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

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 yang diciptakan untuk windows sekitar tahun 2012. Kinect dapat digunakan untuk mendeteksi gerakan isyarat yang diberikan dengan memanfaatkan fitur yang ada. Tugas akhir ini menggunakan fitur skeleton tracking yang ada pada Kinect untuk mendeteksi bahasa isyarat.

Hasil pengujian dari tugas akhir ini menunjukkan bahwa metode Back Propagation Genetic Algorithm Neural Network yang digunakan sebagai classifier gerakan isyarat memiliki akurasi yang baik yaitu sekitar 92,5 persen. Hasil tersebut masih dapat ditingkatkan dengan menambahkan data training yang diambil dari sample yang bervariasi.

Kata kunci: Kinect, 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 which invented by Microsoft in 2012. Kinect can be used to detect motion gesture by utilizing the existing features. This final project using the skeleton tracking feature that availavle on Kinect to detect the sign language.

The test results of this final project show that Back Propagation Genetic Algorithm Neural Network method used as classifier have good accuracy which is about 92,5 percent. These results can be improved by adding training data taken from different sample.

Keywords: : Kinect, 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

DAFTAR GAMBAR

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

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 disampaikannya 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 console* 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*, melainkan melalui *natural user interface* dengan menggunakan *gesture* dan perintah suara) [3].

Sebelumnya sudah ada tugas akhir yang dibuat oleh Risal Andika Tridisaputra tentang pengenalan bahasa isyarat Indonesia menggunakan teknologi leap motion dengan metode *Back Propagation Genetic Algorithm Neural Network*, tetapi tugas akhir tersebut hanya sebatas membaca isyarat abjad [13]. 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 didapat dengan Kinect?
2. Bagaimana menerapkan metode *Back Propagation Genetic Algorithm Neural Network* untuk mendeteksi bahasa isyarat yang diberikan?
3. Bagaimana hasil akurasi gerakan isyarat yang dideteksi oleh Kinect?

1.3 Batasan Masalah

Permasalahan yang dibahas dalam tugas akhir ini memiliki beberapa batasan, di antaranya sebagai berikut:

1. Aplikasi dikembangkan menggunakan Kinect SDK dengan menggunakan bahasa pemrograman C# dengan IDE Microsoft Visual Studio.
2. Versi Kinect sensor yang dipakai adalah Kinect V1.
3. Topologi *Neural Network* adalah jenis *Multilayer Percepton* dengan 1 hidden layer.

4. Bahasa Isyarat yang digunakan berdasarkan pada Sistem Isyarat Bahasa Indonesia (SIBI).
5. Bahasa isyarat yang digunakan hanya yang bersifat statis.
6. Hanya 8 bahasa isyarat pokok yang akan dipakai yaitu Alquran, Bentuk, Gang, Hai, Hamba, Hormat, Ketua, dan Wadah.

1.4 Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat aplikasi pengenalan bahasa isyarat menggunakan teknologi Kinect. Dan dapat mengimplementasikan metode *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. *Decision Tree*
6. *Neural Network*
7. *Back Propagation*
8. *Back Propagation Genetic Algorithm*

B. Perancangan perangkat lunak

Pada tahap ini diawali dengan melakukan analisis awal terharap 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 ektraksi fitur Kinect untuk mendapatkan data bahasa isyarat
4. Perancangan normalisasi fitur
5. Perancangan pembuatan *rule*
6. 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 dilakuka 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 *back propagation*, algoritma *back propagation genetic*, dan pengetahuan umum mengenai bahasa isyarat.

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.

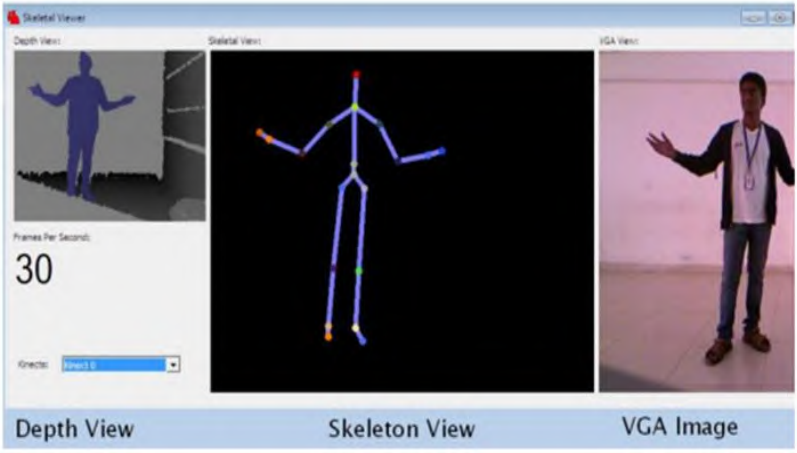
2.2 Kinect

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 2.1.

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 2.2 [5].

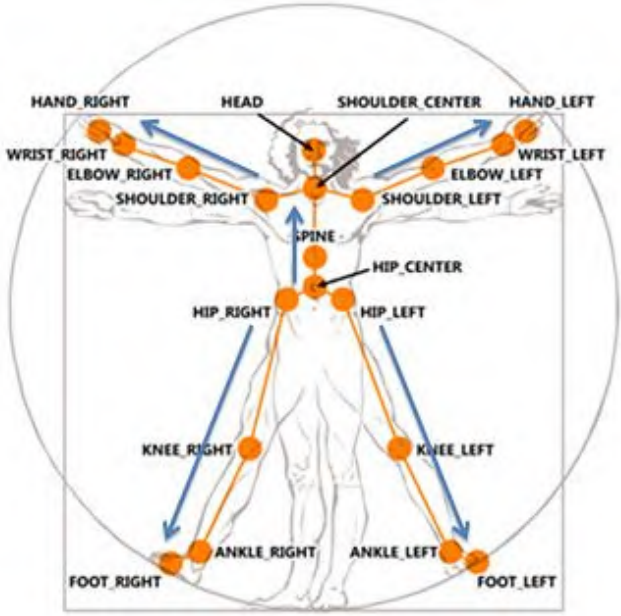
Gambar 2.1 Microsoft Kinect [6]



Gambar 2.2 *Depth, Skeleton,* dan *VGA view* [7]

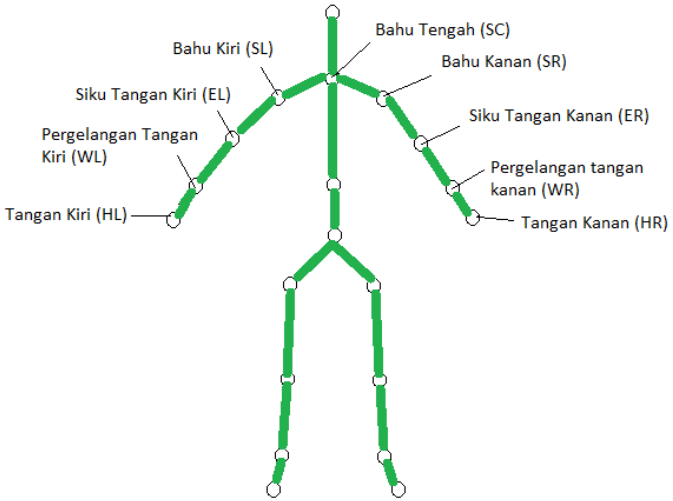
2.3 Kinect SDK

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]. Tugas akhir ini mengunakan fitur *skeleton tracking* Kinect dengan mengambil data *skeleton joint. Skeleton joints* adalah sebuah titik dalam bidang 3 dimensi yang menggambarkan bagian tubuh manusia yang ditangkap oleh Kinect. Pada Kinect SDK yang akan digunakan memiliki 20 *skeleton joints* yang dapat dilihat pada Gambar 2.3.



Gambar 2.3 *Skeleton Joints* yang diketahui Kinect

Dari semua *skeleton joints* yang diketahui, hanya ada 9 yang akan dipakai dalam tugas akhir ini. Hal ini dikarenakan kebanyakan bahasa isyarat memiliki perbedaan satu sama lainnya dengan melihat 9 *skeleton joints* tersebut. Sembilan *skeleton joints* tersebut dapat dilihat pada Gambar 2.4.



Gambar 2.4 *Skeleton Joints* yang Digunakan

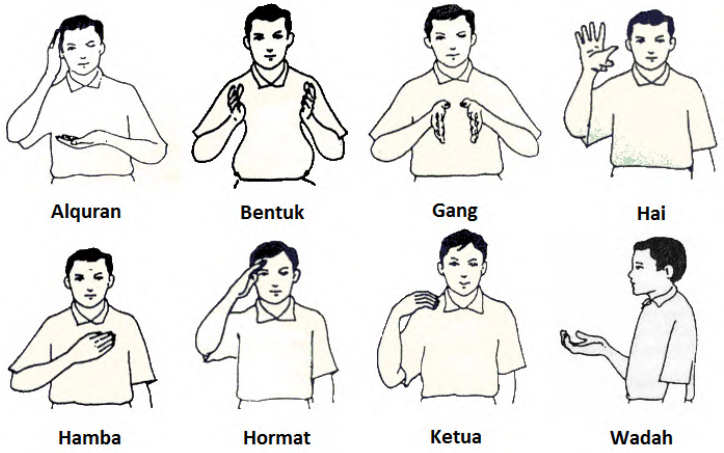
2.4 Bahasa Isyarat

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 Bentukan: Dibentuk dengan menggabungkan isyarat pokok dan isyarat tambahan.
4. Abjad Jari: dibentuk dengan jari-jari untuk mengeja huruf.

