2012 Intel Cup Undergraduate Electronic Design Contest

- Embedded System Design Invitational Contest

Final Report

Intel Cup Embedded System Design Contest

Project Name: Improving Communication Ability of the Disabled — Chinese Sign Language Recognition and Translation System

Students: Chan Chun Kit, Shu Jianfei, Liu Ruifeng

Faculty Mentor: Prof. Xu Qiang

University: The Chinese University of Hong Kong

2012 Intel Cup Undergraduate Electronic Design Contest

- Embedded System Design Invitational Contest

Declaration of Originality

We hereby declare that this thesis and the work reported herein was composed and originated entirely by ourselves. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given in the references.

Team Members Signature: _____

Name (in Block Letters): CHAN CHUN KIT, SHU JIANFEI, LIU RUIFENG

Date: 30 June, 2012

IMPROVING COMMUNICATION ABILITY OF THE DISABLED — CHINESE SIGN LANGUAGE RECOGNITION AND TRANSLATION SYSTEM

ABSTRACT

Sign language enables the deaf and the mute to “talk” to each other. However, most normal people don’t know sign language and it remains difficult for them to communicate with the disabled. On the other hand, teaching deaf or mute children sign language is a challenging task due to their learning difficulties and the short of capable teachers. Motivated by the above, the objective of this project is to bridge the communication gap between the disabled and normal people, by translating Chinese sign languages into other forms such as text and sound. With the help of Intel Atom-based platform and Microsoft Kinect, we record the 3D position of hand gestures of sign languages, recognize them with learning-based techniques, and then translate them into recognized forms by the disabled. Various optimization techniques have been utilized to achieve the above objective under the computational capability constraint of the given platform.

Key words: Chinese Sign Language, 3D Gesture Recognition, Skeleton Tracing, Support Vector Machine, Machine Learning, FPGA

Content

1 Introduction

1.1 Motivation

Sign language is a major communication tool for the deaf and the mute, which uses manual communication and body language to convey meaning. It helps sign language users communicate with each other. Nevertheless, it is also an obstacle for hearing-impaired people to communicate with healthy people.

The communication barrier happens since most people don't know sign language [1], so it is often very difficult for deaf people to communicate with others. On the other hand, deaf children have learning difficulties since they cannot express themselves easily and teachers are limited in China.

In order to improve these situations, we design and implement a translator converting Chinese Sign Language to text and sound which makes learn sign language more interesting, convenient and easier. The aim of this project is to bridge the communication gap between the disabled and other people.

1.2 System Schema

1.2.1 Overview

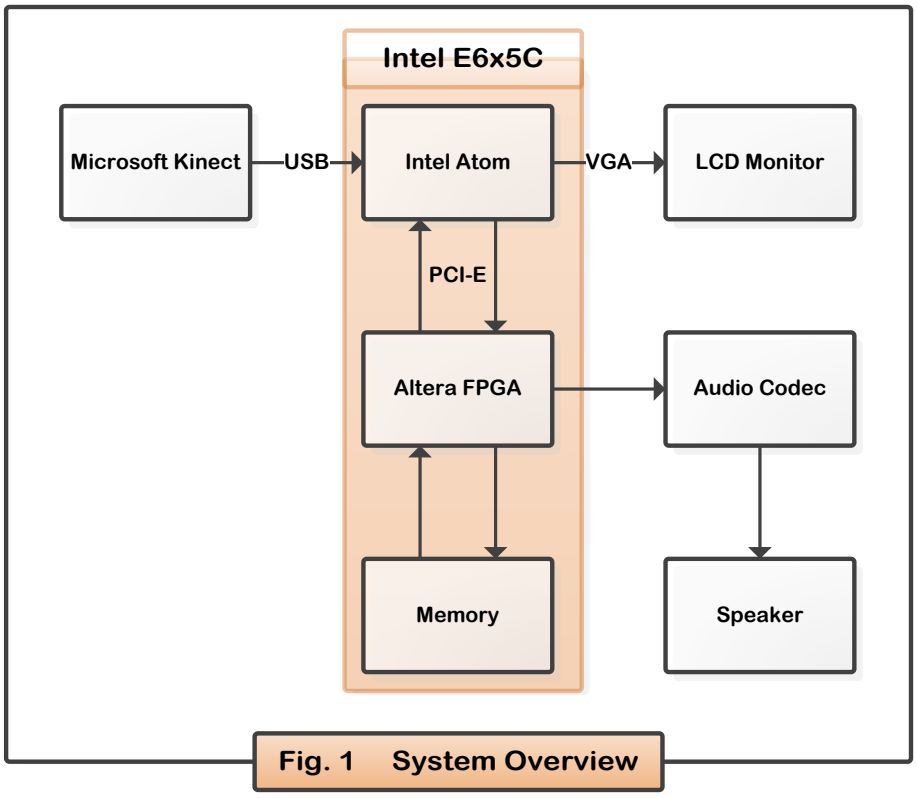
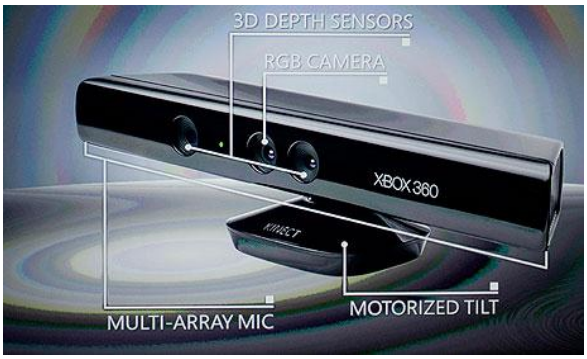


