageTraining;
vector<KeyPoint> KeypointTraining;
Ptr<ORB> DetectORB = ORB::create();
Mat DescriptorTraining;
Mat DescriptorTrainingFloat;
cvtColor(Image,GrayImageTraining,CV_RGB2GRAY);
DetectORB->detect(GrayImageTraining,KeypointTraining);
DetectORB->compute(GrayImageTraining,KeypointTraining,DescriptorTraining);
DescriptorTraining.convertTo(DescriptorTrainingFloat,CV_32FC1);
CollectionOfDescritorTraining.append(DescriptorTrainingFloat);
```

Gambar 3.6 *List* Program *Feature Extraction Training*

Penampung deskriptor semua kelas kemudian digunakan sebagai masukan dalam proses K-Means *clustering.* Langkah awal untuk

melakukan K-Means *clustering* adalah menentukan *centroid* dari deskriptor semua kelas. *Centroid* merupakan titik pusat data, dalam hal ini diasumsikan dengan *keypoint* yang memiliki ciri paling kuat dari suatu objek. Penentuan *centroid* yang dilakukan pada penelitian ini menggunakan bantuan *library* OpenCV. Proses penentuan *centroid* ditunjukkan pada Gambar 3.7.



Gambar 3.7 Diagram Alir Penentuan *Centroid*

Program dimulai dengan adanya masukan penampung deskriptor semua kelas, inisialisasi fungsi K-Means *library* OpenCV, dan menentukan jumlah *centroid* yang digunakan. Dalam hal ini, jumlah *centroid* ditentukan sebanyak empat. Setelah itu dilakukan pemeriksaan kondisi penampung deskriptor semua kelas. Jika penampung deskriptor semua kelas tidak ada data maka proses akan selesai, jika penampung deskriptor semua kelas terdapat data maka dilakukan proses penentuan *centroid* dengan bantuan *library* OpenCV. Penentuan *centroid* ini dilakukan dengan menuliskan fungsi "kmeans" yang terdapat pada *library* OpenCV. Hasil proses ini berupa *centroid*

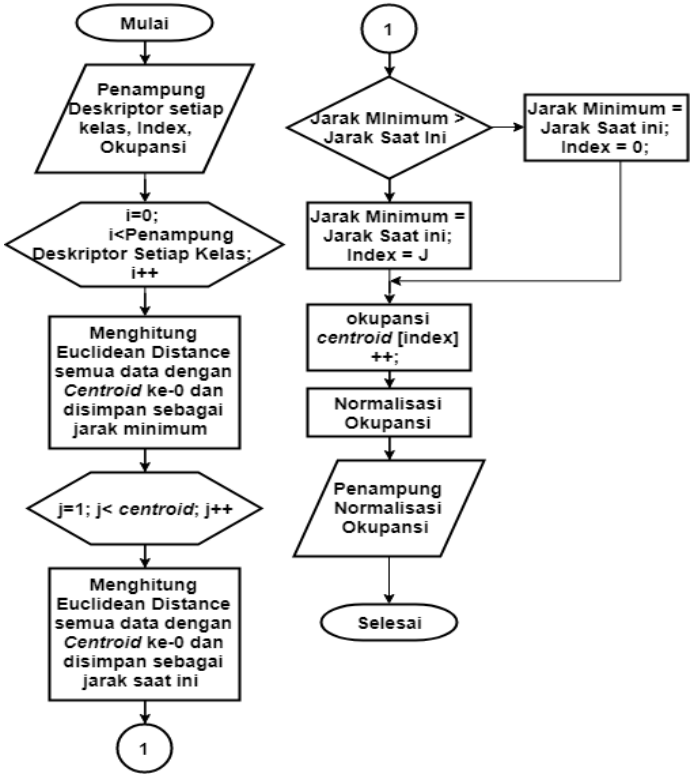
dengan jumlah yang telah ditentukan. Proses penentuan *centroid* dapat dilihat dalam Gambar 3.8 yang menunjukkan *listing* program penentuan *centroid.*

```
int clusterCount = 5;
Mat labels;
int attempts = 5;

if(CollectionDataTrainingAllClass.rows == 0)
{

}
else
{
    kmeans(CollectionDataTrainingAllClass, clusterCount, labels,
    TermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS, 10000, 0.0001),
    attempts, KMEANS_PP_CENTERS, CentersClass );

}
```

Gambar 3.8 *List Program* Proses Penentuan *Centroid*

Proses penentuan *centroid* menggunakan *library* OpenCV dilakukan dengan kriteria maksimum iterasi yang ditentukan dan *epsilon*, yaitu batas nilai akurasi yang ditentukan. Jika, salah satu kondisi tersebut terpenuhi maka iterasi proses penentuan *centroid* akan berhenti. Kemudian, menghitung banyak deskriptor yang berada disekitar masing-masing *centroid.* Untuk melakukan hal tersebut penelitian ini menggunakan *histogram of occupancy* dengan pendeketan Euclidean Distance. *Histogram of occupancy* menunjukkan banyak deskriptor yang berada disekitar masing-masing centroid. Banyak *centroid* ditentukan berdasarkan perhitunga jarak antara setiap *centroid* dengan deskriptor masing-masing data *training.* Perhitungan jarak ini menggunakan metode Euclidean Distance. Dalam melakukan perhitungan Euclidean Distance penelitian ini dibantu *library* OpenCV. *Centroid* yang memiliki jarak terdekat dengan deskriptor, kemudian diberikan okupansi satu. Hasil *histogram of occupancy* berupa data

histogram yang menunjukkan kelompok dari masing-masing *centroid* setiap data *training.* Proses tersebut ditunjukkan pada Gambar 3.9.



Gambar 3.9 Diagram Alir Perhitungan *Histogram of Occupancy*

Program dimulai dengan adanya masukan berupa penampung deskriptor data *taining* masing-masing kelas, penginisialisasian index yang menunjukkan masing-masing *centroid*, dan penampung okupansi yang diperoleh masing-masing *centroid*. Kemudian mencari jarak antara setiap data yang yang tertampung pada penampung deskriptor *data training* masing-masing kelas dengan *centroid.* Jarak setiap data dengan *centroid* ke-0 diinisialisasikan sebagai jarak minimum. Jarak setiap data dengan *centroid* lainnya diinisialisasikan sebagai jarak saat ini. Setelah itu dilakukan perbandingan nilai jarak minimum dengan

jarak saat ini. Jika jarak minimum lebih besar dari jarak saat ini maka nilai jarak minimum digantikan dengan jarak saat ini dan index yang menunjukkan *centroid* dengan jarak minimum akan memiliki okupansi 1. Hal tersebut terjadi secara berulang hingga data terakhir dari penampung deskriptor data *training* masing-masing kelas. Setelah itu dilakukan proses normalisasi okupansi untuk masing-masing *centroid* dengan cara membagi okupansi dengan jumlah *keypoint* yang terdeteksi. Hasil normalisasi okupansi tersebut kemudian akan ditampung dan digunakan pada proses klasifikasi. Program untuk perhitungan *histogram of occupancy* dapat dlihat pada Gambar 3.10 dan untuk program keseluruhan terdapat pada Lampiran A-10.

```
for(int l = 1; l < CentersClass.rows; l++ )
{
    double CurrentDistance = norm(MemberofClass[indexclass].CollectionOfDescritorTraining_()[j].row(k),
    CentersClass.row(l),NORM_L2);
    if(MinimumDistance > CurrentDistance)
    {
        MinimumDistance = CurrentDistance;
        index = l;
    }

}
```

Gambar 3.10 *Listing* Program *Histogram of Occupancy*

3.3 *Testing Data*

Proses untuk menguji suatu sistem yang telah diberikan proses pembelajaran. *Testing Data* meliputi dua proses yaitu, *object detection* dan *feature extraction testing. Object detection* memiliki tujuan untuk memisahkan objek dengan *background*. *Feature Extraction* memiliki tujuan untuk mendeteksi dan mendeskripsikan ciri atau *feature* pada objek.

3.3.1 *Object Detection*

Penelitian ini menggunakan algoritma *region proposal* untuk melakukan *object detection. Region proposal* merupakan algoritma yang membagi gambar masukan menjadi beberapa potongan kecil yang dianggap paling mungkin menjadi objek. Hasil keluaran dari algoritma *region proposal* ini dapat saling tumpang tindih dan mungkin tidak mengandung objek dengan sempurna, tetapi terdapat

region proposal yang akan sangat dekat dengan objek aktual dalam gambar. Salah satu metode untuk melakukan algoritma *region proposal* adalah *selective search.* Proses *selective search* dalam menentukan objek menggunakan proses segmentasi dan unsur kesamaan dengan beberapa parameter. Parameter tersebut adalah kesamaan warna, kesamaan bentuk, kesamaan ukuran, kesamaan tekstur, dan gabungan dari ke-empat parameter. Proses *selective search* pada penelitian ini dilakukan dengan bantuan *library* OpenCV dengan menambahkan *extra module* OpenCV. Diagram alir pemrograman *selective search* dapat dilihat pada Gambar 3.11.

Gambar 3.11 Diagram Alir *Selective Search*

Langkah awal untuk melakukan pemrograman proses *selective search* adalah dengan adanya masukan gambar dan inisialisasi proses *selective search.* Penginisialisasian proses *selective search* dilakukan dengan menuliskan "Ptr <SelectiveSearchSegmentation> selective Search = create Selective Search Segmentation ()". Menurut penginisialisasian, variabel yang mewakili proses *selective seacrh*

diberi nama selective Search. Dari variabel tersebut kemudian proses s*elective search* dipanggil dengan menuliskan program "selective Search -> process (rects);". Proses *selective search* menghasilkan gambar yang terdapat tanda berupa kota pembatas pada bagian yang dianggap sebagai objek. Dari hasil gambar kemudian dipotong sesuai dengan kotak pembatas. Hasil potongan gambar merupakan hasil akhir dari proses *object detection*, dalam hal ini potongan gambar dianggap sebagai objek. Proses *object detection* menggunakan *selective search* ini terdapat pada program *selective search* Gambar 3.12 dan untuk program selengkapnya dapat dilihat pada Lampiran A-10.

```
rects.clear();
Ptr<SelectiveSearchSegmentation> selectiveSearch = createSelectiveSearchSegmentation();
selectiveSearch->setBaseImage(testImage);
selectiveSearch->switchToSingleStrategy();
selectiveSearch->process(rects);
int numShowRect = rects.size();
QVector<Mat>sumCrop;
```

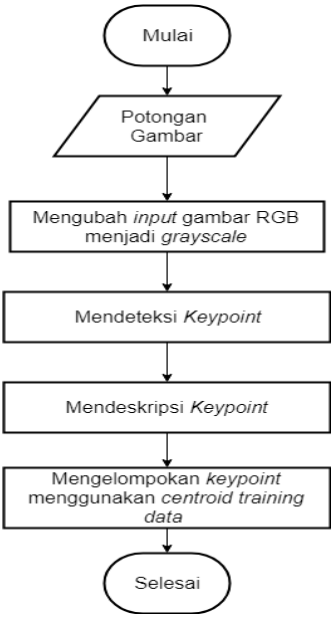
Gambar 3.12 *Listing* Program *Selective Search*

Pada *listing* program *selective search* tersebut, semua potongan gambar disimpan dalam penampung dengan tipe data QVector<Mat> yang disebut dengan sumCrop. Hasil dari proses *object detection* ini, kemudian digunakan sebagai masukan pada proses *feature etxtraction testing.*

3.3.2 *Feature Extraction Testing*

Proses untuk mendeteksi dan mendeskripsikan ciri atau *feature* dari gambar *testing*. Proses *feature extraction testing* menggunakan metode yang sama dengan proses *feature extraction training* yaitu menggunakan metode *Oriented FAST and Rotated BRIEF* (ORB). Proses *feature extraction testing* dilakukan dengan bantuan *library* OpenCV. Adapun proses tersebut dapat dilihat dalam diagram alir yang ditunjukkan pada Gambar 3.13. Pada Gambar 3.13 ditunjukkan masukan dari proses *feature extraction testing* berupa potongan gambar. Dalam hal ini potongan gambar merupakan hasil dari proses *object detection.* Potongan gambar kemudian diubah kedalam bentuk *grayscale* atau keabu-abuan. Setelah itu mendeteksi *keypoint* dari

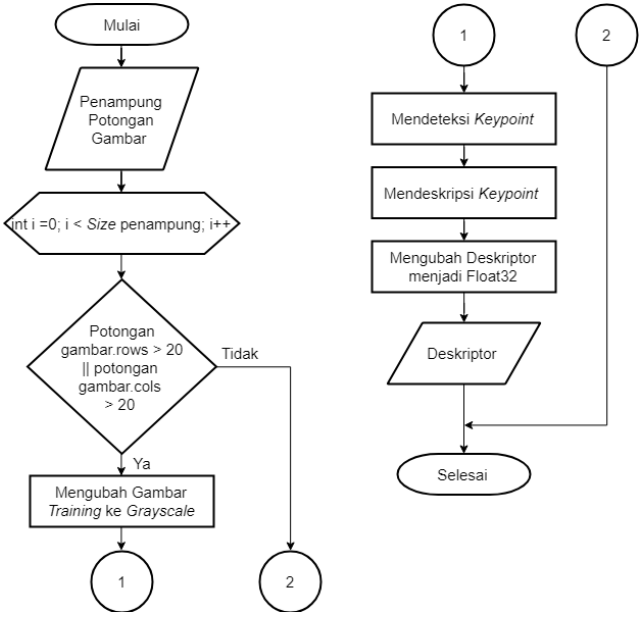
setiap potongan gambar. Hasil deteksi *keypoint*, kemudian dideskripsi sehingga menghasilkan deskriptor. Deskriptor kemudian dibandingkan dengan *centroid* yang dihasilkan dari proses *training data*. Proses membandingkan deskriptor dengan *centroid* ini dilakukan dengan mencari jarak antara deskriptor dengan *cetroid*. Adapun tujuan dari proses ini untuk mencari *histogram of Occupancy* dari gambar *testing* yang dibandingkan dengan gambar *training*.



Gambar 3.13 Proses *Feature Extraction Testing*

Proses *feature extraction testing* ini terdapat pada program yang memiliki diagram alir ditunjukkan pada Gambar 3.14. Pada Gambar 3.14 program dimulai dengan adanya masukan berupa penampung potongan gambar yang diberi nama “sumCrop”.Proses dalam program ini akan terus berlangsung hingga potongan gambar terakhir yang tertampung pada sumCrop. Program akan memeriksa kondisi sumCrop sebelum melakukan proses *feature extraction testing*. Jika potongan gambar yang terdapat penampung sumCrop memiliki ukuran baris dan kolom lebih dari 20 maka proses *feature extraction testing*

menggunakan metode ORB dilakukan. Namun, jika potongan gambar di dalam penampung sumCrop memiliki ukuran baris dan kolom kurang dari 20 maka proses *feature extraction testing* menggunakan metode ORB tidak akan dilakukan.

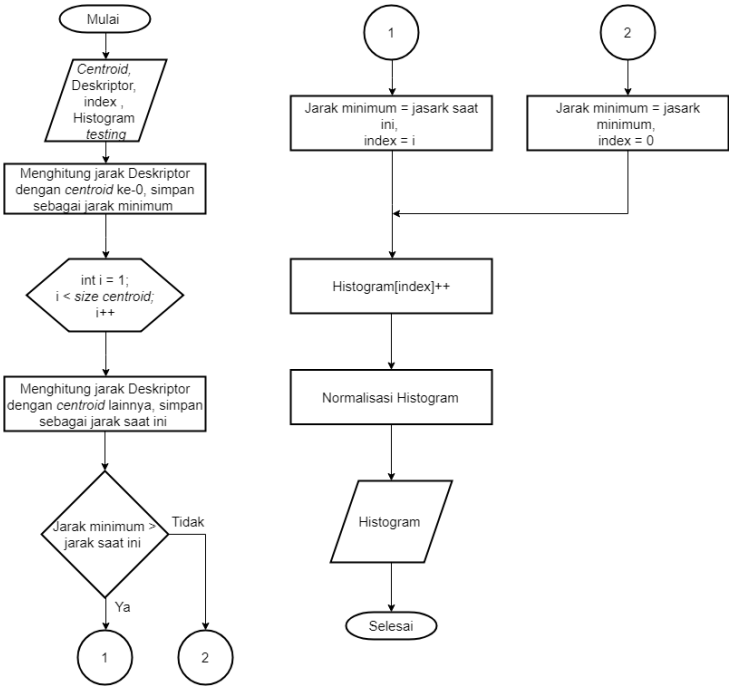


Gambar 3.14 Diagram Alir Program *Feature Extraxtion Testing*

Proses *feature extraction* metode ORB menggunakan *library* OpenCV diawali dengan mengubah potongan gambar menjadi bentuk *grayscale.* Kemudian mencari *keypoint* pada potongan gambar *grayscale. Keypoint* kemudian dideskripsikan sehingga menghasilkan deskriptor. Deskriptor yang dihasilkan memiliki tipe data *binary*. Setelah itu mengubah tipe data deskriptor dari *binary* menjadi tipe data *float*32. Hasil deskriptor tersebut kemudian akan dibandingkan dengan *centroid* yang dihasilkan pada proses *training data.* Program proses *feature extraction testing* ditunjukkan pada Gambar 3.15 yang merupakan *list* program proses *feature extraction testing*. Untuk program keseluruhan terdapat pada Lampiran A-10.