Pada tugas akhir ini ada 8 bahasa isyarat pokok yang digunakan. Semua isyarat tersebut bersifat statis (tidak ada gerakan) karena fitur *skeleton tracking* yang dimiliki Kinect v1 sangat terbatas. Selain itu keterbatasan bahasa isyarat statis yang ada membuat ke 8 bahasa isyarat tersebut dipilih. Gambar 2.5 menunjukkan bahasa isyarat yang dipakai dalam tugas akhir ini.



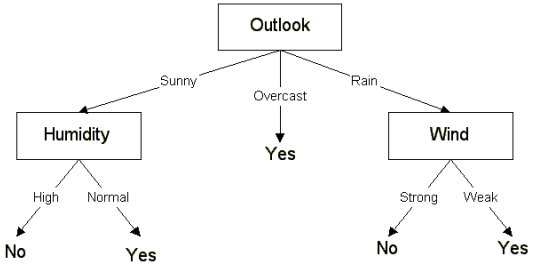
Gambar 2.5 Bahasa Isyarat yang Dipakai

2.5 *Decision Tree*

Decision tree merupakan salah satu teknik *data mining* yang terkenal. *Decision tree* secara sistematis menganalisis informasi yang terkandung dalam *data source* yang besar untuk mengekstrak *valuable rules* dan *relationships* dan biasanya digunakan untuk *classification.* Dibandingkan dengan teknik *data mining* lainnya, *Decision tree* secara luas diterapkan dalam

berbagai area karena cukup kokoh untuk *data scales* dan *distributions.*

Algoritma *decision tree* secara rekursif membagi *dataset* menggunakan pendekatan *depth-first greedy* atau pendekatan *breadth-first*, sampai semua item dimiliki oleh *class* yang diketahui. Struktur dari *decision tree* terbuat dari *root, internal,* dan *leaf nodes.* Kebanyakan *decision tree classifiers* melakukan *classification* dalam 2 tahap yaitu *tree-growing* dan *tree-pruning*. *Tree-growing* dilakukan dengan cara *top-down*. Dalam tahap ini *tree* secara rekursif dibagi sampai semua data termasuk dalam *class* yang sama. Dalam tahap *tree-pruning tree* yang sudah dibentuk dipotong. Hal ini dilakukan untuk meningkatkan prediksi dan akurasi dari *decision tree* dengan meminimalisasi *over-fitting* (*nouse* atau banyaknya data dalam *dataset training*) [8]. Contoh *decision tree* dapat dilihat pada Gambar 2.6.



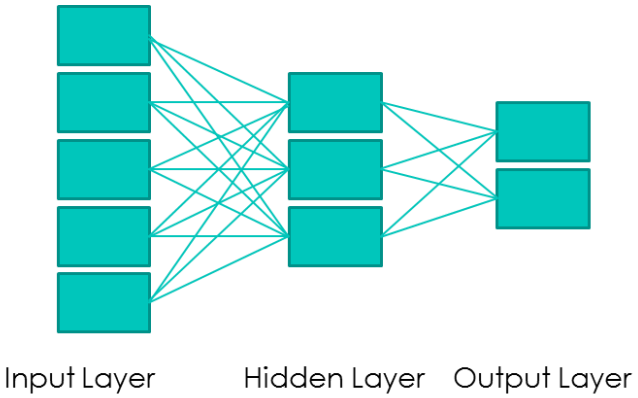
Gambar 2.6 Contoh *Decision Tree*

Pada tugas akhir ini dilakukan pembagian *rule* menggunakan metode *decision tree* dengan mengambil level 1 *tree* dari hasil *decision tree* yang didapat, kemudian setiap *leaf* akan menjadi 1 golongan. Hal ini dilakukan untuk menyederhanakan model *classifier* yang akan dibentuk.

2.6 *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 [9]. Contoh dari bentuk *neural network* dengan 1 lapisan *hidden* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Contoh *Neural Network* Dengan 1 *Hidden Layer*

2.7 *Back Propagation*

Back Propagation (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.* Pada tugas akhir ini, data yang di *traning* oleh algoritma Back Propagation adalah data *skeleton joints* yang diambil oleh Kinect. 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.

$$Error_j = (Target_j - Input_j) \quad (2.1)$$

2. Update bobot pada lapisan *hidden* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *hidden* dan *output*) dan α adalah *learning rate*.

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

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

$$Error_j = Input_j(1 - Input_j)(Error_a\, W_{ja} + Error_b\, W_{jb}) \quad (2.3)$$

4. Update bobot pada lapisan *input* menggunakan rumus pada Persamaan 2.2. *Wij* adalah bobot di antara lapisan i dan j (antara lapisan *input* dan *hidden*), α adalah *learning rate* dan *Input* adalah fitur *skeleton joints* yang digunakan (dibahas dalam subbab 3.2.6.1)[10].

2.8 *Back Propagation Genetic Algorithm Neural Network*

Metode *Back Propagation Genetic Algorithm Neural Network* (BPGANN) merupakan metode 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 [11]. Langkah-langkah BPGANN yang dilakukan dalam tugas akhir ini adalah seperti berikut:

1. Inisialisasi kromosom.
2. Mencari *fitness value* setiap kromosom dengan algoritma *Back Propagation* dan data *training* yang ada.
3. Melakukan proses *selection, cross-over,* dan *mutation.*
4. Memberikan kromosom terbaik untuk tahap *training* selanjutnya.

[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 atau 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 model *Back Propagation Genetic Algorithm Neural Network* (BPGANN) 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 model BPGANN 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 iysarat 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 sampai 1.8 meter.

3.1.3 Identifikasi Pengguna

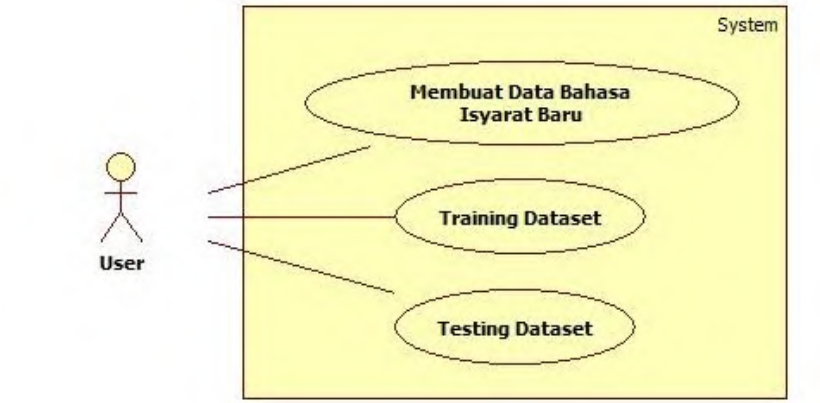
Dalam aplikasi tugas akhir ini, pengguna yang akan terlibat hanya terdapat satu orang saja, yakni orang yang akan melakukan pengenalan bahasa isyarat.

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.
2	UC-02	*Training Dataset*	Pengguna mengaktifkan mode pelatihan *dataset*
3	UC-03	*Testing Dataset*	Pengguna melakukan percobaan *dataset* 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	1. Pengguna memasukkan nama bahasa isyarat yang akan dibuat di dalam *textbox* yang sudah disediakan. 2. Pengguna menekan tombol kontrol "Create File" 3. Sistem menerima inputan dan menunggu 5 detik untuk perangkat Kinect menemukan *skeleton* pengguna. 4. Pengguna masuk ke dalam daerah yang dapat ditangkap Kinect dan mencari lokasi yg tepat sehingga Kinect mendapatkan *skeleton* pengguna, kemudian melakukan gerakan bahasa isyarat yang akan dibuat. A1. Kinect tidak menemukan *skeleton* pengguna. 5. Kinect menemukan *skeleton* pengguna, 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* pengguna. 1. Sistem memberikan notifikasi *skeleton* pengguna 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	*Training Dataset*
Nomor	UC-02
Deskripsi	Kasus penggunaan aktor untuk melakukan pelatihan data set
Aktor	Pengguna
Kondisi Awal	pengguna sudah menjalankan aplikasi dan sudah ada data *training* yang dibuat aplikasi pada tahap sebelumnya
Alur Normal	1. Pengguna menekan tombol kontrol “Train Dataset” yang ada pada aplikasi 2. Sistem memulai tahap *training dataset* yang sudah ada

	3. Sistem membuat model BPGANN yang disimpan dalam format .xml
Alur Alternatif	-
Kondisi Akhir	Aplikasi membuat model BPGANN yang disimpan dalam format .xml

