

# Accelerometer Based Gesture Recognition for Real Time Applications

CS6235: Real Time Systems  
Instructor: Calton Pu

MS CS Georgia Institute of Technology

Kaustubh Beedkar (kbeedkar@gatech.edu)  
Dhanik Shah (dhanikshah@gatech.edu)

## Introduction

A primary goal of gesture recognition research is to create a system which can identify specific human gestures and use them to convey information or for device control.

In recent years gesture is becoming more common for building interfaces. Controlling applications in Real Time using Gesture recognition has become a main stream area of research. It is more convenient to use gestures to control applications as compared to the conventional interfaces like keyboard, mouse, etc.

There are various ways to capture gesture (like accelerometer, video camera, sensors, etc)

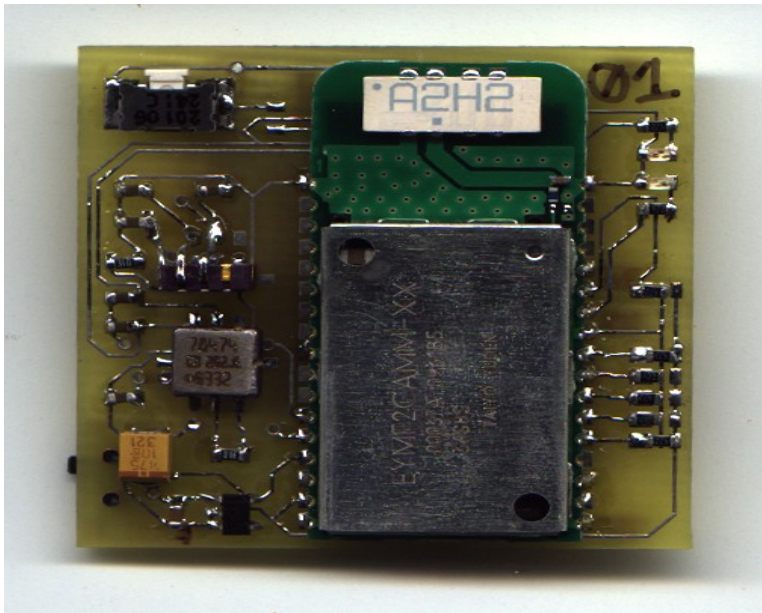
We will use an accelerometer to capture gestures and identify them

These gestures can then be used to perform various tasks (using them instead of standard input devices)

The real time application becomes much more convenient and easy to use

## Accelerometer

An accelerometer is an electromechanical device that is used to measure acceleration forces. These forces may be static (like the force of gravity), or they could be dynamic (caused by moving or vibrating the accelerometer). In our project, accelerometer collects information about the users' physical movements.



TAIYO SPP Bluetooth Accelerometer

## Specifications:

3 axis 2 g accelerometer made by Taiyo SPP  
Has a bluetooth module  
There can be up to 7 accelerometers on a single network (a limitation of the Bluetooth implementation)  
Sampled at approximately 50Hz  
Bluetooth serial port profile (SPP) for standardized interface  
3-Axis accelerometer data, 3.9mg resolution (ADXL202JE)  
Reprogrammable PIC (16LF873-04I)  
17 free I/O lines  
In-Circuit programming connector  
Over-the-air programming via bootloader  
Battery life ~60hrs on 840mAh 3.7v battery, full TX mode  
13mA @ 3.7v TX mode, 3mA @ 3.7v standby  
Very Simple Firmware  
35mm X 35mm X 5mm

## Data Format:

The device sends ASCII data the form of:

**AAAAXXXYYYZZZ**\n where

**AAAA**=16-bit sequence number, hexadecimal

**XXX** = 10-bit X-axis data, hexadecimal

**YYY** = 10-bit Y-axis data, hexadecimal

**ZZZ** = 10-bit Z-axis data, hexadecimal

These values are sent around 100hz, or every 10 milliseconds, whichever is faster.  
Axis data ranges from 0000 to 03FF

There are two LEDs on the board that indicate the status of the bluetooth connection.