```
Mat GrayTestImage;
vector<KeyPoint> KeypointTesting;
Ptr<ORB> DetectORBTesting = ORB::create();
Mat DescriptorTesting;
cvtColor(sumCrop[i],GrayTestImage,CV_RGB2GRAY);
```

Gambar 3.15 *List* Program Proses *Feature Extraction Testing*



Gambar 3.16 Diagram Alir Proses Penentuan *Centroid*

Proses membandingkan deskriptor dengan *centroid* bertujuan untuk menghitung *histogram of occupancy* dari deskriptor *testing* terhadap *centroid training*. Program untuk menghitung *histogram of occupancy* dapat dilihat dari Gambar 3.16. Pada Gambar 3.16 ditunjukkan program dimulai dengan adanya masukan berupa *centroid*, deskriptor, penginisialisasian variabel index, dan penginisialisasian

penampung *histogram of occupancy testing*. Deskriptor terlebih dahulu dibandingkan dengan *centroid* ke -0 dengan menghitung jarak antara keduanya menggunakan Euclidean Distance dengan bantuan *library* OpenCV. Jarak tersebut diasumsikan sebagai jarak minimum. Setelah itu deskriptor dibandingkan dengan *centroid* lainnya dan diasumsikan sebagai jarak saat ini. Setelah mendapatkan jarak minimum dan jarak saat ini, kedua jarak tersebut dibandingkan dan dicari nilai jarak yang paling kecil. Jarak tersebut diasumsikan sebagai jarak minimum menggantikan jarak minimum sebelumnya. *Centroid* yang memiliki jarak minimum tersebut diberikan okupansi dengan nilai satu. Proses tersebut berlangsung hingga *centroid* terakhir. Program yang menjelaskan proses tersebut terdapat pada Gambar 3.17. Program keselurahan terdapat pada Lampiran A-10.

```
for(int k = 1; k < CentersClass.rows; k++ )
{

    double CurrentDistance = norm(DescriptorTestingFloat.row(j),
                                  CentersClass.row(k),NORM_L2);
    if(MinimumDistance > CurrentDistance)
    {
        MinimumDistance = CurrentDistance;
        index = k;
    }
}
```

Gambar 3.17 *Listing* Program *Histogram of Occupancy Testing*

3.4 Klasifikasi Naive Bayes

Proses untuk mengidentifikasi objek *testing* dengan menghitung nilai peluang setiap data *testing* terhadap data *trainig* menggunakan metode Naive Bayes. Metode Naive Bayes memiliki persamaan sebagai berikut:

$$\boldsymbol{P(H|X) = \frac{P(X|H) * P(H)}{P(X)}} \qquad (\mathbf{3.1})$$

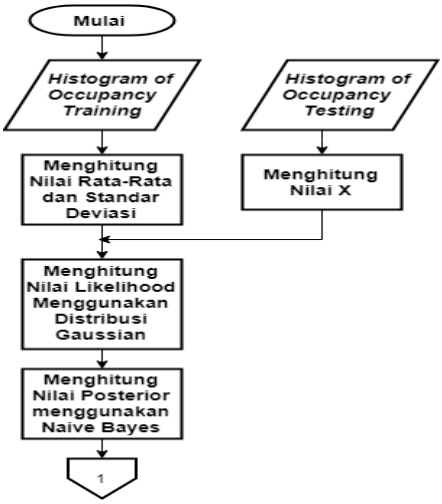
Dengan,
$P(H|X)$ = Posterior
$P(X|H)$ = Likelihood

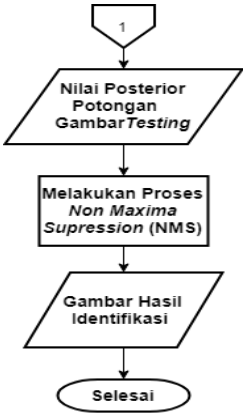
$P(H)$ = Prior
$P(X)$ = Evidence

Pada penelitian ini evidence dan prior memiliki nilai yang selalu tetap untuk semua kelas. Sehingga pada penelitian ini dapat dituliskan persamaan :

$$\boldsymbol{Posterior} \approx \boldsymbol{Likelihood} \qquad (\mathbf{3.2})$$

Dengan persamaan tersebut probabilitas yang mempengaruhi identifikasi dan klasifikasi objek adalah nilai likelihood. Pada penelitian ini proses klasifikasi Naive Bayes menggunakan pendekatan metode Distribusi Gaussian untuk mendapatkan nilai likelihood. Metode Distribusi Gaussian memerlukan parameter nilai rata-rata, standar deviasi, dan varibel x yang mempengaruhi nilai hasil Distribusi Gaussian. Proses klasifikasi Naive Bayes pada penelitian ini menggunakan masukan *histogram of occupacy training* dan *histogram of occupancy testing* untuk menentukan nilai rata-rata, standar deviasi, dan nilai x. *Histogram of occupancy training* digunakan untuk menghitung nilai rata-rata dan nilai standar deviasi, sedangkan *histogram of occupancy testing* digunakan untuk nilai x. Gambar 3.18 menunjukan alur proses yang terjadi pada klasifikasi Naive Bayes.





Gambar 3.18 Diagram Alir Proses Klasifikasi Naive Bayes

Nilai rata-rata dan standar deviasi dihitung berdasarkan *histogram of occupancy training* yang terdiri dari okupansi setiap *centroid* terhadap data *training.* Nilai rata-rata dan standar deviasi dihitung setiap *centroid* dari semua data *training*. Jika terdapat empat *centroid* maka akan dihasilkan nilai rata-rata dan standar deviasi berjumlah empat seperti jumlah *centroid.* Sedangkan nilai x merupakan nilai *histogram of occupancy testing* terhadap *centroid.* Nilai rata-rata, nilai standar deviasi, dan nilai x yang telah didapatkan kemudian digunakan untuk masukan Distribusi Gaussian yang memiliki rumus sebagai berikut:

$$\boldsymbol{Distribusi\ Gaussian} = \frac{\mathbf{1}}{\sqrt{\mathbf{2}\pi\sigma}} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)} \qquad (\mathbf{3.3})$$

Hasil dari Distribusi Gaussian berupa nilai likelihood data testing terhadap setiap *centroid.* Untuk menghitung nilai posterior maka, semua nilai likelihood setiap *centroid* yang didapatkan dikalikan. Nilai posterior yang bukan nol kemudian disimpan dalam penampung. Penampung tersebut berisikan semua nilai posterior potongan gambar yang dianggap sebagai objek dan memiliki nilai posterior bukan nol. Dari penampung posterior kemudian dilakukan proses *non-maxima supression* (NMS) untuk membuang posterior bukan objek namun

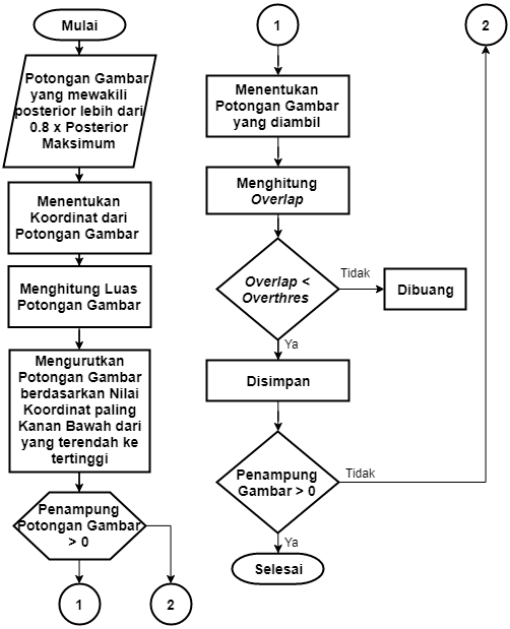
dianggap sebagai objek. Adapun program untuk menghitung nilai rata-rata dan standar deviasi proses klasifikasi Naive Bayes ditunjukkan pada Gambar 3.19. Program keseluruhan terdapat pada Lampiran A-10.

```
for (int i = 0; i < CentersClass.rows; i++)
{
    for (int j = 0; j < CollectionTrainingHistogram.size(); j++)
    {
        AllTraining += CollectionTrainingHistogram[j][i];
    }
    sumMeans[i] = AllTraining/CollectionTrainingHistogram.size();
    for(int k=0; k < CollectionTrainingHistogram.size(); k++)
    {
        AllTrainingMeans += ((CollectionTrainingHistogram[k][i]-sumMeans[i])
                            *(CollectionTrainingHistogram[k][i]-sumMeans[i]));
    }
    sumDeviasi[i] = sqrt(AllTrainingMeans/CollectionTrainingHistogram.size());
    AllTraining = 0;
    AllTrainingMeans = 0;

}
```

Gambar 3.19 *Listing* Program Klasifikasi Naive Bayes

NMS pada penelitian ini memiliki alur proses yang ditunjukkan pada Gambar 3.20. Dari penampung posterior dicari nilai posterior maksimum. Kemudian posterior lainnya yang ada di penampung dibandingkan dengan posterior maksimum. Jika terdapat posterior yang nilainya kurang dari 0.8 X posterior maksimum, maka nilai posterior tersebut dibuang. Potongan gambar yang memiliki nilai probabilitas posterior yang lebih dari 0.8 X posterior maksimum digunakan sebagai masukan proses NMS. Setelah itu menghitung nilai koordinat setiap potongan gambar dan menghitung luas area dari setiap potongan gambar. Kemudian mengurutkan potongan gambar tersebut berdasarkan nilai koordinat yang paling kanan bawah dan disimpan dalam penampung. Setelah itu melakukan perulangan selama penampung memiliki potongan gambar. Dalam perulangan tersebut potongan gambar yang memiliki nilai koordinat paling kanan bawah diasumsikan sebagai potongan gambar yang dianggap sebagai objek saat ini. Dari potongan gambar tersebut kemudian dibandingkan dengan potongan gambar lainnya dan dihitung *overlap* dari setiap potongan gambar.

Potongan gambar yang memiliki nilai *overlap* kurang dari *threshold* yaitu 0.4 akan dibuang, sedangkan potongan gambar yang memiliki nilai *overlap* lebih dari 0.4 akan disimpan. Jika potongan gambar yang ada dalam penampung masih lebih dari 0 maka proses tersebut akan berlangsung. Hasil akhir dari proses NMS berupa potongan gambar yang diidentifikasi dan diklasifikasikan sebagai objek. Adapun sebagian proses NMS ditunjukkan pada Gambar 3.21. Untuk program selengkapnya terdapat pada Lampiran A-10.



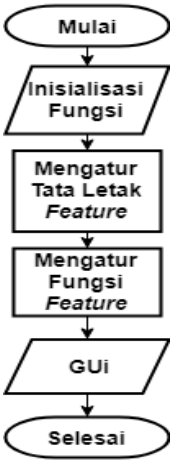
Gambar 3.20 Diagram Alir Proses *Non- Maxima Supression*

```
int x1, y1, x2, y2;
x1 = tempScaleRect[indexRect[i]].x;
y1 = tempScaleRect[indexRect[i]].y;
x2 = x1 + tempScaleRect[indexRect[i]].width;
y2 = x1 + tempScaleRect[indexRect[i]].height;

float area;
area = (x2 - x1 + 1)*(y2-y1 +1);
```

Gambar 3.21 *Listing* Program *Non-Maxima Supression*

3.5 Perancangan *Graphical User Interface* (GUI)

Perancangan GUI memiliki tujuan untuk menampilkan hasil identifikasi dan klasifikasi objek. GUI yang dirancang memiliki *feature* meliputi label untuk menampilkan video asli, label untuk menampilkan video hasil identifikasi objek, *pushbutton* "play" untuk mengakses video asli dari kamera *webcam personal computer*, *pushbutton* "classify" untuk mengakses dan mengidentifikasi objek dalam video asli, *pushbutton* "stop" untuk menghentikan akses video dan proses identifikasi objek. Perancangan GUI ini memanfaatkan fungsi-fungsi pada *framework* Qt yaitu *QLabel*, *QPushbutton* dan *QLayout*. *QLabel* digunakan untuk membuat judul GUI yaitu "*Obstacle Identification* ", membuat tulisan pada video asli yaitu "Video Target", membuat tulisan pada video hasil identifikasi yaitu "Video Result", menampilkan video asli, dan menampilkan video hasil identifikasi objek. *QPushbutton* digunakan untuk membuat *pushbutton* "play", "classify" , dan "stop". *Qlayout* digunakan untuk mangatur tata letak *feature.* Perancangan GUI ini meliputi pengolahan tata letak *feature* yang digunakan dan pengaturan fungsi dari setiap *feature.*Pemrosesan GUI dapat dilihat pada Gambar 3.22.



Gambar 3.22 Diagram Alir Perancangan GUI

Proses dimulai dengan penginisialisasian *library* Qt yang akan digunakan. *Library* Qt digunakan untuk membuat *feature* yang

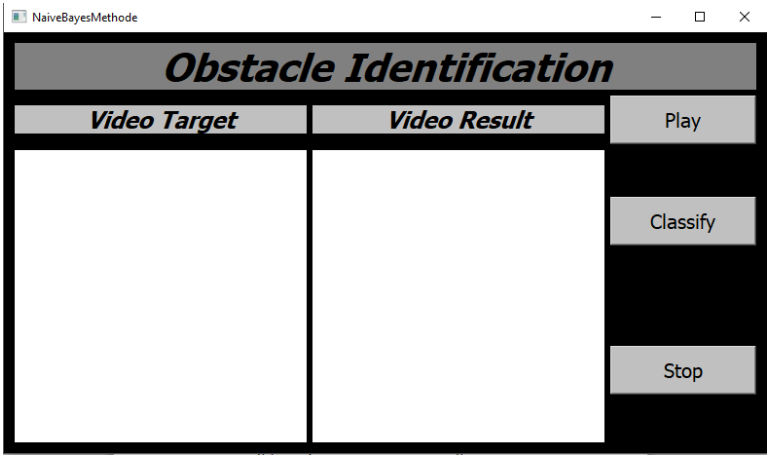
memiliki beberapa fungsi. Setelah *feature* dibuat selanjutnya adalah mengatur tata letak dari *feature*. Tata letak GUI direpresentasikan sebagai baris dan kolom. Dalam perancangan GUI ini menggunakan ukuran 3 kolom dan 5 baris. Proses pengaturan tata letak meliputi: gaya penulisan judul, pemilihan *font* dan *size* font yang digunakan, pengaturan warna *background* dan pengaturan ukuran setiap *feature*. *QLabel* dan *QPushbutton* dapat diberikan tulisan dengan menuliskan program "setText". Untuk mengatur warna tulisan dan *background* dapat dilakukan dengan menulisakan "setStyleSheet". Pengaturan *size* dan *font* dapat dilakukan dengan menuliskan "setFont". Tata letak *feature* dapat diatur dengan menuliskan "setAlignment". Dalam hal ini, *feature* dapat diatur akan diletakkan pada *center, right, left, bottom* atau *top*. Sedangkan untuk mengatur letak *feature* pada *layout* mengunakan tulisan "addWidget" dengan menuliskan *feature* akan diletakkan pada baris kebera,kolom keberapa, jumlah baris yang digunakan, jumlah kolom yang digunakan. Program untuk mengatur tata letak dapat dilihat dalam Gambar 3.23. Program keseluruhan terdapat pada Lampiran A-8.

