

TUGAS AKHIR - KI141502

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT INDONESIA MENGGUNAKAN TEKNOLOGI KINECT DAN METODE BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK

Yohanes Aditya Sutanto
NRP 5112100135

Dosen Pembimbing
Wijayanti Nurul Khotimah, S.Kom., M.Sc.
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]



TUGAS AKHIR - KI141502

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT INDONESIA MENGGUNAKAN TEKNOLOGI KINECT DAN METODE BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK

Yohanes Aditya Sutanto
NRP 5112100135

Dosen Pembimbing
Wijayanti Nurul Khotimah, S.Kom., M.Sc.
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

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

[Halaman ini sengaja dikosongkan]



FINAL PROJECT - KI141502

DESIGN AND IMPLEMENTATION OF INDONESIA SIGN LANGUAGE MODULE USING KINECT TECHNOLOGY AND BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK METHOD

YOHANES ADITYA SUTANTO
NRP 5112100135

Advisor
Wijayanti Nurul Khotimah, S.Kom., M.Sc.
Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

INFORMATICS DEPARTMENT
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2016

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI KINECT DAN METODE *BACK PROPAGATION-GENETIC ALGORITHM* *NEURAL NETWORK*

TUGAS AKHIR

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

Oleh:

YOHANES ADITYA SUTANTO
NRP. 5112100135

Disetujui oleh Pembimbing Tugas Akhir:

1. Wijayanti Nurul Khotimah, S.Kom., M.Sc.
NIP: 19860312 201212 2 004 (pembimbing 1)
2. Dr.Eng. Nanik Suciati, S.Kom., M.Kom.
NIP: 19870213 201404 1 001 (pembimbing 2)

SURABAYA
JUNI, 2016

[Halaman ini sengaja dikosongkan]

RANCANG BANGUN MODUL PENGENALAN BAHASA ISYARAT MENGGUNAKAN TEKNOLOGI KINECT DAN METODE BACK PROPAGATION-GENETIC ALGORITHM NEURAL NETWORK

Nama Mahasiswa : Yohanes Aditya Sutanto
NRP : 5112100135
Jurusan : Teknik Informatika FTIf-ITS
Dosen Pembimbing I : Wijayanti Nurul Khotimah, S.Kom, M.Sc.
Dosen Pembimbing II : Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRAK

Bahasa isyarat adalah hal yang penting dalam komunikasi bagi orang yang menderita gangguan pendengaran. Kecepatan menguasai bahasa dan kemampuan mereka berinteraksi sangat dibutuhkan. Mereka membutuhkan bahan pembelajaran yang tidak hanya berisi tentang komponen aural saja, namun juga secara visual karena lebih nyata.

Di sisi lain, teknologi berkembang pesat di segala aspek kehidupan. Berbagai macam terobosan teknologi baru telah diciptakan oleh manusia, salah satunya perangkat Kinect Sensor yang diciptakan untuk windows sekitar tahun 2012. Dengan menggunakan Kinect Sensor, manusia dapat melakukan interaksi dengan komputer tanpa sentuhan sama sekali.

Oleh karena itu, penulis memiliki ide untuk membuat aplikasi modul pengenalan Bahasa isyarat Indonesia berdasarkan Sistem Bahasa Isyarat Indonesia (SIBI) dengan menggunakan Kinect Sensor. Harapan aplikasi ini agar dapat membantu pengguna untuk belajar mengenai bahasa isyarat.

Kata kunci: Kinect Sensor, bahasa isyarat, SIBI.

[Halaman ini sengaja dikosongkan]

DESIGN AND IMPLEMENTATION OF INDONESIA SIGN LANGUAGE MODULE USING KINECT TECHNOLOGY AND BACK PROPAGATION GENETIC ALGORITHM NEURAL NETWORK METHOD

Name : Yohanes Aditya Sutanto
NRP : 5112100135
Major : Informatics Department, FTIf-ITS
Advisor I : Wijayanti Nurul Khotimah, S.Kom, M.Sc.
Advisor II : Dr.Eng. Nanik Suciati, S.Kom., M.Kom.

ABSTRACT

Sign language is an important thing in the communication of deaf people. The speed of learning sign language and their skills to interact each other is very necessary. They need a media and the learning materials is not only about aural matter, but also visual.

On the other hand, technology growth very quickly. The new Many technology are invented, one of them is Kinect Sensor which invented by Microsoft in 2012. With Kinect Sensor, human can interact with computer touchless.

Therefore, the author had the idea to create application that can be used to learn about Indonesian sign language based on Sistem Isyarat Bahasa Indonesia (SIBI). The expectation of this application is to help people know and learn about Indonesian sign language.

Keywords: : Kinect Sensor, Sign Language, SIBI

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Puji dan syukur kepada Tuhan Yang Maha Esa atas rahmat dan kasih-Nya yang menyertai penulis selama proses pengerjaan tugas akhir ini sampai selesai.

Penulis ingin menyampaikan rasa hormat dan terima kasih yang setinggi-tingginya kepada pihak-pihak yang telah membantu penulis dalam penyelesaian tugas akhir ini, terutama kepada: Kedua orang tua penulis, yang selalu mendukung penulis mulai dari awal kuliah sampai lulus.

1. Ibu Wijayanti Nurul Khotimah, S.Kom., M.Sc. dan Ibu Dr.Eng. Nanik Suciati, S.Kom., M.Kom. yang telah bersedia untuk menjadi dosen pembimbing tugas akhir sehingga penulis dapat mengerjakan tugas akhir dengan arahan dan bimbingan yang baik dan jelas.
2. Teman-teman Mahasiswa Teknik Informatika 2012 yang telah berjuang bersama-sama selama menempuh pendidikan di Jurusan ini.
3. Kakak kelas Mahasiswa Teknik Informatika 2011 yang telah membantu dalam segala proses perkuliahan di TC ini.
4. Serta pihak-pihak lain yang turut membantu penulis baik secara langsung maupun tidak, yang namanya tidak penulis sebutkan disini.

Penulis telah berusaha sebaik-baiknya dalam menyusun tugas akhir ini, mohon maaf apabila ada kesalahan dan kata-kata yang dapat menyinggung perasaan. Penulis berharap tugas akhir ini dapat menjadi media pembelajaran bahasa isyarat Indonesia.

Surabaya, Juni 2016

Penulis

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

LEMBAR PENGESAHAN.....	vii
ABSTRAK	ix
ABSTRACT	xi
KATA PENGANTAR.....	xiii
DAFTAR ISI	xv
DAFTAR GAMBAR	xvii
DAFTAR TABEL	xix
DAFTAR KODE SUMBER	xxi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Batasan Masalah.....	2
1.4 Tujuan.....	3
1.5 Manfaat.....	3
1.6 Metodologi	3
1.7 Sistematika Penulisan.....	5
BAB II DASAR TEORI.....	7
2.1 Tunarungu	7
2.2 Bahasa Isyarat	7
2.3 Kinect	8
2.4 Kinect SDK	10
2.5 <i>Neural Network</i>	11
2.6 <i>Backpropagation</i>	11
2.7 <i>Backpropagation-Genetic Algorithm Neural Network</i>	12
BAB III ANALISIS DAN PERANCANGAN SISTEM.....	15
3.1 Analisis Perangkat Lunak.....	15
3.1.1 Deskripsi Umum Perangkat Lunak	16
3.1.2 Spesifikasi Kebutuhan Perangkat Lunak.....	16
3.1.3 Identifikasi Pengguna	17
3.2 Perancangan Perangkat Lunak	17
3.2.1 Model Kasus Penggunaan	18
3.2.2 Definisi Aktor.....	18
3.2.3 Definisi Kasus Penggunaan.....	19

3.2.4 Arsitektur Umum Sistem	23
3.2.5 Rancangan Antarmuka Aplikasi	24
3.2.6 Rancangan Proses Aplikasi	25
BAB IV IMPLEMENTASI	29
4.1 Lingkungan Pembangunan	29
4.1.1 Lingkungan Pembangunan Perangkat Keras	29
4.1.2 Lingkungan Pembangunan Perangkat Lunak	29
4.2 Implementasi Antarmuka	29
4.3 Implementasi Aplikasi	30
4.3.1 Implementasi Pendeteksian Skeleton Pengguna	31
4.3.2 Implementasi Proses Ekstraksi Fitur	32
4.3.3 Implementasi Pembuatan Data Bahasa Isyarat Baru	34
4.3.4 Implementasi Proses <i>Training</i> dan <i>Testing</i>	36
4.3.4 Implementasi Proses Normalisasi Fitur	41
4.3.6 Implementasi Proses Pembagian <i>Neural Network</i>	42
BAB V PENGUJIAN DAN EVALUASI	45
5.1 Lingkungan Pembangunan	45
5.2 Skenario Pengujian	45
5.2.1 Pengujian Skenario A1 dan Analisis	47
5.2.2 Pengujian Skenario A2 dan Analisis	48
5.2.3 Pengujian Skenario B1 dan Analisis	49
5.2.4 Pengujian Skenario B2 dan Analisis	50
5.3 Evaluasi	52
BAB VI KESIMPULAN DAN SARAN	53
6.1 Kesimpulan	53
6.2 Saran	54
DAFTAR PUSTAKA	55
LAMPIRAN A KODE SUMBER[12]	57
LAMPIRAN B SCREENSHOT APLIKASI	71
BIODATA PENULIS	75

DAFTAR GAMBAR

Gambar 2.1 Contoh Bahasa Isyarat yang ada didalam SIBI	8
Gambar 2.2 Microsoft Kinect[6]	9
Gambar 2.3 Depth, Skeleton, dan VGA view[7].....	9
Gambar 2.4 <i>Skeleton Joints</i> yang diketahui Kinect.....	10
Gambar 2.5 Contoh <i>Neural Network</i> Dengan 1 <i>Hidden Layer</i> ...	11
Gambar 3.1 Diagram Kasus Penggunaan Aplikasi	18
Gambar 3.2 Arsitektur Sistem (<i>Training</i>)	23
Gambar 3.3 Arsitektur Sistem (<i>Testing</i>).....	23
Gambar 3.4 Rancangan Antarmuka Aplikasi.....	24
Gambar 3.5. Diagram Alir Proses Ekstraksi Fitur.....	26
Gambar 3.6 <i>Skeleton Joints</i> yang Digunakan.....	27
Gambar 4.1 Antarmuka Aplikasi.....	30
Gambar 5.1 Bahasa Isyarat yang Dipakai	46
Gambar 5.2 Hasil Pengujian Tingkat Akurasi <i>Neural Network</i> ..	52
Gambar B.1 Aplikasi Mendeteksi Bahasa Isyarat Hai	71
Gambar B.2 Aplikasi Mendeteksi Bahasa Isyarat Ketua	71
Gambar B.3 Aplikasi Mendeteksi Bahasa Isyarat Hormat.....	72
Gambar B.4 Aplikasi Mendeteksi Bahasa Isyarat Bentuk	72
Gambar B.5 Aplikasi Mendeteksi Bahasa Isyarat Wadah.....	73
Gambar B.6 Aplikasi Mendeteksi Bahasa Isyarat Alquran.....	73
Gambar B.7 Aplikasi Mendeteksi Bahasa Isyarat Gang	74
Gambar B.8 Aplikasi Mendeteksi Bahasa Isyarat Hamba	74

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 3.1 Definisi Kasus Penggunaan.....	18
Tabel 3.2 Definisi Kasus Penggunaan.....	19
Tabel 3.3 Spesifikasi Kasus Penggunaan Membuat Data Bahasa Isyarat Baru	20
Tabel 3.4 Spesifikasi Kasus Penggunaan <i>Training Dataset</i>	21
Tabel 3.5 Spesifikasi Kasus Penggunaan <i>Testing Dataset</i>	22
Tabel 3.6 Atribut <i>Dataset</i>	27
Tabel 3.7 <i>Rule</i> Pembagian <i>Neural Network</i>	28
Tabel 5.1 Skenario Pengujian A1	47
Tabel 5.2 Terjemahan Isyarat Kata Skenario Pengujian A1	48
Tabel 5.3 Skenario Pengujian A2.....	48
Tabel 5.4 Terjemahan Isyarat Kata Skenario Pengujian A2	49
Tabel 5.5 Skenario Pengujian B1	50
Tabel 5.6 Terjemahan Isyarat Kata Skenario Pengujian B1	50
Tabel 5.7 Skenario Pengujian B2	51
Tabel 5.8 Terjemahan Isyarat Kata Skenario Pengujian B2.....	51

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

Kode Sumber 4.1 Kode Sumber Integrasi Kinect	31
Kode Sumber 4.2. Kode Sumber Deteksi <i>Skeleton</i> Pengguna	32
Kode Sumber 4.3 Kode Sumber Ekstraksi Fitur <i>Skeleton</i>	34
Kode Sumber 4.4 Implementasi Pembuatan Data Bahasa Isyarat Baru	36
Kode Sumber 4.5 Implementasi Proses <i>Traning</i>	39
Kode Sumber 4.6 Implementasi Proses Testing	41
Kode Sumber 4.7 Fungsi Normalisasi Fitur	42
Kode Sumber 4.8 Implementasi fungsi FeatureNorm() pada class Features	43
Kode Sumber 4.9 Implementasi Pembagian <i>Neural Network</i>	43
Kode Sumber A.1 Kelas BackPropagation.cs	59
Kode Sumber A.2 Kelas Chromosom.cs	60
Kode Sumber A.3 Kelas ClassificationClass.cs	61
Kode Sumber A.4 Kelas DataSet.cs	62
Kode Sumber A.5 Kelas FeedForward.cs	64
Kode Sumber A.6 Kelas GeneticAlgorithm.cs	68
Kode Sumber A.7 Kelas Neural Network.cs	70

BAB I

PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan, batasan permasalahan, dan manfaat.

1.1 Latar Belakang

Bahasa isyarat adalah media bagi pada penderita Tunarungu untuk berkomunikasi dengan sekitarnya. Gerakan visual tubuh sangat membantu penderita agar yang ingin disampaikan lebih mudah dimengerti oleh pasangan komunikasinya. Jika penderita berkomunikasi dengan gerakan bibir, tingkat keakuratan untuk mengartikan gerakan bibir tersebut lebih rendah dibandingkan dengan gerakan tubuh. Hal tersebut menjadi alasan untuk mengembangkan bahasa isyarat di Indonesia [1].

Dari survei yang dilakukan Multi Center Study di Asia Tenggara, Indonesia termasuk dalam 4 negara dengan prevalensi ketulian yang cukup tinggi yaitu 4,6 persen. Sedangkan 3 negara lainnya yakni Srilangka (8,8 persen), Myanmar (8,4 persen) dan India (6,3 persen)[2].

Di Indonesia ada dua dasar bahasa isyarat yang digunakan, salah satunya adalah Sistem Isyarat Bahasa Indonesia (SIBI). SIBI sudah menjadi bahasa isyarat Indonesia yang resmi. Di dalamnya terdapat posisi jari dan gerakan tangan untuk menggantikan kosa kata Bahasa Indonesia. Gerakan isyarat yang ada di dalam SIBI sudah diatur secara sistematis.

Di jaman sekarang, teknologi sudah berkembang sangat pesat. Sudah banyak hardware yang bisa membaca isyarat (gesture) yang ada dengan algoritma tertentu. Salah satunya adalah Kinect. Kinect adalah motion sensing input device yang dibuat oleh Microsoft untuk game consolenya Xbox 360 dan Xbox One serta untuk windows PC. Dengan bentuk seperti webcam, Kinect dapat membuat pengguna mengontrol dan berinteraksi dengan console

atau komputernya tanpa perlu controller, tapi melalui natural user interface (menggunakan gesture dan perintah suara). Generasi pertama Kinect diperkenalkan bulan November tahun 2010[3].

