Sonic Object Localization for Reconstruction in Virtual Reality

Anonymous
Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY 14586

Abstract—Real-time localization has many use-cases; from tracking a pet in a house to inventory in a warehouse. Out of the many localization technologies, ultra-wide band stands out to be promising contender. This capstone report describes the effort in designing a hardware system that uses ultra-wide band technology to track sound emitting objects in real-time. This positional data is used for an eventual reconstruction of the object motion in a virtual space. An ultra-wide band chip made by Decawave is used for real-time localization and a Raspberry Pi for exporting that data to a remote server. Centimeter-level localization accuracy was achieved.

Index Terms—Real-time Indoor Localization, Ultra Wide Band, Decawave

I. INTRODUCTION

Localization is a technique of identifying the position of an object in an environment. Accuracy in localization depends on where it is performed: indoors or outdoors. Both forms of localization are dependent on the characteristics of the tracked object; such as object movement speed, dimensions, etc. It also depends on the need to track the objects in real-time. Especially, outdoor localization technology (like the GPS) is plagued by environmental factors such as altitude, cloud cover, etc.

Many researchers [1][2] approach to localization is to use Wi-Fi technology or in some cases [3], Bluetooth. GPS has been known to have issues [4] with indoor tracking and therefore is not viable. As suggested by [5], this capstone project aims to use ultra-wide band technology for indoor and outdoor localization of sound-emitting (sonic) objects in real-time. This localization data will be transferred to a remote server for reconstruction in virtual reality.

Motivation for this capstone project is mentioned in section II. The benefits and drawbacks of various localization technologies are compared in this section as well. Section III defines the solution architecture that utilizes ultra-wide band technology for real-time localization. Section IV and V talk about the project results and its future expansion respectively.

II. BACKGROUND AND MOTIVATION

On | Off project is a joint collaboration effort between Peter Ferry, Xuan and Danny Clay. It is supported by the Paul R. Judy Center for Applied Research at the Eastman School of Music. This project is an audience-collaborative work where the participants are given a homemade music synthesizer with which they create harmonious music while being guided by video projections. On | Off aims to improve the interaction between audience by enhancing their listening skills and by appreciating the unique contributions of others.

Fig. 1: Home-made music synthesizer [6]

The music synthesizers(sound-boxes) used here are roughly 4 x 4 inches in size and can be as many as 40 in number spread across a room that is roughly 25 x 25 feet. These boxes have simple knobs for changing volume and frequency. Currently, they have no way of communicating with each other. A localization accuracy of atleast 20-30 centimetres is required for this project. GPS has been known to have a localization error of a “few metres in open areas to 80 metres in urban canyons” [4]. It might not be feasible in this scenario as we are not sure whether the sound-boxes will be deployed indoors or outdoors.

Some researchers [1] have used Wi-Fi sensors for indoor localization and were able to achieve accuracy under 10 centimeters using commodity hardware. However, this comes at the cost of losing real-time tracking and the initial hurdle of fingerprinting every environment that the objects reside in. While Bluetooth localization technology does offer real-time tracking [3], its positioning error ranged between 50 centimetres to 1 metre. Bluetooth also has a limit on the number of devices being tracked simultaneously and is not
suitable for this project as we require roughly 40 devices being tracked at any given point of time.

Ultra-wide band technology, standardized in 2007, offers low power requirements for high-bandwidth communications. According to a recent survey [5], ultra-wide band is the most accurate indoor positioning technology by far. The survey also suggests that centimeter level accuracy can be achieved by this technology in real-time. Due to these benefits, this capstone project aims to localize the sound-boxes using hardware based on ultra-wide band chips and transmit the location of these sound-boxes to a remote server in real-time.

III. Solution

Since the sound-boxes are tiny (4 x 4 inches) in size and the communication hardware should typically go inside the box, there is a small footprint to work with. Therefore, a localization technology that is small and portable will be required as it needs to be placed inside the sound-box. It should also be power-efficient as it needs to run on the same power source as the sound-box. Localization chips that utilize Ultra Wide Band (UWB) technology satisfy these requirements. Currently, many companies like Pozyx, Senion, etc. offer indoor localization technology. Chips designed by Decawave Ltd. were chosen due to their competitive price, documentation and extensive feature set.