3.2.3.3 Kasus *Penggunaan Testing Dataset*

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

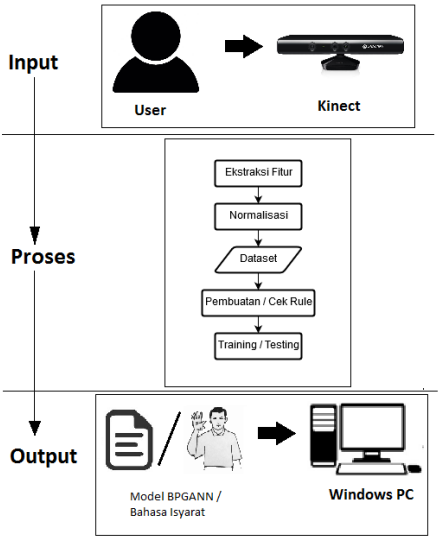
Tabel 3.5 Spesifikasi Kasus Penggunaan *Testing Dataset*

Nama Kasus Penggunaan	*Testing Dataset*
Nomor	UC-03
Deskripsi	Kasus penggunaan aktor untuk melakukan percobaan *dataset*
Aktor	Pengguna
Kondisi Awal	pengguna sudah menjalankan aplikasi dan sudah ada data *training* yang dibuat aplikasi pada tahap sebelumnya
Alur Normal	1. Pengguna menekan tombol kontrol "Start *Testing*" yang ada pada aplikasi 2. Sistem memulai tahap *testing dataset* dengan model BPGANN 3. Pengguna melakukan bahasa isyarat didepan Kinect. 4. Kinect mengambil data skeleton yang didapat 5. Sistem mengekstrak data skeleton yang kemudian akan

	dijadikan input untuk model BPGANN 6. Sistem mendapatkan *output class* dari model BPGANN kemudian menampilkannya
Alur Alternatif	-
Kondisi Akhir	Aplikasi memberikan output bahasa isyarat

3.2.4 Arsitektur Umum Sistem

Arsitektur umum pada aplikasi ini memiliki perangkat tambahan Kinect sebagai *input device*. Implementasi aplikasi dibuat menggunakan Microsoft Visual Studio. Arsitektur umum sistem pada aplikasi yang akan dibuat dapat dilihat pada Gambar 3.2.

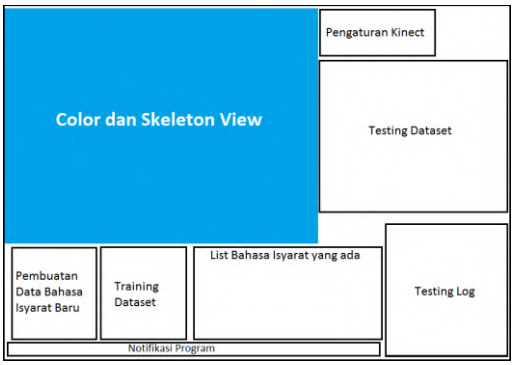


Gambar 3.2 Arsitektur Umum Sistem

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 apikasi 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 3.3.



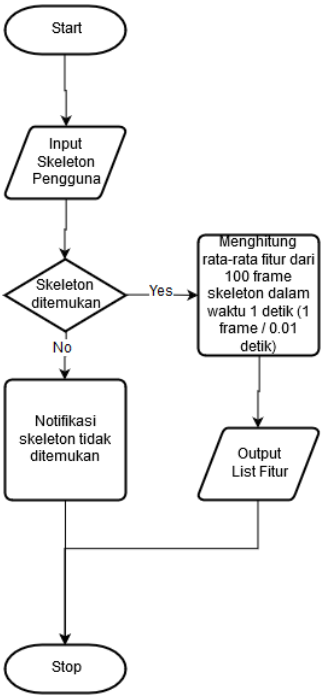
Gambar 3.3 Rancangan Antarmuka Aplikasi

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 3.4.



Gambar 3.4 Diagram Alir Proses Ekstraksi Fitur

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 2.4. 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 [12]. Dua puluh depalan fitur yang dimaksud dapat dilihat pada Tabel 3.6.

Tabel 3.6 Fitur yang Digunakan

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

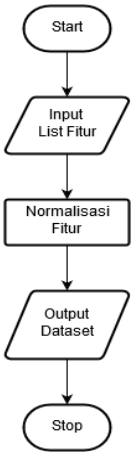
3.2.6.2 Rancangan Proses Normalisasi Fitur

Normalisasi fitur dilakukan setelah mendapatkan semua fitur yang ada di Tabel 3.6. 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. Diagram alir proses normalisasi fitur dapat dilihat pada Gambar 3.5.

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

Keterangan:
X = nilai fitur sebelum normalisasi
X' = nilai fitur setelah proses normalisasi
X_{min} = nilai fitur minimum
X_{max} = nilai fitur maksimum.



Gambar 3.5 Diagram Alir Proses Normalisasi Fitur

Untuk fitur *vector3* memiliki nilai minimum -2 dan nilai maksimum 2 untuk setiap koordinat. Nilai tersebut didapat dari hasil perhitungan *vector3 u*(1,1,1) dengan *vector3 v*(-1,-1,-1). Nilai minimum didapatkan dari perhitungan *v – u*, sedangkan nilai maksimum didapatkan dari perhitungan *u – v*. Dengan nilai tersebut persaamaan untuk normalisasi fitur *vector3* dapat dilihat pada Persamaan 3.2.

$$n_{baru} = \frac{n_{lama} + 2}{4} \qquad (3.2)$$

Untuk fitur *angle* memiliki nilai minimum 0 dan nilai maksimum π. Dengan nilai tersebut persamaan untuk normalisasi fitur *angle* dapat dilihat pada Persamaan 3.3

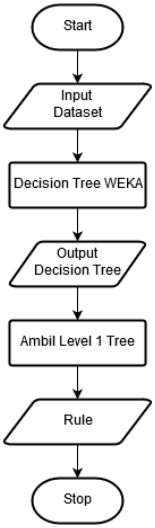
$$n_{baru} = \frac{n_{lama}}{\pi} \qquad (3.3)$$

Untuk fitur *distance* memiliki nilai minimum 0 dan nilai maksimum 3.4641. Nilai maksimum didapatkan dari perhitungan *vector3 u* dan *vector3 v* yang memiliki jarak terjauh. Dengan nilai tersebut persamaan untuk normalisasi fitur *distance* dapat dilihat pada Persamaan 3.4

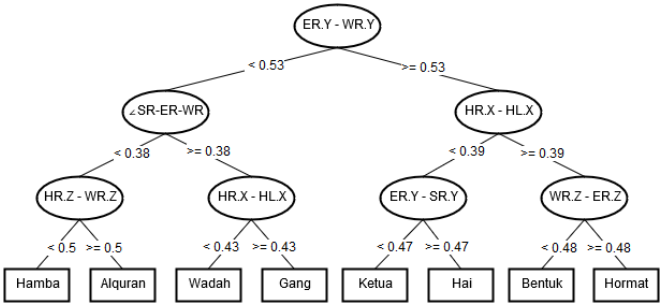
$$n_{baru} = \frac{n_{lama}}{3.4641} \qquad (3.4)$$

3.2.6.3 Rancangan Proses Pembuatan *Rule*

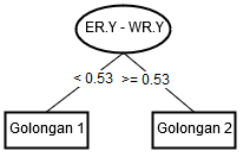
Pembuatan *rule* dilakukan untuk menyederhanakan model BPGANN. Hal ini dilakukan untuk meningkatkan akurasi model yang dibuat karena semakin sedikit target *class* yang diinginkan maka akurasi dari model tersebut akan semakin baik. Diagram alir proses pembuatan *rule* dapat dilihat pada Gambar 3.6. Tahap pertama pembuatan *rule* yaitu menggunakan metode *Decision Tree* yang ada pada aplikasi *WEKA* dengan data *training* yang sudah ditentukan. Hasil dari metode Decision Tree dapat dilihat pada Gambar 3.7. Dari hasil tersebut diambil level 1 *tree*. Hasil dari level 1 *tree* dapat dilihat pada Gambar 3.8. Selanjutnya, dari level 1 *tree* dibuat rule yang dapat dilihat pada Tabel 3.7.