Fig. 1 System Overview

1.2.2 Microsoft Kinect



Kinect consists of different sensors such as depth camera, RGB camera and microphones. We mainly use the depth camera which can extract the 3D image from the scene. OpenNI is an open source framework which provides APIs to control supported natural interaction devices. In this project, we use OpenNI to control and get the skeletons from Kinect [2]. It provides a middleware API called NiTE which can analyze raw data and extract the positions of skeletons [3].

1.2.3 FPGA-based MP3 word speaker

In order to help a mute person “speak words” or make people hear gesture words, we design an MP3 word speaker using FPGA and peripheral audio to play the sound of gestures. Once the gesture is recognized, its corresonding sound will be played.

1.2.4 Equipment List

Name	Function	Port
Intel Atom	Run Operation System	
Altera FPGA	Decode Audio	
LCD monitor	Display Result	VGA
Microsoft Kinect	Get 3D Depth Data	USB
Speaker	Play sound of words	3.5mm Jack

2 User Interface

2.1 Status Bar

The status bar shows the current state of the Kinect. For example, “Recording” means the Kinect is recording the gestures. “Please Do Ready Pose” tells the user to do the ready pose.

2.2 Word Panel

After the Kinect records the action of the hand sign, it will translate the hand sign into Chinese words and sound will be played automatically. The right part shows the history.

2.3 Sentence Panel

The sentence panel can record several hand sign actions and output a sentence. As it might have some similar signs, several possible sentences are available for user to choose while the most possible one displays on top of the list.

2.4 Word List

The word list shows all the words in our database. It also shows gestures of the words in the right part.

2.5 Hand Sign Learning Game

After you click the start button, a picture comes out and shows the gesture of a Chinese word accordingly. If you do the correct action, it will move to the next one. If not, it will stop until you do a correct action.

3 Working Principle and Algorithm

3.1 Feature Extraction

The resolution of depth map is 640 x 480 where its depth is represented by 16 bit integer. It means that there are 17.6 MB data per second, so it is not possible for saving such huge data set. At the same time, raw depth data is not meaningful to the movement and many useless data such as background are included to the raw data. Therefore, we have to extract useful information from the raw depth map.

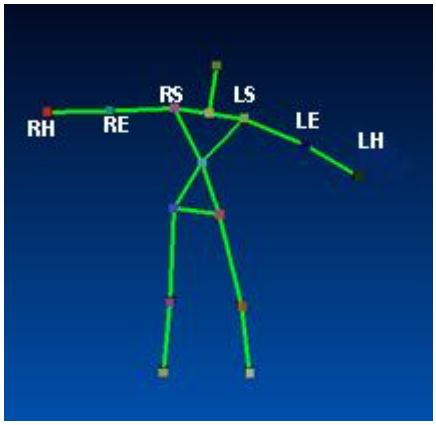


Fig. 2 Image of Position of Skeleton

Skeleton is one of the useful information that can be used to represent hand signs. We use Kinect and OpenNI to capture skeleton joints of the user and get the x, y and z coordinates of these joints. We have 6 joints in total -- left hand, left elbow, left shoulder, right hand, right shoulder, right elbow and right hand. For each frame, we compute 14 features from these joints to represent the hand sign. After normalization, each word has 121 frames, thus a word has 1694 features.