```
button_1->setText("Play");
button_1->setStyleSheet("QPushButton {background-color : silver}");
button_1->setFont(QFont("lucida", 15));
button_1->setSizePolicy(QSizePolicy::Fixed,QSizePolicy::Fixed );
button_1->setFixedSize(150,50);
layout->addWidget(button_1, 1, 2, 1, 1, Qt::AlignCenter);
```

Gambar 3.23 *Listing* Program Perancangan GUI.

Hasil GUI pada penelitian ini dapat dilihat pada Gambar 3.24. Setelah tata letak selesai dilakukan selanjutnya adalah mengatur fungsi dari setiap *feature.* Dalam hal ini tidak semua *feature* yang memiliki fungsi. *Feature* yang memiliki fungsi adalah *feature pushbutton* "play", *pushbutton* "classify", *pushutton* "stop", label penampilan video asli, dan label penampilan video hasil identifikasi. Untuk memberikan fungsi pada *feature* dilakukan pemrograman signal dan slot. Pada proses menampilkan video asli dan video hasil identifikasi menggunakan *QThread*. Ketika *pushbutton* "play" ditekan maka video target akan mengakses kamera *webcam personal computer*. Ketika *pushbutton* "classify" ditekan maka video result akan mengakses

kamera dan mengidentifikasi setiap *frame* dalam video dan ditampilkan. Ketika *pushbutton* "stop" ditekan proses akses video dan identifikasi akan berhenti. Program yang menjelaskan proses pengaturan fungsi ditunjukkan pada Gambar 3.25.Program keseluruhan terdapat pada lampiran A-6, A-7 dan A-8.



Gambar 3.24 Tampilan GUI

```
stop_ = true;
threadCam->stop_();
objectClassify->stop_();
connect(objectClassify, SIGNAL(finished()), thread_, SLOT(quit()));
connect(objectClassify, SIGNAL(finished()), objectClassify, SLOT(deleteLater()));
connect(objectClassify, SIGNAL(finished()), thread_, SLOT(deleteLater()));
thread_->exit();
```

Gambar 3.25 *List Program* Pengaturan Fungsi

Gambar 3.25 merupakan program pengaturan fungsi pada *pushbutton* "Stop".

3.6 Kriteria Pengujian

Pengujian aplikasi sonar untuk mengidentifikasi *obstacle* menggunakan data *training* berjumlah 30 gambar wajah seperti yang

ditunjukkan pada Gambar 3.26. Gambar *training* merupakan wajah asia dengan ekspresi *emotion.* Gambar tersebut digunakan untuk mendapatkan *feature* yang sama antara gambar *training* dan gambar *testing*. Hal tersebut dikarenakan gambar *testing* yang akan digunakan merupakan wajah asia.

Gambar 3.26 Gambar Data *Training* [50]

Pengujian menggunakan data *testing* berupa gambar dari kamera *webcam personal computer* secara *realtime*. Data *testing* yang digunakan untuk pengujian berjumlah 200 data dengan 100 data gambar postif dan 100 data gambar negatif. Gambar positif merupakan gambar yang terdapat objek, sedangkan gambar negatif merupakan gambar yang tidak terdapat objek. Data gambar positif dapat dilihat pada Gambar 3.27, sedangkan data gambar negatif dapat dilihat pada Gambar 3.28. Untuk mengetahui perbedaan pengujian yang dilakukan, pengujian ini menggunakan parameter *recall, precision,* dan *accuracy.* Analisis pengujian menggunakan metode *confussion matrix* untuk mencari nilai parameter tersebut.

Gambar 3.27 Data Gambar Positif

Gambar 3.28 Data Gambar Negatif

Fungsi aplikasi sonar dengan menggunakan metode naive bayes diuji tingkat keberhasilannya dalam mengidentifikasi dan mengklasifikasikan objek. Pengujian dilakukan dengan memberikan data *testing* secara acak dan menghitung jumlah data yang diklasifikasikan secara benar. Dari 100 data positif, jika dapat mengidentifikasi dan mengklasifikasikasikan secara benar maka dianggap sebagai *true* positif. Jika mengidentifikasi dan mengklasifikasikan secara salah maka dianggap sebagai *false* positif. Gambar *true* positif dan *false* positif ditunjukkan pada Gambar 3.29.

Gambar 3.29 Data Gambar *True* Postif dan *False* Positif

Dari 100 data negatif, jika tidak mengidentifikasi dan mengklasifikasikan objek maka dianggap dengan *true* negatif. Jika mengidentifikasi dan mengklasifikasikan objek maka dianggap dengan *false* negatif. Gambar *true* negatif dan *false* negatif ditunjukkan pada Gambar 4.5.

Gambar 3.30 Data Gambar *True* Negatif dan *False* Negatif

Hasil dari proses *testing* data tersebut kemudian dihitung jumlah data yang termasuk dalam *true* positif, *false* postif, *true* negatif, dan *false* negatif. Ke-empat data tersebut kemudian dimasukkan dalam tabel *confussion matrix,* seperti yang ditunjukkan pada Tabel 3.1.

Tabel 3.1 Tabel *Confussion Matrix*

Data	**Positif (Hasil Klasifikasi)**	**Negatif (Hasil Klasifikasi)**
Positif (Aktual)	*True* Positif	*False* Positif
Negatif (Aktual)	*False* Negatif	*True* Negatif

Dari tabel tersebut, kemudian dapat dihitung parameter *recall*, *precission*, dan *accuracy*. Persamaan untuk menghitung nilai parameter tersebut dapat dilihat sebagai berikut:

$$\boldsymbol{Recall = \frac{TP}{TP + FN}} \qquad (\mathbf{3.4})$$

$$\boldsymbol{Precision = \frac{TP}{TP + FP}} \qquad (\mathbf{3.5})$$

$$\boldsymbol{Accuracy = \frac{TP + TN}{TP + FN + TN + FP}} \qquad (\mathbf{3.6})$$

Setelah mengetahui proses untuk melakukan pengujian aplikasi sonar, selanjutnya melakukan pengujian dengan membandingkan 3 metode. Pengujian ini dilakukan dengan masukan berupa potongan gambar *testing.* Metode pertama adalah *histogram of occupancy*, dimana metode tersebut digunakan dalam penelitian ini. Metode kedua adalah *histogram of distance* Normal. Metode ketiga adalah *histogram of distance softweight.* Perbedaan antara ketiga metode tersebut terletak pada perhitungan nilai histogram dalam proses *clustering*, pendekatan metode klasifikasi Naive Bayes, dan pengambilan keputusan dalam menentukan nilai *threshold.* Selain membandingkan metode, dilakukan pengujian dengan *background* berwarna dan wajah yang berbeda. Data tersebut menggunakan data positif berjumlah 100 dan data negatif berjumlah 100. Pengujian tersebut dilakukan hanya pada metode *histogram of occupancy.*

BAB IV
PENGUJIAN DAN ANALISIS

Bab ini berisikan pengujian sistem aplikasi sonar untuk mengidentifikasi *obstacle*. Pengujian dilakukan dengan membandingkan cara untuk menentukan anggota dari setiap *centroid*. Adapun metode yang diuji meliputi *histogram of occupancy*, *histogram of distance* normal, dan *histogram of distance softweight*. Selain itu, terdapat pengujian tambahan untuk meyakinkan aplikasi sonar mampu mengidentifikasi *obstacle* berupa wajah manusia. Pengujian tambahan tersebut dilakukan dengan mengganti gambar *testing* dengan wajah yang berbeda dan *background* warna selain putih. Hasil dari pengujian tambahan, kemudian dibandingkan dengan pengujian dengan data *testing* berupa gambar 1 wajah manusia dan *background* warna selain putih. Parameter yang digunakan untuk pengujian pada penelitian ini meliputi : *recall*, *precision*, dan *accuracy*. *Recall* merupakan tingkat keberhasilan sistem dalam menemukan kembali suatu informasi. *Precision* merupakan tingkat ketepatan sistem memberikan informasi yang diminta. *Accuracy* merupakan tingkat kedekatan nilai yang diprediksi dengan nilai aktual. Ketiga parameter tersebut akan digunakan untuk menganalisis hasil pengujian yang dilakukan.

4.1 Pengujian Metode *Histogram of Occupancy*

Histogram of occupancy menghitung anggota kelompok dari setiap *centroid* dengan mempertimbangkan jarak minimum. Setiap *centroid* dihitung jaraknya terhadap deskriptor atau *feature* objek dari data *testing*. *Centroid* yang memiliki jarak minimum dengan deskriptor, akan diberikan nilai okupansi satu. Hal tersebut dilakukan secara berulang hingga deskriptor terakhir. Jumlah okupansi dari setiap *centroid*, kemudian dinormalisasi. Hasil normalisasi digunakan untuk nilai histogram masing-masing *centroid*. Kemudian menghitung nilai peluang menggunakan Distribusi Gaussian. Nilai peluang kemudian difilter dengan membuang nilai posterior 0 dan nilai posterior yang kurang dari 0.8 x posterior maksimum.

Pengujian ini menggunakan 200 data *testing* yang dibagi menjadi 100 data positif dan 100 data negatif. Hasil dari pengujian ini dapat dilihat dalam Tabel 4.1 dan lebih jelasnya dapat dilihat pada Lampiran C-1. Untuk hasil gambar terdapat pada Lampiran B-1.

Tabel 4.1 Hasil Pengujian Metode *Histogram of Occupancy*

Jumlah Data	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
Total	93	7	84	16

Hasil pengujian menunjukkan terdapat data *true* positif sebanyak 93, *false* positif sebanyak 7, *true* negatif sebanyak 84, dan *false negatif* sebanyak 16. Hasil pengujian tersebut dimasukan kedalam tabel *confussion matrix* untuk dilakukan analisis dari nilai parameter yang ditentukan. Tabel *confussion matrix* pengujian menggunakan metode *histogram of occupancy* dapat dilihat pada Tabel 4.2.

Tabel 4.2 Tabel *Confussion Matrix Histogram of Occupancy*

Data	**Positif (Hasil diklasifikasikan)**	**Negatif (Hasil Klasifikasi)**
Positif (Aktual)	93	7
Negatif (Aktual)	16	84

Dari Tabel 4.2 dilakukan analisis untuk mengetahui keberhasilan sistem dengan parameter yang telah ditentukan. Untuk prameter *recall*, *precision*, *accuracy* memiliki perhitungan sebagai berikut :

$$Recall = \frac{93}{93 + 16} = 0{,}853\ x\ 100\% = 85.3\%$$

$$Precision = \frac{93}{93 + 7} = 0,93\ x\ 100\% = 93\%$$

$$Accuracy = \frac{177}{93 + 16 + 84 + 7} = 0{,}885\ x\ 100\% = 88.5\ \%$$

4.2 Pengujian Metode *Histogram of Distance* Normal

Pada bagian ini dilakukan pengujian dengan metode lain untuk membandingkan hasil pengujian metode yang digunakan dalam penelitian dengan metode lain. Salah satu metode lain tersebut adalah metode *histogram of distance* normal. *Histogram of distance* normal

menghitung anggota kelompok dari setiap *centroid* dengan nilai rata-rata. Setiap *centroid* dihitung jaraknya terhadap deskriptor dari data yang berupa potongan gambar *testing. Centroid* yang memiliki jarak minimum akan menyimpan nilai jarak minimum tersebut. Hal tersebut dilakukan secara berulang hingga deskriptor terkhir. Semua jarak minimum yang tersimpan di setiap *centroid* dihitung jumlah nilai rata-ratanya. Jumlah tersebut digunakan sebagai nilai histogram masing-masing *centroid.* Setelah mendapatkan *histogram of distance* selanjutnya adalah mencari nilai peluang. Dalam metode ini menggunakan pedekatan metode *error function* untuk menghitung nilai peluang. Dari nilai peluang yang didapatkan kemudian diminimalisir dengan membuang nilai posterior 0 dan kurang dari 0.3233. Hasil pengujian ini dapat dilihat pada Tabel 4.3 dan lebih jelasnya pada Lampiran C-2.

Tabel 4.3 Pengujian Metode *Histogram of Distance* Normal

Jumlah Data	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
Total	40	60	100	0

Semua hasil pengujian kemudian dimasukkan dalam tabel *cofussion matrix* yang berisikan data *true* positif, *false* positif, *true* negatif, dan *false* negatif. Dari data tersebut akan dilakukan analisis untuk menghitung nilai parameter yang ditentukan. Tabel *confusision matrix histogram of distance* normal ditunjukkan pada Tabel 4.4.

Tabel 4.4 Tabel *Confussion Matrix Histogram of Distacne* Normal

Data	**Positif (Hasil Klasifikasi)**	**Negatif (Hasil Klasifikasi)**
Positif (Aktual)	40	60
Negatif (Aktual)	0	100

Dari data Tabel 4.4 kemudian digunakan untuk menghitung parameter yaitu: *recall, precision,* dan *accuracy.* Perhitungan tersebut ditunjukkan pada persamaan berikut :

$$Recall = \frac{40}{40+0} = 1\ x\ 100\% = 100\ \%$$

$$Precision = \frac{40}{40+60} = 0{,}4\ x\ 100\% = 40\%$$

$$Accuracy = \frac{140}{40+0+100+60} = 0{,}7\ x\ 100\% = 70\%$$

4.3 Pengujian Metode *Histogram of Distance Softweight*

Histogram of distance softweight menghitung anggota kelompok dari setiap *centroid* dengan mempertimbangkan nilai jarak untuk diberikan bobot. Setiap *centroid* dihitung jaraknya dengan semua deskriptor. *Centroid* yang memiliki jarak paling minimum diberikan bobot satu. Jarak minimum kedua dan seterusnya diberikan bobot kelipatan 0.5 dari bobot jarak sebelumnya. Bobot setiap *centroid* kemudian disimpan dan digunakan untuk mengkalikan nilai jarak yang didapatkan. Hasil kali bobot dengan jarak tersebut kemudian disimpan dan digunakan untuk nilai histogram. Hasil *histogram of distance softweight* kemudian digunakan untuk menghitung nilai peluang menggunakan metode *error function.* Hasil nilai peluang kemudian diminimalisirkan dengan membuang nilai peluang 0 dan kurang dari peluang batas 0.3233. Hasil pengujian ini dapat dilihat pada Tabel 4.5 dan lebis jelasnya pada Lampiran C-3.

Tabel 4.5 Pengujian *Histogram of Distance Softweight*l

Jumlah Data	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
Total	58	42	95	5

Dari hasil pengujian tersebut kemudian dimasukkan dalam tabel *confussion matrix* yang menunjukkan data *true* positif, *false* positif, *true* negatif, dan *false* negatif. Tabel *confussion* ditunjukkan pada Tabel 4.6.

Tabel 4.6 *Confussion Matrix Histogram of Distance Softweight*

Data	**Positif (Hasil Klasifikasi)**	**Negatif (Hasil Klasifikasi)**
Positif (Aktual)	58	42
Negatif (Aktual)	5	95

Setelah mendapatkan data *true* positif, *false* positif, *true* negatif, dan *false* negatif, data tersebut digunakan untuk menghitung parameter *recall*, *precision*, dan *accuracy*. Perhitungan parameter tersebut dapat dilihat pada persamaan berikut :

$$Recall = \frac{58}{58+5} = 0.92\ x\ \ 100\% = 92\%$$

$$Precision = \frac{58}{58+42} = 0{,}58\ \ x\ 100\% = 58\%$$

$$Accuracy = \frac{153}{58+5+95+42} = 0{,}765\ \ x\ 100\% = 76.5\%$$

4.4 Pengujian dengan *Background* Warna dan Wajah Berbeda

Dalam hal ini dilakukan pengujian dengan mengganti data *testing*. Pengujian yang dilakukan dengan membandingkan metode menggunakan data *testing* gambar 1 wajah manusia dan *background* warna putih. Untuk meyakinkan metode *histogram of occupancy* pada aplikasi identifikasi *obstacle* mampu melakukan klasifikasi, maka dilakukan pengujian dengan *background* warna selain putih dan gambar wajah manusia yang berbeda. Hal tersebut dilakukan mengingat gambar *training* yang digunakan pada aplikasi ini berupa gambar wajah dengan *background* putih. Pengujian ini menggunakan gambar 4 wajah manusia, dimana masing-masing wajah memiliki 25 data. Data tersebut digunakan sebagai data positif, sedangkan data negatif didapatkan dari *background* yang berwarna selain putih. Adapun hasil dari pengujian tersebut dapat dilihat pada Tabel 4.7

untuk hasil keseluruhan dapat dilihat pada Lampiran C-4. Untuk hasil gambar pada pengujian ini dapat dilihat pada Lampiran B-3

Tabel 4.7 Pengujian Wajah Berbeda dan *Background* Warna

Jumlah Data	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
Total	89	11	66	34

Dari hasil pengujian yang ditunjukkan pada Tabel 4.7 didapatkan data *true* positif sebanyak 89, data *false* positif sebanyak 11, data *true* negatif sebanyak 66, dan data *false* negatif sebanyak 34. Hasil pengujian tersebut kemudian diolah dalam *confussion matrix* untuk mendapatkan nilai parameter *recall*, *precision*, dan *accuracy*. Dalam menghitung nilai parameter tersebut hasil pengujian dimasukkan kedalam tabel *confussion matrix* yang ditunjukkan pada Tabel 4.8.

Tabel 4.8 *Confussion Matrix* Wajah Berbeda dan *Background* Warna

Data	**Positif (Hasil Klasifikasi)**	**Negatif (Hasil Klasifikasi)**
Positif (Aktual)	89	11
Negatif (Aktual)	34	66

Dari Tabel 4.8 kemudian dilakukan perhitungan nilai *recall*, *precision*, dan *accuracy*. Perhitungan tersebut dapat dilihat dalam persamaan berikut :

$$Recall = \frac{89}{89 + 34} = 0.72\ x\ 100\% \ = 72\%$$

$$Precision = \frac{89}{89 + 11} = 0{,}89\ x\ 100\% = 89\%$$

$$Accuracy = \frac{155}{89 + 11 + 66 + 34} = 0.775\ x\ 100\% = \ 77.5\%$$

4.5 Analisis Hasil Pengujian

Pengujian yang dilakukan dengan menggunakan tiga metode menghasilkan nilai parameter *recall, precision,* dan *accuracy.* Pada bagian ini akan ditampilkan tabel data hasil pengujian ketiga metode dan grafik perbandingan nilai parameter yang didapatkan. Tabel 4.9 menunjukkan tabel data hasil pengujian ketiga metode yang berisikan nilai parameter *recall, precision,* dan *accuracy.*

Tabel 4.9 Tabel Hasil Pengujian 3 Metode

Metode	**Parameter**		
	Recall	***Precision***	***Accuracy***
Histogram of Occupancy	85,3%	93%	88.5%
Histogram of Distamce Normal	100%	40%	70%
Histogram of Distance Softweight	92%	58%	76.5%

Berdasarkan hasil pengujian tiga metode yang ditunjukkan pada Tabel 4.9, metode *histogram of occupancy* memiliki nilai *accuracy* yang lebih tinggi dibandingkan dengan kedua metode lainnya. Metode *histogram of occupancy* memiliki nilai *accuracy* yang lebih tinggi dikarenakan hasil nilai posterior dari metode *histogram of occupancy* memiliki perbedaan yang cukup jauh antara bagian objek yang sebenarnya dengan bagian yang dianggap objek. Nilai posterior yang merupakan objek sebenarnya mendekati nilai posterior maksimum sehingga ditentukan nilai batas ambang 0.8 x posterior maksimum. Jika menggunakan metode *histogram of distance* normal dan *histogram of distance softweight* nilai posterior dari objek yang sebenarnya dengan bagian yang dianggap objek sangatlah mirip. Hal tersebut mengakibatkan kesulitan dalam menentukan batas ambang untuk menentukan bagian yang sebenarnya objek.

Dalam bab ini juga dibahas mengenai hasil pengujian data gambar 1 wajah dengan *background* warna putih yang dibandingkan dengan hasil pengujian data gambar 4 wajah dengan *background* warna selain

putih. Hasil kedua pengujian didapatkan dari metode *histogram of occupancy*. Perbandingan pengujian ini dilakukan untuk mengetahui kinerja aplikasi identifikasi *obstacle* pada gambar *testing* yang berbeda. Hasil perbandingan dari pengujian dengan gambar 1 wajah dengan *background* warna putih dan gambar 4 wajah dengan *background* warna selain putih dapat dilihat pada Tabel 4.10.

Tabel 4.10 Tabel Hasil Pengujian dengan Data *Testing* Berbeda

Pengujian	**Parameter**		
	Recall	***Precision***	***Accuracy***
dengan 1 Wajah dan *Background* Warna Putih	85.3%	93%	88.5%
dengan 4 Wajah dan *Background* Warna selain Putih	72%	83%	77.5%

Dari Tabel 4.10 menunjukkan nilai *recall*, *precision*, dan *accuracy* yang didapatkan pengujian dengan 1 wajah dan *background* warna putih lebih tinggi dibandingkan dengan pengujian 4 wajah dan *background* warna selain putih. Nilai parameter tersebut dipengaruhi oleh kemampuan aplikasi untuk melakukan identifikasi secara benar. Kemampuan aplikasi ditunjukkan pada jumlah data *true* positif , *true* negatif, *false* positif, dan *false* negatif. Dengan nilai yang didapatkan seperti pada Tabel 4.1 dan Tabel 4.7 pengujian dengan 1 wajah dan *background* warna putih lebih mampu untuk mengidentifikasi *obstacle* berupa wajah. Hal tersebut dapat dibuktikan dengan nilai parameter yang didapatkan lebih baik. Pengujian dengan 1 gambar wajah dan *background* putih memiliki hasil yang lebih baik dipengaruhi oleh data *training* yang digunakan. Dalam membuat aplikasi ini dilakukan proses *training* data untuk mendapatkan *feature* sebagai ciri dari data acuan. Banyaknya *feature* yang mirip antara data gambar *training* dan gambar *testing* akan mempengaruhi nilai posterior yang didapatkan. Semakin banyak *feature* yang mirip nilai posterior akan semakin tinggi dan tingkat akurasi dalam mendeteksi *obstacle* juga semakin tinggi.

4.6 Validasi Pengujian

Setelah melakukan pengujian dengan membandingkan metode yang digunakan dengan metode lain, berikut adalah validasi pengujian sesuai target yang dituliskan:

Tabel 4.11 Validasi Pengujian

Target	Hasil
Dapat melakukan identifikasi objek berupa manusia menggunakan pengenalan wajah.	Pengujian dapat mengidentifikasi adanya manusia dengan nilai *recall* 85,3%, *precision* 93%, dan *accuracy* 88.5%.
Dapat menampilkan hasil identifikasi ke dalam GUI dengan memberikan tanda berupa kotak pembatas pada bagian yang dianggap objek.	GUI dapat menampilkan hasil identifikasi dengan menandai objek dengan kotak pembatas yang diberi keterangan manusia dan nilai posterior.
Dapat digunakan untuk mengidentifikasi *obstacle* dari data Sonar.	Aplikasi yang dibuat dapat mengidentifikasi objek dengan gambar dari kamera *webcam personal computer*. Dengan hal ini aplikasi dapat dipastikan mampu mengidentifikasi *obstacle* dari data sonar, dikarenakan data *training* dan data *testing* yang digunakan memiliki jenis yang sama yaitu gambar. Selain itu dalam proses klasifikasi selama data *training* yang digunakan memiliki jenis yang sama dengan data *testing*, proses klasifikasi dapat dilakukan.

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Pada Proyek Akhir ini telah dibuat aplikasi sonar untuk mengidentifikasi dan mengklasifikasikan *obstacle* berupa objek manusia dengan pengenalan wajah. Aplikasi ini mampu mendeteksi objek menggunakan metode *histogram of occupancy*. Aplikasi yang telah dibuat mampu melakukan identifikasi dan klasifikasi objek manusia dari kamera *webcam personal computer* secara *realtime* dengan nilai *recall* 85.3%, *precision* 93%, dan *accuracy* 88.5%. Nilai *precision* dan *accuracy* dalam aplikasi dengan menggunakan *histogram of occupancy* ini lebih tinggi dibandingkan dengan metode *histogram of distance* normal dan *histogram of distance softweight.* Hal ini karena metode yang digunakan untuk mencari nilai peluang pada *histogram of occupancy* menghasilkan nilai jauh berbeda antara objek sebenarnya dengan bukan objek yang dianggap objek. Dengan demikian dapat mempermudah proses *filter* nilai peluang.

5.2 Saran

Berdasarkan perencanaan dan pengujian yang dilakukan oleh penulis, dengan hasil aplikasi yang dibuat mampu mengidentifikasi dan mengklasifikasi objek. Dengan hal tersebut aplikasi ini dapat juga digunakan untuk mengidentifikasi dan mengklasifikasikan data sonar, hal ini dikarenakan data *training* dan data *testing* yang digunakan memiliki jenis yang sama yaitu gambar. Selain itu dalam proses klasifikasi selama data *training* yang digunakan memiliki jenis yang sama dengan data *testing*, proses klasifikasi dapat dilakukan. Pengembangan selanjutnya dari sistem ini meliputi pengidentifikasian dan pengklasifikasian lebih dari satu objek. Karena pada perancangan yang dilakukan penulis hanya mengidentifikasi dan mengklasifikasikan satu objek.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] G.Conte, S.M. Zanoli, L.Gambella. “Accoustic and Localization of an ROV,”. 2006.

[2] W. W. Wijonarko, B. Sasmito, and A. L. Nugraha, “Kajian Pemodelan Dasar Laut Menggunakan Side Scan Sonar dan Singlebeam Echosounder,” .2016.

[3] -, “Sonar”, [online]. Available: https://nanopdf.com/. 2018.

[4] I. Setiawan, “Simulasi Model Sensor Sonar Untuk keperluan sistem navigasi robot mobile,” .2006.

[5] D.A. Khrisna, A. Hidayatno, R.R. Isnanto. “Identifikasi Objek Berdasarkan Bentuk dan Ukuran,”. 2016.

[6] S. L. Siregar and S. Ariswoyo, “Pengambilan keputusan menggunakan metode bayes pada ekspetasi fungsi utilitas” .2014.

[7] -, “Kelas Kilo Kapal Selam”, [online] . Available : https://www.jejaktapak.com/2018/03/20/kelas-kilo-kapal-selam-paling-sukses-dari-negeri-beruang / [Acessed 11 Maret 2019]

[8] Nugroho, Wibowo Harso. “Perancangan Kapal Selam Berasarkan Kajian Berat, Daya Apung dan Stabiliytas Statisnya,”. 2007.

[9] Setiawan, Iwan. “Simulasi Model Sensor Sonar Untuk Keperluan Sistem Navigasi Robot Mobile,” . 2006.

[10] Brennan, C. W. “Basic Acoustic Theory,” 1-10. R2 Sonic LLC. 2009.

[11] -, “Forward Looking Sonar”, [online] . Available : https://dosits.org/galleries/technology-gallery/locating-objects-using-sonar/forward-looking-sonar/ [Acessed 20 Maret 2019]

[12] Norbit. “Wideband Multibeam Sonar.”, [online]. Available: https://www.seabed.nl/products/sonar-bathymetry/forward-looking-sonar/. 2014.

[13] -, “Ethernet dan IEEE 802.3 LAN Standard”.no Llc:1-14.

[14] Neno, Mikael, A Sumardin, vAri Primanedi. “Ethernet,” 2011.

[15] Lionardo, Michael. “Memahami Konsep OOP dengan C++,”.2003.

[16] Wachid, Turahe Nur. “Koding C ++ Dengan Qt.”. 2018.

[17] -, “QWidget Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qwidget.html#details [Acessed 23 Maret 2019]

[18] -, “QString Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qstring.html#details [Acessed 24 Maret 2019]
[19] -, “QVector Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qvectort.html#details [Acessed 23 Maret 2019]
[20] -, “QLabel Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qlabel t.html#details [Acessed 24 Maret 2019]
[21] -, “QGridLayout Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qgridlayout.html#details [Acessed 24 Maret 2019]
[22] -, “QPushButtonClass Reference”, [online] . Available : https://doc.qt.io/qt-5/qpushbutton.html#details [Acessed 24 Maret 2019]
[23] -, “QThread Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qthread.html#details [Acessed 24 Maret 2019]
[24] -, “QTimer Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qtimer.html#details [Acessed 24 Maret 2019]
[25] -, “QMutex Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qmutex.html#details [Acessed 24 Maret 2019]
[26] -, “QImage Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qimage.html#details [Acessed 24 Maret 2019]
[27] -, “QWaitCondition Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qWaitCondition.html#details [Acessed 24 Maret 2019]
[28] -, “QByteArray Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qbytearray.html#details [Acessed 24 Maret 2019]
[29] -, “QBitArray Class Reference”, [online] . Available : https://doc.qt.io/qt-5/qbitarray.html#details [Acessed 24 Maret 2019]
[30] -, “QTime Reference”, [online] . Available : https://doc.qt.io/qt-5/qtimehtml#details [Acessed 24 Maret 2019]

[31] -, “OpenCV Documentation”, [online] . Available : https://docs.OpenCV.org/3.2.0/d1/dfb/intro.html [Acessed 25 Maret 2019]
[32] -, “Journal UMM”, [online] . Available : http://eprints.umm.ac.id/34169/2/jiptummpp-gdl-rendradwiw-43950-2-bab1.pdf [Acessed 25 Maret 2019]
[33] -, “OpenCV Documentation Core”, [online] . Available : https://docs.OpenCV.org/2.4/modules/core/doc/intro.html [Acessed 25 Maret 2019]
[34] -, “Module ImgCodecs”, [online] . Available : https://docs.OpenCV.org/3.0-beta/modules/imgcodecs/doc/imgcodecs.html [Acessed 25 Maret 2019]
[35] -, “Object detection”, [online] . Available : http://sinta.ukdw.ac.id/sinta/resources/sintasrv/getintro/22105006/0882d9173250b9b53bec692b03fb72ab/intro.pdf [Acessed 27 Maret 2019]
[36] -, “Segmentasi pada Citra”, [online] . Available : https://www.academia.edu/7279845/Segmentasi_pada_citra [Acessed 27 Maret 2019]
[37] -, “Selective Search”, [online] . Available : https://www.learnOpenCV.com/selective-search-for-object-detection-cpp-python/ [Acessed 27 Maret 2019]
[38] Rinaldi, Munir. n.d. “Pengenalan Pola,”.
[39] -, “Feature Extraction”, [online] . Available : https://slideplayer.info/slide/3666429/ [Acessed 27 Maret 2019]
[40] Mirza, Muhammad, Tjokorda Agung Budi W, Sa’adah Siti. “Analisis Dan Implementasi Contet Based Image Retrieval Menggunakan Metode ORB” . 2015.
[41] -, “K-Means Clustering”, [online] . Available : https://www.datascience.com/blog/K-Means-clustering[Acessed 28 Maret 2019]
[42] -, “Understanding K-Means Clustering”, [online] . Available : https://towardsdatascience.com/understanding-K-Means-clustering-in-machine-learning-6a6e67336aa1 [Acessed 28 Maret 2019]
[43] Manik, Fuzy Yustika, Kana Saputra Saragih. “Klasifikasi Belimbing Menggunakan Naïve Bayes Berdasarkan Fitur Warna RGB” . 2017.1

[44] Handayani, Fitri, and Feddy Setio Pribadi. “Implementasi Algoritma Naive Bayes Classifier Dalam Pengklasifikasian Teks Otomatis Pengaduan Dan Pelaporan Masyarakat Melalui Layanan Call Center 110” .2015.
[45] Taheri, Sona, and Musa Mammadov. “Learning the Naive Bayes Classifier with Optimization Models,”. 2015.
[46] -, “Naive Bayes”, [online] . Available : https://repository.widyatama.ac.id/xmlui/bitstream/handle/123456789/3523/Bab%202.pdf?sequence=4 [Acessed 28 Maret 2019]
[47] Buil, Mikel Diez. “Non-Maxima Supression”.2011.
[48] Neubeck, Alexander, and Luc Van Gool. “Efficient Non-Maximum Suppression,” . 2016.
[49] Wang, Derui, Chaoran Li, Sheng Wen, and Yang Xiang. n.d. “Daedalus : Breaking Non-Maximum Suppression in Object Detection via Adversarial Examples.”
[50] MJ Lyons, M. Kamachi dan J. Gyoba, "Ekspresi Wajah Wanita Jepang (JAFFE)," Database gambar digital. 1997.

LAMPIRAN

A-1 Program *Headers Akses* Kamera *Webcam Personal Computer*