**Green/Red alternating** : board is on, no rfcomm connection

**Green solid/Red blinking** : rfcomm connection established with host

## Working:

The accelerometers are registered with the host computer with the Bluetooth Serial Port Protocol (SPP)

They are read using standard serial port software. The accelerometer sensor provides post-processing synchronization of the data from multiple Bluetooth accelerometers and generates a sample of  $\Delta x$ ,  $\Delta y$ , and  $\Delta z$  indicating changes in acceleration for each axis

Data from the ADC conversion is sent to a remote computer using the PIC's UART in conjunction with a drop-in bluetooth serial part

Data from the ADC conversion is sent to a remote computer using the PIC's UART in conjunction with a drop-in bluetooth serial port

Even with two dual-axis accelerometers onboard there are up to 17 free I/O lines and two additional ADC channels depending on the device configuration

Over-the-air programming allows for easy firmware updates and rapid prototyping without the need to have a PIC programmer or special cable

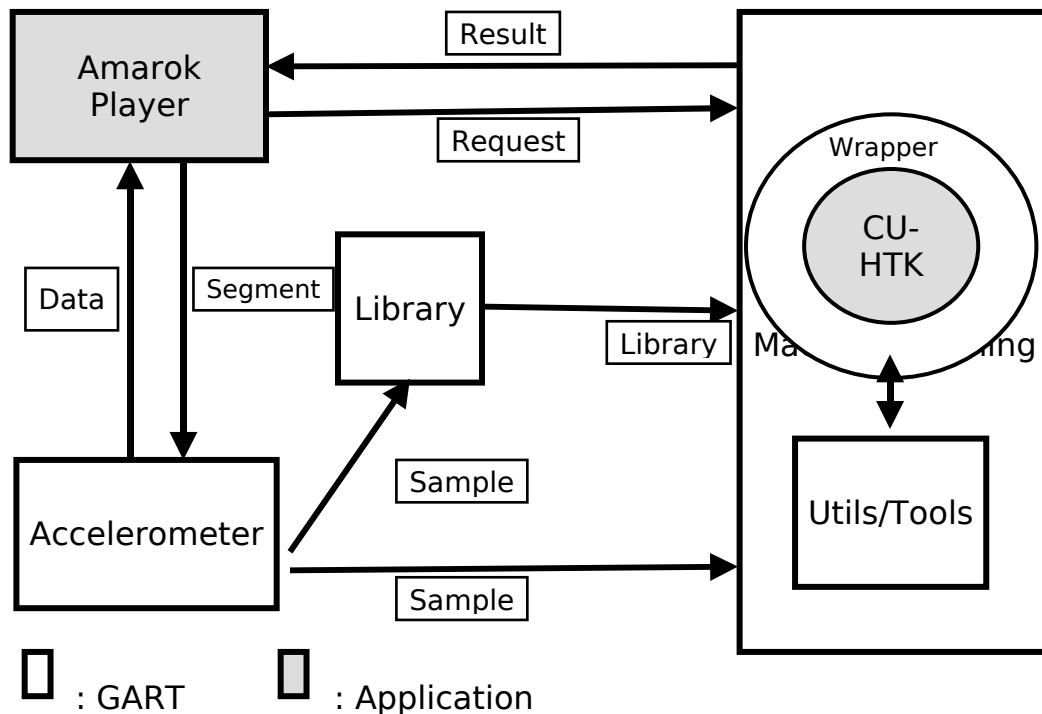
## GART

It stands for Gesture and Activity Recognition Toolkit

It is a user interface toolkit designed to enable the development of gesture-based applications. It is written in Java for cross-platform use and easy integration into graphical user interface development tools. GART is written in JAVA and has graphical user interface. It is built on top of HTK (the Hidden Markov Model toolkit).

## Architecture overview

---



## Requirements:

- Java version 1.5 or higher
- HTK (Hidden Markov Model Toolkit)
- Linux Platform
- Java Communications API (Java Comm)

## Working:

The application has two modes namely Collection mode and recognition. Basic gestures like a punch, hand movements are stored in a library. A minimum of 5 samples per gesture are stored in the library

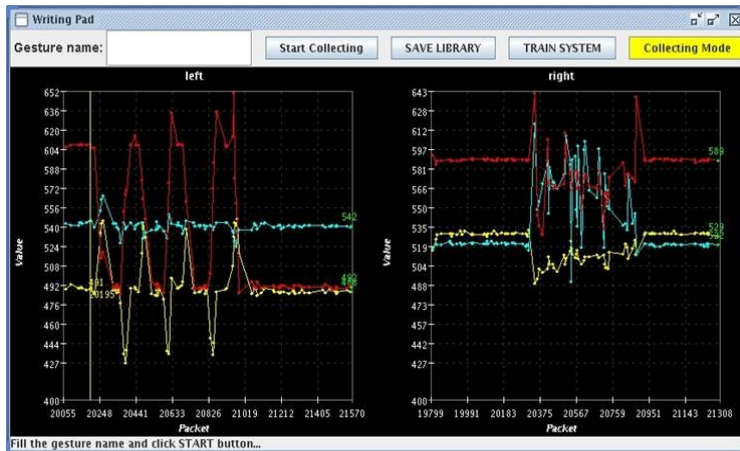
When the user starts the application, they select the number of accelerometers they have. Then they select the location of the accelerometer on the body. And lastly they select the COM port used by each accelerometer