Sebelumnya sudah ada Tugas Akhir yang dibuat oleh Risal Andika Tridisaputra tentang pengenalan Bahasa isyarat Indonesia menggunakan teknologi leap motion, tetapi Tugas Akhir tersebut hanya sebatas membaca isyarat abjad. Oleh karena itu, Dengan kemampuan yang dimiliki Kinect, munculah ide untuk membuat software pengenalan bahasa isyarat menggunakan Kinect. Bahasa isyarat yang digunakan akan berpaku pada Sistem Isyarat Bahasa Indonesia (SIBI).

1.2 Rumusan Masalah

Rumusan masalah yang diangkat dalam Tugas Akhir ini adalah sebagai berikut:

1. Bagaimana menentukan fitur yang bisa digunakan untuk mendeteksi Bahasa isyarat yang diberikan dengan Kinect?
2. Bagaimana menerapkan metode *Back Propagation Genetic Algorithm Neural Network* untuk mendeteksi Bahasa isyarat yang diberikan?
3. Bagaimana menentukan ketepatan gerakan isyarat yang dilakukan oleh penderita Tunarungu?.

1.3 Batasan Masalah

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

1. Aplikasi yang akan dibuat adalah aplikasi yang berjalan pada sistem operasi Windows 7, Windows 8/8.1, dan Windows 10.
2. Aplikasi akan dikembangkan menggunakan *Kinect SDK* dengan menggunakan bahasa pemrograman C# dengan IDE Microsoft Visual Studio.
3. Versi Kinect sensor yang dipakai adalah Kinect V1

4. Algoritma *Neural Network* yang akan digunakan adalah *Back Propagation Genetic Algorithm*.
5. Topologi *Neural Network* adalah jenis *Multilayer Perceptron* dengan 1 hidden layer.
6. Bahasa Isyarat yang digunakan berdasarkan pada Sistem Isyarat Bahasa Indonesia (SIBI).
7. Bahasa isyarat yang dikenali adalah kata.
8. Bahasa isyarat yang digunakan hanya yang bersifat statis (tidak terlalu banyak gerak).
9. Hanya 8 bahasa isyarat yang akan dipakai yaitu Alquran, Bentuk, Gang, Hai, Hamba, Hormat, Ketua, dan Wadah.

1.4 Tujuan

Tujuan dari pembuatan Tugas Akhir ini antara lain:

1. Membuat aplikasi pengenalan bahasa isyarat menggunakan teknologi *Kinect*
2. Dapat mengimplementasikan algoritma *Back Propagation Genetic Algorithm Neural Network* sebagai *classifier* gerakan *skeleton* pengguna yang akan didapat dengan *Kinect*

1.5 Manfaat

Manfaat dari Tugas Akhir ini adalah membuat aplikasi pengenalan bahasa isyarat yang nantinya akan digunakan sebagai media pembelajaran bahasa isyarat bagi penderita Tunarungu.

1.6 Metodologi

Pembuatan Tugas Akhir ini dilakukan menggunakan metodologi sebagai berikut:

- A. Studi literatur
Tahap Studi literature merupakan tahap pembelajaran dan pengumpulan informasi yang digunakan untuk

mengimplementasikan Tugas Akhir. Tahap ini diawali dengan pengumpulan literature, diskusi eksplorasi teknologi dan pustaka, serta pemahaman dasar teori yang digunakan pada topik Tugas Akhir. Literatur-literatur yang dimaksud disebutkan sebagai berikut:

1. Tunarungu
2. Bahasa Isyarat
3. Kinect
4. Kinect SDK
5. *Neural Network*
6. *Backpropagation*
7. *Backpropagation-Genetic Algorithm*

B. Perancangan perangkat lunak

Pada tahap ini diawali dengan melakukan analisis awal terhadap permasalahan utama yang muncul pada topik Tugas Akhir. Kemudian dilakukan perancangan perangkat lunak yang meliputi penentuan data yang digunakan an proses-proses yang akan dilaksanakan. Langkah yang digunakan pada tahap ini adalah sebagai berikut:

1. Pendataan bahasa isyarat yang akan digunakan untuk aplikasi
2. Perancangan integrasi aplikasi dengan perangkat Kinect
3. Perancangan ekstraksi fitur Kinect untuk mendapatkan data bahasa isyarat
4. Perancangan proses *training* dan *testing* data.

C. Implementasi dan pembuatan system

Pada tahap ini dilakukan implementasi integrase aplikasi dengan perangkat Kinect. Kemudian dilakukan implementasi ekstraksi fitur Kinect. Akhirnya dilakukan implementasi proses *training* dan *testing* data, Aplikasi ini dibangun menggunakan Microsoft Visual Studio 2015 dan Kinect SDK.

- D. Uji coba dan evaluasi
Pada tahap ini dilakukan uji coba dengan menggunakan gerakan bahasa isyarat yang dilakukan penulis untuk mencoba aplikasi bisa berjalan atau tidak. Uji fungsionalitas untuk mengetahui apakah sudah memenuhi kebutuhan fungsional.
- E. Penyusunan laporan Tugas Akhir
Pada tahap ini dilakukan penyusunan laporan yang berisi dasar teori, dokumentasi dari perangkat lunak, dan hasil-hasil yang diperoleh selama pengerjaan Tugas Akhir.

1.7 Sistematika Penulisan

Buku Tugas Akhir ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut:

BAB I PENDAHULUAN

Bab ini berisi latar belakang masalah, rumusan dan batasan masalah, tujuan dan manfaat pembuatan Tugas Akhir, metodologi yang digunakan, dan sistematika penyusunan Tugas Akhir.

BAB II TINJAUAN PUSTAKA

Bab ini membahas dasar pembuatan dan beberapa teori penunjang yang berhubungan dengan pokok pembahasan yang mendasari pembuatan Tugas Akhir ini.

BAB III ANALISIS DAN PERANCANGAN

Bab ini membahas analisis dari sistem yang dibuat meliputi analisis permasalahan, deskripsi umum perangkat lunak, spesifikasi kebutuhan, dan identifikasi pengguna. Kemudian membahas rancangan dari sistem yang dibuat meliputi rancangan skenario kasus penggunaan, arsitektur, data, dan antarmuka.

BAB IV IMPLEMENTASI

Bab ini membahas implementasi dari rancangan sistem yang dilakukan pada tahap perancangan. Penjelasan implementasi meliputi implementasi antarmuka aplikasi dan pembuatan kebutuhan fungsional aplikasi

BAB V PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dari aplikasi yang dibuat dengan melihat keluaran yang dihasilkan oleh aplikasi dan evaluasi untuk mengetahui kemampuan aplikasi.

BAB VI PENUTUP

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan serta saran untuk pengembangan aplikasi selanjutnya.

BAB II

DASAR TEORI

Pada bab ini akan dibahas mengenai dasar teori yang menjadi dasar pembuatan tugas akhir ini. Pokok permasalahan yang akan di bahas mengenai teknologi yang mendukung dalam pembuatan tugas akhir seperti Kinect, Kinect SDK, *neural network*, algoritma *backpropagation*, algoritma *backpropagation-genetic*, dan pengetahuan umum mengenai bahasa isyarat huruf.

2.1 Tunarungu

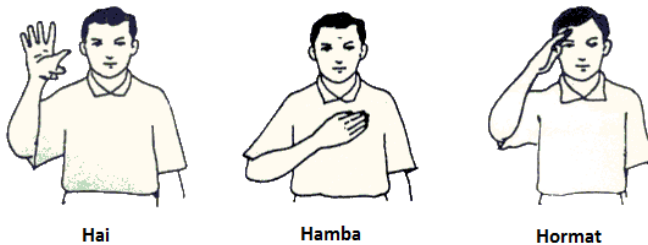
Tunarungu dapat diartikan sebagai keterbatasan yang dimiliki seseorang dalam mendengar sesuatu karena tidak berfungsinya organ pendengaran yang dimiliki. Ketunarunguan dapat dibedakan menjadi dua kategori yaitu tuli (*deaf*) dan kurang dapat mendengar (*low hearing*)[4]. Tuli adalah keadaan di mana organ pendengaran telah mengalami kerusakan yang sangat parah dan mengakibatkan tidak berfungsinya pendengaran. Sedangkan kurang dapat mendengar adalah keadaan di mana organ pendengaran mengalami kerusakan tetapi masih dapat berfungsi untuk mendengar.

2.2 Bahasa Isyarat

Bahasa isyarat adalah sarana komunikasi bagi penderita tunarungu. Bahasa Isyarat berkembang dan memiliki karakteristik sendiri di berbagai negara. Di Indonesia, bahasa isyarat yang digunakan berdasarkan pada Sistem Isyarat Bahasa Indonesia (SIBI)[1]. Ada 4 jenis bahasa isyarat dalam SIBI yaitu:

1. Isyarat Pokok: melambangkan sebuah kata atau konsep
2. Isyarat Tambahan: melambangkan awalan, akhiran, dan partikel (Imbuhan)
3. Isyarat Bentuk: Dibentuk dengan menggabungkan isyarat pokok dan isyarat tambahan.
4. Abjad Jari: dibentuk dengan jari – jari untuk mengeja huruf.

Pada Tugas Akhir ini, isyarat yang akan digunakan adalah isyarat pokok. Gambar 2.1 menunjukkan contoh Bahasa isyarat yang ada didalam SIBI.



Gambar 2.1 Contoh Bahasa Isyarat yang ada didalam SIBI

2.3 Kinect

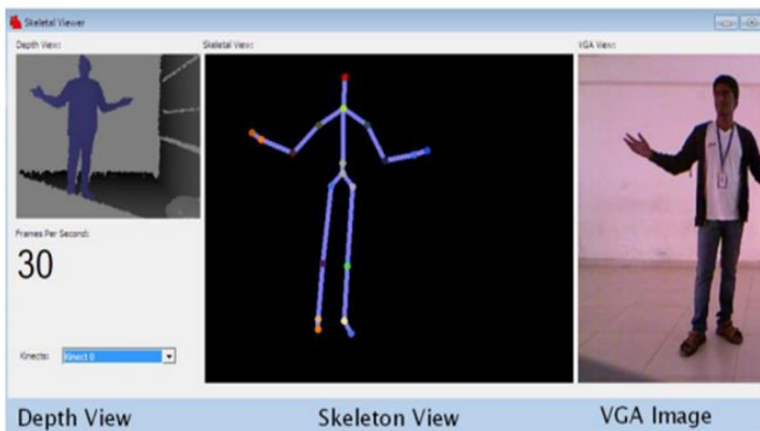
Kinect adalah sebuah perangkat keras buatan Microsoft yang menggantikan input game controller pada Xbox dengan *natural user interface* berupa *gesture* dan perintah suara. *RGB camera*, *3d depth sensing system*, *multi-array microphone*, dan *motorized tilt* merupakan komponen dasar dari Kinect yang dapat dilihat pada Gambar 2.2.

Kinect dapat mendapatkan area sekitarnya dalam 3D dengan mengkombinasikan informasi dari depth sensor dan standard RGB camera. Hasil dari penggabungan tersebut adalah RGBD image dengan resolusi 640x480, dimana setiap *pixel* memiliki *color information* dan *depth information*. Pada bagian kiri Kinect

memiliki *laser infrared light source* yang menghasilkan gelombang elektromagnetik dengan frekuensi 830 nm. Informasi akan dikodekan sebagai light pattern yang akan di “*deformed*” menjadi *light reflects* dari objek di depan Kinect. Berdasarkan proses *deformation* yang didapat oleh sensor di sisi kanan *RGB camera* sebuah *depth map* dibuat seperti pada Gambar 2.3[5].



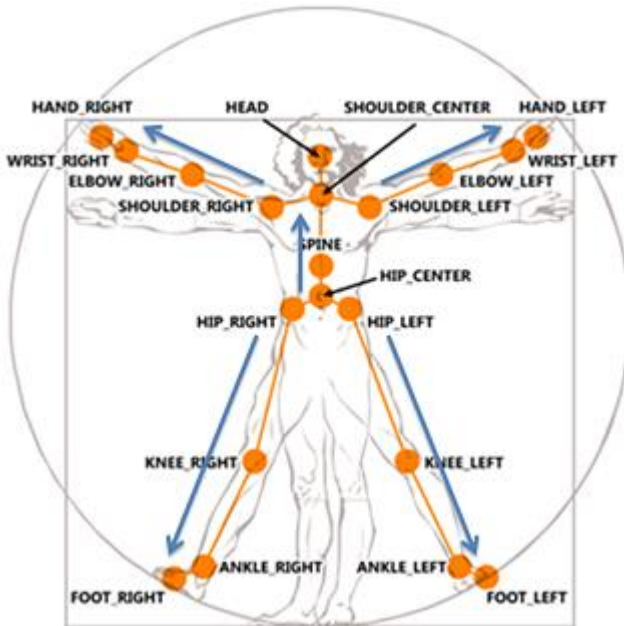
Gambar 2.2 Microsoft Kinect[6]



Gambar 2.3 Depth, Skeleton, dan VGA view[7]

2.4 Kinect SDK

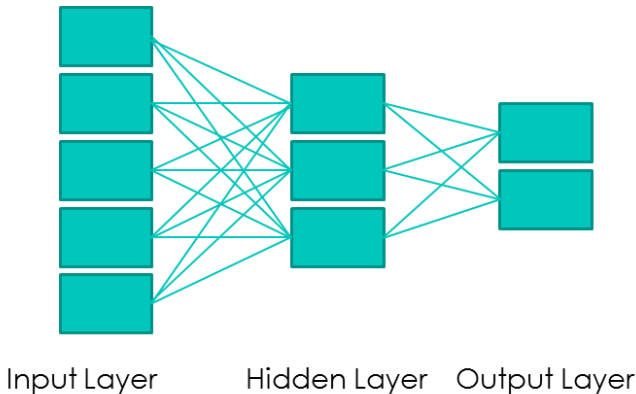
Kinect SDK adalah library yang dibuat oleh Microsoft untuk pengembangan aplikasi perangkat lunak yang menggunakan Kinect sebagai alat input utama. Kinect SDK dapat diimplementasikan dengan bahasa pemrograman C#, C++, dan JavaScript. Library ini memiliki beberapa fitur diantaranya skeleton tracking, Thumb Tracking, end of hand tracking, open/close hand gesture dan lainnya [3]. Pada Kinect SDK yang akan digunakan di Tugas Akhir ini, memiliki 20 skeletoh joint yang dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Skeleton Joints* yang diketahui Kinect

2.5 Neural Network

Neural Network dibentuk dengan 3 *layer*. Setiap *layer* dibentuk oleh beberapa *interconnected nodes* yang memiliki *activation function*. *Input layer* berkomunikasi dengan satu atau lebih *hidden layer* dimana pengolahan yang sebenarnya dilakukan melalui sistem *weighted connections*. *Hidden layer* kemudian terhubung ke *output layer* yang akan menghasilkan output[8]. Contoh dari bentuk *neural network* dengan 1 lapisan *hidden* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Contoh Neural Network Dengan 1 Hidden Layer

2.6 Backpropagation

Backpropagation (BP) adalah algoritma *training* yang menggunakan *forward network* atau biasanya disebut *Multi Layer Perceptron* (MLP). Algoritma BP mencari nilai minimum dari *error function* dalam *weight space* dengan menggunakan *gradient descent*. Kombinasi dari berat yang meminimalisir *error function* dianggap sebagai sebuah solusi dari *learning problem*. Langkah algoritma BP yaitu:

1. Mencari *error* di *node* ke-*j* pada lapisan *output* menggunakan rumus pada Persamaan 2.1. *Target* adalah kelas yang diinginkan.

$$Error_j = Output_j(1 - Output_j)(Target_j - Output_j) \quad (2.1)$$

2. Update bobot pada lapisan *hidden* menggunakan rumus pada Persamaan 2.2. *W_{ij}* adalah bobot di antara lapisan *i* dan *j* (antara lapisan *hidden* dan *output*).

$$W_{ij} = W_{ij} + (Error_j * Output_i) \quad (2.2)$$

3. Mencari *error* di *node* ke-*j* pada lapisan *hidden* menggunakan rumus pada Persamaan 2.3. *Node A* terhubung ke *node B* dan *node C*. Error dari *node B* dan *node C* dibutuhkan untuk menghasilkan *error* pada *node A*.

$$Error_a = Output_a(1 - Output_a)(Error_b W_{ab} + Error_c W_{ac}) \quad (2.3)$$

4. Update bobot pada lapisan *input* menggunakan rumus pada Persamaan 2.2. *W_{ij}* adalah bobot di antara lapisan *i* dan *j* (antara lapisan *input* dan *hidden*)[9].

