Computer Science 3307A

ASSIGNMENT 5: FINAL GROUP PROJECT SUBMISSION

Mike Katchebaw
December 5, 2019

Group 35
Suhail Shakya
Bradley Assaly
Drew Barot
Abdullah Khan
Prabjot Lakhesar

Computer Science 3307A: Final Project Documentation

Group 35

December 5, 2019

1 Description

1.1 Project Overview

A fundamental task in Computer Science and Physics is determining the position of an object in a space. The development of the Global Positioning System (GPS) is an excellent example of the use of Computer Science in the application of Physics concepts, and was a revolutionary development allowing accurate positioning across the globe. While GPS is an effective solution outdoors, indoor positioning is a much more difficult task. Various solutions have been investigated, using radio, optical, heat, and audio-based solutions. For the CS3307 project, our group decided to implement an indoor positioning system using audio to effectively locate a target in a 2D space.

The goal of this project was to create an indoor positioning system capable of accurately and precisely positioning an audio source in a 2D space, like the GPS. This was to be accomplished with various Physics techniques (trilateration, ultra-high frequency sound emissions, Kalman Smoothing), and applied through Computer Science hardware (microphones and a Raspberry Pi) and software. The project was to be built in C++, using standard and open-source libraries to accomplish its core functions.

In a sample test case, a user would be able to compute the relative location of an audio emission source using 3 microphones and display it on a GUI that a central administrator could control. Past work has indicated that a high degree of accuracy is possible with specialised equipment and advanced mathematical models. Consideration was initially given to using more advanced positioning methods such as Fast Fourier Transforms, Kalman Smoothing, and Doppler Shift, which could all improve the signal to noise ratio and detect separate wavelengths.

1.2 Original Project Specifications

The project was originally split into three major components: a client front-end (audio source), a processing back-end, and a display function.

The client front-end required the design of a web page or application that a user could

interact with to create a unique identity and emit a constant tone from a speaker. An HTML webpage accessed from a phone was thought to be the simplest solution. The frequency of emitted audio was planned to be beyond the audible hearing range (18000+ Hz) as a courtesy to users. Audio Multiplexing was investigated to implement support for multiple users, allowing for a unique frequency for each user identity.

On the processing back-end, the audio would be accepted, processed, and displayed using a Raspberry Pi and three microphones. Audio would be collected via three microphones and flow into the Raspberry Pi, capturing the audio location of the emission source. Audio processing would occur using Kalman Smoothing and Fourier Transforms to improve the signal to noise ratio. Then, using trilateration, a coordinate position would be calculated. Concurrently, the Raspberry Pi would also hold two SQL databases - one to store unique user identities, and another to track positions of the users. The final output of this system would be a coordinate in 2D space corresponding to the location of a specific audio source. It was thought that this audio pipeline could easily be implemented in C++ and would work on the Raspberry Pi normally.

On the final display portion, a simple GUI would control the system and display the output position from the Raspberry Pi to a window within the OS. Additional optional features proposed included more front-end tie in (allowing users to customize their unique identities and propagating this data to the display), and optimization using threading to increase performance and a feed-forward neural network to improve accuracy.

1.3 Final Version

The final version of the application deviated from the original. The final version did not allow for live audio streaming via microphones, instead accepting integer decibel arguments and computing position via this method. In the final version, there is a single portal with three principal components: the audio processing script, the plotting tool, and the GUI system.

The primary computational component of the application is the audio processing script. The GUI would accept arguments on behalf of the script. The script accepts three beacon coordinate locations (as microphones) and outputs an ordered pair (source coordinate). The conversion from the decibel input at each beacon to the final coordinate occurs using trilateration and the inverse square method. After coordinates are generated, they are written to a SQL location table to be displayed later. The code can be found in the final submission files.

On the user side, the first prompt in the GUI is to create unique user profiles in an SQL table stored along with the program. User creation and deletion are implemented with SQL table commands. This allows for unique identification of users and tracking of their position via another SQL table.