Features	Description	Vector Count
A	Distance between RH and LH over average of length of two forearms	1
B	3D Unit vector from LS to LH	3
C	3D Unit vector from RS to RH	3
D	3D Unit vector from RH to LH	3
E	Angle in radian between vector LE→LH and LE→LS	1
F	Angle in radian between vector RE→RH and RE→RS	1
G	Angle in radian between vector LE→LS and vertical	1
H	Angle in radian between vector RE→RS and vertical	1

Table 1 List of Skeleton Features

3.2 Normalization for Raw Data

We adopt two-stepped normalization which is distribution and interpolation.

Distribution is trying to evenly distribute the raw data. In Figure 3, the first and last data is mapped to the same location in processed data. The second raw data is mapped to most suitable location which is the third position in processed one.

Interpolation is to find the most suitable estimated value for the empty intervals. We make use of quadratic interpolation which estimates values by using the nearest three raw data.

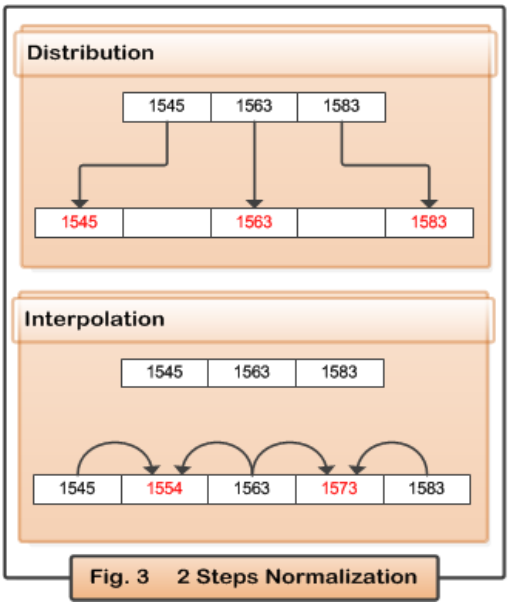


Fig. 3 2 Steps Normalization

3.3 Recognizing Gestures

Support Vector Machine (SVM) is a concept in Statistics and Computer Science for a set of related supervised learning methods that analyze data, used for classification and regression analysis. Given a set of training examples and each marked as belonging to one of the classes, an SVM training algorithm builds a model that assigns new examples into one of the classes.

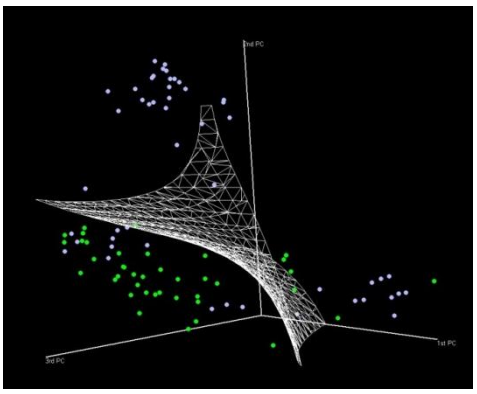
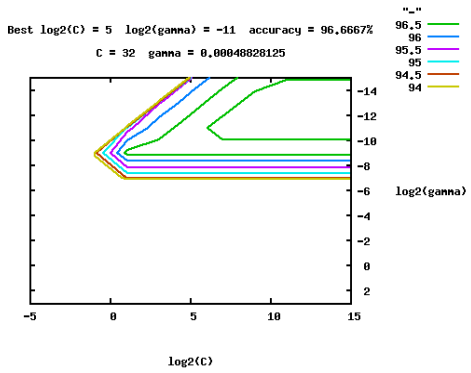


Fig. 4 Visual Classification of SVM

An SVM model is a representation of the examples as points in space so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a class based on which side of the gap they fall on.

LIBSVM has four basic kernels and the Radial Basis Function (RBF) kernel is often the best choice. It can map sample data into high dimensional space so that even when the relation between class labels and attributes is nonlinear, it still works well. [4]

There are two parameters for an RBF kernel: C and γ. We can use Cross-validation (which can prevent over-fitting problem) and Grid-search (which can discretize the diagram) to get them. LIBSVM provides a tool to compute them, so we can use the library to get the desirable result.