Gambar 3.6 Diagram Alir Proses Pembuatan *Rule*



Gambar 3.7 Hasil Metode *Decision Tree*



Gambar 3.8 Level 1 *Decision Tree*

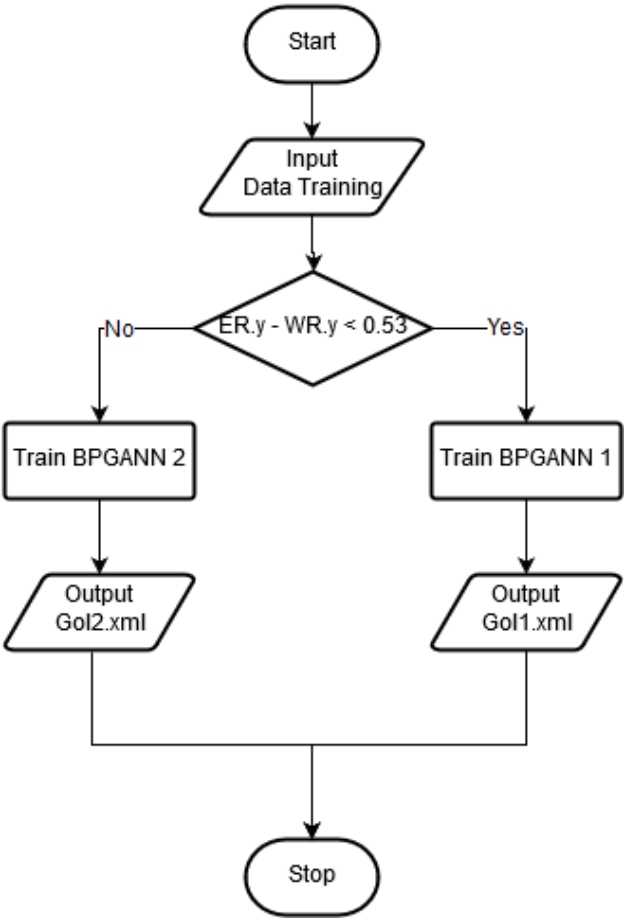
Tabel 3.7 *Rule* Pembagian Model BPGANN

Rule	*If*	*Then*	*Class*
1	ER.Y – WR.Y < 0.53f	Gol1.xml	Alquran, Gang, Hamba, Wadah
2	*else*	Gol2.xml	Bentuk, Hai, Hormat, Ketua

3.2.6.4 Rancangan Proses *Training*

Proses *training* diawali inisialisasi model BPGANN. inisialisasi model BPGANN diawali membagi data *training* yang sudah di normalisasi menjadi 2 kelompok sesuai dengan *rule* pada Tabel 3.7. Kemudian *training* dilanjutkan dengan mencari kromosom terbaik menggunakan *Genetic Algorithm*. Setelah mendapatkan kromosom tersebut, dilakukan seleksi terharap fitur dengan kromosom yang didapatkan. Jika nilai gen ke ‘x’ pada kromosom bernilai 0 maka fitur ke ‘x’ tidak dipakai sebagai *input* untuk model. Sebaliknya jika gen ke ‘x’ pada kromosom bernilai 1 maka fitur ke ‘x’ dipakai sebagai *input* untuk model yang akan dibentuk. Setelah proses seleksi selesai, model diinisialisasi dengan jumlah *input node* sebanyak fitur yang dipakai, kemudian jumlah *hidden node* sebanyak ½ dari jumlah *input node*, dan jumlah *output node* sebanyak 2 *node* yang ditentukan dengan melihat jumlah bit pada *binary code* banyaknya *class* dimana setiap model memiliki output sebanyak 4 *class*. Setelah itu dilakukan proses *update weight* dengan menggunakan metode *Back Propagation*. Nilai awal *weight* pada *input layer* dan *hidden layer* ditentukan dengan mengambil angka acak antara 0.0 sampai 0.3.

Setelah model diinisialisasi *traning* dilanjutkan dengan input data *training* yang ada, setelah itu dilakukan pemeriksaan *rule*. Untuk data *training* yang memenuhi rule pertama, akan dilatih dengan BPGANN pertama dan menghasilkan output model BPGANN “Gol1.xml”. Untuk data *training* yang tidak memenuhi rule pertama, akan dilatih dengan BPGANN kedua dan menghasilkan output model *BPGANN* “Gol2.xml”. Diagram alir proses *training* dapat dilihat pada Gambar 3.9.

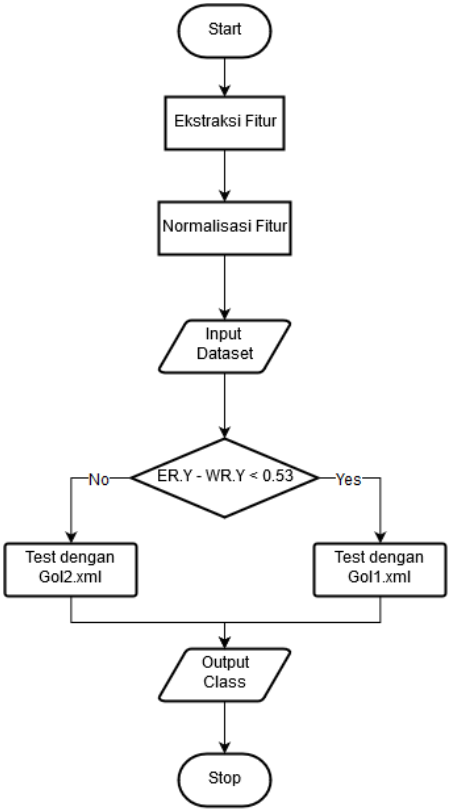


Gambar 3.9 Diagram Alir Proses *Training*

3.2.6.5 Rancangan Proses *Testing*

Proses *testing* dilakukan dengan membuat *dataset* baru yang dibentuk dari proses ektraksi fitur, setelah itu dilakukan normalisasi fitur. Setelah *dataset* di normalisasi, *dataset* masuk dalam pemeriksaan *rule*. Jika *dataset* memenuhi *rule* pertama

maka *dataset* tersebut akan diuji menggunakan model BPGANN "Gol1.xml". Jika *dataset* tidak memenuhi *rule* pertama maka *dataset* tersebut akan diuji menggunakan model BPGANN "Gol2.xml". Diagram alir proses *testing* dapat dilihat pada Gambar 3.10.



Gambar 3.10 Diagram Alir Proses *Testing*

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

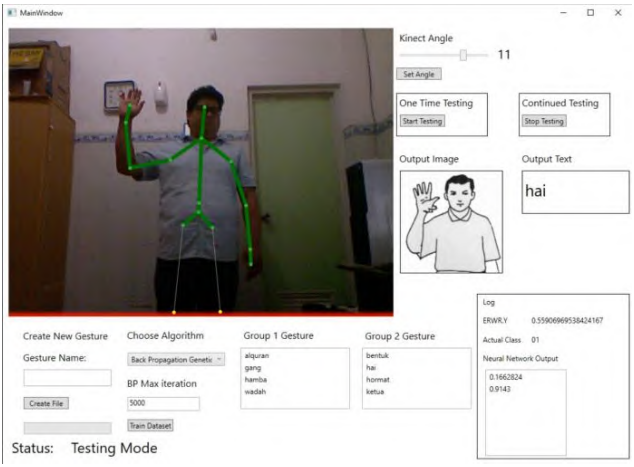
Spesifikasi perangkat lunak yang digunakan untuk membuat aplikasi ini sebaga berikut.

- 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  Microsoft.Samples.Kinect.WpfViewers.KinectSensorManager
    kinectManager = new
    Microsoft.Samples.Kinect.WpfViewers.KinectSensorManager();
16          kinectManager.KinectSensor = mainSensor;
17          kinectManager.ColorStreamEnabled = true;

18  kinectManager.SkeletonStreamEnabled = true;
20          kinectManager.DepthStreamEnabled = true;
21          kinectManager.ElevationAngle = KinectDefaultAngle;
22          ColorView.KinectSensorManager =
23          SkeletonView.KinectSensorManager = kinectManager;
24          StatusDetail.Content = "Idle";
25          mainSensor.AllFramesReady +=
    MainSensor_AllFramesReady;
26          InitClassificationClass();
27       }
28      else StatusDetail.Content = "Kinect Not Found";
29  }
                }
```

Kode Sumber 4.1 Kode Sumber Integrasi Kinect

```
1   private Skeleton GetFirstSkeleton(AllFramesReadyEventArgs e)
2   {
3       using (SkeletonFrame skeletonFrameData =
    e.OpenSkeletonFrame())
```

```
4       {
5           if (skeletonFrameData == null)
6               return null;
8
8   skeletonFrameData.CopySkeletonDataTo(allSkeletons);
9
10          //get the first tracked skeleton
11          Skeleton first = (from s in allSkeletons
                                        where s.TrackingState ==
    SkeletonTrackingState.Tracked
                                        select s).FirstOrDefault();
12
13          return first;
14      }
15  }
```

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           result = false;
5           return;
6       }
8
8       //Right Body
9   SR.SetVector(first.Joints[JointType.ShoulderRight].Position
10  );
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].Positio
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 Proses Normalisasi Fitur

Setelah proses ekstraksi fitur, semua fitur yang dihasilkan harus di normalisasi agar *range* nilai semua fitur sama, yaitu 0 sampai 1. Normalisasi fitur dilakukan dengan memanggil fungsi AllFeatureNormalize() di baris 101 pada Kode Sumber 4.3. Implementasi fungsi tersebut dapat dilihat pada Kode Sumber 4.4. Kemudian setiap fitur memiliki persamaan tersendiri untuk

normalisasi sesuai dengan tahap perancangan proses normalisasi fitur. Normalisasi fitur *vector3* diimplementasikan dengan fungsi VectorFeatureNorm(). Normalisasi fitur *angle* diimplementasikan dengan fungsi AngleFeatureNorm(). Normalisasi fitur *distance* diimplementasikan dengan fungsi DistanceFeatureNorm(). Implementasi ketiga fungsi tersebut dapat dilihat pada Kode Sumber 4.5.

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

Kode Sumber 4.4 Fungsi Normalisasi Fitur