A. Hardware

Decawave Ltd produces a communication module known as DWM1001 with an embedded firmware that enables accurate UWB-based real time localization. These modules also require little power and come with a development board that can be soldered to the GPIO module of the Raspberry Pi. More detailed installation and setup instructions can be found at Appendix A. The Raspberry Pi can communicate with DWM1001 using their provided API library written in C programming language. This serial communication can also be done using PySerial. The full extent of the API can be found in Appendix B.

B. Architecture

To localize the position of a sound-box, four R-D modules functioning as anchors are placed in the corners of the room. Another R-D module goes inside the sound-box and behaves like a tag. The distance between the anchors needs to be measured as accurately as possible. This ensures that the localization of the tag is accurate. The first powered-on R-D module is known as the initiator. When the initiator anchor is turned on, it searches for other anchors to establish a network. If it does not find any other anchors, the module will go to sleep and periodically wake up to establish a network.

![Fig. 3: Transmitting tag co-ordinates to a remote server](image)

When all the anchors are turned on, an android application provided by Decawave can be used to name the anchors and create a named network. When establishing the network, all four anchors need to be in line of sight of each other. There is an auto-positioning feature built into the android application. However, it requires a specific order of device activation and positioning and takes an accuracy hit upto $50\, \text{cm}$. A better way to create the network would be to measure the accurate distance between the anchors on a grid and set the coordinates directly using the C-API or PySerial.

As shown in Fig. 4, a named network called *Capstone* is created using 4 anchors (*Crust, Plate, Pan and Tin*) and one tag (*Sauce*). The tag named *Sauce* can be placed inside the sound-box and when it moves, the application displays the movement of the tag along with its co-ordinates. These co-ordinates can be extracted from the DWM1001 chip using the C API and transferred to an external server using the Raspberry Pi. A sample code of how this could be done is provided in Appendix C.

C. Trilateration Algorithm

Decawave chips utilize trilateration algorithm for localization. Trilateration is the process of calculating the position of
a point with respect to three other points using linear algebra. As shown in Fig. 5 let us assume that the position of three anchors are given by the co-ordinates \((x_1, y_1)\), \((x_2, y_2)\) and \((x_3, y_3)\) and the position of the tag as \((x, y)\). If the distance between the anchors and tag are \(d_1\), \(d_2\) and \(d_3\), we can solve the following three equations using linear algebra and calculate the position of the tag.

\[
(x-x_1)^2 + (y-y_1)^2 = d_1^2 \\
(x-x_2)^2 + (y-y_2)^2 = d_2^2 \\
(x-x_3)^2 + (y-y_3)^2 = d_3^2
\]

Since the Time of Arrival (TOA) can be obtained by each R-D module, it is easy to calculate the distance \(d_i\) between any two modules using the formula

\[
distance = speed\_of\_light \times time\_of\_arrival
\]

\[D. \text{ Virtual Reality}\]

Localization data can be constantly polled from the tag and transmitted to a remote server. This data could be saved offline for future use or can be reconstructed in 3D space for virtual reality. Fig. 6 shows a demo application made in Unity Game Engine that tracks an object in real time. The bright cube is the tag and the orange cubes are the anchors.

\[\text{Fig. 6: Real-time localization in Unity}\]

\[\text{IV. Results}\]

The performance benefits of ultra-wide band technology, as mentioned in the survey [5], were apparent during experimentation. Localization accuracy of \(\pm 10 \text{ cm}\) was achieved during testing with an added benefit of real-time tracking. With the help of the Decawave API, the location of the R-D module was polled three times per second and the data was saved to a local text file. Provisions were made in the code so that this localization data can also be transmitted to a remote server over UDP. The server-side feature of storing this data and processing it for virtual space representation is evaluated by Nikhilesh Kshirsagar (nmk4698@rit.edu) in his capstone project.

\[\text{V. Future Work}\]

Decawave is currently in the process of upgrading their C API so that exporting the localization data of the tag becomes easier. This would also enable Raspberry-Pi’s to be used as a gateway for data transmission natively rather using custom solutions. They have also promised to release their Bluetooth-API by the end of Fall 2018 which would enable programmers to develop their own bluetooth Decawave networking applications for any operating system. Next logical step would be to reduce the footprint of the R-D module by
using Raspberry-Pi Zero or Arduino with the Decawave chip. Better power sources for the R-D module can be designed as well. The sound card that are currently used in the soundboxes could be soldered to the Raspberry Pi and their data can be transmitted along with the location to a remote server.