After a user is chosen and the audio details are plugged in, the final operating screen of the GUI displays the final 'Display Control Panel'. The display control panel is where the

audio inputs can be entered, and there is also a plot displayed in one corner, showing where the audio emission was calculated to be relative to the three beacons.

This final version omits the use of physical microphones, and as such, also omits the user front-end that would have emitted an audio signal to be picked up. Instead, a single GUI accepts static literal audio arguments. This simplified version can be expanded easily to accept decibel levels instead of user entered values with a simple additional script and modified hardware.

1.4 User Stories and UML

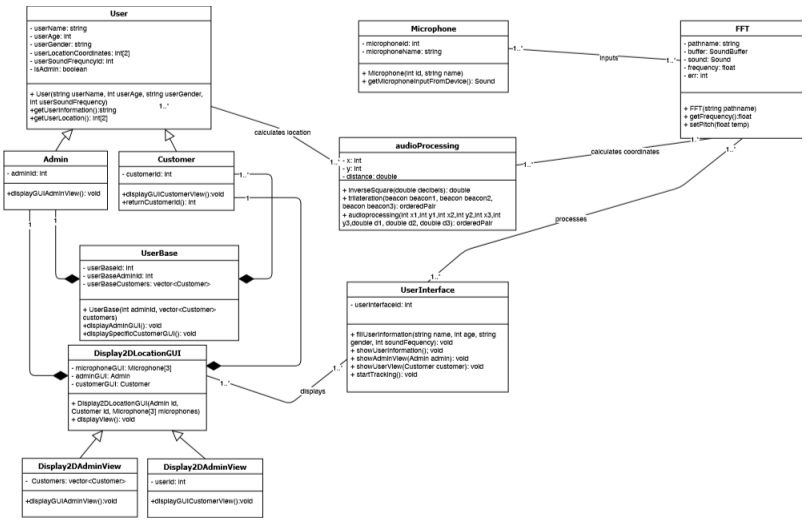
1.4.1 User Stories

There are nine total required features for this project (See appendix). Each required feature related to a user story - a path a user could take.

1.5 UML Diagram

A UML Diagram was created to display the different components of the final application.

The central GUI for the application takes the form of the Locationtracker script. This script presents a UI for a user to interact with and provides controls to the user. This central script interfaces with 3 supporting scripts. The first script is the audioProcessing script, which computes the location of a sound source after being provided beacon locations, exporting an ordered pair coordinate (which is fed into the Locationtracker). The second script is the qcustomplot script, which plots the ordered pair coordinates in a 3D plane and provides a display for the central application. This script is supported by the displayGraphics script, which displays the generated plot. The final script is the GUI implementation, MainWindow. This is the actual window of the application, which Locationtracker opens, allowing the user to open the application and interact with the various components. The UML Diagram is presented below.



2 Development Process and Key Accomplishments

2.1 Project Completion Timeline

This subsection will discuss the timeline of completing the project. A later subsection will cover key accomplishments.

The first task undertaken after group formation was planning and design of the project. After evaluating multiple ideas, the audio-location idea was selected as it was both interesting and feasible for all group members. Thereafter, research was conducted into existing positioning system and various components of audio engineering. Special attention was paid to fields such as audio filtering, position calculation, and plotting. This allowed major portions of the application to be laid out at an early stage, and for members to discuss how best to implement them.

During subsequent group meetings, potential features were proposed and evaluated for feasibility. This led to the creation of several user stories - allowing a clear user path to form leading to easier Acceptance Testing later. Each user story followed the path of a potential user, and test cases were also laid out. Once a framework was developed, potential UML diagrams and Design Patterns were explored. Analysing the most effective software architecture for the problem took some time - especially considering the uniqueness of the idea, which looked at only one effective user path. Some initial work at this stage allowed for implementation of design patterns later in the process.