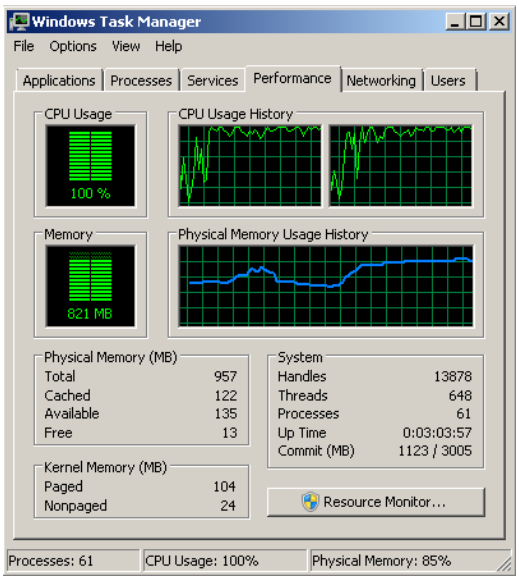
$$(\text{RBF}):\ K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2),\ \gamma > 0.$$

Fig. 5 Formula of RBF

4 Challenges and Solutions

4.1 The Challenge

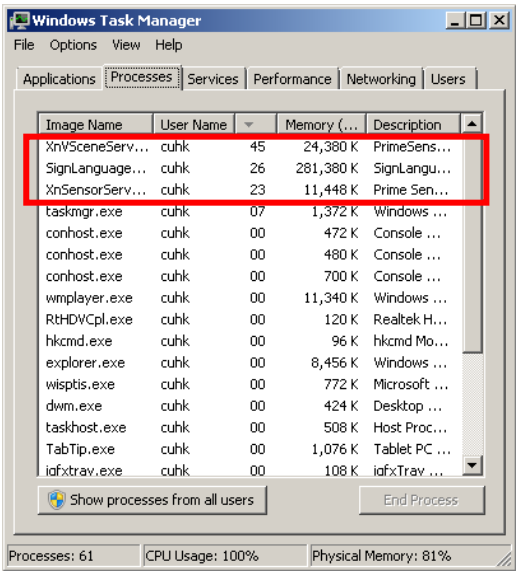
The recommended hardware requirement for operating Kinect application is using dual-core 2.66GHz or faster CPU and 2GB RAM [5]. It is clear that our platform cannot fulfill this requirement.



Even we use Linux to run the application, CPU usage is still 100% and the frame per second (FPS) is 2 to 4 which is much lower than the normal 30 FPS. It is also not possible for us to process the raw data in CPU as CPU is occupied by the thread capturing raw data form Kinect.

4.2 The Bottleneck

We have to verify which part is the source of high CPU usage. After disabling the GUI, the CPU usage dropped to about 85% which is a bit lower but FPS is still the same as before. We observe the reduction is due to a single-thread process by OpenNI which occupying whole thread. As our CPU has two threads, this process is bounded by 50% where another thread uses 35% CPU.



What we next to do is disabling the skeleton extraction, it is surprised that CPU usage dropped to about 25%. After profiling, we confirm that the bottleneck is extracting skeleton. However, the algorithm is protected by PrimeSence and is not open-sourced. That's why we cannot move the code to FPGA in order to reduce the CPU usage.

4.3 The Proposed Solution

To solve the problem, we use a trick that selectively extracts the skeleton. Original data flow is User generator (which provides skeleton) directly retrieves raw data from Depth generator. Now we add a generator called Mock Depth generator which simulates the behaviors of original Depth generator and then User generator retrieves raw data from Mock Depth one instead.

Once Kinect has new data, it will transfer to Depth Generator and redirect raw data to Mock Depth one. Only for every other frame, Mock Depth generator will transfer the data to User generator. When depth data comes, Mock Depth generator will redirect data to User generator for every two frame while another frame is dropped and not processed as shown in Figure 6.

The measures can maintain skeleton extraction at 15 FPS and 67% CPU usage. It makes using Kinect on our platform feasible and even provides spare computation power of CPU for other processing works.

	Original	Disable GUI	Disable Skeleton	Modified with Skeleton
FPS	3	3	30	**15**
Thread 1	50 %	50 %	20 %	**44 %**
Thread 2	50 %	35 %	5 %	**23 %**
Total	**100 %**	**85 %**	**25 %**	**67 %**