```
#ifndef ACCESSVIDEO_H
#define ACCESSVIDEO_H
#include <opencv2/core/core.hpp>
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/video.hpp>
#include <opencv2/videoio.hpp>

#include <QThread>
#include <QTimer>
#include <QWaitCondition>
#include <QMutex>
#include <QImage>


using namespace cv;
using namespace std;
class AccessVideo : public QThread
{
    Q_OBJECT
public:
    explicit AccessVideo(QObject *parent =
nullptr);
    ~AccessVideo();
    void start_();
    void stop_();
    void AccesCamera();

private :

    VideoCapture capWebcame;
    Mat pictureFrame;
    QWaitCondition waitThread;
    bool stop;
    QMutex mutex;
    int frameRate;
```

```
int delay;


signals :
    void frameCapture(QImage);

public slots :

protected:
        void run();
        void msleep(int ms);

};

#endif // ACCESSVIDEO_H
```

A-2 Program *Headers* Akses Kamera dan Klasifikasi

```
#ifndef CLASSIFYTHREAD_H
#define CLASSIFYTHREAD_H
#include <opencv2/video.hpp>
#include <opencv2/videoio.hpp>
#include "nbayesorbclassify.h"
#include "nbayesclassorb.h"

#include <QObject>
#include <QImage>

using namespace cv;
using namespace std;
class ClassifyThread : public QObject
{
    Q_OBJECT
public:
    ClassifyThread(QObject *parent = 0);
    ~ClassifyThread();
    void start_();
    void stop_();

public slots:
    void start();


signals :
    void getCame(QImage, QString);
    void finished();

private :
    VideoCapture capImage;
    VideoWriter videoResult;
    Mat matFrame;
    QImage currentFrame;
    NBayesOrbClassify classifier;
    NBayesClassORB Square_Class;
    NBayesClassORB Star_Class;
    NBayesClassORB Angry_Class;
```

```
    NBayesClassORB Neutral_Class;
    NBayesClassORB Human_Class;
    NBayesClassORB Cup_Class;
    NBayesClassORB Finger_Class;
    bool play;
    bool stop;
    int counter = 0;
    bool indexStop;


};

#endif // CLASSIFYTHREAD_H
```

A-3 Program *Headers Design GUI*

```
#ifndef DESIGNGUI_H
#define DESIGNGUI_H

#include <QWidget>
#include "accessvideo.h"
#include "classifythread.h"

#include <QLabel>
#include <QGridLayout>
#include <QPushButton>
#include <QThread>

using namespace std;
using namespace cv;

class DesignGUI : public QWidget
{
    Q_OBJECT

public:
    DesignGUI(QWidget *parent = 0);
    ~DesignGUI();

    QPushButton *button_1;
    QPushButton *button_2;
    QPushButton *button_3;
    QGridLayout *layout;
    QLabel *imageTarget;
    QLabel *imageResult;
    QLabel *title;
    QLabel *result;
    QLabel *showTheResult;
    QLabel *titleTarget;
    QLabel *titleResult;

public slots :
    void thread(QImage picture);
    void play();
```

```
void stop();
    void classify();
    void getClassify(QImage picture, QString
time);

private :
    AccessVideo* threadCam;
    QThread *thread_;
    ClassifyThread *objectClassify;


    Mat imageTest;
    Mat pictureObject;
    bool flag = 0;
    bool stop_;

};

#endif // DESIGNGUI_H
```

A-4 Program *Headers Training Data*

```
#ifndef NBAYESCLASSORB_H
#define NBAYESCLASSORB_H
#include<opencv2/features2d.hpp>
#include<opencv2/imgcodecs.hpp>
#include<opencv2/highgui.hpp>
#include<opencv2/imgproc.hpp>

#include<QVector>
#include<QDebug>
#include<QByteArray>
#include<QBitArray>

using namespace std;
using namespace cv;
class NBayesClassORB
{
public:

    NBayesClassORB(QString NameClass);
    ~NBayesClassORB();
    NBayesClassORB();
    void GetDescriptor(Mat Image);
    void GetMeansAndDeviasi(int index);
    Mat Centers();
    QString NameClass();

QVector<Mat>CollectionOfDescritorTraining_();

private :
    QString NameClass_;
    QVector<Mat>CollectionOfDescritorTraining;
    Mat centers;
    float means[100];
    float Deviasi[100];
    float ProbabilityLikelihood;
};

#endif // NBAYESCLASSORB_H
```

A-5 Program *Headers Testing Data*

```
#ifndef NBAYESORBCLASSIFY_H
#define NBAYESORBCLASSIFY_H
#include"nbayesclassorb.h"
#include "opencv2/ximgproc/segmentation.hpp"
#include <opencv2/dnn.hpp>
#include <QImage>
#include <QObject>

using namespace cv::ximgproc::segmentation;
using namespace dnn;
using namespace dnn::experimental_dnn_34_v11;
class NBayesOrbClassify
{
public:
    NBayesOrbClassify();
    ~NBayesOrbClassify();
    void GetClassification(Mat testImage);
    void addClass(NBayesClassORB Class);
    void GetCentroid();
    void GetHistogramTesting();
    void GetHistogramTraining(int indexclass);
    void GetHistogramTestingofAccupancy();
    void GetHistogramTrainingofAccupancy(int
indexclass);
    void GetHistogramTestingofDistance();
    void GetHistogramTrainingofDistance(int
indexclass);
    void GetMeansandDeviasi();
    void GetMeansDeviasiTraining(int
indexclass);
    float GetEvidence(int index, float
ValueOfIndex);
    float GetPrior();
    float GetPosterior(int indexclass);
    float GetLikelihood(int indexclass, int
index,float ValueOfIndex);
    float GetLikelihoodDistGaus(int
indexclass,
```