When coding of the project began, group members expressed their strengths and took on appropriate tasks. A Kanban board was set-up in JIRA, allowing tasks to be tracked and allocated. The group eventually moved off JIRA and elected to inform the group via chat which component/feature they would be working on. Each member selected one component laid out in the Project Proposal/UML and completed it individually. As such, a Minimum Viable Product was built function by function and adapted overtime. Design Patterns such as Singleton, Factory, Builder and Prototype were implemented. Behavioural Patterns like Command were also implemented. Major course components led to more effective solutions and interface design as opposed to simply implementation. When individual components were tested and approved, all components were linked. Structural Patterns such as Adapters between components, and Composites were used. The final application was then tested on MacOS, Windows, and Raspberry Pi Desktop.

2.2 Implementation Details

As required, the application was built in C++ using the Raspberry Pi environment. Members of the group were comfortable with C++, while they had limited experience with a Raspberry Pi.

When initial planning had concluded, each member who was assigned a specific portion of the application researched the most effective way of implementing it. While most com-

ponents started by being implemented from scratch, using only standard C++ libraries, it was found at later stages that far more efficient and usable open source libraries were available for use. These libraries were utilized to complete various aspects of the software. An example of using an open source library was the use of Qt, a GUI builder. Building a GUI from scratch in C++ was outside the scope of the group's programming abilities, so an existing library was used to simplify the process by the member responsible for GUI development.

Programming in C++ started off in each member's individual desktop environments. This allowed for ease of development and testing. The use of BitBucket allowed for the individual members to work together and keep the project updated. Further down the line, testing and tying the application together occurred on the Raspberry Pi.

2.3 Features and Testing

2.3.1 Features

The features of the final application were laid out in the Project Summary (Subsection Final Version). All User Stories were passed, although slight modifications were necessary.

2.3.2 Testing

Testing was accomplished by evaluating the original test cases for each functional required feature laid out in the user story stage of development. Test cases were analysed, and test scripts were designed. This allowed for debugging and evaluation of the components of the software. Each test case for the required components was successfully passed from a functionality perspective. However, certain components were not as effective as was originally expected.

Quality Assurance and Quality Checks were done by various members of the group. Test scripts were executed within an IDE in a Raspberry Pi environment (natively or through the VM). Each member of the group was responsible for scrutinizing a specific component by executing the tests, reporting inconsistencies by adding them to the Kanban board, and then resolving them.

2.4 Accomplishments

There were three major accomplishments the group had throughout the course of this project.

The result was a functional application that implemented all original required user story steps. The application did not work as well as originally proposed due to a variety of reasons. These included poor sound quality and processing (due to lack of expertise), high-level implementation of locating algorithms, and discrepancies between testing environments.

2.4.1 Researching Solution to Problem and Implementing

Our group felt we effectively were able to find an interesting problem in the field of Computer Science, with a real-world application. We were also able to research this problem and find several interesting solutions which could be meshed together to improve precision, accuracy and functionality. We were also able to avoid the use of pre-built solutions and implement our own mathematical approach from scratch (albeit ineffective). This allowed us to further appreciate the process of analysing a problem and looking at existing solutions when implementing our own blended solution.

2.4.2 Using Open-Source Libraries and the Raspberry Pi Environment

Our group had previous experience in scripting languages such as Python and JavaScript, where there are a myriad of open-source libraries implementing numerous algorithms. We felt a major accomplishment we had was familiarizing ourselves with the C++ object-oriented environment, and building an application using open source libraries (such as Qt). Using open source libraries is a critical part of the modern software development process, and we were effectively able to do so.

We also felt using the Linux environment through the Raspberry Pi was a significant accomplishment. In the past, members had used either Windows or MacOS for their personal and development computing. After using the Raspberry Pi VM and physical machine, members were far more comfortable in the Linux environment when it came to development, and two members switched all development over to this environment.

2.4.3 Building a Working, Functional Application

The most significant accomplishment we felt was in building a native desktop application using C++ that was functional. In the past, members had built web-enabled applications or back-end scripts, but never a full-stack application (front-end and back-end). We felt this was another major accomplishment and saw how we could use these skills in the future to develop subsequent applications.

3 Key Problem Areas

3.1 Teamwork and Distribution of Work

As with any group project, a prominent challenge our group faced came to team communication. Throughout the term of the project, two members became seriously ill and required time away from the assignment. During this period, there was a lack of communication, resulting in delays. When all team members resolved their medical issues, the team was able to complete the project.