REFERENCES


Appendix

A. Installation & Setup

Decawave DWM1001-Dev chips can be ordered from semiconductorstore.com. They take around two weeks to get delivered as they need to be shipped from Ireland. There are other variants of the chip known as MDEK1001 which have a better housing and battery solution for the decawave chip. They might be cheaper to buy in bulk as well. Any Raspberry Pi from v2.0 and above are good for combining with the DWM1001-Dev. Ensure that the GPIO header pins on the decawave chip are less than or equal to the number of pins on the Raspberry-Pi. Viable Raspberry Pi models are Pi Zero, V3.0 and V2.0. While Arduinos are good for tags, they might be underpowered for anchors or gateways. The operating system on the Raspberry Pi is Raspbian. Make sure that the SPI and Remote GPIO are enabled in the interfacing options of the Raspberry Pi. This can be accessed by entering the terminal command `sudo raspi-config`. This is the only way to make the Raspberry Pi communicate with the Decawave chip.

A 2x13 header stack should be soldered on to the DWM1001-Dev chip and then inserted onto the GPIO pins of the Raspberry Pi. Insertion of the header stack should be done such that the first 26 pins of the RPi GPIO are occupied by the DWM1001-Dev and the Decawave chip should point away from the USB drives of the Raspberry Pi.

The device should not be powered on without ensuring the correct insertion of these pins. Also, the UWB chip on the DWM1001-Dev module should not be obstructed by any metallic contraption that could block its signals. The power requirements for the R-D module is 5.1 volts and 2.5 amperes. A portable power bank may be used but ensure that the power bank supports low power devices like the decawave chips.

Before reading the next set of instructions, follow the installation video on Decawave’s youtube channel: MDEK Quickstart guide - (URL: https://www.youtube.com/watch?v=hl8EaU5nOmI). Atleast 4 anchors and one tag is required for forming a localization network. After the four R-D modules are turned on, open the Decawave Android app and turn on its bluetooth. All 4 anchors would show up on the application. Create a network, add all 4 anchors to the network and give them names. Activate the anchor and its LED and ensure only one of the anchors is selected as an initiator.

After this, position the anchors in the four corners of the room. The anchors need to be at a high position and should be in clear line of sight without any obstructions. Assume the room dimensions on a 2D axis and enter the co-ordinates of each anchor in the android application. For a quick setup, use the auto-positioning feature on the android application. This is not always accurate but can save some time in a pinch. When using the auto-positioning feature, ensure that the anchors are ordered in anti-clockwise direction with the initiator at the top.

Power on the tag R-D module and add it to the network using the android application. After enabling it as the tag and setting the quickest update rate possible, restart the application. Select the previously created network and you will see the tag being localized.

B. API Reference

A detailed API manual can be found on Decawave’s product website. These API functions are written in C language. An example code on how to call the API is listed in Appendix C. Few of the important APIs are listed below.

- `dwm_pos_set`: Used to set the position of an R-D module.
- `dwm_pos_get`: Used to get the position of an R-D module.
- `dwm_cfg_set`: Used to set the position of an R-D module.
- `dwm_cfg_get`: Used to get the current configuration of the module like update rate, name, location etc.
- `dwm_cfg_tag_set`: Used to set configuration for a tag module.
- `dwm_cfg_anchor_set`: Used to set configuration for an anchor module.
- `dwm_loc_get`: Used to get the last known distance from a tag to its connecting anchors.
- `dwm_status_get`: Used to get the current status like location data and network information of a module.

C. Sample Code

The following is an example code that can be run on an R-D module that is connected to the decawave network to get its current x,y,z location.

```c
#include "dwm_api.h"
#include "hal.h"

int main( void )
{
    HAL_Print( "dwm_init()\n" , HAL_DevNum() );
    dwm_init ( );
    dwm_pos_t pos;
    dwm_pos_get(&pos);
    printf("x:%u, y:%u, z:%u\n", pos.x, pos.y, pos.z);
    return 0;
}
```