```
int index, float ValueOfIndex);
    QString ResultOfImage_();
    QVector<NBayesClassORB>MemberofClass_();
    QImage pictDetect_();
    Mat objectDetect_();
    Mat objectClear_();
    void getNMS(QVector<int> indexRect, float
overThres, vector<int> &indexOut);
    QString timeCount_();
    QVector<Mat>tempClearPict_();

private:
   QVector<Mat>CollectionOfTestImage;
   QVector<NBayesClassORB>MemberofClass;
   QVector<QVector<float>>ArrayHistogramTesting
   ;
   QVector<QVector<float>>trainingHistogram;
   QVector<QVector<float>>CollectionTrainingHis
   togram;
   QVector<QVector<float>>MeansAllClass;
   QVector<QVector<float>>DeviasiAllClass;
   Mat CollectionDataTrainingAllClass;
   Mat CentersClass;
   Mat DescriptorTestingFloat;
   float SumLikelihood;
   QVector<float>Likelihood;
   QVector<float>posteriorMaxAll;
   QVector<int>currentIndexALl;
   QVector<int>indexRectAll;
   QVector<int>indexPictClear;
   float SumEvidence;
   float sumMeans[10000];
   float sumDeviasi[10000];
   float Deviasi[1000];
   float ProbabilityEvidence;
   float ProbabilityPrior;
   float ProbabilityPosteior;
   float ProbabilityLikelihood;
   QString ResultOfImage;
   Mat objectDetect;
```

```
    Mat objectClear;

     QImage pictDetect;
     vector<Rect> rects;
     QVector<float>posteriorSet;
     QVector<int>setIndexClass;
     QVector<int>setIndexRect;
     QVector<float> setPosterior;
     QVector<int> tempCurrentIndex;
     QVector<int> tempIndexRect;
     QVector<int>indexFinish;
     vector<Rect>outputAll;
     vector<Rect> tempScaleRect;
     int counter = 1;
     int counterImage =0;
     QString timeCount;
     QVector<Mat>tempClearPict;



};

#endif // NBAYESORBCLASSIFY_H
```

A-6 Program Cpp Akses Kamera *Webcam Personal Computer*

```
#include "accessvideo.h"
#include <QTime>

AccessVideo::AccessVideo(QObject *parent)
    :QThread(parent)

{

    capWebcame.open(1);

}

AccessVideo::~AccessVideo()
{

    mutex.lock();
    stop = true;
    capWebcame.release();
    waitThread.wakeOne();
    mutex.unlock();
    wait();
}


void AccessVideo::run()
{
    frameRate = 25;
    delay = 1000/frameRate;

    while(stop == false)
    {

        if(capWebcame.isOpened())
        {
            capWebcame.read(pictureFrame);
            Mat pictureRGB;
            cvtColor(pictureFrame, pictureRGB,
CV_BGR2RGB);
```

```
QImage imageFrame =
QImage((uchar*)pictureRGB.data,
pictureRGB.cols, pictureRGB.rows,
pictureRGB.step, QImage::Format_RGB888);
            emit frameCapture(imageFrame);
            this->msleep(delay);
        }
    }

}

void AccessVideo::msleep(int ms)
{
    struct timespec ts = {(ms/1000), (ms %
1000) * 1000 * 1000};
    nanosleep(&ts, NULL);
}

void AccessVideo::start_()
{
    if(!isRunning())
    {
        stop = false;
        start(LowPriority);
    }

}

void AccessVideo::stop_()
{
    stop = true;
}
```

A-7 Program Cpp Akses Kamera dan Klasifikasi

```
#include "classifythread.h"
ClassifyThread::ClassifyThread(QObject *parent)
    : QObject(parent)
    , Human_Class("Human")
    , classifier()
{
}

ClassifyThread::~ClassifyThread()
{
    stop = true;

}

void ClassifyThread::start_()
{
    stop = false;
    play = true;
}

void ClassifyThread::stop_()
{
    stop = true;
}

void ClassifyThread::start()
{
    QString file_path4 = "E:/TUGAS AKHIR
SEMANGATTT/Tugas
Akhir/faces/expression/angryTraining/";
    QString ekstensi2 = ".tiff";
    for (int i = 1; i<30; i++)
    {
        QString full_path1 = file_path4
+QString::number(i) + ekstensi2;
        Mat Image1 = imread
(full_path1.toStdString());
Human_Class.GetDescriptor(Image1);
```

```
    }
    classifier.addClass(Human_Class);

    if (capImage.open(1))
    {
        qDebug() << "Camera device is open.";
    }
    else
    {
        qDebug() << "Failed to open camera
device.";
    }

    while(capImage.isOpened() && stop == false)
    {
        if (capImage.read(matFrame))
        {

classifier.GetClassification(matFrame);
            counter++;
            Mat pictureRGB;
            Mat pictureRGB_flipped;
            flip(pictureRGB, pictureRGB_flipped,
1);
            cvtColor(classifier.objectDetect_(),
pictureRGB_flipped, CV_BGR2RGB);
            QImage
imageFrame(pictureRGB_flipped.data,
pictureRGB_flipped.cols,
pictureRGB_flipped.rows,
pictureRGB_flipped.step, QImage::Format_RGB888);
            currentFrame = imageFrame.copy(0, 0,
imageFrame.width(), imageFrame.height());
            QString a = QString::number(counter)
+ ".jpg";
            currentFrame.save(a);
QString time = classifier.timeCount_();
            emit getCame(currentFrame,time);
        }
```

```
        else
        {
            qDebug() << "Camera device read
empty frame.";
        }

    }
    emit finished();

}
```

A-8 Program Cpp *Design GUI*

```
#include "designgui.h"

#include <QFileDialog>
#include <QMessageBox>
#include <QThread>
#include "classifythread.h"

DesignGUI::DesignGUI(QWidget *parent)
    :QWidget(parent)

{
    threadCam = new AccessVideo();
    objectClassify = new ClassifyThread;
    thread_ = new QThread;

    layout = new QGridLayout(this);
    title =  new QLabel(this);
    titleTarget = new QLabel(this);
    imageTarget = new QLabel(this);
    result = new QLabel(this);
    titleResult = new QLabel(this);
    imageResult = new QLabel (this);
    button_1 = new QPushButton(this);
    button_3 = new QPushButton(this);
    button_2 = new QPushButton(this);
    showTheResult = new QLabel(this);

    layout->setRowStretch(0, 2);
    layout->setRowStretch(1, 2);
    layout->setRowStretch(2, 2);
    layout->setRowStretch(3, 2);
    layout->setRowStretch(4, 2);
    layout->setRowStretch(5, 2);


    title->setText("Obstacle Identification");
    title->setStyleSheet("QLabel {background-
color : gray; color : black}");
```

```
    title->setFont(QFont("lucida", 30,
QFont::Bold, true));
    title->setAlignment(Qt::AlignHCenter |
Qt::AlignTop);
    title-
>setSizePolicy(QSizePolicy::Expanding,QSizePolic
y::Fixed );
    layout->addWidget(title, 0, 0, 1, 3);
    titleTarget->setText("Video Target");
    titleTarget->setStyleSheet("QLabel
{background-color : silver; color black}");
    titleTarget->setFont(QFont("lucida", 18,
QFont::Bold, true));
    titleTarget->setAlignment(Qt::AlignHCenter |
Qt::AlignTop);
    titleTarget-
>setSizePolicy(QSizePolicy::Expanding,QSizePolic
y::Fixed);
    layout->addWidget(titleTarget,1,0,1,1);

    imageTarget->setStyleSheet("QLabel
{background-color : white}");
    imageTarget->setAlignment(Qt::AlignVCenter);
    imageTarget-
>setSizePolicy(QSizePolicy::Fixed,QSizePolicy::E
xpanding );
    imageTarget->setAlignment(Qt::AlignCenter);
    imageTarget->setFixedSize(300,300);
    layout->addWidget(imageTarget, 2, 0, 2, 1,
Qt::AlignCenter);

    result->setText("Time of Processing :");
    result->setStyleSheet("QLabel {color :
silver}");
    result->setFont(QFont("lucida", 18,
QFont::Bold,true));
    result->setAlignment(Qt::AlignTop);

result-
>setSizePolicy(QSizePolicy::Fixed,QSizePolicy::F
```

```
ixed );
    layout->addWidget(result, 4, 0, 1, 2,
Qt::AlignCenter);

    titleResult->setText("Video Result");
    titleResult->setStyleSheet("QLabel
{background-color : silver; color black}");
    titleResult->setFont(QFont("lucida", 18,
QFont::Bold, true));
    titleResult->setAlignment(Qt::AlignHCenter |
Qt::AlignTop);
    titleResult-
>setSizePolicy(QSizePolicy::Expanding,QSizePolic
y::Fixed);
    layout->addWidget(titleResult,1,1,1,1);

    imageResult->setStyleSheet("QLabel
{background-color : white}");
    imageResult->setAlignment(Qt::AlignVCenter);
    imageResult-
>setSizePolicy(QSizePolicy::Fixed,QSizePolicy::E
xpanding );
    imageResult->setAlignment(Qt::AlignCenter);
    imageResult->setFixedSize(300,300);
    layout->addWidget(imageResult, 2, 1, 2, 1,
Qt::AlignCenter );


    button_1->setText("Play");
    button_1->setStyleSheet("QPushButton
{background-color : silver}");
    button_1->setFont(QFont("lucida", 15));
    button_1-
>setSizePolicy(QSizePolicy::Fixed,QSizePolicy::F
ixed );
    button_1->setFixedSize(150,50);
    layout->addWidget(button_1, 1, 2, 1, 1,

Qt::AlignCenter);
```

```
    button_3->setText("Classify");
    button_3->setStyleSheet("QPushButton
{background-color : silver}");
    button_3->setFont(QFont("lucida", 15));
    button_3-
>setSizePolicy(QSizePolicy::Fixed,QSizePolicy::F
ixed );
    button_3->setFixedSize(150,50);
    layout->addWidget(button_3, 2, 2, 1, 1,
Qt::AlignCenter);

    button_2->setText("Stop");
    button_2->setStyleSheet("QPushButton
{background-color : silver}");
    button_2->setFont(QFont("lucida", 15));
    button_2-
>setSizePolicy(QSizePolicy::Fixed,QSizePolicy::F
ixed );
    button_2->setFixedSize(150,50);
    layout->addWidget(button_2, 3, 2, 1, 1,
Qt::AlignCenter);

    showTheResult->setFont(QFont("luicida", 18,
QFont::Bold,true));
    showTheResult->setStyleSheet("QLabel
{background-color : white}");
    showTheResult-
>setSizePolicy(QSizePolicy::Fixed,QSizePolicy::F
ixed );
    showTheResult->setFixedSize(300,50);
    layout->addWidget(showTheResult, 4, 2, 1,
1);

    setLayout(layout);

    imageTarget->clear();



QObject::connect(threadCam,
```

```
SIGNAL(frameCapture(QImage)), this,
SLOT(thread(QImage)));

    objectClassify->moveToThread(thread_);
    connect(thread_, SIGNAL(started()),
objectClassify, SLOT(start()));
    connect(objectClassify,
SIGNAL(getCame(QImage,QString)), this,
SLOT(getClassify(QImage,QString)));


    connect(button_1, SIGNAL(clicked(bool)),
this, SLOT(play()));
    connect(button_3, SIGNAL(clicked(bool)),
this, SLOT(classify()));
    connect(button_2, SIGNAL(clicked(bool)),
this, SLOT(stop()));

}

DesignGUI::~DesignGUI()
{
   thread_->exit();
}

void DesignGUI::thread(QImage picture)
{

    imageTarget-
>setPixmap(QPixmap::fromImage(picture));
    imageTarget->setScaledContents(true);
    imageTarget->update();
    stop_ = false;
}
void DesignGUI::play()
{
```

```
stop_ = false;
    threadCam->start_();
}

void DesignGUI::stop()
{
    stop_ = true;
    threadCam->stop_();
    objectClassify->stop_();
    connect(objectClassify, SIGNAL(finished()),
thread_, SLOT(quit()));
    connect(objectClassify, SIGNAL(finished()),
objectClassify, SLOT(deleteLater()));
    connect(objectClassify, SIGNAL(finished()),
thread_, SLOT(deleteLater()));
    thread_->exit();

}

void DesignGUI::classify()
{
    stop_ = false;
    thread_->start();
    objectClassify->start_();

}

void DesignGUI::getClassify(QImage picture,
QString time)
{
    imageResult-
>setPixmap(QPixmap::fromImage(picture));
    imageResult->setScaledContents(true);
    imageResult->update();
    showTheResult->setText(time);
    showTheResult->update();
    stop_ = false;

}
```

A-9 Program *Training Data*

```
#include "nbayesclassorb.h"

NBayesClassORB::NBayesClassORB()
{

}
NBayesClassORB::NBayesClassORB(QString
NameClass)
{
   NameClass_ = NameClass;
}

NBayesClassORB::~NBayesClassORB()
{

}
void NBayesClassORB::GetDescriptor(Mat Image)
{
    Mat GrayImageTraining;
    vector<KeyPoint> KeypointTraining;
    Ptr<ORB> DetectORB = ORB::create();
    Mat DescriptorTraining;
    Mat DescriptorTrainingFloat;

cvtColor(Image,GrayImageTraining,CV_RGB2GRAY);
    DetectORB-
>detect(GrayImageTraining,KeypointTraining);
    DetectORB-
>compute(GrayImageTraining,KeypointTraining,De
scriptorTraining);

DescriptorTraining.convertTo(DescriptorTrainin
gFloat,CV_32FC1);

CollectionOfDescritorTraining.append(Descripto
rTrainingFloat);

}
```

```
QString NBayesClassORB::NameClass()
{
    return NameClass_;
}


QVector<Mat>
NBayesClassORB::CollectionOfDescritorTraining_
()
{
    return CollectionOfDescritorTraining;
}
```

A-10 Program *Testing Data*

```
#include "nbayesorbclassify.h"
#include <QTime>
#include <QTimer>


NBayesOrbClassify::NBayesOrbClassify()
{

}


NBayesOrbClassify::~NBayesOrbClassify()
{

}

void NBayesOrbClassify::addClass(NBayesClassORB
Class)
{

    MemberofClass.append(Class);


    qDebug()<< "before : "
<<CollectionDataTrainingAllClass.rows << " , "
<<CollectionDataTrainingAllClass.cols;
    Mat container1;

vconcat(Class.CollectionOfDescritorTraining_()[
0], container1);
    for (int j=1; j <
Class.CollectionOfDescritorTraining_().size();
j++)
    {


vconcat(Class.CollectionOfDescritorTraining_()[
j], container1, container1);
```

```
    }
    if (CollectionDataTrainingAllClass.cols !=
container1.cols)
    {

container1.copyTo(CollectionDataTrainingAllClas
s);
    }
    else
    {
        vconcat(container1,
CollectionDataTrainingAllClass,
CollectionDataTrainingAllClass);
    }

    qDebug()<< "after : "
<<CollectionDataTrainingAllClass.rows << " , "
<<CollectionDataTrainingAllClass.cols;


    GetCentroid();
    MeansAllClass.clear();
    DeviasiAllClass.clear();
    for (int a=0; a < MemberofClass.size();
a++)
    {
        GetHistogramTrainingofAccupancy(a);
        GetMeansDeviasiTraining(a);
    }

}

void NBayesOrbClassify::GetClassification(Mat
testImage)
{

    QTime time;


    time.start();
```

```
    Mat resizeTestImage;
    resize(testImage, resizeTestImage,
cvSize(256,256));
    rects.clear();
    tempScaleRect.clear();
    Ptr<SelectiveSearchSegmentation>
selectiveSearch =
createSelectiveSearchSegmentation();
    selectiveSearch-
>setBaseImage(resizeTestImage);
    selectiveSearch->switchToSingleStrategy();
    selectiveSearch->process(rects);

    int numShowRect = rects.size();

    float factorW = (float)testImage.cols /
(float)resizeTestImage.cols;
    float factorH = (float)testImage.rows /
(float)resizeTestImage.rows;

    for (int i = 0 ; i < rects.size(); i++)
    {
        float x = rects[i].x * factorW;
        float y = rects[i].y * factorH;
        float w = rects[i].width * factorW;
        float h = rects[i].height * factorH;

        Rect scaleRect(x, y, w, h);
        tempScaleRect.push_back(scaleRect);

    }
    QVector<Mat>sumCrop;
    Mat imOut = testImage.clone();
    sumCrop.clear();
    for(int i = 0; i < tempScaleRect.size();
i++)
    {
        if(tempScaleRect[i].empty())

{
```

```
        }
        else
        {
            if (i < numShowRect)
            {
              Mat crop =
imOut(tempScaleRect[i]);
              sumCrop.append(crop);
            }
            else
            {

            }
        }


    }

    float posteriorMax_= 0;
    float currentIndex_ = -1;
    float IndexCrop = -1;
    posteriorMaxAll.clear();
    currentIndexALl.clear();
    indexRectAll.clear();
    indexPictClear.clear();

    for (int i = 0; i < sumCrop.size(); i++)
    {
        if (sumCrop[i].rows < 20 ||
sumCrop[i].cols < 20)
        {