Beyond group member reliability, working in the prescribed team environment was at times

difficult, with tools such as JIRA, and BitBucket. Group members worked well independently, focusing on their strength and engaging in discussions in person or via chat to resolve issues. As such, using tools like JIRA made bug tracking more visible to the team, but less effective as they required a significant time and energy contribution to set-up and track. JIRA was cumbersome and difficult to use, and the team arrived at creating a physical Kanban board during multiple group programming sessions. BitBucket was used for ultimate submission of the application, however, alternatives were used for better bug tracking and revision updating. As such, the team struggled with using these standard software development tools.

3.2 Use of Course Components and Design Patterns/Principles

As per course requirements, strict specifications had to be met, and it was expected that certain principles would be implemented. Due to the unique nature of the application, it was at times difficult to accomplish this. The app was effectively implemented, not designed. This means that each component was programmed individually and in such a way where only basic design patterns were considered, while more advanced things like coupling and cohesion were not considered. Work was done to simply get the application working - not to optimize it. This fragmented programming style and subsequent linking led to poor coupling and cohesion. More attention should have been paid to key design decisions and set-up of the project - like building event factories and states.

Another challenge was abiding by course-specific requirements like the use of user stories. Due to the unique nature of this project, there were not multiple clear paths/stories a user could follow. The application performed only one primary function and there was one path to be taken. As such, initial discussion regarding user stories were clouded. This was resolved after working with a TA to more clearly understand how to frame the user stories.

3.3 Results Deviation

The most significant challenge faced while implementing the application was deviation from expected results. As this application was more experimental in nature, it was expected that there would be some error when it came to processing the final position. There were however significant errors in the results from the expected result.

While the application implemented trilateration and inverse square operations to get the effective position, the output was consistently incorrect. This is likely attributed to poor implementation of the location algorithms. A simplified version of trilateration was used, not accounting for several factors as they relate to geometry. Moreover, several factors in the calculations were simplified, meaning multiple assumptions greatly skewed results. Attempting to implement the algorithms in C++ may also have raised problems unknown to the group. When a proof-of-concept version was implemented in Python and JavaScript, results were far more accurate, and various open-source computational libraries were used to implement the location calculation.

Another issue faced was using multiple microphones with the Raspberry Pi hardware and linking them to the program. Developing a script to capture audio in C++ with a decibel output is trivial but building a script to handle three separate inputs in the Raspberry Pi platform proved to be more difficult. A potential option was to set up multiple Raspberry Pi modules, however this was not feasible due to a lack of units. As such, the project was altered. The program instead accepts user-defined decibel integer arguments and outputs a static position in a 2D plane instead of displaying a dynamic position based on sound input from microphones as originally planned.

3.4 Resolution

3.4.1 Redesign of Application Components and Architecture

A potential resolution to some problems would be a ground-up redesign of the application, improving cohesion and coupling, and using more design patterns to improve scalability. Implementing states and events would allow components to interact more seamlessly and improve overall performance. This approach would also likely make it easier to determine where errors are occurring in the positioning calculation.

3.4.2 Implementation in Different Language and Library

While C++ is an effective language to implement this application, Python is a suitable alternative with a strong scripting capability and numerous open source libraries. This application used classes but could be implemented equally well using the class structure of Python. This would also allow the use of Python libraries like NumPy and SciPy which are built for numerical and signal processing and built in C (for similar performance). A hybrid solution may be to have location script components run in Python and call these scripts in C++ through Boost Python, part of the open source Boost library. This module allows for execution of Python scripts natively in C++.

3.4.3 Use Real-Time Audio Stream and Optimization of Processing

Lastly, implementing real-time audio streaming through microphones and altering hardware would allow the application to more closely resemble the initially proposed application. Also, optimizing the trilateration scripts or introducing a new computational algorithm would help to improve accuracy.