2.7 Backpropagation-Genetic Algorithm Neural Network

Algoritma *Backpropagation – Genetic Algorithm Neural Network* (BPGANN) merupakan algoritma gabungan dari *Back Propagation* dan *Genetic Algorithm* (GA) dengan menggunakan metode GA untuk mencari bobot inisial dari suatu *neural network*

dan mempercepat konvergensi. Hal ini digunakan karena algoritma BP membuat proses konvergensi menjadi lambat. Langkah-langkah algoritma genetik seperti berikut:

1. Algoritma GA akan mencari kromosom paling unggul yang akan digunakan untuk bobot awal pada *neural network*.
2. Algoritma BP akan melatih *neural network* dengan bobot awal yang sudah ada.
3. Jika performa *neural network* semakin berbeda maka lanjut ke langkah 4, jika tidak maka lakukan langkah 5.
4. *Update* bobot dari *neural network* sebagai populasi awal dari GA dan menggunakannya untuk mencari kromosom paling unggul dari bobot yang dilatih oleh algoritma BP, lalu kembali ke langkah 2.
5. Tentukan apakah kondisi pada saat iterasi memuaskan atau tidak, jika sudah memuaskan maka proses pelatihan berhenti, jika tidak maka kembali ke langkah 2[10].

[Halaman ini sengaja dikosongkan]

BAB III

ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan tugas akhir. Analisis kebutuhan mencantumkan kebutuhan-kebutuhan yang diperlukan perangkat lunak. Selanjutnya dibahas mengenai perancangan sistem yang dibuat.

3.1 Analisis Perangkat Lunak

Bahasa isyarat merupakan sarana komunikasi untuk penderita tunarungu. Walaupun agak sulit untuk mengartikan isyarat yang diberikan, hal tersebut telah membantu penderita tunarungu untuk berkomunikasi dengan sekitar. Namun masih banyak yang belum mengerti apa arti isyarat bagi yang diberi ato memberi isyarat.

Aplikasi ini dibangun dengan tujuan membantu pengguna untuk mempelajari Bahasa isyarat yang ada. Isyarat yang dipakai sesuai dengan SIBI yang nantinya akan didapat dari training data yang didapat. Dengan menggunakan Microsoft Visual Studio dan Kinect, penulis mengekstraksi *skeleton joints* yang ditangkap oleh Kinect kemudian akan dikalkulasi menjadi fitur – fitur yang akan diolah. Kemudian sebuah *classifier* digunakan untuk mengolah fitur yang telah dikalkulasi dan akan menjadi sebuah *Neural Network* yang akan digunakan untuk menentukan arti Bahasa Isyarat yang diberikan.

Untuk tahap testing data tidak jauh berbeda dengan tahap training data. Setelah mengolah fitur yang didapat dari kalkulasi *skeleton joints*, fitur akan dimasukkan sebagai input *Neural Network* dan akan menghasilkan output arti Bahasa Isyarat yang diberikan

3.1.1 Deskripsi Umum Perangkat Lunak

Tugas akhir yang akan dikembangkan adalah sebuah modul pengenalan Bahasa isyarat dengan menggunakan Kinect. Aplikasi ini menggunakan Kinect SDK dan dijalankan dengan perangkat keras Kinect V1.

Pengguna utama adalah semua orang yang ingin belajar Bahasa isyarat. Pengguna dapat mempelajari isyarat yang sudah ada dalam aplikasi ataupun memberikan isyarat baru sesuai dengan SIBI. Jika isyarat yang diberikan tidak ada dalam aplikasi maka tidak ada output yang diberikan.

3.1.2 Spesifikasi Kebutuhan Perangkat Lunak

Kebutuhan sistem yang akan dibuat ini melibatkan dua hal, yakni kebutuhan fungsional maupun kebutuhan non-fungsional. Dimana masing-masing berhubungan dengan keberhasilan dalam pembuatan aplikasi tugas akhir ini.

3.1.2.1 Kebutuhan Fungsional Perangkat Lunak

Pada sistem ini, terdapat beberapa kebutuhan fungsional yang mendukung untuk jalannya aplikasi. Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:.

- a) Mendeteksi *Skeleton* Pengguna
Aplikasi dapat mendeteksi pengguna yang sedang berdiri di depan Kinect .
- b) Mengekstraksi Fitur *Skeleton*
Aplikasi dapat mendeteksi lokasi objek berupa *Skeleton* yang akan diekstraksi menjadi fitur-fitur untuk proses klasifikasi pada saat melakukan *training* dan *testing*.
- c) Menerjemahkan Bahasa Isyarat
Aplikasi dapat menerjemahkan bahasa isyarat yang sesuai dengan fitur-fitur dari skeleton pengguna pada saat melakukan *testing*.

3.1.2.2 Kebutuhan Non-Fungsional

Pada sistem ini, terdapat beberapa kebutuhan non-fungsional yang mendukung dan menambah performa untuk jalannya aplikasi. Fungsi yang terdapat dalam aplikasi ini adalah sebagai berikut:

a) **Penyesuaian Intensitas Cahaya**

Intensitas cahaya merupakan salah satu hal penting yang perlu diperhatikan ketika kita akan menggunakan sebuah sensor. Kinect juga sangat terpengaruh dengan intensitas cahaya. Jika terlalu gelap atau terang, Kinect tidak dapat mendeteksi pengguna. Jika kurang terang, Data skeleton yang diambil Kinect tidak stabil. Maka dari itu, untuk penggunaan aplikasi ini sebaiknya berada di ruangan yang memiliki Intensitas cahaya yang cukup dan usahakan tidak langsung terkena matahari.

b) **Posisi Peletakan Kinect**

Posisi peletakan Kinect disesuaikan dengan pengguna sehingga menghasilkan fitur – fitur yang sempurna. Jarak optimal pengguna dengan Kinect adalah 0.6 meter – 1.8 meter.

3.1.3 Identifikasi Pengguna

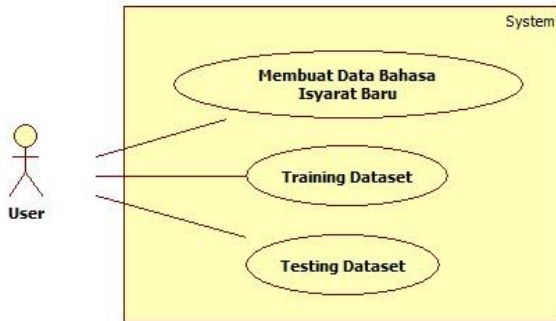
Dalam aplikasi tugas akhir ini, pengguna yang akan terlibat hanya terdapat satu orang saja, yakni orang yang akan melakukan pengenalan bahasa isyarat.

3.2 Perancangan Perangkat Lunak

Subbab ini membahas bagaimana rancangan dari aplikasi tugas akhir ini. Hal yang dibahas meliputi model kasus penggunaan, definisi aktor, definisi kasus penggunaan, arsitektur umum sistem, rancangan antarmuka aplikasi, dan rancangan proses aplikasi.

3.2.1 Model Kasus Penggunaan

Dari hasil analisa deskripsi umum perangkat lunak dan spesifikasi kebutuhan perangkat lunak yang telah dijelaskan, maka model kasus penggunaan untuk aplikasi pengenalan bahasa isyarat pada Gambar 3.1.



Gambar 3.1 Diagram Kasus Penggunaan Aplikasi

3.2.2 Definisi Aktor

Aktor yang terdapat dalam sistem aplikasi ini terlihat pada Tabel 3.1.

Tabel 3.1 Definisi Kasus Penggunaan

No	Nama	Deskripsi
1	Pengguna	Merupakan aktor yang bertugas untuk menambahkan data pelatihan dan melakukan latihan gerakan isyarat statis, seluruh fungsionalitas yang ada di dalam sistem dapat digunakan oleh pengguna.

3.2.3 Definisi Kasus Penggunaan

Pada Gambar 3.1 telah dijelaskan bahwa aktor yang dalam hal ini disebut pengguna mempunyai tiga kasus penggunaan, yakni menambahkan data bahasa isyarat baru, melakukan pelatihan data set dan percobaan *dataset*. Detail mengenai kasus penggunaan tersebut dapat dilihat pada Tabel 3.2.

Tabel 3.2 Definisi Kasus Penggunaan

No	Kode Kasus Penggunaan	Nama Kasus Penggunaan	Keterangan
1	UC-01	Membuat Data Bahasa Isyarat Baru	Pengguna membuat data Bahasa isyarat baru. Dapat berupa Bahasa isyarat baru atau yang sudah ada.
2	UC-02	<i>Training Dataset</i>	Pengguna mengaktifkan mode pelatihan <i>dataset</i>
3	UC-03	<i>Testing Dataset</i>	Pengguna melakukan percobaan <i>dataset</i> dengan melakukan gerakan bahasa isyarat yang ada.

3.2.3.1 Kasus Penggunaan Membuat Data Bahasa Isyarat Baru

Spesifikasi kasus penggunaan membuat data bahasa isyarat baru dapat dilihat pada Tabel 3.3.

Tabel 3.3 Spesifikasi Kasus Penggunaan Membuat Data Bahasa Isyarat Baru

Nama Kasus Penggunaan	Membuat Data Bahasa Isyarat Baru
Nomor	UC-01
Deskripsi	Kasus penggunaan aktor untuk membuat data Bahasa isyarat baru
Aktor	Pengguna
Kondisi Awal	Pengguna sudah menjalankan aplikasi dan perangkat Kinect telah tersambung
Alur Normal	<ol style="list-style-type: none"> 1. User memasukkan nama Bahasa isyarat yang akan dibuat didalam textbox yang sudah disediakan. 2. User menekan tombol kontrol "Create File" 3. Sistem menerima inputan dan menunggu 5 detik untuk perangkat Kinect menemukan skeleton user. 4. User masuk kedalam daerah yang dapat ditangkap Kinect dan mencari lokasi yg tepat sehingga Kinect mendapatkan skeleton user, kemudian melakukan gerakan Bahasa isyarat yang akan dibuat. <ol style="list-style-type: none"> A1. Kinect tidak menemukan skeleton user. 5. Kinect menemukan skeleton user, sistem mengekstrak data skeleton selama 10 detik untuk dikalkulasi.

	A2. Kinect kehilangan skeleton saat proses kalkulasi 6. Sistem menyimpan hasil kalkulasi dalam sebuah file.
Alur Alternatif	A1. Kinect tidak menemukan skeleton user. 1. Sistem memberikan notifikasi skeleton user tidak ditemukan. A2. Kinect kehilangan skeleton saat proses kalkulasi 1. Sistem menghentikan proses kalkulasi 2. Sistem memberikan notifikasi skeleton hilang
Kondisi Akhir	Aplikasi akan membuat data set baru

3.2.3.2 Kasus Penggunaan *Training Dataset*

Spesifikasi kasus penggunaan *training dataset* dapat dilihat pada Tabel 3.4.

Tabel 3.4 Spesifikasi Kasus Penggunaan *Training Dataset*

Nama Kasus Penggunaan	<i>Training Dataset</i>
Nomor	UC-02
Deskripsi	Kasus penggunaan aktor untuk melakukan pelatihan data set
Aktor	Pengguna
Kondisi Awal	pengguna sudah menjalankan aplikasi dan sudah ada <i>dataset</i> yang dibuat aplikasi pada tahap sebelumnya

Alur Normal	<ol style="list-style-type: none"> 1. User menekan tombol kontrol “Train Dataset” yang ada pada aplikasi 2. Sistem memulai tahap training <i>dataset</i> yang sudah ada 3. Sistem membuat file <i>neural network</i> dalam format .xml
Alur Alternatif	-
Kondisi Akhir	Aplikasi membuat file <i>neural network</i> dalam format .xml

3.2.3.3 Kasus *Penggunaan Testing Dataset*

Spesifikasi kasus penggunaan *Testing Dataset* dapat dilihat pada Tabel 3.5.

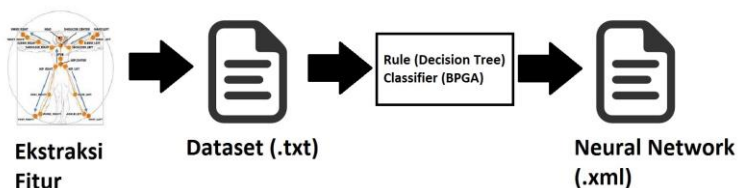
Tabel 3.5 Spesifikasi Kasus Penggunaan *Testing Dataset*

Nama Kasus Penggunaan	<i>Testing Dataset</i>
Nomor	UC-03
Deskripsi	Kasus penggunaan aktor untuk melakukan percobaan <i>dataset</i>
Aktor	Pengguna
Kondisi Awal	pengguna sudah menjalankan aplikasi dan sudah ada <i>dataset</i> yang dibuat aplikasi pada tahap sebelumnya
Alur Normal	<ol style="list-style-type: none"> 1. User menekan tombol kontrol “Start Testing” yang ada pada aplikasi 2. Sistem memulai tahap <i>testing dataset</i> dengan <i>neural network</i> yang sebelumnya dibentuk 3. User melakukan Bahasa isyarat didepan Kinect.

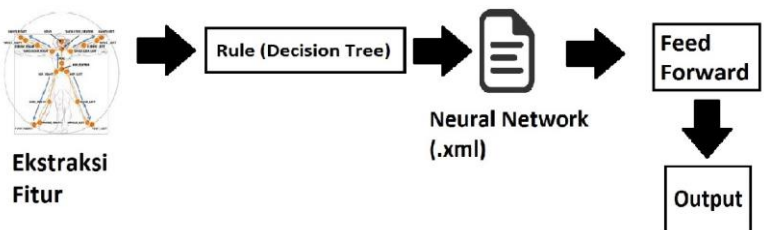
	<ol style="list-style-type: none"> Kinect mengambil data skeleton yang didapat Sistem mengekstrak data skeleton yang kemudian akan dijadikan input dapat <i>neural network</i> Sistem mendapatkan output dari <i>neural network</i> dan kemudian menampilkan.
Alur Alternatif	-
Kondisi Akhir	Aplikasi memberikan output bahasa isyarat

3.2.4 Arsitektur Umum Sistem

Arsitektur sistem pada aplikasi yang akan dibuat dapat dilihat pada Gambar 3.2 dan **Gambar 3.3**.



Gambar 3.2 Arsitektur Sistem (Training)

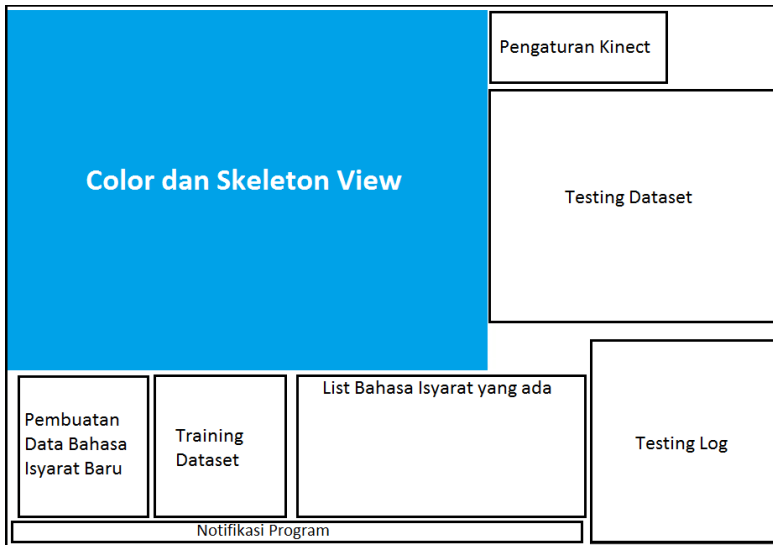


Gambar 3.3 Arsitektur Sistem (Testing)

3.2.5 Rancangan Antarmuka Aplikasi

Rancangan antarmuka aplikasi diperlukan untuk memberikan gambaran umum kepada pengguna bagaimana sistem yang ada dalam aplikasi ini berinteraksi dengan pengguna. Selain itu rancangan ini juga memberikan gambaran bagi pengguna apakah tampilan yang sudah disediakan oleh aplikasi mudah untuk dipahami dan digunakan, sehingga akan muncul kesan *user experience* yang baik dan mudah.