        }
        else
        {
            Mat GrayTestImage;
            vector<KeyPoint> KeypointTesting;
```

```
            Ptr<ORB> DetectORBTesting =
ORB::create();
            Mat DescriptorTesting;

cvtColor(sumCrop[i],GrayTestImage,CV_RGB2GRAY);
            if (GrayTestImage.rows < 20 &&
KeypointTesting.empty() || GrayTestImage.cols <
20)
            {

            }
            else
            {

                DetectORBTesting-
>detect(GrayTestImage,KeypointTesting);
                DetectORBTesting-
>compute(GrayTestImage,KeypointTesting,Descript
orTesting);


DescriptorTesting.convertTo(DescriptorTestingFl
oat,CV_32FC1);

GetHistogramTestingofAccupancy();
GetMeansandDeviasi();
float PosteriorMax = 0;
int currentIndex = -1;
int indexRect = -1;
float Prior[100];
float Posterior[100];
Likelihood.clear();

for( int e = 0; e < MemberofClass.size(); e++)
{SumLikelihood = 1;
for(int b = 0; b < CentersClass.rows; b++)
{SumLikelihood = SumLikelihood *
GetLikelihoodDistGaus(e,b,ArrayHistogramTesting

[e][b]);
```

```
                    }

Likelihood.append(SumLikelihood);

                    Prior[e]=GetPrior();
                    Posterior[e]=
GetPosterior(e);

                    if (PosteriorMax <
Posterior[e])
                    {
                        PosteriorMax =
Posterior [e];
                        currentIndex = e;
                    }

                }
                indexRect = i;
                if (PosteriorMax != 0)
                {

posteriorMaxAll.append(PosteriorMax);

currentIndexALl.append(currentIndex);

indexRectAll.append(indexRect);
                }
                else
                {

indexPictClear.append(indexRect);
                }

            }

        }

    }
```

```
    float temPosteriorMax = 0;
    posteriorSet.clear();
    setIndexClass.clear();
    setIndexRect.clear();
    for(int i = 0; i <posteriorMaxAll.size();
i++)
    {
        if(temPosteriorMax <
posteriorMaxAll[i])
        {
           temPosteriorMax =
posteriorMaxAll[i];
        }
    }

    for (int i = 0; i < posteriorMaxAll.size();
i++)
    {
        if (posteriorMaxAll[i] > 0.8 *
temPosteriorMax )
        {

posteriorSet.append(posteriorMaxAll[i]);

setIndexClass.append(currentIndexALl[i]);

setIndexRect.append(indexRectAll[i]);
        }
        else
        {

indexPictClear.append(indexRectAll[i]);
        }
    }


    vector<int> indexOutput;
    if (setIndexRect.size() == 0)

    {
```

```
    }
    else
    {
      getNMS(setIndexRect,0.4,indexOutput);
    }

    objectDetect = testImage.clone();

    int baseLine = 0;
    counterImage++;

    for (int i=0; i <indexOutput.size(); i++)
    {


rectangle(objectDetect,tempScaleRect[setIndexRe
ct[indexOutput[i]]], Scalar(0,0,255));

        int x =
tempScaleRect[setIndexRect[indexOutput[i]]].x;
        int y =
tempScaleRect[setIndexRect[indexOutput[i]]].y;
        QString posteriorScore =
QString::number(posteriorSet[indexOutput[i]]);
        Size labelposterior =
getTextSize(posteriorScore.toStdString(),
FONT_HERSHEY_SIMPLEX,3,4,&baseLine);
        ResultOfImage =
MemberofClass[setIndexClass[indexOutput[i]]].Na
meClass();
        Size labelSize =
getTextSize(ResultOfImage.toStdString(),
FONT_HERSHEY_SIMPLEX,2,3,&baseLine);
        putText(objectDetect,
ResultOfImage.toStdString(), Point(x, y +
labelSize.height), FONT_HERSHEY_SIMPLEX, 1,
Scalar (0,255,0),1, LINE_AA);
```

```
        putText(objectDetect,
posteriorScore.toStdString(), Point(x, y +
labelposterior.height), FONT_HERSHEY_SIMPLEX,
1, Scalar (255,0,0),1,LINE_8);
    }

    counter++;
    time.elapsed();
    timeCount =
QString::number(time.elapsed());

}


void
NBayesOrbClassify::GetHistogramTraining(int
indexclass)
{
    trainingHistogram.clear();
    for (int j = 0; j <
MemberofClass[indexclass].CollectionOfDescritor
Training_().size(); j++)
    {

QVector<float>currentHistogram(CentersClass.row
s,0);

QVector<double>tempSumMinimumDistance(CentersCl
ass.rows,0);

QVector<float>averageMinimumDistance(CentersCla
ss.rows,0);

QVector<float>normalisasiMinimumDistance(Center
sClass.rows,0);
        for(int k = 0 ; k <
MemberofClass[indexclass].CollectionOfDescritor
Training_()[j].rows; k++)

        {
```

```
            int index = 0;
            double MinimumDistance =
norm(MemberofClass[indexclass].CollectionOfDesc
ritorTraining_()[j].row(k),
CentersClass.row(0),NORM_L2);


            for(int l = 1; l <
CentersClass.rows; l++ )
            {
                double CurrentDistance =
norm(MemberofClass[indexclass].CollectionOfDesc
ritorTraining_()[j].row(k),
CentersClass.row(l),NORM_L2);
                if(MinimumDistance >
CurrentDistance)
                {
                    MinimumDistance =
CurrentDistance;
                    index = l;
                }

            }

            tempSumMinimumDistance[index] +=
MinimumDistance;
            currentHistogram[index]++;
            averageMinimumDistance[index] =
tempSumMinimumDistance[index]/currentHistogram[
index];

        }

        float totalAverageMinimumDistance = 0;
        for (int i = 0; i<CentersClass.rows;
i++)
        {
            totalAverageMinimumDistance +=

averageMinimumDistance[i];
```

```
        }
        for (int j = 0; j < CentersClass.rows;
j++)
        {

normalisasiMinimumDistance[j]=averageMinimumDis
tance[j]/totalAverageMinimumDistance;
        }


trainingHistogram.append(normalisasiMinimumDist
ance);

    }


    for(int p=0;
p<trainingHistogram.size();p++)
    {

CollectionTrainingHistogram.append(trainingHist
ogram[p]);
    }

}



void
NBayesOrbClassify::GetHistogramTrainingofAccupa
ncy(int indexclass)
{
    trainingHistogram.clear();
    for (int j = 0; j <
MemberofClass[indexclass].CollectionOfDescritor
Training_().size(); j++)
    {


QVector<float>currentHistogram(CentersClass.row
```

```
s,0);

QVector<float>NormalisasiCurrentHistogram(Cente
rsClass.rows,0);
        for(int k = 0 ; k <
MemberofClass[indexclass].CollectionOfDescritor
Training_()[j].rows; k++)
        {
            int index = 0;
            double MinimumDistance =
norm(MemberofClass[indexclass].CollectionOfDesc
ritorTraining_()[j].row(k),
CentersClass.row(0),NORM_L2);


            for(int l = 1; l <
CentersClass.rows; l++ )
            {
                double CurrentDistance =
norm(MemberofClass[indexclass].CollectionOfDesc
ritorTraining_()[j].row(k),
CentersClass.row(l),NORM_L2);
                if(MinimumDistance >
CurrentDistance)
                {
                    MinimumDistance =
CurrentDistance;
                    index = l;
                }

            }

            currentHistogram[index]++;
            NormalisasiCurrentHistogram[index]
=
currentHistogram[index]/MemberofClass[indexclas
s].CollectionOfDescritorTraining_()[j].rows;


        }
```

```
trainingHistogram.append(NormalisasiCurrentHist
ogram);
    }

    for(int p=0;
p<trainingHistogram.size();p++)
    {

CollectionTrainingHistogram.append(trainingHist
ogram[p]);
    }
}


void
NBayesOrbClassify::GetHistogramTrainingofDistan
ce(int indexclass)
{
    trainingHistogram.clear();
    for (int j = 0; j <
MemberofClass[indexclass].CollectionOfDescritor
Training_().size(); j++)
    {

QVector<float>currentHistogram(CentersClass.row
s,0);

QVector<double>tempSumMinimumDistance(CentersCl
ass.rows,0);
        QVector<float>tempCurrentDistance;

QVector<float>averageMinimumDistance(CentersCla
ss.rows,0);

QVector<float>normalisasiMinimumDistance(Center
sClass.rows,0);
        for(int k = 0 ; k <

MemberofClass[indexclass].CollectionOfDescritor
```

```
Training_()[j].rows; k++)
        {

            Mat MAttempCurrentDistance;
            tempCurrentDistance.clear();
            int index = 0;
            for(int l = 0; l <
CentersClass.rows; l++ )
            {

                double CurrentDistance =
norm(MemberofClass[indexclass].CollectionOfDesc
ritorTraining_()[j].row(k),
CentersClass.row(l),NORM_L2);

MAttempCurrentDistance.push_back(CurrentDistanc
e);

tempCurrentDistance.append(CurrentDistance);

            }

            Mat tempDistance;

sortIdx(MAttempCurrentDistance,tempDistance,
SORT_ASCENDING + SORT_EVERY_COLUMN);
            QVector<int>indexSortingDistance;
            for(int i = 0 ; i <
tempDistance.rows; i++)
            {

indexSortingDistance.append(tempDistance.at<int
>(i,0));

            }

            float valueIndex = 1;


            for(int i = 0 ; i <
```

```
indexSortingDistance.size(); i++)
            {

                index = indexSortingDistance[i];
                tempSumMinimumDistance[index] +=
tempCurrentDistance[index] *valueIndex;
                currentHistogram[index] +=
valueIndex;
                valueIndex = valueIndex * (0.5);
                averageMinimumDistance[index] =
tempSumMinimumDistance[index]/currentHistogram[
index];
            }

        }


        float totalAverageMinimumDistance = 0;
        for (int i = 0; i<CentersClass.rows;
i++)
        {
            totalAverageMinimumDistance +=
averageMinimumDistance[i];

        }

        for (int j = 0; j < CentersClass.rows;
j++)
        {

normalisasiMinimumDistance[j]=averageMinimumDis
tance[j]/totalAverageMinimumDistance;
        }


trainingHistogram.append(normalisasiMinimumDist
ance);


}
```

```
    for(int p=0;
p<trainingHistogram.size();p++)
    {

CollectionTrainingHistogram.append(trainingHist
ogram[p]);
    }
}


void
NBayesOrbClassify::GetMeansDeviasiTraining(int
indexclass)
{

    float sumCentroid = 0;
    float sumMeansCentroid = 0;

    QVector<float>Means(CentersClass.rows,0);
    QVector<float>Deviasi(CentersClass.rows,0);
    for(int d=0; d < CentersClass.rows; d++)
    {
        for(int a=0; a <
trainingHistogram.size(); a++)
        {
            sumCentroid +=
trainingHistogram[a][d];
        }

        Means[d]=
sumCentroid/trainingHistogram.size();

        for (int b=0; b <
trainingHistogram.size(); b++)
        {
            sumMeansCentroid +=
```

```
((trainingHistogram[b][d]-
Means[d])*(trainingHistogram[b][d]-Means[d]));
        }
        Deviasi[d]=
sqrt(sumMeansCentroid/trainingHistogram.size())
;

        sumCentroid = 0 ;
        sumMeansCentroid = 0 ;

    }
    MeansAllClass.append(Means);
    DeviasiAllClass.append(Deviasi);
}

void NBayesOrbClassify::GetHistogramTesting()
{
    ArrayHistogramTesting.clear();
    for(int z=0; z<MemberofClass.size();z++)
    {

QVector<float>HistogramTesting(CentersClass.row
s,0);

QVector<float>NormailisasiHistogramTesting(Cent
ersClass.rows,0);

QVector<double>sumMinimumDistance(CentersClass.
rows, 0);

QVector<float>averageMinimumDistance(CentersCla
ss.rows , 0);

        for(int j = 0 ; j
<DescriptorTestingFloat.rows; j++)
        {
            int index = 0;
            double MinimumDistance =

norm(DescriptorTestingFloat.row(j),
```

```
CentersClass.row(0),NORM_L2);

            for(int k = 1; k <
CentersClass.rows; k++ )
            {

                double CurrentDistance =
norm(DescriptorTestingFloat.row(j),CentersClass
.row(k),NORM_L2);
                if(MinimumDistance >
CurrentDistance)
                {
                    MinimumDistance =
CurrentDistance;
                    index = k;
                }
            }

            HistogramTesting[index]++;

sumMinimumDistance[index]+=MinimumDistance;
            averageMinimumDistance[index] =
sumMinimumDistance[index]/HistogramTesting[inde
x];


            index = 0;
        }
        float totalAverageMinimumDistance = 0;
        for(int i = 0; i<CentersClass.rows;
i++)
        {
            totalAverageMinimumDistance +=
averageMinimumDistance[i];
        }
        for(int j = 0; j<CentersClass.rows;
j++)
        {

            NormailisasiHistogramTesting[j] =
```

```
averageMinimumDistance[j]/totalAverageMinimumDi
stance;
        }



ArrayHistogramTesting.append(NormailisasiHistog
ramTesting);
        NormailisasiHistogramTesting.clear();
        HistogramTesting.clear();
        sumMinimumDistance.clear();
        averageMinimumDistance.clear();
    }

}
void
NBayesOrbClassify::GetHistogramTestingofAccupan
cy()
{
    ArrayHistogramTesting.clear();
    for(int z=0; z<MemberofClass.size();z++)
    {

QVector<float>HistogramTesting(CentersClass.row
s,0);

QVector<float>NormailisasiHistogramTesting(Cent
ersClass.rows,0);

        for(int j = 0 ; j
<DescriptorTestingFloat.rows; j++)
        {
            int index = 0;
            double MinimumDistance =
norm(DescriptorTestingFloat.row(j),
CentersClass.row(0),NORM_L2);

            for(int k = 1; k <

CentersClass.rows; k++ )
```

```
            {

                double CurrentDistance =
norm(DescriptorTestingFloat.row(j),CentersClass
.row(k),NORM_L2);
                if(MinimumDistance >
CurrentDistance)
                {
                    MinimumDistance =
CurrentDistance;
                    index = k;
                }
            }

            HistogramTesting[index]++;
            NormailisasiHistogramTesting[index]
=
HistogramTesting[index]/DescriptorTestingFloat.
rows;

            index = 0;
        }


ArrayHistogramTesting.append(NormailisasiHistog
ramTesting);
        NormailisasiHistogramTesting.clear();
        HistogramTesting.clear();

    }
}

void
NBayesOrbClassify::GetHistogramTestingofDistanc
e()
{
    ArrayHistogramTesting.clear();
    for(int z=0; z<MemberofClass.size();z++)

    {
```

```
QVector<float>HistogramTesting(CentersClass.rows,0);

QVector<float>NormailisasiHistogramTesting(CentersClass.rows,0);

QVector<float>averageMinimumDistance(CentersClass.rows , 0);
        QVector<float>tempCurrentDistance;

QVector<double>sumMinimumDistance(CentersClass.rows, 0);


        for(int j = 0 ; j <DescriptorTestingFloat.rows; j++)
        {
            int index = 0;
            tempCurrentDistance.clear();
            Mat matTempCurrentDistance;
            for(int k = 0; k < CentersClass.rows; k++ )
            {
                double CurrentDistance = norm(DescriptorTestingFloat.row(j),CentersClass.row(k),NORM_L2);


matTempCurrentDistance.push_back(CurrentDistance);

tempCurrentDistance.append(CurrentDistance);
            }


            Mat tempDistance;


sortIdx(matTempCurrentDistance,tempDistance,
```

```
SORT_ASCENDING + SORT_EVERY_COLUMN);

            QVector<int>indexSortingDistance;
            for(int i = 0 ; i <
tempDistance.rows; i++)
            {


indexSortingDistance.append(tempDistance.at<int
>(i,0));
            }


            float valueIndex = 1;
            for(int i =0 ; i
<indexSortingDistance.size(); i++)
            {
                index =
indexSortingDistance[i];
                sumMinimumDistance[index] +=
tempCurrentDistance[index] * valueIndex;
                HistogramTesting[index] +=
valueIndex;
                valueIndex = valueIndex*(0.5);
                averageMinimumDistance[index] =
sumMinimumDistance[index]/HistogramTesting[inde
x];

            }


            index = 0;
        }


        float totalAverageMinimumDistance = 0;

        for(int i = 0; i<CentersClass.rows;

i++)
```

```
        {

            totalAverageMinimumDistance +=
averageMinimumDistance[i];
        }

        for(int j = 0; j<CentersClass.rows;
j++)
        {
            NormailisasiHistogramTesting[j] =
averageMinimumDistance[j]/totalAverageMinimumDi
stance;
        }



ArrayHistogramTesting.append(NormailisasiHistog
ramTesting);

        NormailisasiHistogramTesting.clear();
        HistogramTesting.clear();
        sumMinimumDistance.clear();
        averageMinimumDistance.clear();
    }

}

void NBayesOrbClassify::GetMeansandDeviasi()
{
    float AllTraining = 0;
    float AllTrainingMeans = 0;

    for (int i = 0; i < CentersClass.rows; i++)
    {
        for (int j = 0; j <
CollectionTrainingHistogram.size(); j++)
        {
            AllTraining +=

CollectionTrainingHistogram[j][i];
```

```
        }
        sumMeans[i] =
AllTraining/CollectionTrainingHistogram.size();


        for(int k=0; k <
CollectionTrainingHistogram.size(); k++)
        {
            AllTrainingMeans +=
((CollectionTrainingHistogram[k][i]-
sumMeans[i])*(CollectionTrainingHistogram[k][i]
-sumMeans[i]));
        }
        sumDeviasi[i] =
sqrt(AllTrainingMeans/CollectionTrainingHistogr
am.size());