```
1   public class Features
2   {
3       public static Vector3 VectorFeatureNorm(Vector3 u)
4       {
5           Vector3 newVector = new Vector3();
6           newVector.X = (u.X + 2) / 4;
7           newVector.Y = (u.Y + 2) / 4;
8           newVector.Z = (u.Z + 2) / 4;
9           return newVector;
10      }
11
12      public static double AngleFeatureNorm(double n)
13      {
14          return n / Math.PI;
15      }
16
17      public static double DistanceFeatureNorm(double n)
18      {
19          return n / 3.4641f;
```

```
20          }
21      }
```

Kode Sumber 4.5 Implementasi fungsi VectorFeatureNorm(), AngleFeatureNorm(), dan DistanceFeatureNorm() pada class Features

4.3.4 Implementasi Proses Pembuatan *Rule*

Proses pembuatan rule pada aplikasi ini dilakukan dengan membentuk 2 model BPGANN. Untuk model pertama ditentukan dengan melihat nilai Y pada *vector3* ER – WR. Jika Y kurang dari 0.53 maka bahasa isyarat termasuk model 1. Selain itu bahasa isyarat akan masuk model 2. Pada aplikasi ini diberlakukan penyimpanan *dataset* yang sudah ada didalam 2 folder untuk membedakan *dataset* saat proses pelatihan. Folder pertama yaitu "gol1-normalize" yang berisi semua *dataset* yang termasuk dalam model 1 dan "gol2-normalize" yang berisi semuada *dataset* yang termasuk dalam model 2. Implementasi pembuatan *rule* dapat dilihat pada Kode Sumber 4.6.

```
1       int gol = 0;
2       if (ERWR.Y < 0.53)
3           gol = 1;
4       else
5           gol = 2;
6
7       result = true;
8
9       int count = 0;
10      string fullPath;
11      if (gol == 1)
12      {
13          filename = string.Format("gol1.{0}", filename);
14          fullPath = @path + "gol1-normalize\\" + filename;
15      }
16      else
17      {
18          filename = string.Format("gol2.{0}", filename);
19          fullPath = @path + "gol2-normalize\\" + filename;
20      }
21      while (File.Exists(fullPath))
22      {
23          string[] temp = filename.Split('.');
```

```
24          count++;
25          if(gol == 1)
26              filename = String.Format("gol1.{0}.{1}.txt",
27      temp[1], count.ToString());
28          else
29              filename = String.Format("gol2.{0}.{1}.txt",
30      temp[1], count.ToString());
31
32          if (gol == 1)
33              fullPath = @path + "gol1-normalize\\" + filename;
34          else
35              fullPath = @path + "gol2-normalize\\" + filename;
36      }
37      File.AppendAllText(fullPath, sb.ToString());
```

Kode Sumber 4.6 Implementasi Proses Pembuatan Rule

4.3.5 Implementasi Proses, *dan Testing*

Pada proses t*raining* dan *testing* penulis menggunakan *classifier* dengan metode *Back Propagation Genetic Algorithm Neural* Network yang pada Tugas Akhir sebelumnya sudah diimplementasikan oleh Risal Andika Tidisaputra. Dengan melakukan beberapa modifikasi kode, aplikasi ini dapat menggunakannya. Untuk implementasi proses *training* dapat dilihat pada Kode Sumber 4.7. Output dari proses *training* adalah sebuah model BPGANN disimpan dalam format .xml yang nantinya akan digunakan untuk proses *testing*. Implementasi proses *testing* dapat dilihat pada Kode Sumber 4.8.

```
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    GeneticAlgorithm ga1 = new GeneticAlgorithm();
15    GeneticAlgorithm ga2 = new GeneticAlgorithm();
16    GeneticAlgorithm ga1 = new GeneticAlgorithm();
17    GeneticAlgorithm ga2 = new GeneticAlgorithm();
18    ga1.Init(dsl1, cc1);
19    ga2.Init(dsl2, cc2);
20
21    //Chromosom fitChrom1 = ga1.Run();
22    //Chromosom fitChrom2 = ga2.Run();
23
24    loadNet = new NeuralNetwork();
25    NNtoXMLReader nnr = new NNtoXMLReader(loadNet);
26    algoTest = nnr.read("gol1ga.xml");
27    seleksi1 = nnr.chromosom;
28
29    loadNet = new NeuralNetwork();
30    nnr = new NNtoXMLReader(loadNet);
31    algoTest = nnr.read("gol2ga.xml");
32    seleksi2 = nnr.chromosom;
33
34    for(int i = 0; i < dsl1.Count; i++)
35    {
36        int popCount = 0;
37        for(int j = 0; j < seleksi1.Count(); j++)
38        {
39            if(seleksi1[j] == 0)
40            {
41                dsl1[i].RemoveBit(j - popCount);
42                popCount++;
43            }
44        }
45    }
46
47    for (int i = 0; i < dsl2.Count; i++)
48    {
49        int popCount = 0;
50        for (int j = 0; j < seleksi2.Count(); j++)
51        {
52            if (seleksi2[j] == 0)
53            {
54                dsl2[i].RemoveBit(j - popCount);
55                popCount++;
56            }
57        }
58    }
59
60    nn1.InitNetwork(dsl1[0].AttributeCount,
61    dsl1[0].AttributeCount / 2, cc1.TargetCount);
62
```

```
63  nn2.InitNetwork(dsl2[0].AttributeCount,
64  dsl2[0].AttributeCount / 2, cc2.TargetCount);
65  nn1.Seed = 0;
66  nn2.Seed = 0;
67  nn1.InitWeight();
68  nn2.InitWeight();
69  BackPropagation bp1 = new BackPropagation();
70  BackPropagation bp2 = new BackPropagation();
71  bp1.Init(nn1, dsl1, cc1);
72  bp2.Init(nn2, dsl2, cc2);
73  avgError1 = bp1.Run(Int32.Parse(iteration_text.Text));
74  avgError2 = bp2.Run(Int32.Parse(iteration_text.Text));
75  NNtoXMLWriter nnw1 = new NNtoXMLWriter(nn1, avgError1);
76  NNtoXMLWriter nnw2 = new NNtoXMLWriter(nn2, avgError2);
77  nnw1.Write("gol1ga.xml", algorithm, seleksi1);
    nnw2.Write("gol2ga.xml", algorithm, seleksi2);
```

Kode Sumber 4.7 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          xmlName = "gol1ga.xml";
36          gol = 1;
37      }
38      else
39      {
40          xmlName = "gol2ga.xml";
41          gol = 2;
42      }
43      DataSet ds = new DataSet(fitur.Count);
44      for (int i = 0; i < fitur.Count; i++)
45      {
46          ds[i] = fitur[i];
47      }
48
49      dsl.Add(ds);
50
51      FeedForward ff = new FeedForward();
52      loadNet = new NeuralNetwork();
53      NNtoXMLReader nnr = new NNtoXMLReader(loadNet);
54      Translate t = new Translate();
55
56      algoTest = nnr.read(xmlName);
57      for (int i = 0; i < dsl.Count; i++)
58      {
59          if (algoTest.Equals("BPGA"))
60          {
61              int popCount = 0;
62              for (int j = 0; j < nnr.chromosom.Length; j++)
63              {
64                  if (nnr.chromosom[j] == 0)
65                  {
66                      dsl[i].RemoveBit(j - popCount);
67                      popCount++;
68                  }
69              }
70          }
71      }
72
73      loadNet = new NeuralNetwork();
74      nnr = new NNtoXMLReader(loadNet);
75      nnr.read(xmlName);
76
77      ff = new FeedForward();
78      ff.Init(loadNet, dsl);
79      for (int i = 0; i < dsl.Count; i++)
80          ff.Run(i);
```

```
81
82      int[] actualClass = new int[loadNet.OutputLayer.Count];
83      actualClass = ff.GetActualClass();
84
85      t = new Translate(actualClass);
86      string actualClassLog = "";
87      for(int i = 0; i < actualClass.Length; i++)
88          actualClassLog += actualClass[i];
89
90      if (gol == 1)
91          outputText.Content = t.Result(cc1);
92      else if (gol == 2)
93          outputText.Content = t.Result(cc2);
94      string imageFullPath = @imagePath + outputText.Content +
        ".bmp";
95      if (File.Exists(imageFullPath))
96          outputImage.Source = (ImageSource)new
        ImageSourceConverter().ConvertFrom(imageFullPath);
97      Log_ERWRY.Content = ERWR.Y.ToString();
98      outputList.Items.Clear();
99
100     for (int i = 0; i < loadNet.OutputLayer.Count(); i++)
101     {
102         outputList.Items.Add(loadNet.OutputLayer[i].Input);
103     }
```

Kode Sumber 4.8 Implementasi Proses Testing

[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 2.5. Pembagian golongan tersebut bersadarkan ketentuan pada Tabel 3.7.