Rancangan antarmuka aplikasi ini hanya memiliki 1 windows dengan beberapa elemen didalamnya seperti *color view* dan *skeleton view* dari Kinect, dan kontrol – kontrol lainnya yang sekiranya bisa dipahami pengguna. Rancangan antarmuka aplikasi dapat dilihat pada Gambar 3.4.



Gambar 3.4 Rancangan Antarmuka Aplikasi

3.2.6 Rancangan Proses Aplikasi

Pada rancangan proses aplikasi akan dijelaskan mengenai proses yang terjadi dalam sistem untuk memenuhi fungsionalitas yang ada pada aplikasi. Proses ini penting agar aplikasi dapat berjalan secara baik dan benar.

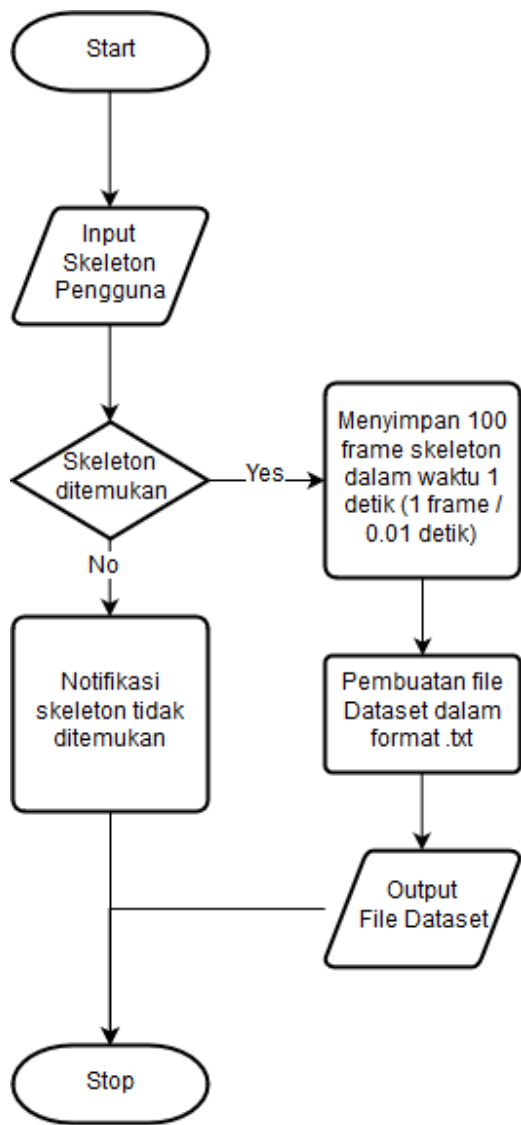
3.2.6.1 Rancangan Proses Ekstraksi Fitur

Proses ekstraksi fitur skeleton sangat dibutuhkan bagi pengguna untuk melakukan *training* maupun *testing*. Saat pembuatan *dataset* baru, aplikasi akan menunggu 5 detik untuk mendeteksi skeleton pengguna, kemudian akan ada 2 detik tambahan untuk melakukan gerakan isyarat. Kemudian aplikasi akan mengambil total 100 data dalam waktu 1 detik yang akan dihitung dan dicari rata – ratanya kemudian disimpan dalam file .txt. Rancangan proses pengambilan fitur skeleton dapat dilihat pada Gambar 3.5.

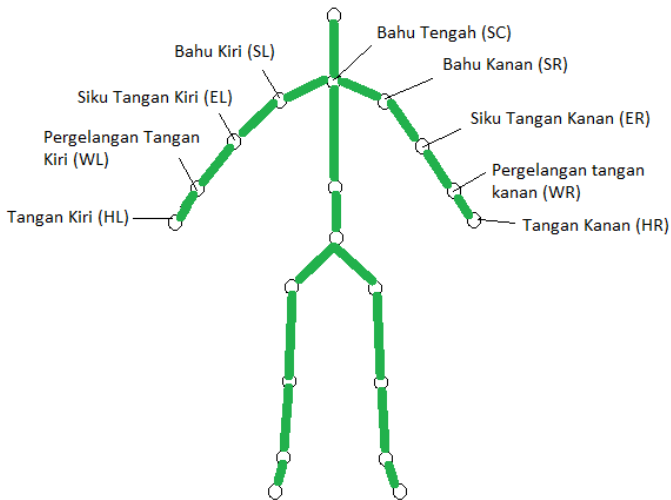
Untuk melakukan ekstraksi skeleton pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 9 *skeleton joints* yang diketahui oleh Kinect yang dapat dilihat pada Gambar 3.6. Sembilan *skeleton joints* tersebut adalah sebagai berikut:

1. Bahu kanan (Vektor SR)
2. Siku tangan kanan (Vektor ER)
3. Pergelangan tangan kanan (Vektor WR)
4. Telapak tangan kanan (Vektor HR)
5. Bahu kiri (Vektor SL)
6. Siku tangan kiri (Vektor EL)
7. Pergelangan tangan kiri (Vektor WL)
8. Telapak tangan kiri (Vektor HL)
9. Bahu tengah (Vektor SC)

Kemudian *skeleton joints* tersebut diolah untuk dijadikan fitur yang berjumlah 28 buah[11]. Dua puluh delapan fitur yang dimaksud dapat dilihat pada Tabel 3.6.



Gambar 3.5. Diagram Alir Proses Ekstraksi Fitur



Gambar 3.6 Skeleton Joints yang Digunakan

Tabel 3.6 Fitur yang Digunakan

Vector3 (x,y,z)	Angle (float)	Distance (float)
ER -> SR	\angle SC - SR - ER	HR - HL
WR -> ER	\angle SR - ER - WR	
HR -> WR	\angle ER - WR - HR	
EL -> SL	\angle SC - SL - EL	
WL -> EL	\angle SL - EL - WL	
HL -> WL	\angle EL - WL - HL	
HL -> HR		

3.2.6.2 Rancangan Proses Pembuatan Rule

Proses pembuatan rule pada aplikasi ini dilakukan dengan membagi bahasa isyarat menjadi 2 golongan sesuai dengan percobaan penulis menggunakan metode *Decision Tree* yang ada pada aplikasi *WEKA* dengan *dataset* yang sebelumnya sudah

dibuat. Dengan melihat level 1 *tree* dari hasil metode *Decision tree* didapatkan rule yang dapat dilihat pada Tabel 3.7.

Tabel 3.7 Rule Pembagian Neural Network

Rule	If	Then	Class
1	$ER.Y - WR.Y < 0.53f$	Gol1.xml	Alquran, Gang, Hamba, Wadah
2	<i>else</i>	Gol2.xml	Bentuk, Hai, Hormat, Ketua

3.2.6.3 Rancangan Proses Normalisasi Fitur

Normalisasi fitur dilakukan dengan menggunakan Persamaan 3.1. Dilakukan proses *feature scalling* untuk membuat fitur dalam *range* tertentu sehingga fitur lebih proporsional. Nilai maksimum dan minimum setiap fitur didapatkan dari hasil pengamatan yang dilakukan oleh penulis. *Pseudocode* untuk normalisasi fitur seperti pada Persamaan 3.1.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}} \quad (3.1)$$

Keterangan:

X = nilai fitur sebelum normalisasi

X' = nilai fitur setelah proses normalisasi

X_{min} = nilai fitur minimum

X_{max} = nilai fitur maksimum.

BAB IV IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan sistem. Bab ini berisi proses implementasi dari setiap kelas pada modul. Namun, pada hasil akhir mungkin saja terjadi perubahan kecil. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C# dengan tambahan menggunakan Kinect SDK.

4.1 Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Lingkungan pembangunan dijelaskan sebagai berikut.

4.1.1 Lingkungan Pembangunan Perangkat Keras

Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah PC dengan spesifikasi sebagai berikut.

- Prosesor Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
- Memori (RAM) 8,00 GB
- Kinect Sensor

4.1.2 Lingkungan Pembangunan Perangkat Lunak

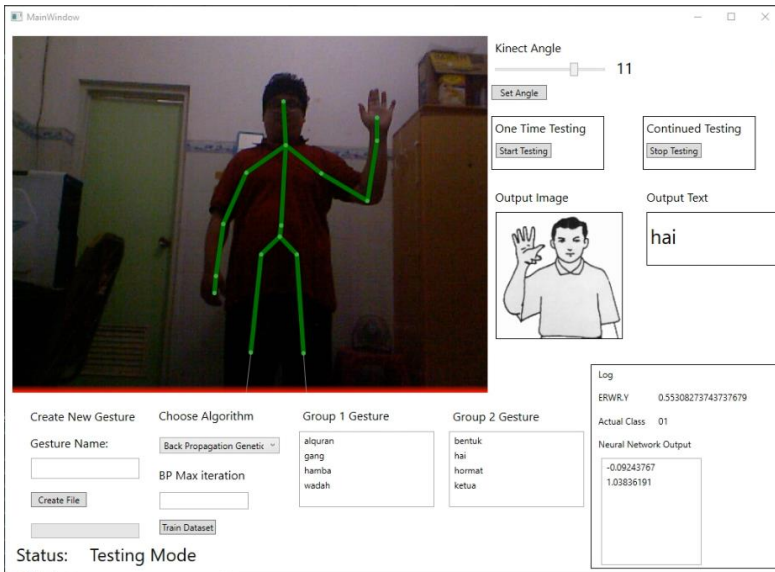
Spesifikasi perangkat lunak yang digunakan untuk membuat aplikasi ini sebagai berikut.

- Microsoft Visual Studio 2015
- Windows 10 64 bit sebagai sistem operasi
- Kinect SDK

4.2 Implementasi Antarmuka

Seperti yang telah dijelaskan pada subbab rancangan antarmuka aplikasi, aplikasi modul pengenalan bahasa isyarat yang

akan dibuat hanya akan memiliki 1 window utama yang sudah mencakup semua fungsionalitas aplikasi yang dibutuhkan. Tampilan antarmuka aplikasi dapat dilihat pada Gambar 4.1.



Gambar 4.1 Antarmuka Aplikasi

4.3 Implementasi Aplikasi

Pada subbab ini akan dibahas mengenai implementasi aplikasi dari kasus penggunaan ke dalam baris kode. Dijelaskan juga dengan fungsi yang dibutuhkan untuk menunjang aplikasi ini agar dapat berjalan sebagaimana mestinya. Implementasi ini dilakukan menggunakan Microsoft Visual Studio 2015 dengan bahasa pemrograman C#.

4.3.1 Implementasi Pendeteksian Skeleton Pengguna

Untuk menjalankan aplikasi ini tentunya membutuhkan perangkat keras Kinect, sehingga dibutuhkan suatu proses untuk mendeteksi *skeleton* pengguna. Sebelum mendeteksi *skeleton*, Kinect harus diintegrasikan dengan program terlebih dahulu.

Kode sumber proses integrasi Kinect dapat dilakukan seperti pada Kode Sumber 4.1. Sementara kode sumber untuk mendeteksi *skeleton* pengguna dapat dilakukan seperti pada Kode Sumber 4.2

```

1      if (KinectSensor.KinectSensors.Count > 0)
2      {
3          StatusDetail.Content = "Kinect Found";
4          mainSensor = KinectSensor.KinectSensors[0];
5
6          if (mainSensor.IsRunning)
7          {
8              StatusDetail.Content = "Reset Kinect";
9              StopKinect(mainSensor);
10         }
11
12         if (mainSensor.Status == KinectStatus.Connected)
13         {
14             StatusDetail.Content = "Setup Kinect";
15
16             Microsoft.Samples.Kinect.WpfViewers.KinectSensorManager
17             kinectManager = new
18             Microsoft.Samples.Kinect.WpfViewers.KinectSensorManager();
19             kinectManager.KinectSensor = mainSensor;
20             kinectManager.ColorStreamEnabled = true;
21
22             kinectManager.SkeletonStreamEnabled = true;
23             kinectManager.DepthStreamEnabled = true;
24             kinectManager.ElevationAngle = KinectDefaultAngle;
25             ColorView.KinectSensorManager =
26             SkeletonView.KinectSensorManager = kinectManager;
27             StatusDetail.Content = "Idle";
28             mainSensor.AllFramesReady +=
29             MainSensor_AllFramesReady;
30             InitClassificationClass();
31         }
32         else StatusDetail.Content = "Kinect Not Found";
33     }
34 }

```

Kode Sumber 4.1 Kode Sumber Integrasi Kinect

```

1 private Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
2 {
3     using (SkeletonFrame skeletonFrameData =
4         e.OpenSkeletonFrame())
5     {
6         if (skeletonFrameData == null)
7             return null;
8
9         skeletonFrameData.CopySkeletonDataTo(allSkeletons);
10
11         //get the first tracked skeleton
12         Skeleton first = (from s in allSkeletons
13                         where s.TrackingState ==
14                             SkeletonTrackingState.Tracked
15                             select s).FirstOrDefault();
16
17         return first;
18     }
19 }

```

Kode Sumber 4.2. Kode Sumber Deteksi *Skeleton* Pengguna

4.3.2 Implementasi Proses Ekstraksi Fitur

Untuk melakukan ekstraksi *skeleton* pengguna dalam melakukan *training* maupun *testing*, penulis mengambil sebanyak 9 *skeleton joints* yang kemudian dijadikan 7 fitur *vector3* dan 14 fitur *double*. Pada Kode Sumber 4.3 dijelaskan mengenai proses ekstraksi fitur pada setiap *frame* dari Kinect. Setiap atribut akan diambil rata-ratanya dari 100 *frame* *skeleton* yang ditangkap oleh Kinect.

```

1 for (int i = 0; i < 100; i++)
2 {
3     if (first == null)
4     {
5         result = false;
6         return;
7     }
8
9     //Right Body
10    SR.SetVector(first.Joints[JointType.ShoulderRight].Position);
11    ER.SetVector(first.Joints[JointType.ElbowRight].Position);
12    WR.SetVector(first.Joints[JointType.WristRight].Position);
13    HR.SetVector(first.Joints[JointType.HandRight].Position);

```

```

14      //Left Body
15      SL.SetVector(first.Joints[JointType.ShoulderLeft].Position)
16      ;
17      EL.SetVector(first.Joints[JointType.ElbowLeft].Position);
18      WL.SetVector(first.Joints[JointType.WristLeft].Position);
19      HL.SetVector(first.Joints[JointType.HandLeft].Position);
20      //Center Body
21      SC.SetVector(first.Joints[JointType.ShoulderCenter].Position);
22      n);
23
24      SRER += ER - SR;
25      ERWR += WR - ER;
26      WRHR += HR - WR;
27      SLEL += EL - SL;
28      ELWL += WL - EL;
29      WLHL += HL - WL;
30      HRHL += HL - HR;
31
32      Vector3 v1, v2;
33      double res;
34
35      //SC-SR-ER
36      v1 = SC - SR;
37      v1 = v1.Normalize();
38      v2 = ER - SR;
39      v2 = v2.Normalize();
40      res = Vector3.DotProduct(v1, v2);
41      SCSRER += (double)Math.Acos(res);
42
43      //SR-ER-WR
44      v1 = SR - ER;
45      v1 = v1.Normalize();
46      v2 = WR - ER;
47      v2 = v2.Normalize();
48      res = Vector3.DotProduct(v1, v2);
49      SRERWR += (double)Math.Acos(res);
50
51      //ER-WR-HR
52      v1 = ER - WR;
53      v1 = v1.Normalize();
54      v2 = HR - WR;
55      v2 = v2.Normalize();
56      res = Vector3.DotProduct(v1, v2);
57      ERWRHR += (double)Math.Acos(res);
58
59      //SC-SL-EL
60      v1 = SC - SL;
61      v1 = v1.Normalize();
62      v2 = EL - SL;