        AllTraining = 0;
        AllTrainingMeans = 0;

    }

}

float NBayesOrbClassify::GetEvidence(int index,
float ValueOfIndex)
{
    float A;
    float B;

    if(sumDeviasi[index]==0)
    {
        if(ValueOfIndex==sumMeans[index])
        {
            ProbabilityEvidence = 1.0;
        }
    }
    else

    {
```

```
        A = 1.0/
((sumDeviasi[index])*(sqrt((2*3.1416))));
        B = -(((ValueOfIndex-
sumMeans[index])*(ValueOfIndex-
sumMeans[index]))/(2*(sumDeviasi[index]*sumDevi
asi[index])));

        float e = 2.7183;
        ProbabilityEvidence = (A*(pow(e,B)));
    }


    return ProbabilityEvidence;

}

float NBayesOrbClassify::GetPrior()
{
    ProbabilityPrior =
(float)1.0/(float)MemberofClass.size();

    return ProbabilityPrior;
}

float NBayesOrbClassify::GetPosterior(int
indexclass)
{

    ProbabilityPosteior=
(Likelihood[indexclass]*ProbabilityPrior);
    return ProbabilityPosteior;
}

float NBayesOrbClassify::GetLikelihood(int
indexclass, int index, float ValueOfIndex)
{
    if (DeviasiAllClass[indexclass][index] ==
0)

    {
```

```
        if(ValueOfIndex ==
MeansAllClass[indexclass][index])
        {
            ProbabilityLikelihood = 1.0 ;
        }
        else
        {
            ProbabilityLikelihood = 0.0;
        }
    }
    else
    {
        float std_dev =
DeviasiAllClass[indexclass][index];
        float inv_std_dev = 1.0 / std_dev;
        float mean_value =
MeansAllClass[indexclass][index];
        float lower_cdf = 0.5 + (0.5 *
std::erf(ValueOfIndex - mean_value - std_dev) *
inv_std_dev * M_SQRT1_2);
        float upper_cdf = 0.5 + (0.5 *
std::erf(ValueOfIndex - mean_value + std_dev) *
inv_std_dev * M_SQRT1_2);

        ProbabilityLikelihood = upper_cdf -
lower_cdf;
    }

    return ProbabilityLikelihood;
}

float
NBayesOrbClassify::GetLikelihoodDistGaus(int
indexclass, int index, float ValueOfIndex)
{
    float A;
    float B;
    if (DeviasiAllClass[indexclass][index] ==

)
```

```
    {
        if(ValueOfIndex ==
MeansAllClass[indexclass][index])
        {
            ProbabilityLikelihood = 1.0 ;
        }
        else
        {
            ProbabilityLikelihood = 0.0;
        }
    }
    else
    {
        A = 1.0/
((DeviasiAllClass[indexclass][index])*(sqrt((2*
3.1416))));
        B = -(((ValueOfIndex-
MeansAllClass[indexclass][index])*(ValueOfIndex
-
MeansAllClass[indexclass][index]))/(2*(DeviasiA
llClass[indexclass][index]*DeviasiAllClass[inde
xclass][index])));

        float e = 2.7183;
        ProbabilityLikelihood = (A*(pow(e,B)));

    }
    return ProbabilityLikelihood;
}

QString NBayesOrbClassify::ResultOfImage_()
{
    return ResultOfImage;
}

QVector<NBayesClassORB>
NBayesOrbClassify::MemberofClass_()
{

    return MemberofClass;
```

```
}


Mat NBayesOrbClassify::objectDetect_()
{
    return objectDetect;
}

Mat NBayesOrbClassify::objectClear_()
{
    return objectClear_();
}


void NBayesOrbClassify::getNMS(QVector<int>
indexRect, float overThres, vector<int>
&indexOut)
{
    indexOut.clear();
    vector<int>koordinatRectx1;
    vector<int>koordinatRectx2;
    vector<int>koordinatRecty1;
    vector<int>koordinatRecty2;
    Mat koordinatY2;
    vector<float>tempArea;
    for (int i = 0; i < indexRect.size(); i++)
    {

        int x1, y1, x2, y2;
        x1 = tempScaleRect[indexRect[i]].x;
        y1 = tempScaleRect[indexRect[i]].y;
        x2 = x1 +
tempScaleRect[indexRect[i]].width;
        y2 = x1 +
tempScaleRect[indexRect[i]].height;
```

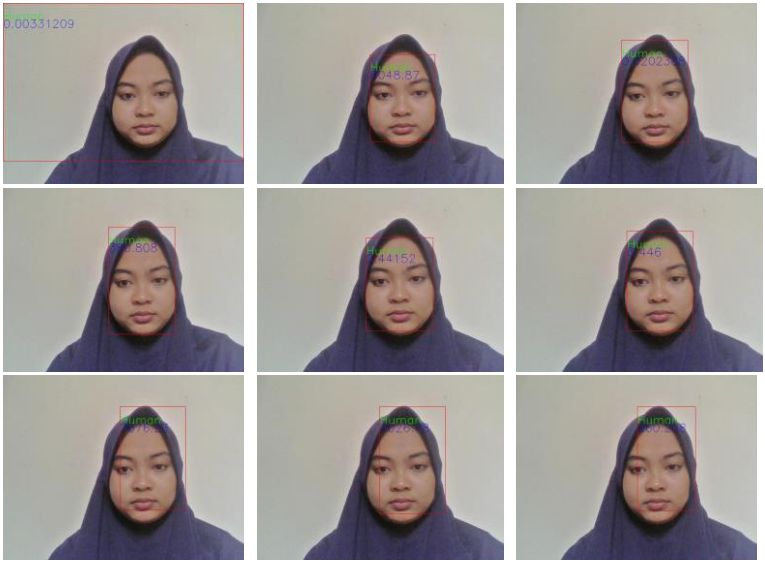
```
        float area;
        area = (x2 - x1 + 1)*(y2-y1 +1);

        koordinatRectx1.push_back(x1);
        koordinatRecty1.push_back(y1);
        koordinatRectx2.push_back(x2);
        koordinatRecty2.push_back(y2);
        koordinatY2.push_back(y2);
        tempArea.push_back(area);
    }


    Mat koordinatSortingy2;
    sortIdx(koordinatY2, koordinatSortingy2,
SORT_ASCENDING + SORT_EVERY_COLUMN);

    vector<int>indexKoordinatSorting;
    for(int i = 0; i < koordinatSortingy2.rows;
i++)
    {

indexKoordinatSorting.push_back(koordinatSortin
gy2.at<int>(i,0));

    }


    while (indexKoordinatSorting.size() > 0)
    {

        // menentukan index terakhir

        int indexLast =
indexKoordinatSorting.size() - 1;
        int last =
indexKoordinatSorting[indexLast];
        indexOut.push_back(last);
        vector<float>overlap_All;
```

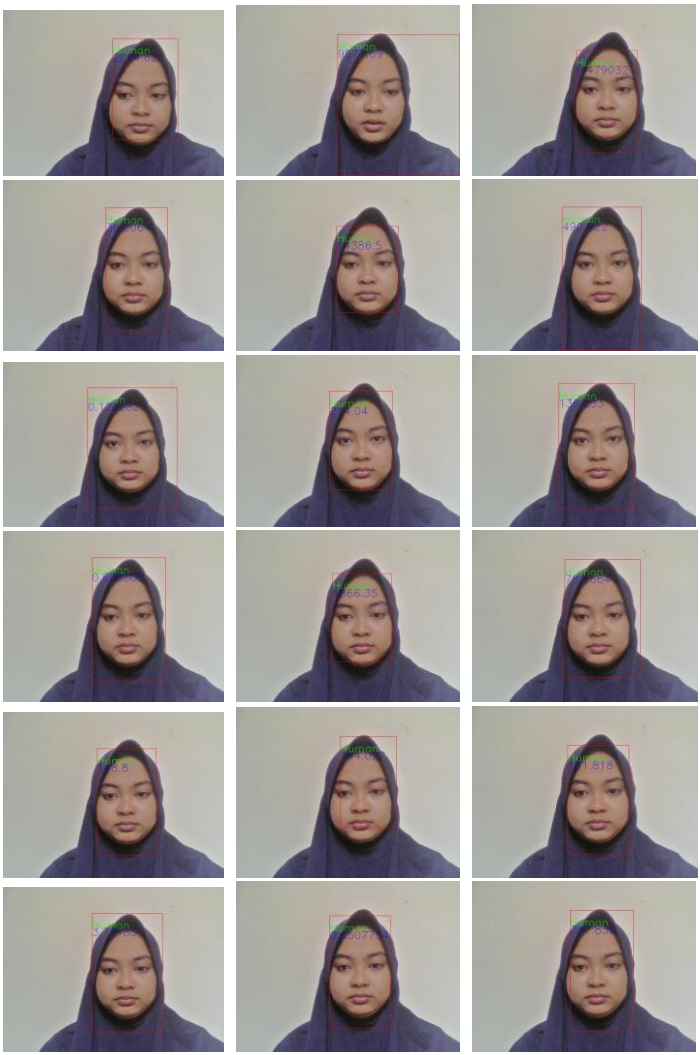
```
        for (int i = 0; i < indexLast; i++)
        {

            int currentIndex =
indexKoordinatSorting[i];
            int xx1_ =
max(koordinatRectx1[last],
koordinatRectx1[currentIndex]);
            int yy1_ =
max(koordinatRecty1[last],
koordinatRecty1[currentIndex]);
            int xx2_ =
min(koordinatRectx2[last],
koordinatRectx2[currentIndex]);
            int yy2_ =
min(koordinatRecty2[last],
koordinatRecty2[currentIndex]);


            float w = max(0, (xx2_ - xx1_) +1);
            float h = max(0, (yy2_ - yy1_) +1);
            float overlap = (w *
h)/(tempArea[currentIndex] - tempArea[last] +
(w*h));
            overlap_All.push_back(overlap);


       }

        std::vector<int>new_sorted_indexes;
        indexKoordinatSorting.pop_back();

        for(int i = 0 ; i < indexLast ; i++)
        {

            if (overlap_All[i] < overThres ||
setIndexClass[last] !=
setIndexClass[indexKoordinatSorting[i]])

            {
```

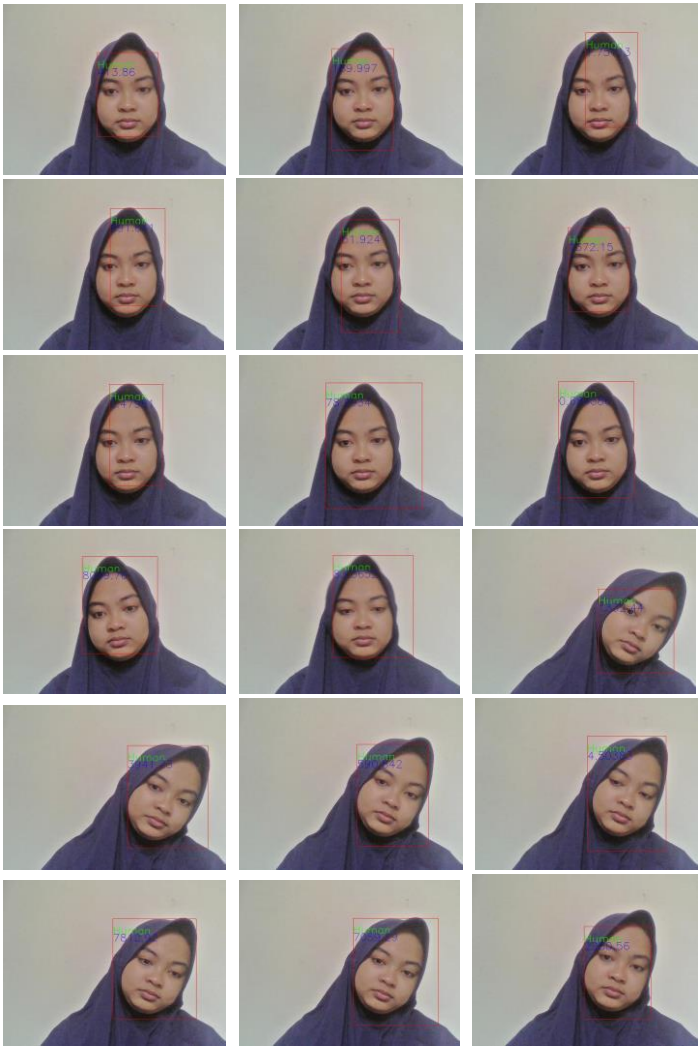
```
new_sorted_indexes.push_back(indexKoordinatSort
ing[i]);
             }

         }
         indexKoordinatSorting =
new_sorted_indexes;



    }
}

QString NBayesOrbClassify::timeCount_()
{
    return timeCount;
}

QVector<Mat>
NBayesOrbClassify::tempClearPict_()
{
    return tempClearPict;
}


void NBayesOrbClassify::GetCentroid()
{
    int clusterCount = 4;
    Mat labels;
    int attempts = 5;

    if(CollectionDataTrainingAllClass.rows ==
0)
    {

        qDebug()<<"Descriptor Empty";
```

```
    }
    else
    {
        kmeans(CollectionDataTrainingAllClass,
clusterCount, labels,
TermCriteria(CV_TERMCRIT_ITER|CV_TERMCRIT_EPS,
10000, 0.0001), attempts, KMEANS_PP_CENTERS,
CentersClass );