4 Lessons Learned

4.1 Key Takeaways

4.1.1 Things Done Well

Despite having to adjust the project, there were still some things that went well and were completed to a high degree. Firstly, despite the fact that physical microphones were not used, we were still able to emulate them to a high degree and were successfully able to use

the trilateration formula to output the coordinates in a 2D space. This was done via a C++ audio script, which is a bulk of the work required to get the microphones working. Another thing that was done well was the GUI component of the project. This was done via the Qt library for C++, which allowed for us to create a native C++ application which is easy to follow along and was in the scope of our groups programming abilities.

4.1.2 Things to Change

We as a group had to adjust and cut some key components out of the project, which led to the final submission deviating from the original project proposal. One of those components being the use of physical microphones. We successfully developed a C++ script to capture audio, however due to various issues we faced with the Raspberry Pi, we opted to replace the physical microphones with user-defined integer decibel arguments to simulate the microphones. Another area we feel we could improve on is the fact that only basic design patterns were considered. With better planning of the overall project, more advanced design patterns such as coupling, cohesion, etc. could also have been considered - which would ultimately have led to improved overall performance.

4.2 Best Practises

Instead of diving straight into programming, we as a group conducted lots of research on indoor positioning systems. We wanted to understand the math behind finding an object in 2D space as it was something none of us had done before. The project was extremely heavy on theory and we made sure everybody had a solid understanding of what we were getting into before starting. Upon the completion of the research phase, group members chose individual components to complete, the result of that was a Minimum Viable Product which was built incrementally one function at a time. This was done with the help of BitBuckets version control allowing all group members to commit changes in an organized way. By utilizing the practices used industry wide, we were able to complete the project seamlessly and on time.

4.3 Potential Improvement: Machine-Learning Optimization

In the project proposal phase, the idea of using a simple feedforward neural network on audio input was evaluated to increase the signal-to-noise ratio and increase overall accuracy. This was not implemented during the process as it required a lot of overhead computation on the Raspberry Pi, the use of various external libraries (TensorFlow, etc.) that were not easily implementable, and lack of training data.

Using Machine Learning to increase the signal-to-noise ratio or to increase the usefulness of a signal is a common practise in image processing, finance, and other fields. To be implemented here, a simple feedforward network could be designed to accept audio streams/files, apply a variety of filters, and come to a more accurate decimal count. An alternative would be a network that could accept audio byte data and output two variable - x and y coordinates in a 2D plane. This could supersede the use of the trilateration formula used and

potentially be more accurate. In this case, a Convolutional Neural Network could be used, as these networks are typically faster (important when tracking in real-time) and allow for various stages and levels of filtering on audio data to design a more complex model than the simplified solution used. Training data would have to be generated for either, in the form of audio to decibel indications (to improve mic accuracy) or audio to coordinate location data (to improve location accuracy).

5 Appendix

5.1 User Stories and UML

Feature 1: User Account Creation Allow a user to create their unique identity in the application's database. The account accepts fields for username, password, and unique userID (distinguishing them from other users).

Feature 2: User Account Deletion Allow a user to delete their unique identity from the application's database. Deletes all fields entered in the Account Creation story.

Feature 3: User Database Allow a user (on the Raspberry Pi) to create, view and maintain a database to store, access and modify information pertaining to user accounts and unique identity keys.

Feature 4: User Login Allow a user to login to the application with their username and password, verified against the user database before accessing audio features.

Feature 5: Audio Processing Allow a user to register three microphone beacons and record audio concurrently, and thereafter processing the audio to measure decibel rating at each beacon.

Feature 6: Trilateration Source Computation Allow a user to interface with a GUI to input beacon coordinates for tracking, and sound recordings of the beacons such that the application computes the source using the algorithms in the location tracking computation feature.

Feature 7: Results Database Allow the user to store user IDs, trial numbers, and coordinates of a trial in a database. Also allow for viewing and maintenance of the database.

Feature 8: Graphical Visualization of Results Allow a user to visually see the coordinates of beacons and the source graphically via an application.

Feature 9: Location Tracking Computation Allow a user to track the current location of an audio source and generate its coordinates. Using inverse squared transform and trilateration algorithms to accomplish the locating.