```

```

63         v2 = v2.Normalize();
64         res = Vector3.DotProduct(v1, v2);
65         SCSLEL += (double)Math.Acos(res);
66
67         //SL-EL-WL
68         v1 = SL - EL;
69         v1 = v1.Normalize();
70         v2 = WL - EL;
71         v2 = v2.Normalize();
72         res = Vector3.DotProduct(v1, v2);
73         SLELWL += (double)Math.Acos(res);
74
75         //EL-WL-HL
76         v1 = EL - WL;
77         v1 = v1.Normalize();
78         v2 = HL - WL;
79         v2 = v2.Normalize();
80         res = Vector3.DotProduct(v1, v2);
81         ELWLHL += (double)Math.Acos(res);
82
83         //Distance HR - HL
84         DisHRHL += Vector3.Distance(HR, HL);
85         Thread.Sleep(10);
86     }
87     SRER /= 100;
88     ERWR /= 100;
89     WRHR /= 100;
90     SLEL /= 100;
91     ELWL /= 100;
92     WLHL /= 100;
93     HRHL /= 100;
94     SCSRER /= 100;
95     SRERWR /= 100;
96     ERWRHR /= 100;
97     SCSLEL /= 100;
98     SLELWL /= 100;
99     ELWLHL /= 100;
100    DisHRHL /= 100;
101    AllFeatureNormalize();

```

Kode Sumber 4.3 Kode Sumber Ekstraksi Fitur *Skeleton*

4.3.3 Implementasi Pembuatan Data Bahasa Isyarat Baru

Dalam implementasi pembuatan data bahasa isyarat baru, hasil ekstraksi fitur akan dibagi menjadi 2 golongan sesuai dengan

syarat pembagian *neural network* pada Tabel 3.7. Implementasi dapat dilihat seperti pada Kode Sumber 4.4

1	StringBuilder sb = new StringBuilder();
2	
3	sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", SRER.X.ToString("0.00000"), System.Environment.NewLine, SRER.Y.ToString("0.00000"), System.Environment.NewLine, SRER.Z.ToString("0.00000")));
4	sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", ERWR.X.ToString("0.00000"), System.Environment.NewLine, ERWR.Y.ToString("0.00000"), System.Environment.NewLine, ERWR.Z.ToString("0.00000")));
5	sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", WRHR.X.ToString("0.00000"), System.Environment.NewLine, WRHR.Y.ToString("0.00000"), System.Environment.NewLine, WRHR.Z.ToString("0.00000")));
6	sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", SLEL.X.ToString("0.00000"), System.Environment.NewLine, SLEL.Y.ToString("0.00000"), System.Environment.NewLine, SLEL.Z.ToString("0.00000")));
7	sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", ELWL.X.ToString("0.00000"), System.Environment.NewLine, ELWL.Y.ToString("0.00000"), System.Environment.NewLine, ELWL.Z.ToString("0.00000")));
8	sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", WLHL.X.ToString("0.00000"), System.Environment.NewLine, WLHL.Y.ToString("0.00000"), System.Environment.NewLine, WLHL.Z.ToString("0.00000")));
9	sb.AppendLine(String.Format("{0}{1}{2}{3}{4}", HRHL.X.ToString("0.00000"), System.Environment.NewLine, HRHL.Y.ToString("0.00000"), System.Environment.NewLine, HRHL.Z.ToString("0.00000")));
10	sb.AppendLine(String.Format("{0}", SCSRER.ToString("0.00000")));
11	sb.AppendLine(String.Format("{0}", SRERWR.ToString("0.00000")));
12	sb.AppendLine(String.Format("{0}", ERWRHR.ToString("0.00000")));
13	sb.AppendLine(String.Format("{0}", SCSLEL.ToString("0.00000")));
14	sb.AppendLine(String.Format("{0}", SLELWL.ToString("0.00000")));
15	sb.AppendLine(String.Format("{0}", ELWLHL.ToString("0.00000")));
16	sb.Append(String.Format("{0}", DisHRHL.ToString("0.00000")));
17	int gol = 0;

```

18  if (ERWR.Y < 0.53)
19      gol = 1;
20  else
21      gol = 2;
22
23  result = true;
24
25  int count = 0;
26  string fullPath;
27  if (gol == 1)
28  {
29      filename = string.Format("gol1.{0}", filename);
30      fullPath = @path + "gol1-normalize\\" + filename;
31  }
32  else
33  {
34      filename = string.Format("gol2.{0}", filename);
35      fullPath = @path + "gol2-normalize\\" + filename;
36  }
37  while (File.Exists(fullPath))
38  {
39      string[] temp = filename.Split('.');
40      count++;
41      if (gol == 1)
42          filename = String.Format("gol1.{0}.{1}.txt",
temp[1], count.ToString());
43      else
44          filename = String.Format("gol2.{0}.{1}.txt",
temp[1], count.ToString());
45
46      if (gol == 1)
47          fullPath = @path + "gol1-normalize\\" + filename;
48      else
49          fullPath = @path + "gol2-normalize\\" + filename;
50  }
51  File.AppendAllText(fullPath, sb.ToString());

```

Kode Sumber 4.4 Implementasi Pembuatan Data Bahasa Isyarat Baru

4.3.4 Implementasi Proses *Training* dan *Testing*

Pada proses *Training* dan *Testing*, penulis menggunakan *classifier* dengan metode *Back Propagation-Genetic Algorithm* yang pada Tugas Akhir sebelumnya sudah diimplementasikan oleh Risal Andika Tidisaputra. Dengan melakukan beberapa modifikasi kode, aplikasi ini dapat menggunakannya. Untuk proses *training*,

Implementasi dapat dilihat pada Kode Sumber 4.5. Output dari proses *training* adalah sebuah *neural network* dalam format .xml yang nantinya akan digunakan untuk proses *testing*. Implementasi proses *Testing* dapat dilihat pada Kode Sumber 4.6.

1	float avgError1 = 0;
2	float avgError2 = 0;
3	NeuralNetwork nn1 = new NeuralNetwork();
4	NeuralNetwork nn2 = new NeuralNetwork();
5	DirectoryInfo info1 = new DirectoryInfo(string.Format("{0}gol1-normalize\\", path));
6	DirectoryInfo info2 = new DirectoryInfo(string.Format("{0}gol2-normalize\\", path));
7	DataSetList dsl1 = new DataSetList();
8	DataSetList dsl2 = new DataSetList();
9	FileReader fr1 = new FileReader();
10	FileReader fr2 = new FileReader();
11	dsl1 = fr1.ReadFile(info1.FullName, cc1);
12	dsl2 = fr2.ReadFile(info2.FullName, cc2);
13	StatusDetail.Content = "Training Dataset";
14	if (algorithm.Equals("BP"))
15	{
16	nn1.InitNetwork(dsl1[0].AttributeCount, dsl1[0].AttributeCount / 2, cc1.TargetCount);
17	nn2.InitNetwork(dsl2[0].AttributeCount, dsl2[0].AttributeCount / 2, cc2.TargetCount);
18	nn1.Seed = nn2.Seed = 0;
19	nn1.InitWeight();
20	nn2.InitWeight();
21	BackPropagation bp1 = new BackPropagation();
22	BackPropagation bp2 = new BackPropagation();
23	bp1.Init(nn1, dsl1, cc1);
24	bp2.Init(nn2, dsl2, cc2);
25	avgError1 = bp1.Run(Int32.Parse(iteration_text.Text));
26	avgError2 = bp2.Run(Int32.Parse(iteration_text.Text));
27	}
28	else if(algorithm.Equals("BPGA"))
29	{
30	GeneticAlgorithm ga1 = new GeneticAlgorithm();
31	GeneticAlgorithm ga2 = new GeneticAlgorithm();
32	ga1.Init(dsl1, cc1);
33	ga2.Init(dsl2, cc2);
34	
35	//Gunakan ga.Run() jika sebelumnya tidak memiliki
36	chromosom yang bagus
37	//Chromosom fitChrom1 = ga1.Run();
38	//Chromosom fitChrom2 = ga2.Run();

```

39
40     //Gunakan chromosom yang sudah ada jika sudah memiliki
41     chromosom yang bagus
42     loadNet = new NeuralNetwork();
43     NNtoXMLReader nnr = new NNtoXMLReader(loadNet);
44     algoTest = nnr.read("gol1ga.xml");
45     seleksi1 = nnr.chromosom;
46
47     loadNet = new NeuralNetwork();
48     nnr = new NNtoXMLReader(loadNet);
49     algoTest = nnr.read("gol2ga.xml");
50     seleksi2 = nnr.chromosom;
51
52     for(int i = 0; i < dsl1.Count; i++)
53     {
54         int popCount = 0;
55         for(int j = 0; j < seleksi1.Count(); j++)
56         {
57             if(seleksi1[j] == 0)
58             {
59                 dsl1[i].RemoveBit(j - popCount);
60                 popCount++;
61             }
62         }
63     }
64
65     for (int i = 0; i < dsl2.Count; i++)
66     {
67         int popCount = 0;
68         for (int j = 0; j < seleksi2.Count(); j++)
69         {
70             if (seleksi2[j] == 0)
71             {
72                 dsl2[i].RemoveBit(j - popCount);
73                 popCount++;
74             }
75         }
76     }
77
78     nn1.InitNetwork(dsl1[0].AttributeCount,
79     dsl1[0].AttributeCount / 2, ccl.TargetCount);
80     nn2.InitNetwork(dsl2[0].AttributeCount,
81     dsl2[0].AttributeCount / 2, cc2.TargetCount);
82     nn1.Seed = 0;
83     nn2.Seed = 0;
84     nn1.InitWeight();
85     nn2.InitWeight();
86     BackPropagation bp1 = new BackPropagation();
87     BackPropagation bp2 = new BackPropagation();

```

86	bp1.Init(nn1, dsl1, cc1);
87	bp2.Init(nn2, dsl2, cc2);
88	avgError1 = bp1.Run(Int32.Parse(iteration_text.Text));
89	avgError2 = bp2.Run(Int32.Parse(iteration_text.Text));
90	}
91	
92	NNtoXMLWriter nnw1 = new NNtoXMLWriter(nn1, avgError1);
93	NNtoXMLWriter nnw2 = new NNtoXMLWriter(nn2, avgError2);
94	if (algorithm.Equals("BP"))
95	{
96	nnw1.Write("gol1.xml", algorithm, null);
97	nnw2.Write("gol2.xml", algorithm, null);
98	}
99	else if(algorithm.Equals("BPGA"))
100	{
101	nnw1.Write("gol1ga.xml", algorithm, seleksi1);
102	nnw2.Write("gol2ga.xml", algorithm, seleksi2);
103	}

Kode Sumber 4.5 Implementasi Proses *Traning*

1	int gol;
2	DataSetList dsl = new DataSetList();
3	List<float> fitur = new List<float>();
4	string xmlName = "";
5	fitur.Add((float)SRER.X);
6	fitur.Add((float)SRER.Y);
7	fitur.Add((float)SRER.Z);
8	fitur.Add((float)ERWR.X);
9	fitur.Add((float)ERWR.Y);
10	fitur.Add((float)ERWR.Z);
11	fitur.Add((float)WRHR.X);
12	fitur.Add((float)WRHR.Y);
13	fitur.Add((float)WRHR.Z);
14	fitur.Add((float)SLEL.X);
15	fitur.Add((float)SLEL.Y);
16	fitur.Add((float)SLEL.Z);
17	fitur.Add((float)ELWL.X);
18	fitur.Add((float)ELWL.Y);
19	fitur.Add((float)ELWL.Z);
20	fitur.Add((float)WLHL.X);
21	fitur.Add((float)WLHL.Y);
22	fitur.Add((float)WLHL.Z);
23	fitur.Add((float)HRHL.X);
24	fitur.Add((float)HRHL.Y);
25	fitur.Add((float)HRHL.Z);
26	fitur.Add((float)SCSRER);
27	fitur.Add((float)SRERWR);
28	fitur.Add((float)ERWRHR);

```

29  fitur.Add((float)SCSLEL);
30  fitur.Add((float)SLELWL);
31  fitur.Add((float)ELWLHL);
32  fitur.Add((float)DisHRHL);
33  if (ERWR.Y < 0.53)
34  {
35      if (algorithm.Equals("BP"))
36          xmlName = "gol1.xml";
37      else if (algorithm.Equals("BPGA"))
38          xmlName = "gol1ga.xml";
39      gol = 1;
40  }
41  else
42  {
43      if (algorithm.Equals("BP"))
44          xmlName = "gol2.xml";
45      else if (algorithm.Equals("BPGA"))
46          xmlName = "gol2ga.xml";
47      gol = 2;
48  }
49  DataSet ds = new DataSet(fitur.Count);
50  for (int i = 0; i < fitur.Count; i++)
51  {
52      ds[i] = fitur[i];
53  }
54
55  dsl.Add(ds);
56
57  FeedForward ff = new FeedForward();
58  loadNet = new NeuralNetwork();
59  NNtoXMLReader nnr = new NNtoXMLReader(loadNet);
60  Translate t = new Translate();
61
62  algoTest = nnr.read(xmlName);
63  for (int i = 0; i < dsl.Count; i++)
64  {
65      if (algoTest.Equals("BPGA"))
66      {
67          int popCount = 0;
68          for (int j = 0; j < nnr.chromosom.Length; j++)
69          {
70              if (nnr.chromosom[j] == 0)
71              {
72                  dsl[i].RemoveBit(j - popCount);
73                  popCount++;
74              }
75          }
76      }
77  }

```

```

78
79     loadNet = new NeuralNetwork();
80     nnr = new NNtoXMLReader(loadNet);
81     nnr.read(xmlName);
82
83     ff = new FeedForward();
84     ff.Init(loadNet, dsl);
85     for (int i = 0; i < dsl.Count; i++)
86         ff.Run(i);
87
88     int[] actualClass = new int[loadNet.OutputLayer.Count];
89     actualClass = ff.GetActualClass();
90
91     t = new Translate(actualClass);
92     string actualClassLog = "";
93     for (int i = 0; i < actualClass.Length; i++)
94         actualClassLog += actualClass[i];
95
96     if (gol == 1)
97         outputText.Content = t.Result(cc1);
98     else if (gol == 2)
99         outputText.Content = t.Result(cc2);
100    string imageFullPath = @imagePath + outputText.Content +
        ".bmp";
101    if (File.Exists(imageFullPath))
        outputImage.Source = (ImageSource)new
102        ImageSourceConverter().ConvertFrom(imageFullPath);
103    Log_ERWR.Y.Content = ERWR.Y.ToString();
104    outputList.Items.Clear();
105
106    for (int i = 0; i < loadNet.OutputLayer.Count(); i++)
107    {
108        outputList.Items.Add(loadNet.OutputLayer[i].Input);
109    }

```

Kode Sumber 4.6 Implementasi Proses Testing

4.3.4 Implementasi Proses Normalisasi Fitur

Proses normalisasi fitur mengimplementasikan *pseudocode* pada Persamaan 3.1, di mana setiap fitur pengguna dibagi dengan nilai maksimum. Nilai maksimum didapatkan penulis pada saat melakukan pengamatan pada setiap fitur. Normalisasi ini dilakukan supaya *training neural network* berjalan dengan cepat. Setiap fitur bernilai antara 0 sampai 1. Implementasi dapat dilihat pada Kode

Sumber 4.7. Implementasi fungsi FeatureNorm() dapat dilihat pada Kode Sumber 4.8.