    }

}
```

B-1 Gambar Pengujian Data Positif *Histogram of Occupancy*







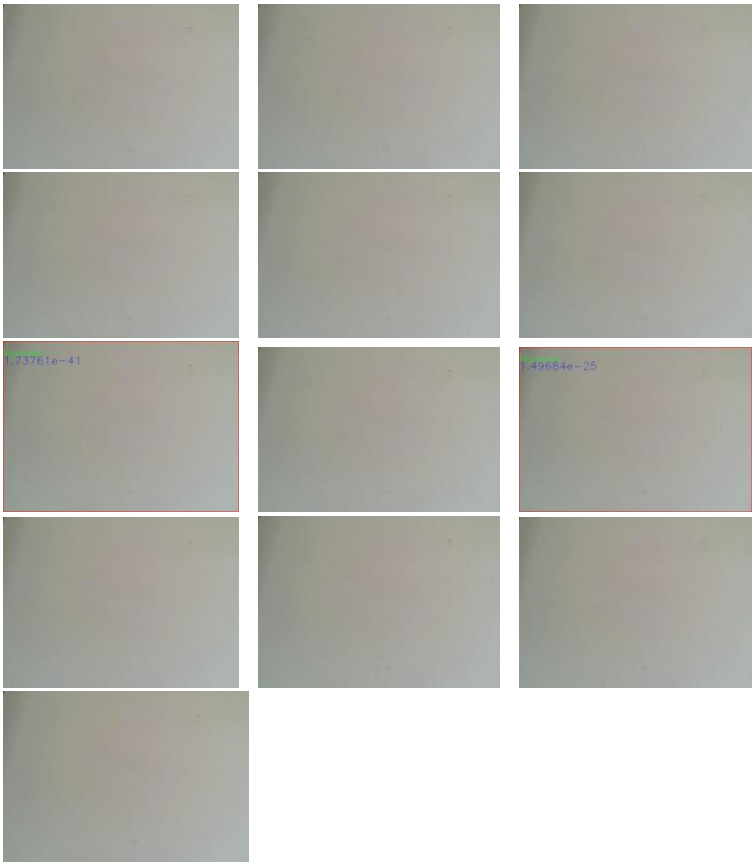
B-2 Gambar Pengujian Data Negatif *Histogram of Occupancy*


1.02337e-15
4.09384e-32
1.14329e-33

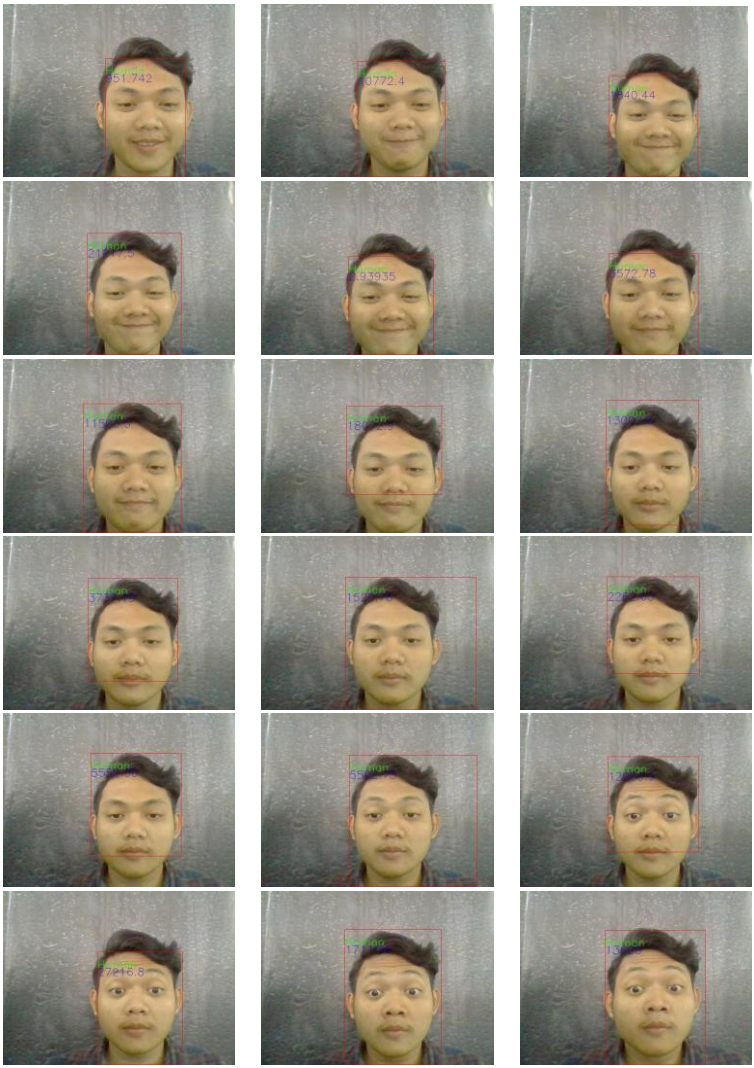

4.09384e-32
1.14329e-33
1.70958e-43


1.10445e-34
4.09384e-32
1.53311e-5

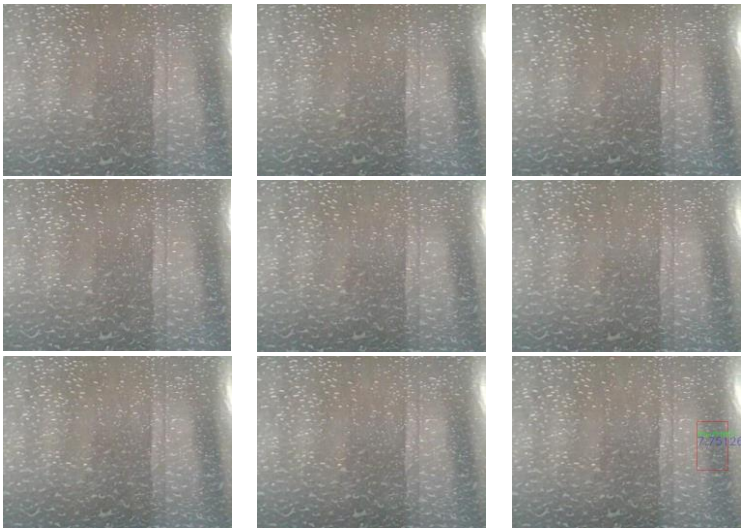

1.02337e-15
1.02337e-15

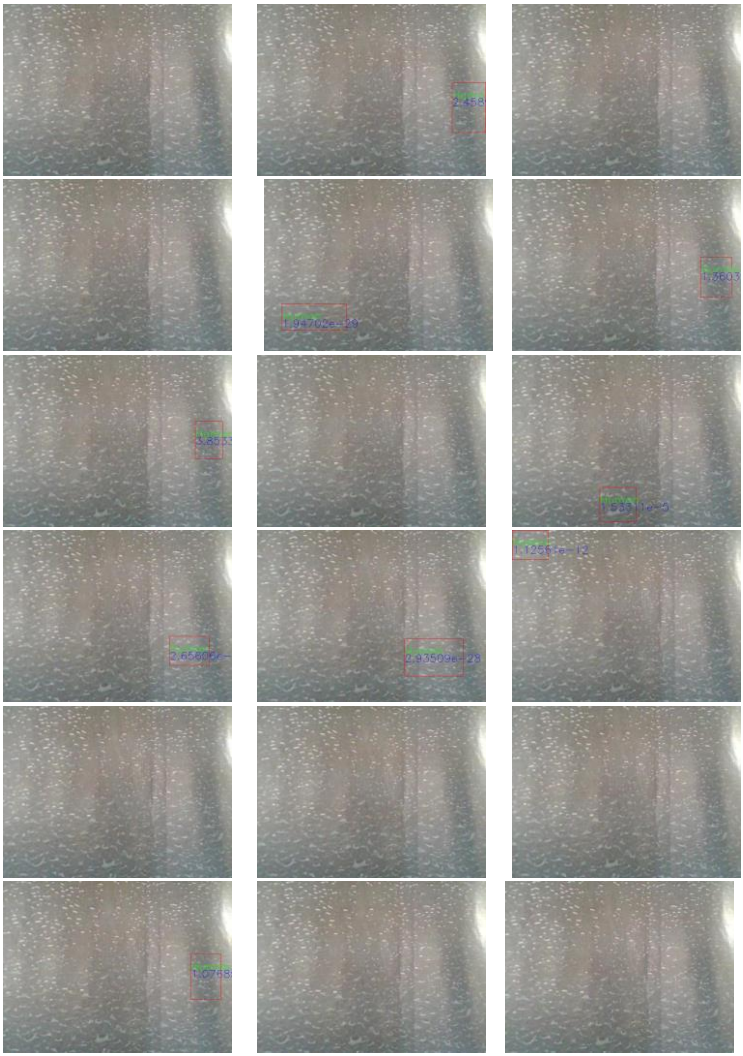


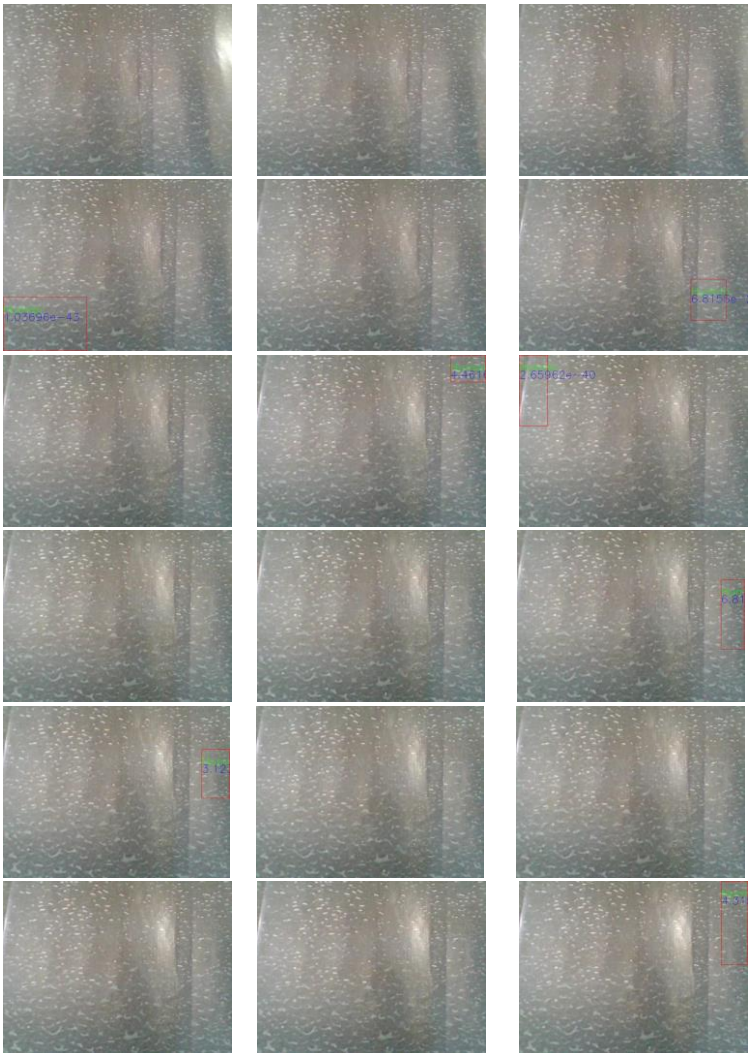
B-3 Gambar Pengujian Data Positif 4 Wajah Berbeda

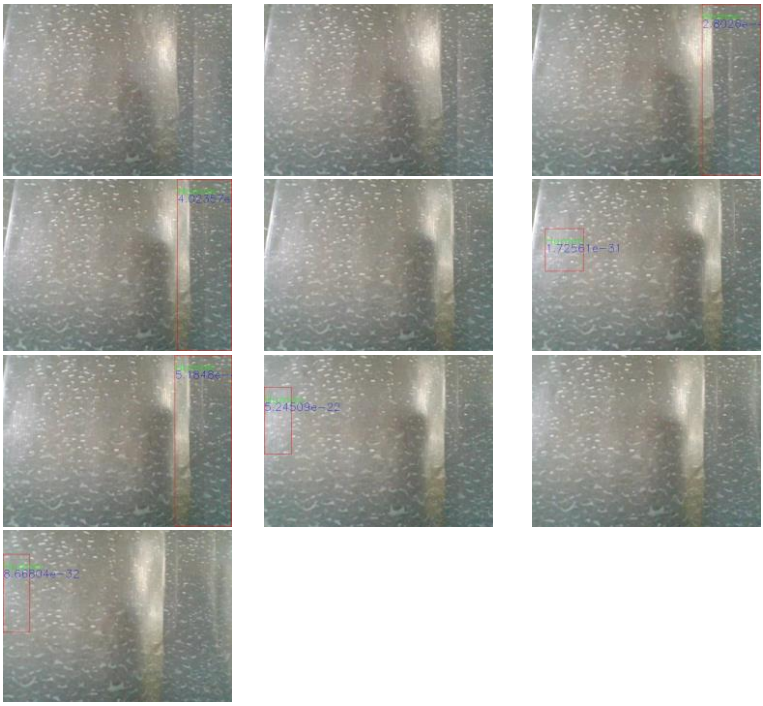


B-3 Gambar Pengujian Data Positif 4 Wajah Berbeda









C-1 Tabel Hasil Pengujian *Histogram of Occupancy*

Data ke -	*True* Positif	*False* Positif	*True* Negatif	*False* Negatif
1		1	1	
2	1			1
3	1		1	
4	1		1	
5	1		1	
6	1		1	
7	1			1
8	1		1	
9	1		1	
10	1		1	
11	1		1	

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
12	1		1	
13	1			1
14	1		1	
15	1		1	
16	1		1	
17	1		1	
18	1		1	
19	1			1
20	1		1	
21	1		1	
22	1		1	
23	1		1	
24	1		1	
25		1	1	
26	1		1	
27	1		1	
28	1		1	
29		1		1
30	1		1	
31	1		1	
32		1	1	1
33	1		1	
34		1	1	
35	1		1	
36	1		1	
37	1		1	
38	1		1	
39	1			1
40	1		1	
41	1		1	
42	1		1	
43	1		1	
44	1		1	
45	1		1	
46	1		1	

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
47		1		
48	1		1	
49	1		1	
50	1			1
51	1			1
52	1			1
53	1			1
54	1		1	
55	1		1	
56	1		1	
57	1		1	
58	1		1	
59	1			1
60	1		1	
61	1		1	
62	1		1	
63	1			1
64	1		1	
65	1		1	
66	1		1	
67	1		1	
68		1	1	
69	1		1	
70	1		1	
71	1		1	
72	1		1	
73	1		1	
74	1		1	
75	1		1	
76	1		1	
77	1		1	
78	1		1	
79	1		1	
80	1		1	
81	1		1	

Data ke -	*True* Positif	*False* Positif	*True* Negatif	*False* Negatif
82	1			1
83	1		1	
84	1		1	
85	1		1	
86	1		1	
87	1		1	
88	1		1	
89	1		1	
90	1		1	
91	1		1	
92	1		1	
93	1		1	
94	1			1
95	1		1	
96	1			1
97	1		1	
98	1		1	
99	1		1	
100	1		1	

C-2 Tabel Hasil Pengujian *Histogram of Distance* Normal

Data ke -	*True* Positif	*False* Positif	*True* Negatif	*False* Negatif
1		1	1	
2		1	1	
3	1		1	
4	1		1	
5	1		1	
6		1	1	
7		1	1	
8		1	1	
9		1	1	
10	1		1	
11	1		1	
12	1		1	

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
13	1		1	
14	1		1	
15		1	1	
16		1	1	
17	1		1	
18		1	1	
19	1		1	
20		1	1	
21	1		1	
22	1		1	
23		1	1	
24	1		1	
25		1	1	
26		1	1	
27		1	1	
28		1	1	
29		1	1	
30		1	1	
31		1	1	
32	1		1	
33	1		1	
34	1		1	
35		1	1	
36	1		1	
37		1	1	
38		1	1	
39		1	1	
40	1		1	
41	1		1	
42		1	1	
43		1	1	
44	1		1	
45		1	1	
46		1	1	
47		1	1	

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
48		1	1	
49		1	1	
50	1		1	
51	1		1	
52		1	1	
53		1	1	
54		1	1	
55		1	1	
56	1		1	
57		1	1	
58	1		1	
59	1		1	
60	1		1	
61	1		1	
62	1		1	
63	1		1	
64		1	1	
65		1	1	
66		1	1	
67	1		1	
68	1		1	
69		1	1	
70		1	1	
71		1	1	
72		1	1	
73		1	1	
74		1	1	
75		1	1	
76		1	1	
77		1	1	
78		1	1	
79		1	1	
80		1	1	
81		1	1	

Data ke -	*True* Positif	*False* Positif	*True* Negatif	*False* Negatif
82		1	1	
83		1	1	
84		1	1	
85		1	1	
86	1		1	
87		1	1	
88	1		1	
89	1		1	
90		1	1	
91	1		1	
92		1	1	
93	1		1	
94	1		1	
95	1		1	
96		1	1	
97		1	1	
98		1	1	
99	1		1	
100	1		1	

C-3 Tabel Hasil Pengujian *Histogram of Distance Softweight*

Data ke -	*True* Positif	*False* Positif	*True* Negatif	*False* Negatif
1		1	1	
2		1	1	
3	1		1	
4		1	1	
5	1		1	
6		1	1	
7	1		1	
8		1		1
9	1		1	
10		1	1	
11		1	1	
12		1	1	

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
13	1		1	
14	1		1	
15		1	1	
16		1	1	
17	1		1	
18	1		1	
19		1	1	
20			1	
21	1		1	
22		1	1	
23	1		1	
24	1		1	
25	1		1	
26		1	1	
27		1	1	
28		1	1	
29		1	1	
30	1		1	
31	1		1	
32		1	1	
33		1	1	
34	1		1	
35	1		1	
36	1		1	
37	1		1	
38		1	1	
39		1	1	
40		1	1	
41		1	1	
42		1	1	
43	1		1	
44	1		1	
45	1			1
46	1		1	
47	1		1	

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
48	1			1
49	1		1	
50	1		1	
51		1	1	
52		1	1	
53			1	
54		1	1	
55	1		1	
56		1	1	
57		1	1	
58		1	1	
59	1			1
60		1	1	
61	1		1	
62	1			1
63		1	1	
64	1		1	
65	1		1	
66		1	1	
67		1	1	
68	1		1	
69	1		1	
70	1		1	
71		1	1	
72	1		1	
73	1		1	
74	1		1	
75	1		1	
76	1		1	
77		1	1	
78	1		1	
79		1	1	
80	1		1	
81	1		1	
82		1	1	

Data ke -	*True* Positif	*False* Positif	*True* Negatif	*False* Negatif
83	1		1	
84	1		1	
85		1	1	
86	1		1	
87		1	1	
88	1		1	
89	1		1	
90	1		1	
91		1	1	
92	1		1	
93	1		1	
94	1		1	
95	1		1	
96	1	1	1	
97	1		1	
98	1		1	
99	1		1	
100	1	1	1	

C-4 Tabel Hasil Pengujian Perbandingan Data *Testing*

Data ke -	*True* Positif	*False* Positif	*True* Negatif	*False* Negatif
1		1	1	
2		1	1	
3	1		1	
4	1		1	
5	1		1	
6		1	1	
7		1	1	
8		1		1
9		1	1	
10	1			1
11	1		1	
12	1		1	
13	1		1	

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
14	1		1	
15		1		1
16		1	1	
17	1			1
18		1	1	
19	1			1
20		1	1	
21	1		1	
22	1		1	
23		1		1
24	1			1
25		1		1
26		1		1
27		1		1
28		1		1
29		1		1
30		1	1	
31		1	1	
32	1		1	
33	1			1
34	1		1	
35		1		1
36	1		1	
37		1	1	
38		1		1
39		1	1	
40	1		1	
41	1		1	
42		1	1	
43		1	1	
44	1			1
45		1	1	
46		1	1	
47		1	1	
48		1		1

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
49		1	1	
50	1		1	
51	1		1	
52		1	1	
53		1	1	
54		1	1	
55		1	1	
56	1		1	
57		1		1
58	1		1	
59	1		1	
60	1		1	
61	1		1	
62	1			1
63	1			1
64		1	1	
65		1		1
66		1	1	
67	1		1	
68	1		1	
69		1	1	
70		1	1	
71		1	1	
72		1	1	
73		1	1	
74		1	1	
75		1		1
76		1	1	
77		1		1
78		1	1	
79		1		1
80		1		1
81		1	1	
82		1	1	
83		1		1

Data ke -	***True* Positif**	***False* Positif**	***True* Negatif**	***False* Negatif**
84		1		1
85		1	1	
86	1		1	
87		1	1	
88	1		1	
89	1			1
90		1	1	
91	1		1	
92		1		1
93	1			1
94	1		1	
95	1			1
96		1		1
97		1		1
98		1		1
99	1			1
100	1			1

RIWAYAT HIDUP

Salma Puspa Mega lahir di Boyolali pada tanggal 2 Agustus 1998. Penulis merupakan anak pertama dari 3 bersaudara. Penulis menenempuh pendidikan dasar di SDN Karangduren 2, kemudian melanjutkan pendidikan di SMPN 1 Kartasura dan menempuh pendidikan tingkat atas di SMAN 1 Teras Boyolali. Penulis melanjutkan pendidikan perguruan tinggi pada tahun 2016 di Departemen Teknik Elektro Otomasi Fakultas Vokasi Institut Teknologi Sepuluh Nopember. Semasa kuliah penulis aktif sebagai anggota Cyber Physical, Industrial Robotics, and Automation Laboratory sejak 2017. Penulis telah menyelesaikan magang dan pengerjaan Proyek Akhir di PT. Bhimasena Research and Development.