Table 2 CPU Usage and FPS with different setting

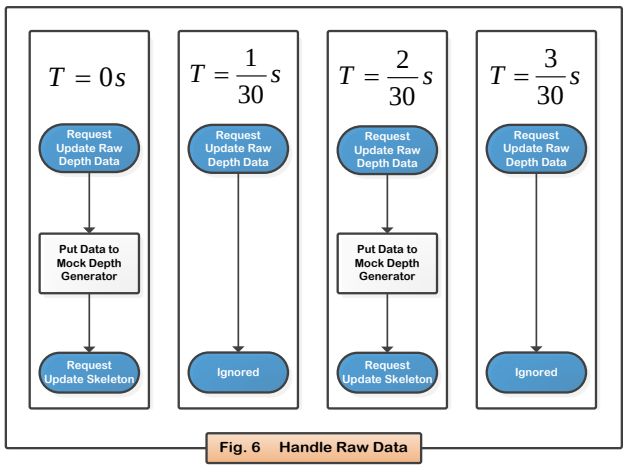
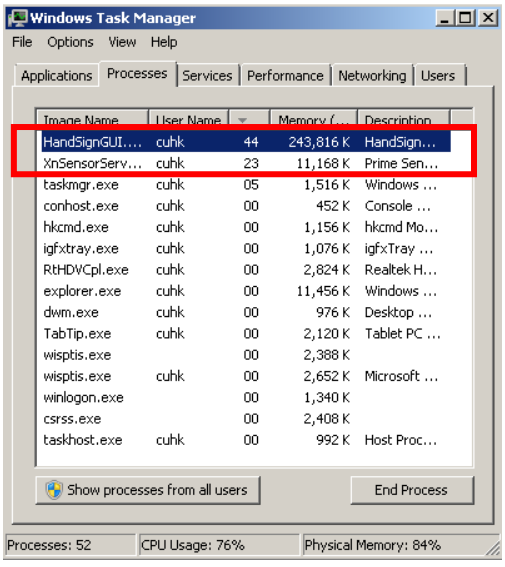


Fig. 6 Handle Raw Data



5 Design and Implementation

5.1 Overview

Once received depth raw data, the area and skeleton of the user will be extracted by NiTE. We only save positions of several skeleton joints interested which are hands, elbows and shoulders. After the user completes whole hand sign, raw data will be normalized and values of different features will be calculated. The system uses these feature vectors to classify which hand sign corresponding to captured data using SVM.

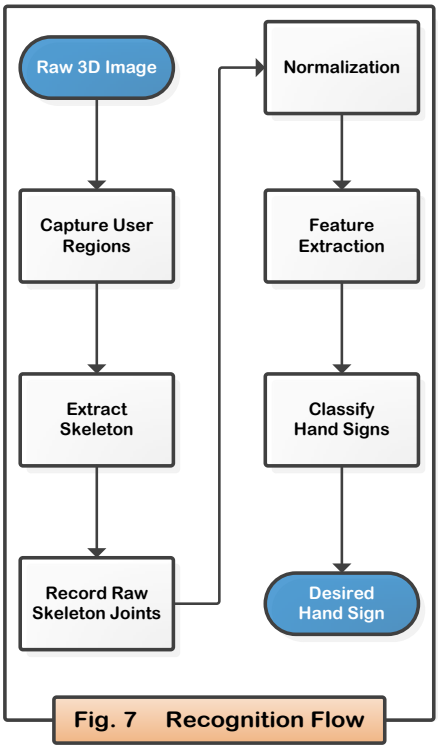


Fig. 7 Recognition Flow

5.2 Capture Sign Language

It is difficult to determine when the hand sign starts and ends so we adopt the following method to solve this problem. We introduce a pose called “Ready Pose” which puts both two hands between torse and hip. Once any hand moves after doing “Ready Pose”, we consider the user has started his sign. In order to tolerate potential errors of skeleton, user must move at least 5 cm in 2 frames (0.13 s).

The movement will be captured until “Ready Pose” is done again and not moving for 2 frames. It considers as end of one sign and the system will start processing captured raw data. The state is repeated so the capture of sign language can be continuous.

If user does a “Ready Pose” and stops for over 100 frames (6.67 s), the capture will be stopped. It is useful when user wants to switch to other functions.