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

1. Pengujian skenario A1 merupakan pengujian akurasi yang dilakukan oleh penulis dengan menggunakan 160 data *training* yang diambil dari data *skeleton* penulis.
2. Pengujian skenario A2 merupakan pengujian akurasi yang dilakukan oleh pengguna dengan menggunakan 160 data *training* yang diambil dari data *skeleton* penulis.
3. Pengujian skenario B1 merupakan pengujian akurasi yang dilakukan oleh penulis dengan menggunakan 160 data *training* yang diambil dari data *skeleton* penulis dan 80 data *training* yang diambil dari data *skeleton* pengguna.
4. Pengujian skenario B2 merupakan pengujian akurasi yang dilakukan oleh pengguna dengan menggunakan 160 data *training* yang diambil dari data *skeleton* penulis dan 80 data *training* 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 data *training* sebanyak 160 buah, semua *dataset* tersebut dilatih dari *skeleton* penulis. Skenario dapat dilihat pada 5.

Tabel 5.1 Skenario Pengujian A1

Nama Skenario Pengujian	Pengujian Akurasi A1
Kode	SP-A1
Algoritma	*Back Propagation Genetic Algorithm*
Jumlah *Dataset*	160 *dataset*
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 data *training* 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	*Back Propagation Genetic Algorithm*
Jumlah *Dataset*	160 *dataset*
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. Lalu pada isyarat Bentuk ketika diterjemahkan mengeluarkan output isyarat Gang. Kemudian pada isyarat Hai ketika dierjemahkan mengeluarkan output isyarat Bentuk. Dan pada isyarat Hamba ketika diterjemahkan mengeluarkan output isyarat wadah. Kesalahan output ini terjadi karena bahasa isyarat yang dilakukan pengguna tidak sempurna. Perbedaan kecil yang terdapat pada gerakan isyarat sangat mempengaruhi model BPGANN dalam menerjemahkan isyarat tersebut. Hasil dari skenario pengujian A2 dapat dilihat pada Tabel 5.4.

Tabel 5.4 Terjemahan Isyarat Kata Skenario Pengujian A2

		Target Kelas							
		Alquran	Bentuk	Gang	Hai	Hamba	Hormat	Ketua	Wadah
Hasil Uji Coba	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 data *training* 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 fitur. 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	*Back Propagation Genetic Algorithm*
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							
		Alquran	Bentuk	Gang	Hai	Hamba	Hormat	Ketua	Wadah
Hasil Uji Coba	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 data *training* 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 data *training* 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	*Back Propagation Genetic Algorithm*

Jumlah *Dataset*	240 *Dataset*
Penguji	Pengguna
Prosedur Pengujian	Pengguna melakukan uji coba 8 kata isyarat sebanyak 5 kali
Hasil yang Diperoleh	Akurasi 95%

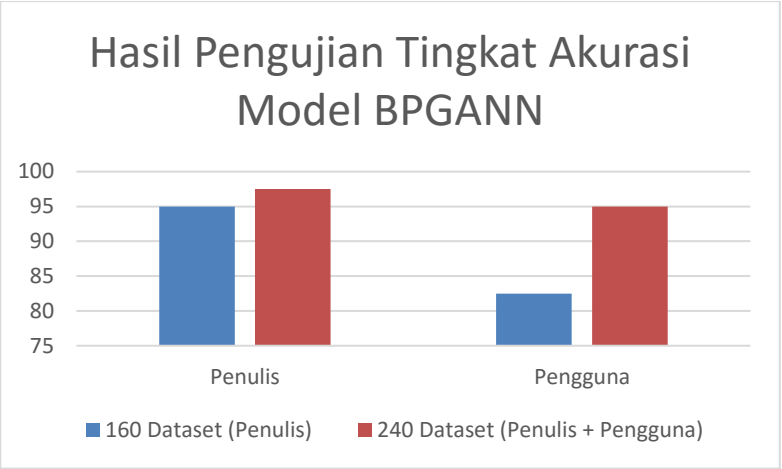
Tabel 5.8 Terjemahan Isyarat Kata Skenario Pengujian B2

		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 model BPGANN yang telah dilakukan sebelumnya.

Evaluasi pengujian akurasi model BPGANN 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 5.1. Dari data yang terdapat pada grafik tersebut, diketahui bahwa aplikasi yang dibuat telah memiliki akurasi rata-rata yang cukup baik yaitu 92.5%.



Gambar 5.1 Hasil Pengujian Tingkat Akurasi Model BPGANN

LAMPIRAN A
KODE SUMBER [13]

```
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;
16
17      public void Init(NeuralNetwork nn, DataSetList dsl,
    ClassificationClass cc)
18      {
20          feedForward = new FeedForward();
21          feedForward.Init(nn, dsl);
22          classificationClass = cc;
23      }
24
25      public float Run(int maxIter)
        {
26          float lastAvgerror = float.MaxValue;
27          float avgError = new float();
28          for(int iter = 0; iter < maxIter; iter++)
29          {
30              avgError = 0;
31              for(int k = 0; k < this.dsl.Count; k++)
32              {
33                  Network.InitInput();
34                  feedForward.Run(k);
35
36                  float[] errorOutput = GetErrorOutput(k);
37                  float totalError = 0;
38                  for (int i = 0; i < errorOutput.Length; i++)
39                      totalError += Math.Abs(errorOutput[i]);
40                  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          }
16          get
17          {
18              return _attribute[index];
20          }
21      }
22
23      public int AttributeCount
24      {
25          get
26          {
27              return _attribute.Count;
28          }
29      }
30
31      public DataSet()
32      {
33          _attribute = new List<float>();
34      }
35
36      public DataSet(int attributeCount)
37      {
38          _attribute = new List<float>();
39          for (int i = 0; i < attributeCount; i++)
40              _attribute.Add(0);
41      }
42
43      public DataSet(DataSet ds)
44      {
45          _attribute = new List<float>();
46          for (int i = 0; i < ds.AttributeCount; i++)
47              _attribute.Add(ds[i]);
48          this.ClassName = ds.ClassName;
49      }
50
51      public void RemoveBit(int index)
52      {
53          _attribute.RemoveAt(index);
54      }
55  }
```

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)
16      {
17          neuralNetwork = nn;
18          _dsl = dsl;
20      }
21
22      public void Run()
23      {
24          for(int i = 0; i < _dsl.Count; i++)
25          {
26              neuralNetwork.InitInput();
27              DoInputLayer(i);
28          }
29      }
30
31      public void Run(int index)
32      {
33          neuralNetwork.InitInput();
34          DoInputLayer(index);
35      }
36
37      private void DoInputLayer(int index)
38      {
39          for (int i = 0; i < _dsl[index].AttributeCount; i++)
40              neuralNetwork.InputLayer[i].Input =
    _dsl[index][i];
41          DoHiddenLayer();
42      }
43      private void DoHiddenLayer()
44      {
45
```

```
            for (int i = 0; i < neuralNetwork.InputLayer.Count;
46      i++)
                for (int j = 0; j <
47      neuralNetwork.HiddenLayer.Count; j++)
                    neuralNetwork.HiddenLayer[j].Input +=
48      neuralNetwork.InputLayer[i].GetOutput(j);
49
            for (int j = 0; j < neuralNetwork.HiddenLayer.Count;
50      j++)
                neuralNetwork.HiddenLayer[j].Input =
51      Sigmoid(neuralNetwork.HiddenLayer[j].Input);
52          DoOutputLayer();
53      }
54
55      private void DoOutputLayer()
56      {
            for (int i = 0; i < neuralNetwork.HiddenLayer.Count;
57      i++)
                for (int j = 0; j <
58      neuralNetwork.OutputLayer.Count; j++)
                    neuralNetwork.OutputLayer[j].Input +=
        neuralNetwork.HiddenLayer[i].Input *
59      neuralNetwork.HiddenLayer[i].GetWeight(j);
60      }
61
62      public int[] GetActualClass()
63      {
            int[] act = new
64      int[neuralNetwork.OutputLayer.Count];
            for(int j = 0; j < neuralNetwork.OutputLayer.Count;
65      j++)
66          {
                if (Math.Abs(neuralNetwork.OutputLayer[j].Input)
67      < 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> chromosoms;
10
11      public void Init(DataSetList dsl, ClassificationClass
    cc)
12      {
13          dataSetList = dsl;
14          classificationClass = cc;
15      }
16
17      public Chromosom Run()
18      {
16          ChromosomInit();
17
18          int iteration = 2;
20          for(int i = 0; i < iteration; i++)
21          {
22              DoBackPropagation();
23              DoSelection();
24              DoCrossOver();
25
26              int chanceMutation = GetRandom();
27              if (chanceMutation < GetRandom())
28                  DoMutation();
29          }
30
31          int index = 0;
32          for(int i = 1; i < chromosoms.Count; i++)
33          {
                if (chromosoms[i].FitnessValue >
34  chromosoms[index].FitnessValue)
                    index = i;
35          }
36
37          Chromosom fittestChromosom = chromosoms[index];
38          return fittestChromosom;
39      }
40
41      private void ChromosomInit()
42      {
43          chromosoms = new List<Chromosom>(individualCount);
```