4.3.6 Implementasi Proses Pembagian *Neural Network*

Proses pembagian *neural network* pada aplikasi ini dilakukan dengan membagi bahasa isyarat yang ada menjadi 2 golongan. Untuk golongan pertama ditentukan dengan melihat nilai Y pada *vector* ER – WR. Jika Y kurang dari 0.53 maka bahasa isyarat termasuk golongan 1. Selain itu bahasa isyarat akan masuk golongan 2. Implementasi pembagian *Neural Network* dapat dilihat pada Kode Sumber 4.9.

1	private void AllFeatureNormalize()
2	{
3	SRER = Features.FeatureNorm(SRER);
4	ERWR = Features.FeatureNorm(ERWR);
5	WRHR = Features.FeatureNorm(WRHR);
6	SLEL = Features.FeatureNorm(SLEL);
7	ELWL = Features.FeatureNorm(ELWL);
8	WLHL = Features.FeatureNorm(WLHL);
9	HRHL = Features.FeatureNorm(HRHL);
10	SCSRER /= Math.PI;
11	SRERWR /= Math.PI;
12	ERWRHR /= Math.PI;
13	SCSLEL /= Math.PI;
14	SLELWL /= Math.PI;
15	ELWLHL /= Math.PI;
16	DisHRHL /= 3.4641f;
17	}

Kode Sumber 4.7 Fungsi Normalisasi Fitur

1	public class Features
2	{
3	public static double Normalize(double x)
4	{
5	return (x + 2) / 4;
6	}
7	
8	public static Vector3 FeatureNorm(Vector3 u)
9	{

10	return new Vector3(Normalize(u.X), Normalize(u.Y),
11	Normalize(u.Z));
12	}

Kode Sumber 4.8 Implementasi fungsi FeatureNorm() pada class Features

1	int gol = 0;
2	if (ERWR.Y < 0.53)
3	gol = 1;
4	else
5	gol = 2;

Kode Sumber 4.9 Implementasi Pembagian *Neural Network*

[Halaman ini sengaja dikosongkan]

BAB V

PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada aplikasi yang dikembangkan. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsional secara keseluruhan. Pengujian dilakukan dengan beberapa skenario. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1 Lingkungan Pembangunan

Dalam membangun aplikasi ini digunakan beberapa perangkat pendukung baik perangkat keras maupun perangkat lunak. Perangkat keras yang digunakan dalam pembuatan aplikasi ini adalah sebuah PC yang memiliki spesifikasi sebagai berikut.

- Prosesor Intel(R) Core(TM) i7-2600K CPU @ 3.40GHz
- Memori (RAM) 8,00 GB
- Kinect

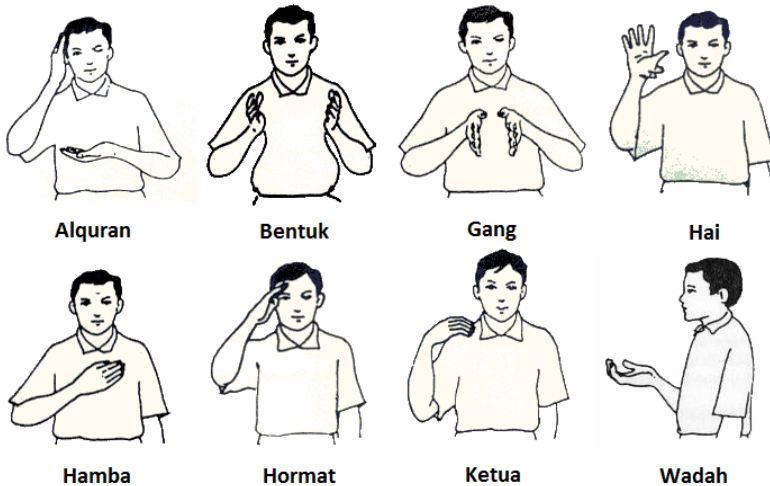
5.2 Skenario Pengujian

Pengujian dilakukan terhadap 8 bahasa isyarat yang sudah disediakan. Ke 8 bahasa isyarat tersebut adalah sebagai berikut:

1. Alquran
2. Bentuk
3. Gang
4. Hai
5. Hamba
6. Hormat
7. Ketua
8. Wadah

Dari ke 8 bahasa isyarat tersebut, dibagi menjadi 2 golongan. Golongan pertama berisikan Alquran, Gang, Hamba, dan Wadah. Sedangkan golongan kedua berisikan Bentuk, Hai, Hormat, dan

Ketua. Gerakan yang dilakukan oleh 8 bahasa isyarat diatas dapat dilihat pada Gambar 5.1. Pembagian golongan tersebut berdasarkan ketentuan pada Tabel 3.7.



Gambar 5.1 Bahasa Isyarat yang Dipakai

Skenario pengujian yang dilakukan dibagi menjadi 2 skenario A dan skenario B. Pada skenario A, *neural network* yang digunakan pada saat *testing* adalah hasil *training* dengan data *skeleton* penulis. Sedangkan skenario B, *neural network* yang digunakan pada saat *testing* adalah hasil *training* dengan data *skeleton* penulis dan satu pengguna.

1. Pengujian skenario A1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh penulis dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis.
2. Pengujian skenario A2 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh pengguna dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis.

3. Pengujian skenario B1 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh penulis dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis dan 80 *dataset* yang diambil dari data *skeleton* pengguna.
4. Pengujian skenario B2 merupakan pengujian akurasi menggunakan algoritma GABP terhadap 8 bahasa isyarat kata yang dilakukan oleh pengguna dengan menggunakan 160 *dataset* yang diambil dari data *skeleton* penulis dan 80 *dataset* yang diambil dari data *skeleton* pengguna.

5.2.1 Pengujian Skenario A1 dan Analisis

Pada pengujian skenario A1 uji coba dilakukan oleh penulis. Uji coba ini menggunakan *dataset* sebanyak 160 buah, semua *dataset* tersebut dilatih dari *skeleton* penulis. Skenario dapat dilihat pada Tabel 5.1.

Tabel 5.1 Skenario Pengujian A1

Nama Skenario Pengujian	Pengujian Akurasi A1
Kode	SP-A1
Algoritma	<i>Backpropagation-Genetic Algorithm</i>
Jumlah Dataset	160 <i>dataset</i>
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 95%

Hasil yang didapat cukup memuaskan. Kesalahan output terjadi pada bahasa isyarat Hamba dan Hormat. Menurut pengamatan penulis, kesalahan tersebut terjadi karena posisi *skeleton* yang kurang tepat dan ketidakstabilan Kinect dalam

mengambil data *skeleton*. Hasil dari skenario pengujian A1 dapat dilihat pada Tabel 5.2.

Tabel 5.2 Terjemahan Isyarat Kata Skenario Pengujian A1

Target Kelas									
Hasil Uji Coba		Alquran	Bentuk	Gang	Hai	Hamba	Hormat	Ketua	Wadah
	Alquran	5							
	Bentuk		5						
	Gang			5					
	Hai				5				
	Hamba					4			
	Hormat						4		
	Ketua						1	5	
	Wadah					1			5

5.2.2 Pengujian Skenario A2 dan Analisis

Pada pengujian skenario A2 uji coba dilakukan oleh pengguna. Uji coba ini menggunakan *dataset* sebanyak 160 buah, semua *dataset* tersebut dilatih dari *skeleton* penulis. Skenario dapat dilihat pada Tabel 5.3.

Tabel 5.3 Skenario Pengujian A2

Nama Skenario Pengujian	Pengujian Akurasi A2
Kode	SP-A2
Algoritma	<i>Backpropagation-Genetic Algorithm</i>
Jumlah Dataset	160 <i>dataset</i>
Penguji	Pengguna
Prosedur Pengujian	Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 82.5%

Hasil yang didapatkan lebih rendah dari hasil pengujian skenario A1. Kesalahan output terjadi pada isyarat Alquran, Bentuk, Hai, dan Hamba. Pada isyarat Alquran ketika diterjemahkan mengeluarkan output isyarat Ketua. Isyarat Alquran dan Ketua berada pada golongan yang berbeda. Ketika melakukan perhitungan fitur dengan gerakan isyarat Alquran, nilai yang dihasilkan oleh fitur $ER.Y-WR.Y$ tidak sesuai sehingga *dataset* yang didapat diterjemahkan dengan *neural network* yang tidak sesuai. Hal yang sama terjadi pada isyarat Ketua. Kemudian pada isyarat Hai terjadi kesalahan penerjemahan menjadi isyarat Bentuk. Hal ini terjadi karena gerakan isyarat Hai yang dibuat tidak sesuai dengan *dataset* yang ada sehingga output yang dihasilkan *neural network* adalah isyarat Bentuk. Hal yang sama terjadi pada isyarat Hambah yang mengalami kesalahan output menjadi isyarat Wadah. Hasil dari skenario pengujian A2 dapat dilihat pada Tabel 5.4.

Tabel 5.4 Terjemahan Isyarat Kata Skenario Pengujian A2

		Target Kelas							
Hasil Uji Coba		Alquran	Bentuk	Gang	Hai	Hamba	Hormat	Ketua	Wadah
	Alquran	4							
	Bentuk		2		2				
	Gang		3	5					
	Hai				3				
	Hamba					4			
	Hormat						5		
	Ketua	1						5	
	Wadah					1			5

5.2.3 Pengujian Skenario B1 dan Analisis

Pada pengujian skenario B1 uji coba dilakukan oleh penulis. Uji coba ini menggunakan *dataset* sebanyak 240 buah, 160 *dataset* dilatih dari *skeleton* penulis dan 80 *dataset* dilatih dari *skeleton* pengguna. Skenario dapat dilihat pada Tabel 5.5.

Hasil yang didapat meningkat 2.5% dari skenario pengujian A1. Kesalahan output terjadi pada bahasa isyarat Hai. Kesalahan yang terjadi karena posisi *skeleton* yang kurang tepat sehingga terjadi perbedaan saat perhitungan. Hasil dari skenario pengujian B1 dapat dilihat pada Tabel 5.6.

Tabel 5.5 Skenario Pengujian B1

Nama Skenario Pengujian	Pengujian Akurasi B1
Kode	SP-B1
Algoritma	<i>Backpropagation-Genetic Algorithm</i>
Jumlah Dataset	240 Dataset
Penguji	Penulis
Prosedur Pengujian	Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 97.5%

Tabel 5.6 Terjemahan Isyarat Kata Skenario Pengujian B1

		Target Kelas							
Hasil Uji Coba		Alquran	Bentuk	Gang	Hai	Hamba	Hormat	Ketua	Wadah
	Alquran	5							
	Bentuk		5						
	Gang			5					
	Hai				4				
	Hamba					5			
	Hormat						5		
	Ketua				1			5	
	Wadah								5

5.2.4 Pengujian Skenario B2 dan Analisis

Pada pengujian skenario B2 uji coba dilakukan oleh pengguna. Uji coba ini menggunakan *dataset* sebanyak 240 buah,

160 *dataset* dilatih dari *skeleton* penulis dan 80 *dataset* dilatih dari *skeleton* pengguna. Skenario dapat dilihat pada Tabel 5.7.

Hasil yang didapat setelah penambahan *dataset* pengguna meningkat cukup signifikan. Hal ini terjadi karena saat proses pelatihan memakai *dataset* yang lebih bervariasi. Kesalahan output yang terjadi pada skenario pengujian B2 disebabkan karena kurang tepatnya posisi *skeleton* saat pengambilan data. Hasil skenario pengujian B2 dapat dilihat pada Tabel 5.8.

Tabel 5.7 Skenario Pengujian B2

Nama Skenario Pengujian	Pengujian Akurasi B2
Kode	SP-B2
Algoritma	<i>Backpropagation-Genetic Algorithm</i>
Jumlah Dataset	240 Dataset
Penguji	Pengguna
Prosedur Pengujian	Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 95%

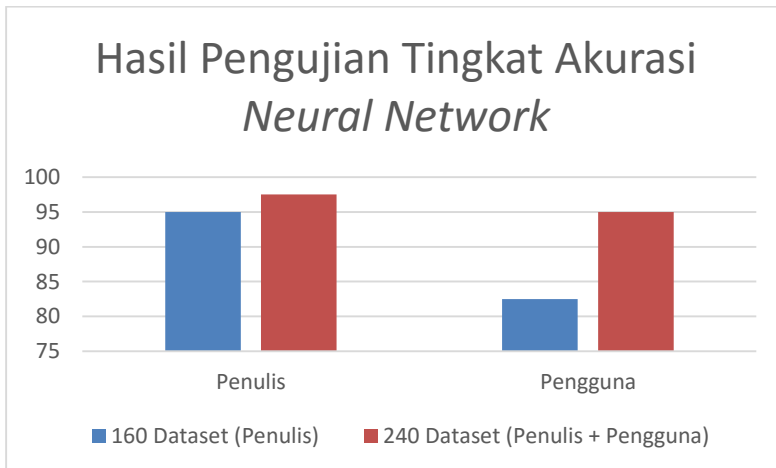
Tabel 5.8 Terjemahan Isyarat Kata Skenario Pengujian B2

		Target Kelas							
Hasil Uji Coba		Alquran	Bentuk	Gang	Hai	Hamba	Hormat	Ketua	Wadah
	Alquran	5							
	Bentuk		5						
	Gang			5					
	Hai				4				
	Hamba					4			
	Hormat				1		5		
	Ketua							5	
	Wadah					1			5

5.3 Evaluasi

Subbab ini membahas mengenai evaluasi terhadap pengujian-pengujian yang telah dilakukan. Dalam hal ini evaluasi menunjukkan data akurasi dari hasil pengujian akurasi *neural network* yang telah dilakukan sebelumnya.

Evaluasi pengujian akurasi *neural network* dilakukan dengan menampilkan data hasil pengujian yang telah dipaparkan pada subbab 5.2. dalam hal ini, pengujian disusun dalam bentuk grafik yang dapat dilihat pada Gambar 5.2. Dari data yang terdapat pada grafik tersebut, diketahui bahwa aplikasi yang dibuat telah memiliki akurasi rata – rata yang cukup baik yaitu 92.5%.



Gambar 5.2 Hasil Pengujian Tingkat Akurasi *Neural Network*

BAB VI

KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari proses pengerjaan selama perancangan, implementasi, dan proses pengujian aplikasi yang dilakukan, dapat diambil kesimpulan sebagai berikut.

1. Untuk menentukan fitur yang bisa digunakan untuk mendeteksi Bahasa isyarat yang diberikan, penulis menggunakan data *skeleton joints* yang diketahui oleh Kinect dengan menggunakan *skeleton* pengguna aplikasi. Untuk mendapatkan *skeleton* pengguna dengan Kinect, pengguna harus memperhatikan intensitas cahaya yang ada di ruangan karena intensitas cahaya sangat mempengaruhi kestabilan Kinect dalam memperoleh data *skeleton joints*.
2. Metode *Back Propagation Genetic Algorithm Neural Network* (BPGANN) yang diterapkan adalah *classifier* yang pada Tugas Akhir sebelumnya sudah diimplementasikan oleh Risal Andika Tidisaputra dan diberikan modifikasi kode sehingga Tugas Akhir ini dapat menjalankannya. Dalam metode BPGANN yang digunakan, jumlah *node* pada *input layer neural network* adalah sebanyak jumlah fitur yang digunakan yaitu 28 *node*. Kemudian penulis menentukan bahwa jumlah *node* pada *hidden layer neural network* adalah $\frac{1}{2}$ dari jumlah *node* pada *input layer* yaitu 14. Kemudian jumlah *node* pada *output layer neural network* ditentukan berdasarkan banyaknya *class* yang terdapat dalam 1 *neural network*. Dalam Tugas Akhir ini setiap *neural network* memiliki 4 *class*. Jumlah *node* pada *output*

layer ditentukan dengan banyak bit pada *binary code* jumlah *class*. Angka desimal 4 jika diubah menjadi *binary code* memiliki 2 bit sehingga banyak *note* pada *output layer* *neural network* sebanyak 2 *node*.

3. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat huruf dengan akurasi rata - rata 92.5%.

6.2 Saran

Berikut saran-saran untuk pengembangan dan perbaikan sistem di masa yang akan datang. Di antaranya adalah sebagai berikut:

1. Kinect yang digunakan dalam Tugas Akhir ini adalah Kinect V1 yang memiliki fitur-fitur yang sangat terbatas. Saat ini sudah ada Kinect V2 yang memiliki fitur tambahan seperti mendeteksi gerakan jadi. Untuk membuat penelitian yang sama lebih baik menggunakan Kinect V2 sehingga fitur yang digunakan akan lebih baik dan akurat.
2. Menggunakan *dataset* dari berbagai macam pengguna sehingga dalam proses *training* menghasilkan *neural network* yang dapat digunakan secara fleksibel.
3. Perlu adanya ekstraksi fitur yang dapat digunakan untuk mendeteksi bahasa isyarat dinamis.

DAFTAR PUSTAKA

- [1] E. Rakun, M. Andriarni, I. W. Wiprayoga, K. Danniswara and A. Tjandra, Combining Depth Image and Skeleton Data from Kinect for Recognizing Words in the Sign System for Indonesian Language (SIBI), IEEE, 2013.
- [2] N. Ulfah, "5.000 Bayi Indonesia Lahir Tuli Setiap Tahun," Detik Health, 9 Januari 2010. [Online]. Available: [http://health.detik.com/read/2010/01/09/155558/1274969/763/..](http://health.detik.com/read/2010/01/09/155558/1274969/763/)
- [3] "Kinect," Wikipedia, 12 Desember 2015. [Online]. Available: <https://en.wikipedia.org/wiki/Kinect>.
- [4] A. W. Yanuardi, P. Samudra and J. A. Purnama P, Indonesian Sign Language Computer Application for The Deaf. 2nd International Conference on Education Technology and Computer (ICETC), IEEE, 2010.
- [5] Sílvia Grasiella Moreira Almeida, Frederico Gadelha Guimarães, Jaime Arturo Ramírez, "Feature extraction in Brazilian Sign Language Recognition based on phonological structure and using RGB-D sensors," *Expert Systems with Applications* 41, p. 7259, 2014.
- [6] K. Sanford, "Smoothing Kinect Depth Frames in Real-Time," CodeProject, 24 Januari 2012. [Online]. Available: <http://www.codeproject.com/Articles/317974/KinectDepthSmoothing>.
- [7] Rajaganapathy. S, Aravind. B, Keerthana. B, Sivagami. M, Conversation of Sign Language to Speech with Human Gestures, Chennai: Elsevier B.V., 2015.
- [8] J. Burger, "A Basic Introduction To Neural Networks," University of Wisconsin-Madison, [Online]. Available: <http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html>. [Accessed 11 Juni 2016].
- [9] C. MacLeod, "The Synthesis of Artificial Neural Network Using Single String Evolutionary Techniques," Robert Gordon University, Aberdeen, 1999.

- [10] S. Wang, S. Yin and M. Jiang, "Hybrid Neural Network Based On GA," in *Fourth International Conference on Natural Computation*, 2008.
- [11] Chao Sun, Tianzhu Zhang, Bing-Kun Bao, Changsheng Xu, "Discriminative Exemplar Coding for Sign Language," *IEEE TRANSACTIONS ON CYBERNETICS*, vol. 43, p. 1418, 2013.
- [12] R. A. T, "Rancang Bangun Modul Pengenalan Bahasa Isyarat Menggunakan Teknologi Leap Motion dan Metode Back Propagation - Genetic Algorithm Neural Network (BPGANN)," Institut Teknologi Sepuluh Nopember, Surabaya, 2015.

LAMPIRAN A

KODE SUMBER[12]

```
1  class BackPropagation
2  {
3      private float learningRate = 0.08f;
4      private FeedForward feedForward;
5      private NeuralNetwork lastStableNetwork;
6
7      public NeuralNetwork Network
8      {
9          set { feedForward.Network = value; }
10         get { return feedForward.Network; }
11     }
12
13     public DataSetList dsl
14     {
15         get { return feedForward.dsl; }
16     }
17
18     private ClassificationClass classificationClass;
19
20     public void Init(NeuralNetwork nn, DataSetList dsl,
21     ClassificationClass cc)
22     {
23         feedForward = new FeedForward();
24         feedForward.Init(nn, dsl);
25         classificationClass = cc;
26     }
27
28     public float Run(int maxIter)
29     {
30         float lastAvgerror = float.MaxValue;
31         float avgError = new float();
32         for(int iter = 0; iter < maxIter; iter++)
33         {
34             avgError = 0;
35             for(int k = 0; k < this.dsl.Count; k++)
36             {
37                 Network.InitInput();
38                 feedForward.Run(k);
39
40                 float[] errorOutput = GetErrorOutput(k);
41                 float totalError = 0;
42                 for (int i = 0; i < errorOutput.Length; i++)
43                     totalError += Math.Abs(errorOutput[i]);
44                 avgError += totalError;
```

```

41         }
42         avgError /= this.dsl.Count;
43     }
44     return avgError;
45 }
46
47 public void Run()
48 {
49     Run(100);
50 }
51
52 private float[] GetErrorOutput(int index)
53 {
54     float[] outputError = new
float[classificationClass.TargetCount];
55
56     for (int i = 0; i < classificationClass.TargetCount;
i++)
57         outputError[i] =
classificationClass.GetTarget(this.dsl[index].ClassName)[i]
- Network.OutputLayer[i].Input;
58
59     UpdateWeightHidden(outputError, index);
60
61     return outputError;
62 }
63
64 private void UpdateWeightHidden(float[] outputError, int
index)
65 {
66     for(int i = 0; i < Network.HiddenLayer.Count; i++)
67     {
68         for(int j = 0; j < Network.OutputLayer.Count;
j++)
69         {
70             float newWeight =
Network.HiddenLayer[i].GetWeight(j) +
                (this.learningRate * outputError[j]
* Network.HiddenLayer[i].Input);
71             Network.HiddenLayer[i].SetWeight(j,
newWeight);
72         }
73     }
74
75     GetErrorHidden(outputError, index);
76 }
77
78 private void GetErrorHidden(float[] outputError,int
index)

```


79	{
80	float[] hiddenError = new
	float[Network.HiddenLayer.Count];
81	
82	for(int i = 0; i < Network.HiddenLayer.Count; i++)
83	{
84	float linear = 0;
85	for (int j = 0; j < Network.OutputLayer.Count;
	j++)
86	linear += outputError[j] *
	Network.HiddenLayer[i].GetWeight(j);
87	
88	hiddenError[i] =
	Network.HiddenLayer[i].Input * (1 -
	Network.HiddenLayer[i].Input) * linear;
89	}
90	
91	UpdateWeightInput(hiddenError, index);
92	}
93	
94	private void UpdateWeightInput(float[] hiddenError, int
	index)
95	{
96	for(int i = 0; i < Network.InputLayer.Count; i++)
97	{
98	for(int j = 0; j < Network.HiddenLayer.Count;
	j++)
99	{
100	float newWeight =
	Network.InputLayer[i].GetWeight(j) +
	(this.learningRate * hiddenError[j]
	* Network.InputLayer[i].Input);
101	Network.InputLayer[i].SetWeight(j,
	newWeight);
102	}
103	}
104	}
105	}

Kode Sumber A.1 Kelas BackPropagation.cs

1	class Chromosom
2	{
3	private int[] bit;
4	
5	public int[] Bit
6	{
7	get { return bit; }
8	set { bit = value; }
9	}

10	
11	public int this[int index]
12	{
13	get { return bit[index]; }
14	set { bit[index] = value; }
15	}
16	
17	public int Length
18	{
16	get { return bit.Length; }
17	}
18	
20	private float fitnessValue = 0;
21	
22	public float FitnessValue
23	{
24	get { return fitnessValue; }
25	set { fitnessValue = value; }
26	}
27	
28	public Chromosom(int[] newBit)
29	{
30	bit = newBit;
31	}
32	}

Kode Sumber A.2 Kelas Chromosom.cs

1	class ClassificationClass
2	{
3	private List<string> classList = new List<string>();
4	private int[] actualClass;
5	
6	public string this[int index]
7	{
8	get { return this.classList[index]; }
9	}
10	
11	public List<string> GetClassList()
12	{
13	return this.classList;
14	}
15	
16	public int TargetCount
17	{
18	get
16	{
17	int factor = 1;
18	while (Math.Pow(2, factor) < classList.Count)
20	factor++;

21	return factor;
22	}
23	}
24	
25	public void Add(string className)
26	{
27	int index = GetIndex(className);
28	if (index == -1)
29	classList.Add(className);
30	}
31	
32	public void Clear()
33	{
34	classList.Clear();
35	}
36	
37	public int GetIndex(string className)
38	{
39	return classList.IndexOf(className);
40	}
41	
42	public int[] GetTarget(int index)
43	{
44	int[] target = new int[TargetCount];
45	int bitCount = target.Length - 1;
46	while(index > 0)
47	{
48	target[bitCount--] = (index % 2);
49	index = index / 2;
50	}
51	return target;
52	}
53	
54	public int[] GetTarget(string className)
55	{
56	return GetTarget(GetIndex(className));
57	}
58	}

Kode Sumber A.3 Kelas ClassificationClass.cs

1	class DataSet
2	{
3	private string className;
4	
5	public string ClassName
6	{
7	get { return className; }
8	set { className = value; }
9	}

```

10
11     private List<float> _attribute;
12
13     public float this[int index]
14     {
15         set
16         {
17             _attribute[index] = value;
18         }
19         get
20         {
21             return _attribute[index];
22         }
23     }
24
25     public int AttributeCount
26     {
27         get
28         {
29             return _attribute.Count;
30         }
31     }
32
33     public DataSet()
34     {
35         _attribute = new List<float>();
36     }
37
38     public DataSet(int attributeCount)
39     {
40         _attribute = new List<float>();
41         for (int i = 0; i < attributeCount; i++)
42             _attribute.Add(0);
43     }
44
45     public DataSet(DataSet ds)
46     {
47         _attribute = new List<float>();
48         for (int i = 0; i < ds.AttributeCount; i++)
49             _attribute.Add(ds[i]);
50         this.ClassName = ds.ClassName;
51     }
52
53     public void RemoveBit(int index)
54     {
55         _attribute.RemoveAt(index);
56     }
57
58     }

```

Kode Sumber A.4 Kelas DataSet.cs

```

1  class FeedForward
2  {
3      private NeuralNetwork neuralNetwork;
4
5      public NeuralNetwork Network
6      {
7          get { return neuralNetwork; }
8          set { neuralNetwork = value; }
9      }
10
11     private DataSetList _dsl;
12
13     public DataSetList dsl
14     {
15         get { return _dsl; }
16     }
17
18     public void Init(NeuralNetwork nn , DataSetList dsl)
19     {
20         neuralNetwork = nn;
21         _dsl = dsl;
22     }
23
24     public void Run()
25     {
26         for(int i = 0; i < _dsl.Count; i++)
27         {
28             neuralNetwork.InitInput();
29             DoInputLayer(i);
30         }
31     }
32
33     public void Run(int index)
34     {
35         neuralNetwork.InitInput();
36         DoInputLayer(index);
37     }
38
39     private void DoInputLayer(int index)
40     {
41         for (int i = 0; i < _dsl[index].AttributeCount; i++)
42             neuralNetwork.InputLayer[i].Input =
43             _dsl[index][i];
44         DoHiddenLayer();
45     }
46
47     private void DoHiddenLayer()
48     {
49
50     }
51 }

```

```

46         for (int i = 0; i < neuralNetwork.InputLayer.Count;
i++)
47             for (int j = 0; j <
neuralNetwork.HiddenLayer.Count; j++)
48                 neuralNetwork.HiddenLayer[j].Input +=
neuralNetwork.InputLayer[i].GetOutput(j);
49
50             for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
j++)
51                 neuralNetwork.HiddenLayer[j].Input =
Sigmoid(neuralNetwork.HiddenLayer[j].Input);
52             DoOutputLayer();
53         }
54
55     private void DoOutputLayer()
56     {
57         for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
i++)
58             for (int j = 0; j <
neuralNetwork.OutputLayer.Count; j++)
59                 neuralNetwork.OutputLayer[j].Input +=
neuralNetwork.HiddenLayer[i].Input *
neuralNetwork.HiddenLayer[i].GetWeight(j);
60         }
61
62     public int[] GetActualClass()
63     {
64         int[] act = new
int[neuralNetwork.OutputLayer.Count];
65         for(int j = 0; j < neuralNetwork.OutputLayer.Count;
j++)
66             {
67                 if (Math.Abs(neuralNetwork.OutputLayer[j].Input)
< neuralNetwork.threshold)
68                     act[j] = 0;
69                 else
70                     act[j] = 1;
71             }
72         return act;
73     }
74
75     private float Sigmoid(float val)
76     {
77         return 1 / (1.0f + (float)Math.Exp(-val));
78     }
}

```

Kode Sumber A.5 Kelas FeedForward.cs

```

1  class GeneticAlgorithm
2  {
3      const int individualCount = 4;
4      private Random random = new Random();
5      private ClassificationClass classificationClass;
6      private BackPropagation backPropagation;
7      private NeuralNetwork network;
8      private DataSetList dataSetList;
9      private List<Chromosom> chromosomes;
10
11     public void Init(DataSetList dsl, ClassificationClass
12     cc)
13     {
14         dataSetList = dsl;
15         classificationClass = cc;
16     }
17     public Chromosom Run()
18     {
19         ChromosomInit();
20
21         int iteration = 2;
22         for(int i = 0; i < iteration; i++)
23         {
24             DoBackPropagation();
25             DoSelection();
26             DoCrossOver();
27
28             int chanceMutation = GetRandom();
29             if (chanceMutation < GetRandom())
30                 DoMutation();
31         }
32
33         int index = 0;
34         for(int i = 1; i < chromosomes.Count; i++)
35         {
36             if (chromosomes[i].FitnessValue >
37             chromosomes[index].FitnessValue)
38                 index = i;
39         }
40
41         Chromosom fittestChromosom = chromosomes[index];
42         return fittestChromosom;
43     }
44
45     private void ChromosomInit()
46     {
47         chromosomes = new List<Chromosom>(individualCount);

```

```

44         for(int i = 0; i < individualCount; i++)
45         {
46             chromosomes.Add(new
47 Chromosom(GetRandomBinary()));
48         }
49
50     private void DoBackPropagation()
51     {
52         for(int i = 0; i < chromosomes.Count; i++)
53         {
54             DataSetList dsl = new
55 DataSetList(this.dataSetList);
56
57             for(int j = 0; j < dsl.Count; j++)
58             {
59                 int popCount = 0;
60                 for(int k = 0; k < chromosomes[i].Length;
k++)
61                 {
62                     if(chromosomes[i][k] == 0)
63                     {
64                         dsl[j].RemoveBit(k - popCount);
65                         popCount++;
66                     }
67                 }
68
69                 NeuralNetwork nn = new NeuralNetwork();
70                 nn.InitNetwork(dsl[0].AttributeCount,
71 dsl[0].AttributeCount / 2, classificationClass.TargetCount);
72                 nn.InitWeight();
73
74                 BackPropagation bp = new BackPropagation();
75                 bp.Init(nn, dsl, classificationClass);
76                 bp.Run(5000);
77
78                 FeedForward ff = new FeedForward();
79                 ff.Init(nn, dsl);
80
81                 int totalCorrect = 0;
82                 for(int j = 0; j < dsl.Count; j++)
83                 {
84                     ff.Run(j);
85                     bool correct = true;
86                     int[] targetClass =
87 classificationClass.GetTarget(dsl[i].ClassName);
88                     for (int k = 0; k <
ff.GetActualClass().Length; k++)