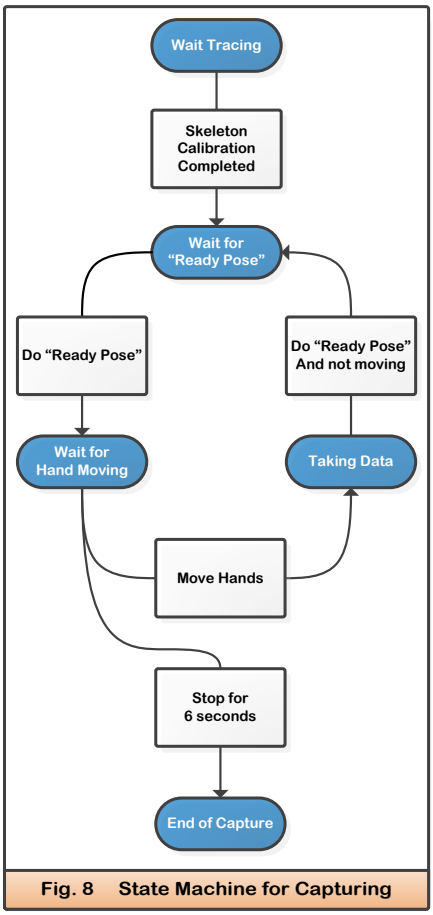


Fig. 8 State Machine for Capturing

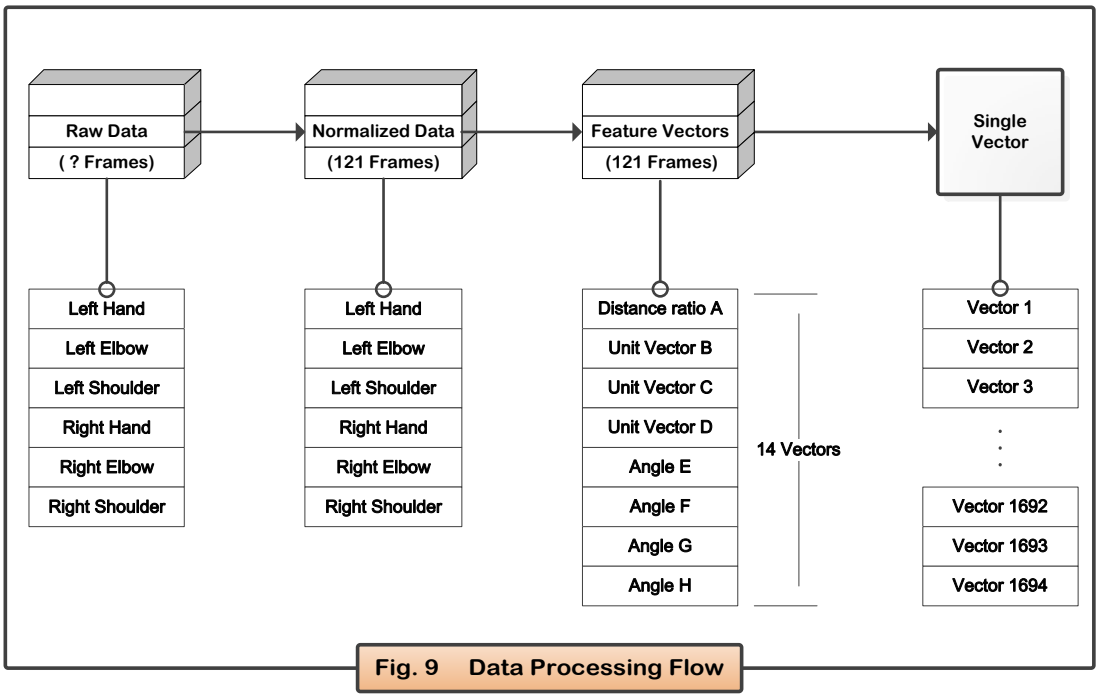


Fig. 9 Data Processing Flow

5.3 Processing Raw Data

After receiving the raw data, we will first normalize data to 121 frames with distribution and interpolation as described previously. The normalized data consist of 2178 vectors. Different features are then calculated and 1694 feature vectors are produced. Finally, all feature vectors is put into the single vector for SVM to classify.

5.4 Classifying and Training Hand Signs

LIBSVM is an integrated software for support vector classification. It helps users from other fields to use SVM as a tool easily. It supports multi-class classification. “svm-train” can read your input files and generate a “Model” file.

Our input file contains 1694 feature vectors for each sample, 20 to 30 samples for each word and in total 30 words up to now. The database can be enlarged after training more words so our database will contain more words after.

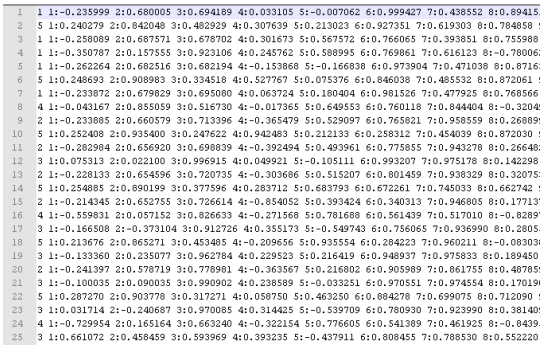


Fig. 10 Input Data

We use the trained model and "svm-predict" to predict which class it belongs to. LIBSVM provides a simple interface for users integrating the library with their own programs. We train the data and store the trained model in our system, and then add the prediction part to our program.