After the configuration is done the user starts the recognition mode

Each hand movement is recognized by the system and it shows a prompt on the screen

For each pattern the system matches them with the patterns stored in the system. Since we know the location of the accelerometer on the body, we can match the movement with the gestures related to this body part

This gesture is recognized by an action listener and given as input to the media player. Gestures can be used to control working of the player



## Hidden Markov Model (HMM)

A hidden Markov model (HMM) is a statistical model in which the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications. A HMM can be considered as the simplest dynamic Bayesian network.

## Applications

The project can be implemented for various purposes like:-

1. To ensure correct body movements while exercising.
2. Interactive boxing game.
3. Learn dancing step by step.
4. Video games based on virtual reality
5. Moving objects on the screen using gestures

## Performance

The application created to incorporate the above mentioned techniques was designed to control a media player with gestures. The media player chosen was *amarok media player* for Linux.

We trained our application for recognizing four gestures. These gestures were then used for PLAY, STOP, NEXT and PAUSE. The following statistics shows the number of samples that were given as input to the HMM for training and those which were successfully recognized. In many cases, the gestures were confused and the recognized gesture was incorrect.

PLAY		STOP		PAUSE		NEXT	
Collected Samples	Recognized Correctly	Collected Samples	Recognized Correctly	Collected Samples	Recognized Correctly	Collected Samples	Recognized Correctly
5	2	5	3	5	4	5	3
10	6	10	7	10	6	10	6
20	17	20	17	20	18	20	17
30	28	30	29	30	30	30	29

Table: Shows the samples collected and number of those which were recognized correctly.

We conducted several experiments varying the number of samples given to the HMM for training and found out that the model required more than 25 samples per gestures to achieve an accuracy of more that 96%. Also, the HMM failed to recognize any sample if the number of samples were less that 5. Also a few inconsistencies were found out when the same gesture was performed with change in Y-coordinate (ie at a different height), as the accelerometer then sent a different value for the Y thus resulting is false gesture recognition. Over all, 96% of the gestures were recognized which was satisfactory.

Our experiments also measured the latency that is the reaction time of the media player after the gesture was performed.

PLAY	STOP	NEXT	PAUSE
------	------	------	-------

300ms	350ms	280ms	~300ms

The average reaction time was around 300ms which was quite reasonable. This was expected as the modified media player's library was made to interact with the GART's library created for gestures, which we plan to work on later reducing the latency. Also since the gesture is basically set of input value at the serial port, we expected few delay in the command listener action. After the gesture was performed it took nearly 4 seconds for the control to reach to the action listener for the next gesture. Thus controlling the media player reacted only when the difference between two gestures was at least 4 seconds.

We also conducted experiments where we trained the application for large number of Gestures. The following table shows the result.

Gestures	Application's Accuracy (%)
1	100
2	100
3	100
4	96
5	90
6 or more	86

Thus to conclude, we restricted ourselves to 4 gestures without compromising on accuracy and also keeping in mind the basic functionalities controlled by gestures.

We attained following parameters.

**Gesture Recognized Correctly - 96%**

**Reaction Time – 300ms**

**Minimum time between gestures for the application to react correctly – 5s**

Over all, the application worked perfectly expects for a few times and the performance was satisfactory.

## Conclusions and Future Scope

The project was indeed very satisfactory and we achieved what we had thought of and proposed. We were able to control the media player with gestures. We trained our application for 4 gestures which had the basic functionalities like play, stop, pause and next. Also the accuracy was ~95% which was decent. We also plan to do the following as our future works.

Faster sampling rates can be implemented to achieve better accuracy

This would enable the recognition of fast movements

The application could also try to identify wrong gestures

## References

<http://www.cybernet.com/~ccohen/>

[http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/COHEN/gesture\\_overview.html](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/COHEN/gesture_overview.html)

<http://wiki.cc.gatech.edu/ccg/projects/gt2k/gt2k>

<http://www-static.cc.gatech.edu/gvu/ccg/resources/btacc/>

<http://htk.eng.cam.ac.uk>

[http://en.wikipedia.org/wiki/Hidden\\_Markov\\_model](http://en.wikipedia.org/wiki/Hidden_Markov_model)

<http://java.sun.com/products/javacomm/>