```



```

88         {
89             if (targetClass[k] !=
ff.GetActualClass()[k])
                correct = false;
90         }
91         if (correct)
92             totalCorrect++;
93     }
94     chromosomes[i].FitnessValue = totalCorrect /
95     (float)dsl.Count;
96     }
97
98     private void DoSelection()
99     {
100         chromosomes.Sort((val1, val2) =>
101     val1.FitnessValue.CompareTo(val2.FitnessValue));
102     }
103
104     private void DoCrossOver()
105     {
106         for(int i = 0; i < chromosomes.Count; i+= 2)
107         {
108             int nextindex = i + 1;
109             if (nextindex >= chromosomes.Count)
110                 nextindex = i - 1;
111             int[] bit1 = chromosomes[i].Bit;
112             int[] bit2 = chromosomes[nextindex].Bit;
113
114             for(int j = bit1.Length / 2; j < bit1.Length;
j++)
115             {
116                 int temp = bit1[j];
117                 bit1[j] = bit2[j];
118                 bit2[j] = temp;
119             }
120
121             chromosomes[i].Bit = bit1;
122             chromosomes[nextindex].Bit = bit2;
123         }
124     }
125
126     private void DoMutation()
127     {
128         for(int i = 0; i < chromosomes.Count; i++)
129         {
130             int chanceChromosomMutation = GetRandom();
131             if(chanceChromosomMutation <= GetRandom())
132             {

```

```

131         int bitIndex =
132         GetRandom(chromosoms[i].Length);
            if (chromosoms[i][bitIndex] == 0)
133                 chromosoms[i][bitIndex] = 1;
134             else
135                 chromosoms[i][bitIndex] = 0;
136         }
137     }
138 }
139
140 private int[] GetRandomBinary()
141 {
142     int[] chromosomTemp = new
143     int[dataSetList[0].AttributeCount];
144     for (int i = 0; i < dataSetList[0].AttributeCount;
145     i++)
146     {
147         int ran = random.Next(0, 2);
148         chromosomTemp[i] = ran;
149     }
150     return chromosomTemp;
151 }
152 private int GetRandom(int max)
153 {
154     return random.Next(0, max);
155 }
156 private int GetRandom()
157 {
158     return random.Next(0, 100);
159 }
160 }
161

```

Kode Sumber A.6 Kelas GeneticAlgorithm.cs

```

1  class NeuralNetwork
2  {
3      private Random random = new Random();
4
5      public int Seed
6      {
7          set { random = new Random(value); }
8      }
9
10     public float threshold = 0.5f;
11
12     public List<Node> InputLayer { set; get; }
13     public List<Node> HiddenLayer { set; get; }

```

```

14     public List<Node> OutputLayer { set; get; }
15
16     public NeuralNetwork()
17     {
18
19     }
20
21     public NeuralNetwork(NeuralNetwork newNN)
22     {
23         newNN.InitInput();
24         InitNetwork(newNN.InputLayer.Count,
25 newNN.HiddenLayer.Count, newNN.OutputLayer.Count);
26         for(int i = 0; i< newNN.InputLayer.Count; i++)
27         {
28             InputLayer[i] = newNN.InputLayer[i];
29         }
30         for(int i = 0; i < newNN.HiddenLayer.Count; i++)
31         {
32             HiddenLayer[i] = newNN.HiddenLayer[i];
33         }
34     }
35
36     public void InitInput()
37     {
38         for (int i = 0; i < InputLayer.Count; i++)
39             InputLayer[i].Input = 0;
40
41         for (int i = 0; i < HiddenLayer.Count; i++)
42             HiddenLayer[i].Input = 0;
43
44         for (int i = 0; i < OutputLayer.Count; i++)
45             OutputLayer[i].Input = 0;
46     }
47
48     public void InitNetwork(int inputLayercount, int
49 hiddenLayerCount, int outputLayerCount)
50     {
51         InputLayer = new List<Node>();
52         HiddenLayer = new List<Node>();
53         OutputLayer = new List<Node>();
54
55         for (int i = 0; i < inputLayercount; i++)
56             InputLayer.Add(new Node(hiddenLayerCount));
57
58         for (int i = 0; i < hiddenLayerCount; i++)
59             HiddenLayer.Add(new Node(outputLayerCount));
60
61         for(int i = 0; i < outputLayerCount; i++)

```

```

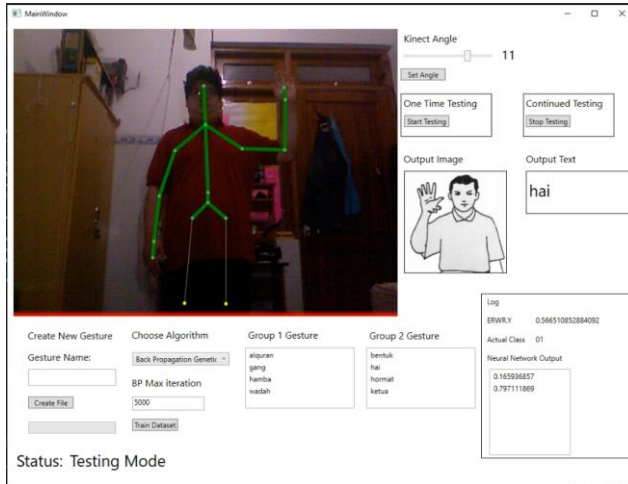
58         {
59             OutputLayer.Add(new Node(1));
60             OutputLayer[i].SetWeight(0, 1);
61         }
62     }
63
64     public void InitWeight()
65     {
66         for (int i = 0; i < InputLayer.Count; i++)
67             for (int j = 0; j < HiddenLayer.Count; j++)
68                 InputLayer[i].SetWeight(j, GetRandom() /
69 100.0f);
70
71         for (int i = 0; i < HiddenLayer.Count; i++)
72             for (int j = 0; j < OutputLayer.Count; j++)
73                 HiddenLayer[i].SetWeight(j, GetRandom() /
74 100.0f);
75     }
76
77     int GetRandom()
78     {
79         return random.Next(0, 30);
80     }

```

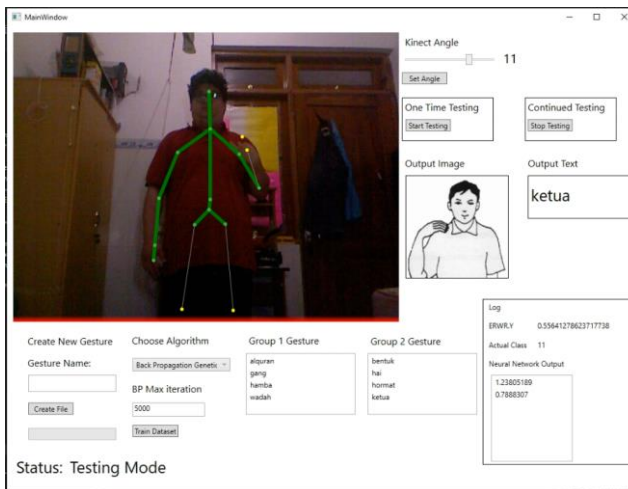
Kode Sumber A.7 Kelas Neural Network.cs

LAMPIRAN B

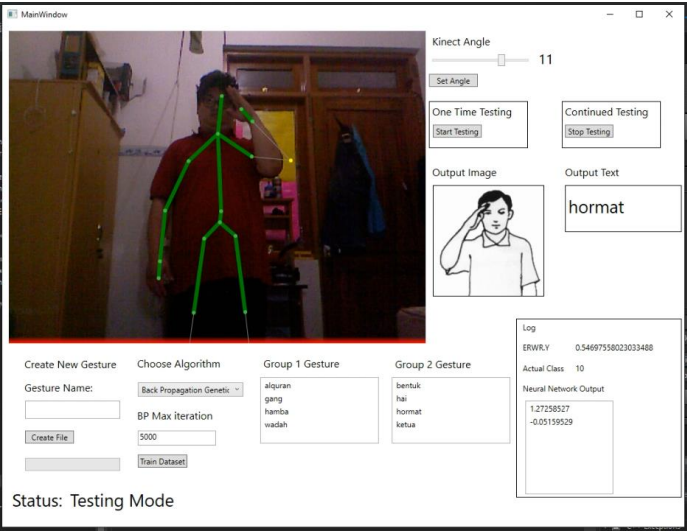
SCREENSHOT APLIKASI



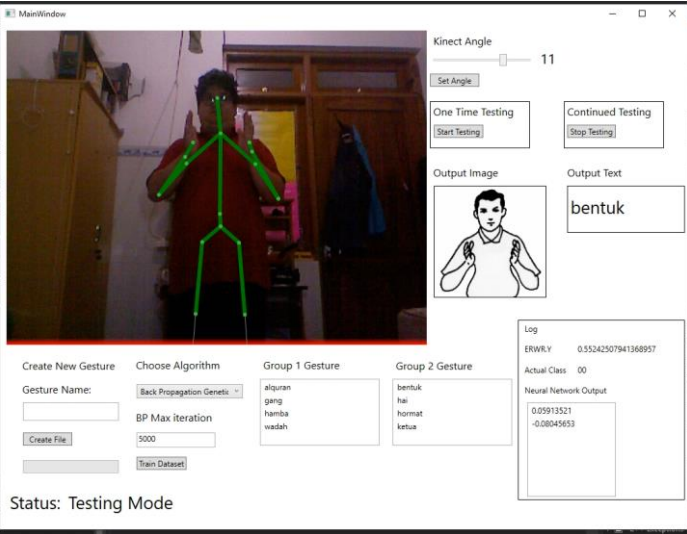
Gambar B.1 Aplikasi Mendeteksi Bahasa Isyarat Hai



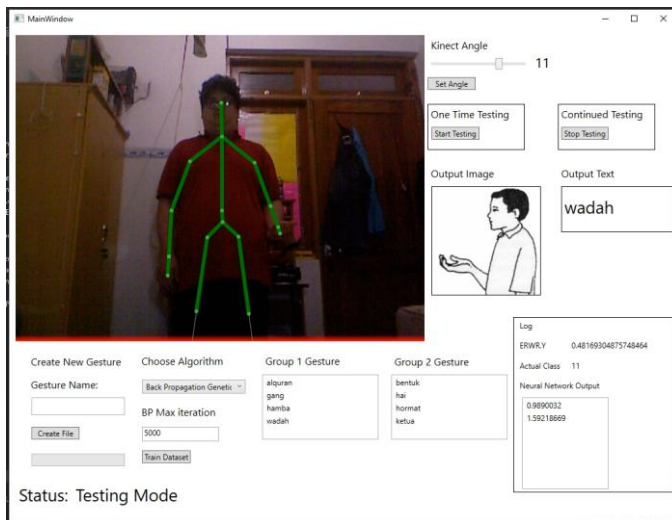
Gambar B.2 Aplikasi Mendeteksi Bahasa Isyarat Ketua



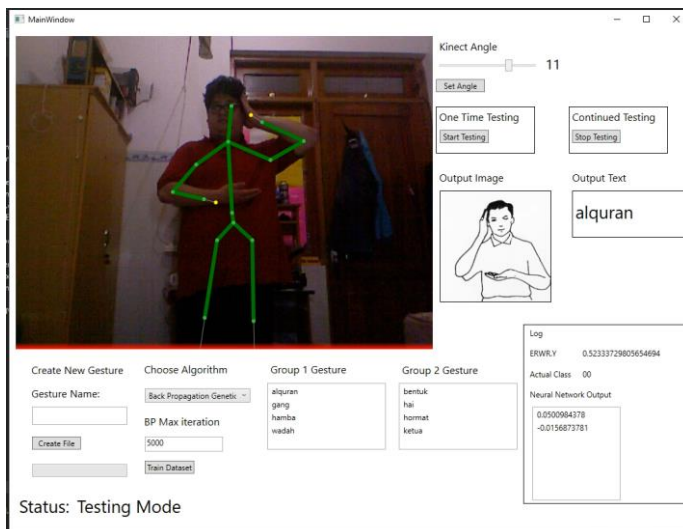
Gambar B.3 Aplikasi Mendeteksi Bahasa Isyarat Hormat



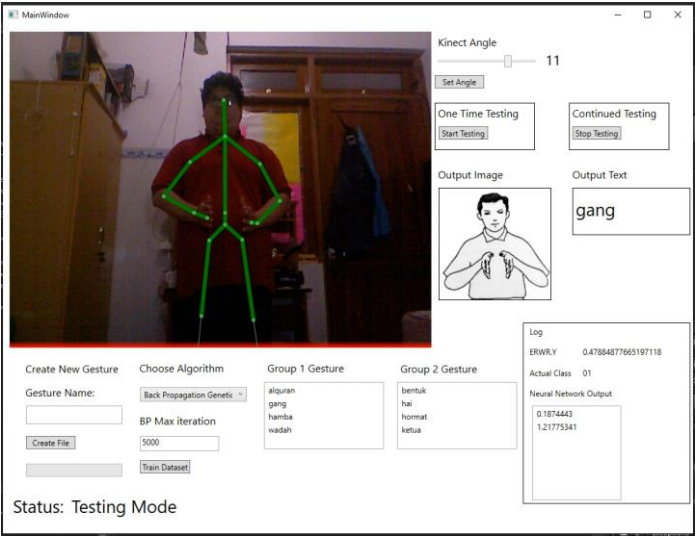
Gambar B.4 Aplikasi Mendeteksi Bahasa Isyarat Bentuk



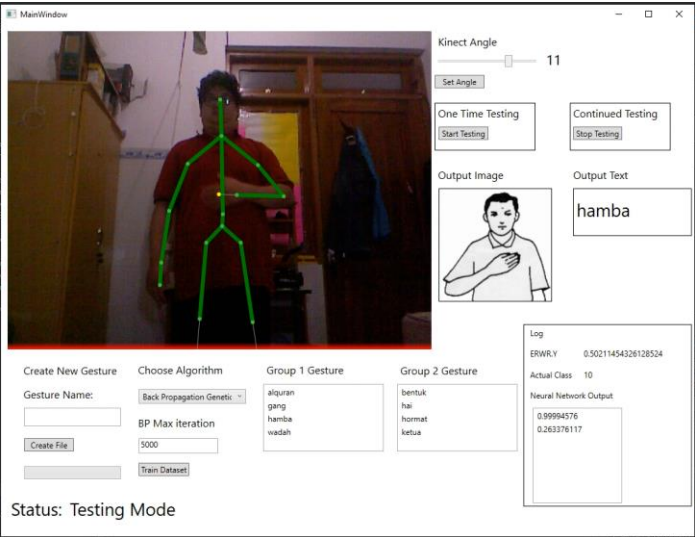
Gambar B.5 Aplikasi Mendeteksi Bahasa Isyarat Wadah



Gambar B.6 Aplikasi Mendeteksi Bahasa Isyarat Alquran



Gambar B.7 Aplikasi Mendeteksi Bahasa Isyarat Gang



Gambar B.8 Aplikasi Mendeteksi Bahasa Isyarat Hamba

BIODATA PENULIS



Yohanes Aditya Sutanto, Lahir pada tanggal 30 Juni 1994. Saat ini sedang menempuh pendidikan perguruan tinggi di Institut Teknologi Sepuluh Nopember Surabaya di jurusan Teknik Informatika Fakultas Teknologi Informasi angkatan 2012.

Pernah mengikuti organisasi dan beberapa kepanitian diantaranya adalah Staf Pengembangan Sumber Daya Mahasiswa Himpunan Mahasiswa Teknik Computer-

Informatika 2013-2014, Staff SCHEMATICS 2013 dan SCHEMATICS 2014.