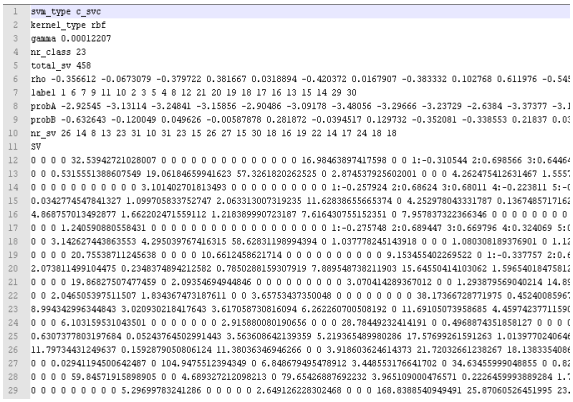


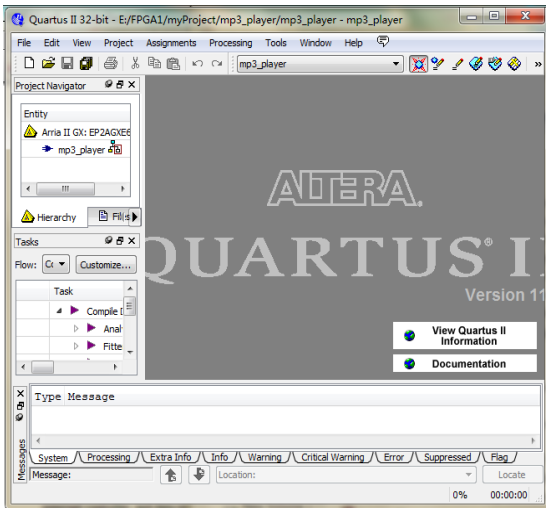
Fig. 11 Trained Model

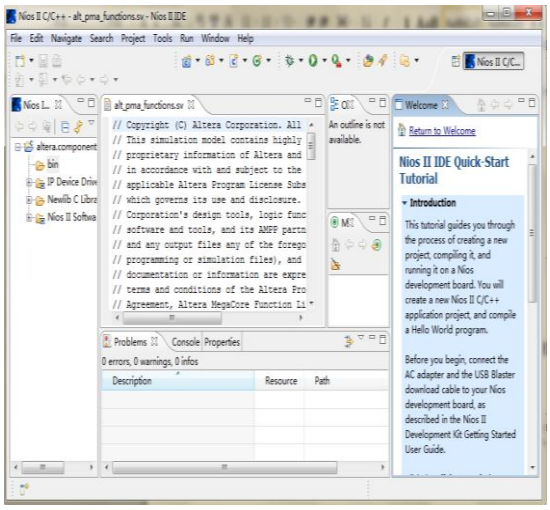
5.5 FPGA-based MP3 word speaker design

The language used for programming the components is VHDL. The language used to program the Nios II chip is C/C++. We refer to the design of Altera MP3 design [6] and other design [7].

5.5.1 Quartus II and Nios II EDS

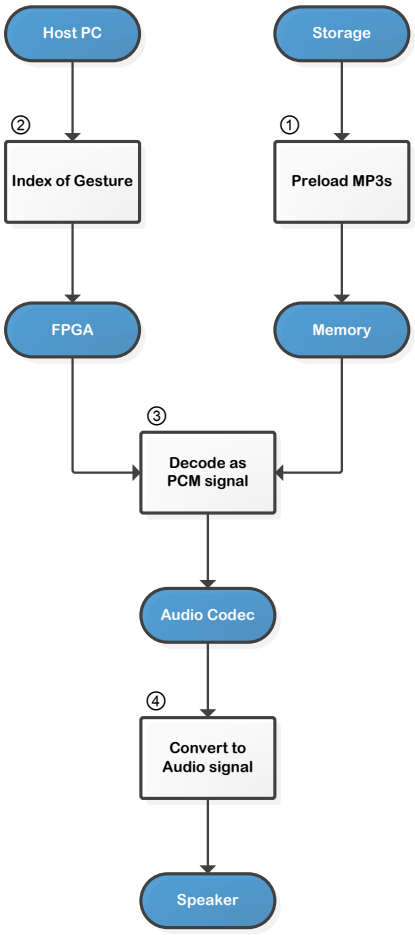
Quartus II is the main software tool to build our FPGA design. We use the Qsys tool that specifies IP cores such as Nios II processor core, memory and other peripherals. We use pin assignment editor to implement the assignment of the pins and then download the design on the board. Qsys generates the Nios II processor system with the desire peripherals, designs the communication interface for the components, specifies the base addresses and interrupts request through Avalon-MM or Avalon-ST.