```
44          for(int i = 0; i < individualCount; i++)
45          {
46              chromosoms.Add(new
47      Chromosom(GetRandomBinary()));
            }
48      }
49
50      private void DoBackPropagation()
51      {
52          for(int i = 0; i < chromosoms.Count; i++)
53          {
54              DataSetList dsl = new
55      DataSetList(this.dataSetList);

56              for(int j = 0; j < dsl.Count; j++)
57              {
58                  int popCount = 0;
59                  for(int k = 0; k < chromosoms[i].Length;
60      k++)
                    {
61                      if(chromosoms[i][k] == 0)
62                      {
63                          dsl[j].RemoveBit(k - popCount);
64                          popCount++;
65                      }
66                  }
67              }
68
69              NeuralNetwork nn = new NeuralNetwork();
70              nn.InitNetwork(dsl[0].AttributeCount,
71      dsl[0].AttributeCount / 2, classificationClass.TargetCount);
                nn.InitWeight();
72
73              BackPropagation bp = new BackPropagation();
74              bp.Init(nn, dsl, classificationClass);
75              bp.Run(5000);
76
77              FeedForward ff = new FeedForward();
78              ff.Init(nn, dsl);
79
80              int totalCorrect = 0;
81              for(int j = 0; j < dsl.Count; j++)
82              {
83                  ff.Run(j);
84                  bool correct = true;
85                  int[] targetClass =
86      classificationClass.GetTarget(dsl[i].ClassName);
                    for (int k = 0; k <
87      ff.GetActualClass().Length; k++)
```

```
                {
                    if (targetClass[k] !=
ff.GetActualClass()[k])
                        correct = false;
                }
                if (correct)
                    totalCorrect++;
            }
            chromosoms[i].FitnessValue = totalCorrect /
(float)dsl.Count;
        }
    }

    private void DoSelection()
    {
        chromosoms.Sort((val1, val2) =>
val1.FitnessValue.CompareTo(val2.FitnessValue));
    }

    private void DoCrossOver()
    {
        for(int i = 0; i < chromosoms.Count; i+= 2)
        {
            int nextindex = i + 1;
            if (nextindex >= chromosoms.Count)
                nextindex = i - 1;
            int[] bit1 = chromosoms[i].Bit;
            int[] bit2 = chromosoms[nextindex].Bit;

            for(int j = bit1.Length / 2; j < bit1.Length;
j++)
            {
                int temp = bit1[j];
                bit1[j] = bit2[j];
                bit2[j] = temp;
            }

            chromosoms[i].Bit = bit1;
            chromosoms[nextindex].Bit = bit2;
        }
    }

    private void DoMutation()
    {
        for(int i = 0; i < chromosoms.Count; i++)
        {
            int chanceChromosomMutation = GetRandom();
            if(chanceChromosomMutation <= GetRandom())
            {
```

```
                int bitIndex =
GetRandom(chromosoms[i].Length);
                if (chromosoms[i][bitIndex] == 0)
                    chromosoms[i][bitIndex] = 1;
                else
                    chromosoms[i][bitIndex] = 0;
            }
        }
    }

    private int[] GetRandomBinary()
    {
        int[] chromosomTemp = new
int[dataSetList[0].AttributeCount];
        for (int i = 0; i < dataSetList[0].AttributeCount;
i++)
        {
            int ran = random.Next(0, 2);
            chromosomTemp[i] = ran;
        }
        return chromosomTemp;
    }

    private int GetRandom(int max)
    {
        return random.Next(0, max);
    }

    private int GetRandom()
    {
        return random.Next(0, 100);
    }
}
```

Kode Sumber A.6 Kelas GeneticAlgorithm.cs

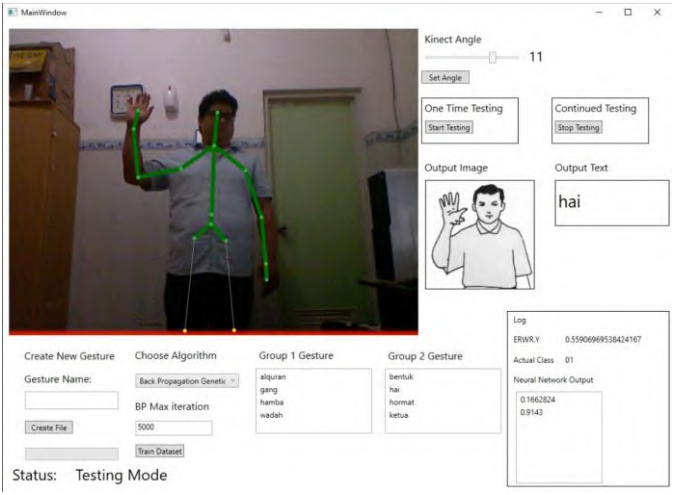
```
class NeuralNetwork
{
    private Random random = new Random();

    public int Seed
    {
        set { random = new Random(value); }
    }

    public float threshold  = 0.5f;

    public List<Node> InputLayer { set; get; }
    public List<Node> HiddenLayer { set; get; }
```

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

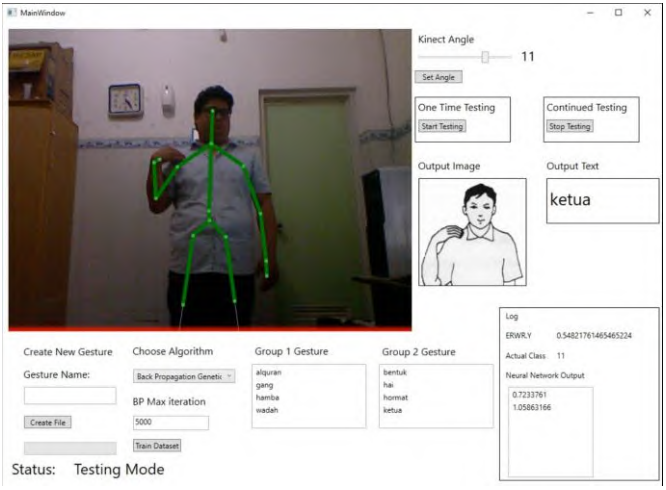
```
58          {
59              OutputLayer.Add(new Node(1));
60              OutputLayer[i].SetWeight(0, 1);
61          }
62      }
63
64      public void InitWeight()
65      {
66          for (int i = 0; i < InputLayer.Count; i++)
67              for(int j = 0; j < HiddenLayer.Count; j++)
68                  InputLayer[i].SetWeight(j, GetRandom() /
69  100.0f);