5.5.2 Audio Data Flow

1. Preload mp3 files onto memory chip.
2. When a gesture is recognized, Host PC sends a signal “index” into FPGA and FPGA starts to decode the corresponding sound of the gesture. PCM format signal is generated.
3. The Audio Codec chip converts the PCM signal into audio signal.
4. The external speaker responds to the audio signal through the audio jack.



6 Test Plans and Result Analysis

6.1 Support Vector Machine Accuracy Test

As mentioned before, we use LIBSVM to predict which hand sign it is after an action is performed. In another word, we use LIBSVM to predict which class it belongs to when we get a new data according to our original database. Now we are going to test the accuracy of LIBSVM for our database.

Database	660 samples for 30 words
Procedure	1. Use cross-validation to get the best parameter C and γ 2. Use the parameter C and γ to train the training database and generate the model file 3. Use the model file and svm-predict to predict the origin database and get the accuracy

Results	1. The parameter C and γ
	`C:\Users\...\tools> grid.py 30.txt`
	`...`
	`32.0 0.00048828125 96.6667`
	`(Best C=32.0, γ=0.00048828125 with five-fold cross-validation rate=96.6667%)`
	2. Train the model file
	`C:\Users\...\windows> svm-train -c 32 -g 0.00048828125 30.txt`
	3. Use the model to predict the origin database
	`C:\Users\...\windows> svm-predict 30.txt 30.txt.model 30.out`
	`Accuracy = 100% (660/660)(classification)`

So you can see the accuracy is very high when we use the training model to predict our original database.

If we use the default C and γ (C = 1, γ = 1)

```
C:\Users\...\windows> svm-train 0.00048828125 30.txt
...
C:\Users\...\windows> svm-predict 30.txt 30.txt.model 30.out
Accuracy = 96.5152% (637/660) (classification)
```

The resulting accuracy is lower than before so with parameter selected we can get higher accuracy. LIBSVM provides a tool to compute them, so we can use the library to get the desirable result.

7 Future Development

7.1 Two-directional Communication

In order to provide natural environment for communication between the mute and other people, voice-to-text translation can be used to convert from sound to text and Google Voice is a possible tool for this translation. It can be further expressed as continuous gestures to provide a real-time communication.

7.2 Sign Language Learning

This application is that you can write or say one sentence, and then you will know the corresponding sign language through this platform.

7.3 Adding Hand Shape features

Besides gestures and actions, hand shape is also an important part in sign language. We can add hand shape features to handle larger vocabulary. One possible approach is to use a glove that can recognize the hand shape but it is not nature enough for common usage. A company called Leap Motion has a new product “Leap” which will be launched in December (2012) or January (2013), it can distinguish your individual fingers and track your movements down to 0.01 millimeter.

8 Conclusion

The deaf and the mute have difficulties in communicating with healthy people who don't know sign language. On the other hand, deaf children have learning disabilities. Our system aims to enhance these situations by real-time translating sign language into text or sound with Kinect. We can develop more interesting games to enhance deaf children's learning abilities. We hope that our system will be able to encourage people to learn more sign language.

9 Reference

1. Chinese Sign Language
   http://baike.baidu.com/view/42806.htm

2. OpenNI
   http://75.98.78.94/About.aspx

3. PrimeSense 3D Sensor Data Sheet
   http://www.primesense.com/press-room/resources/file/4-primesense-3d-sensor-data-sheet?lang=en

4. LIBSVM
   http://www.csie.ntu.edu.tw/~cjlin/libsvm/

5. Hardware Requirements of Kinect
   http://www.microsoft.com/en-us/kinectforwindows/purchase/sensor_setup.aspx

6. FPGA-based MP3 Player
   http://cegt201.bradley.edu/projects/proj2010/fpgamp3/FPGAMP3_Project_Report.pdf

7. Altera wiki - MP3 player
   http://www.alterawiki.com/wiki/MP3_Player