70          for (int i = 0; i < HiddenLayer.Count; i++)
71              for (int j = 0; j < OutputLayer.Count; j++)
72                  HiddenLayer[i].SetWeight(j, GetRandom() /
73  100.0f);
      }
74
75      int GetRandom()
76      {
77          return random.Next(0, 30);
78      }
79  }
80
```

Kode Sumber A.7 Kelas Neural Network.cs

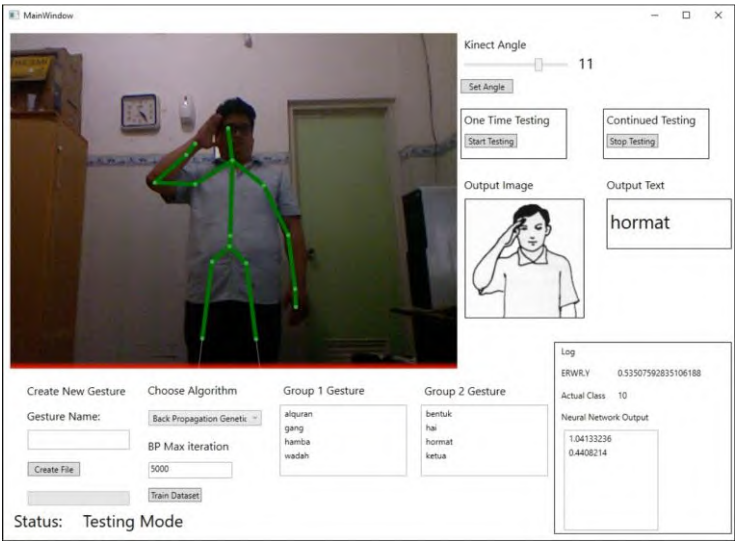
LAMPIRAN B
SCREENSHOT APLIKASI



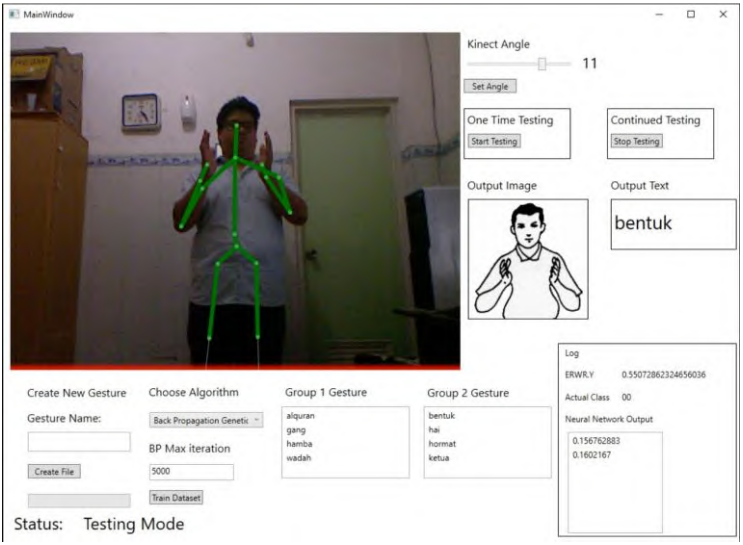
Gambar B.1 Aplikasi Mendeteksi Bahasa Isyarat Hai



Gambar B.2 Aplikasi Mendeteksi Bahasa Isyarat Ketua

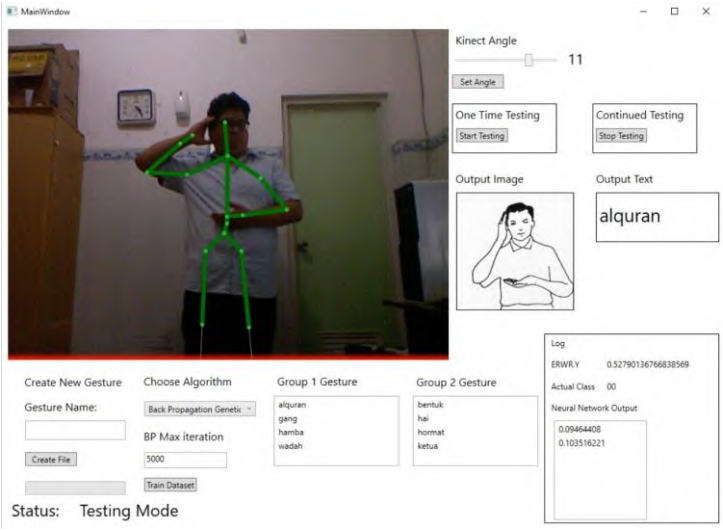


Gambar B.3 Aplikasi Mendeteksi Bahasa Isyarat Hormat

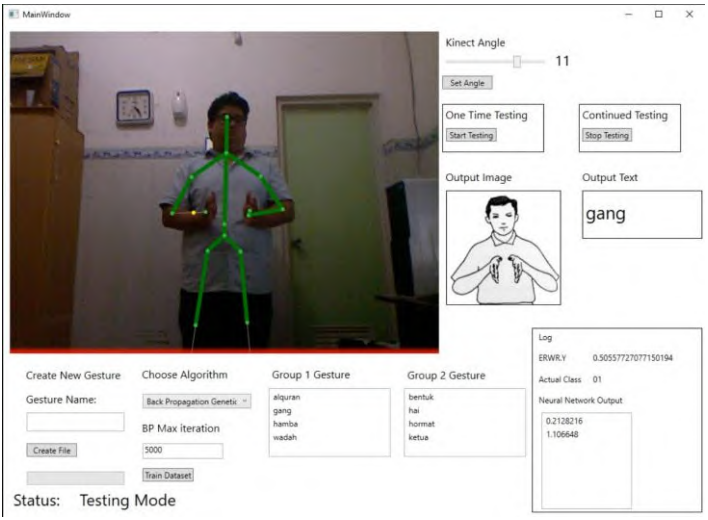


Gambar B.4 Aplikasi Mendeteksi Bahasa Isyarat Bentuk

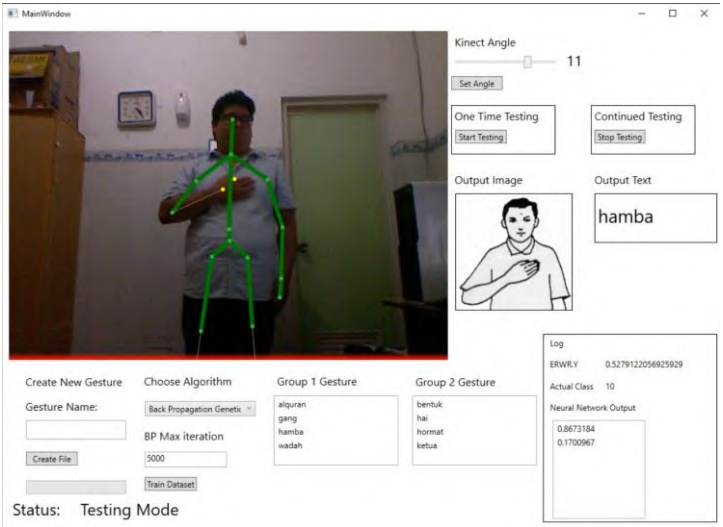
Gambar B.5 Aplikasi Mendeteksi Bahasa Isyarat Wadah



Gambar B.6 Aplikasi Mendeteksi Bahasa Isyarat Alquran



Gambar B.7 Aplikasi Mendeteksi Bahasa Isyarat Gang



Gambar B.8 Aplikasi Mendeteksi Bahasa Isyarat Hamba

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. Fitur yang digunakan untuk mendeteksi bahasa isyarat yang didapat dengan Kinect ada 15 fitur yaitu 7 fitur *vector3*, 6 fitur *angle*, dan 1 fitur *distance*. Semua fitur tersebut didapat dengan mengolah data dari *9 skeleton joints* yaitu bahu tengah, bahu kanan, bahu kiri, siku tangan kanan, siku tangan kiri, pergelangan tangan kanan, pergelangan tangan kiri, tangan kanan, dan tangan kiri. Sembilan *skeleton joints* tersebut dipakai karena kebanyakan bahasa isyarat memiliki perbedaan satu sama lainnya dengan melihat 9 *skeleton joints* tersebut.
2. Untuk menerapkan metode *Back Propagation Genetic Algorithm Neural Network*, jumlah *input neuron* ditentukan berdasarkan jumlah fitur yang sebelumnya telah diseleksi dengan *BPGA*
3. Aplikasi yang dibangun pada tugas akhir ini dapat menerjemahkan bahasa isyarat pokok dengan akurasi rata-rata 92.5%.
4. Perbedaan yang kecil pada gerakan isyarat yang dibentuk dapat mengakibatkan perbedaan output.
5. Semakin bervariasinya data *training* yang digunakan akurasi model BPGANN semakin meningkat.

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 jari. Untuk membuat penelitian yang sama lebih baik menggunakan Kinect V2 sehingga fitur yang digunakan akan lebih baik dan akurat.
2. Menggunakan data *training* dari berbagai macam pengguna sehingga dalam proses *training* menghasilkan model BPGANN yang memiliki akurasi tinggi.
3. Perlu adanya ektraksi fitur yang dapat digunakan untuk mendeteksi bahasa isyarat dinamis.
4. Menggunakan *depth data* yang didapat dengan Kinect unuk mendeteksi bahasa isyarat yang berpengaruh terhadap jarak.

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/..

[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] Venkatadri.M, Lokanatha C. Reddy, "A Comparative Study on Decision Tree Classification Algorithms in Data Mining," *International Journal of Computer Applications in Engineering, Technology and Sciences (IJ-CA-ETS),* vol. 2, no. 2, p. 24, 2010.

[9] J. Burger, "A Basic Introduction To Neural Networks," Universiy of Wisconsin-Madison, [Online]. Available: http://pages.cs.wisc.edu/~bolo/shipyard/neural/local.html. [Accessed 11 Juni 2016].

[10] C. MacLeod, “The Synthesis of Artificial Neural Network Using Single String Evolutionary Techniques,” Robert Gordon University, Aberdeen, 1999.

[11] S. Wang, S. Yin and M. Jiang, "Hybrid Neural Network Based On GA," in *Fourth International Conference on Natural Computation*, 2008.

[12] Chao Sun, Tianzhu Zhang, Bing-Kun Bao, Changsheng Xu, “Discriminative Exemplar Coding for Sign Language,” *IEEE TRANSACTIONS ON CYBERNETICS*, vol. 43, p. 1418, 2013.

[13] R. A. T, "Rancang Bangun Modul Pengenalan Bahasa Isyarat Menggunakan Teknologi Leap Motion dan Metode Back Popagation - Genetic Algorithm Neural Network (BPGANN)," Institut Teknologi Sepuluh Nopember, Surabaya, 2015.

BIODATA PENULIS

Penulis lahir di kota Bekasi pada tanggal 30 Juni 1994, merupakan anak keempat dari 4 bersaudara. Hobi ang dimiliki penulis antara lain bermain game dan mendengarkan musik. Penulis telah menempuh pendidikan formal yaitu TK Marsudirini Bekasi (1998 – 2000), SD Marsudirini Bekasi (2000 – 2006), SMP Marsudirini Bekasi (2006 – 2009), SMA Marsudirini Bekasi (2009 – 2012), dan mahasiswa S1 Jurusan Teknik Informatika Fakultas Teknologi Informasi Institut Teknologi Sepuluh Nopember Surabaya rumpun mata kuliah Interaksi, Grafika, dan Seni.

Penulis 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. Penulis dapat dihubungi melalui surel yohanes.aditya94@gmail.com.