A Low Cost Outdoor Assistive Navigation System for Blind People

Dr. Jizhong Xiao, Kevin Ramdath, Manor Iosilevish, Dharmdeo Sigh, Anastasis Tsakas

Department of Electrical Engineering, The City College, The City University of New York

Convent Ave & 140th Street, New York, NY 10031, USA

Abstract - With over 39 million visually impaired people worldwide, the need for an assistive device that allows the blind user navigate freely is crucial. We have developed an off-line navigation device that uses 3-D sounds to provide navigation instructions to the user. Our device relays directional information to the user through special Audio Bone headphones, which use bone conduction technology. Sounds are recorded and can therefore be selected by the blind user. Navigation processing is handled by a Raspberry Pi. We are using a magnetic compass and gyroscope to calculate the direction that the user is facing. Route queries of the destination address are geocoded using the *Geo-Coder-US* module and passed to the *MoNav* module to generate a pedestrian route. Additional capabilities of the device include speech recognition and voice prompts for obtaining the user's desired destination address. The user can input the address by speaking into a microphone. The entire system is mounted to a pack that sits on the user's waist. It is very light and portable and it does not impede any of the user's senses while it is being used.

I. Introduction

Although there are various technologies available to the blind community to help them navigate, they all either limit the freedom of the user, or are too expensive. For example, the Seeing Eye Dog, with a service life of 7 years, costs as much as $40,000 [3]. This figure may not seem substantial, but one must consider that the average American makes less than $42,000 a year. Current technologies that aim to help a blind person navigate usually consist of vibrotactile or audible feedback to the user [7]. The feedback methodology is more pertinent to a successful device because the blind person needs to be comfortable and fully entrust the device. The main problem with current devices is that a system which gives feedback in the form of sentences may distract the user from sensing obstacles in their environment.

Our product aims to overcome these obstacles by being more affordable and only relaying information when necessary through Audio Bone Headphones that do not impede the user's senses. We understand that the user needs to hear the surrounding environment, therefore, our method does not impede their hearing. Furthermore, the entire device's production cost is only $138, which makes this device very affordable to everyone.

Currently, most of the navigation technology revolves around feedback to the user through synthesized speech. Research at Georgia Tech showed that a much better method of providing feedback was through audio-only output, instead of through synthesized speech. Our goal was to improve on this technology and make it more affordable to blind people [13].

The second method of providing feedback to the user is through vibrotactile tactons (tactile icons). By using vibrotactile technology, the user feels a vibration on a certain part of the body whenever they need to navigate. We decided to avoid this methodology because after some period of repeated use, there are adverse effects, such as Vibration White Finger Syndrome, if the vibration is on the fingers through gloves [7].

II. System Architecture

The entire system consists of 5 modules, controlled by a loader program. The entire system does not require the use of the internet in order to operate. The initializer module verifies that all the libraries are installed and all of the data files for the proper operation of the system exist. The user interface obtains the destination address from the user. The Address Query translates the address to geographic coordinates. The Route Query takes the blind user's current coordinate from the GPS and the destination coordinate, and computes a route. The Route Transversal module provides audible navigation instructions to the user. Each module is described in more detail in the following sections.

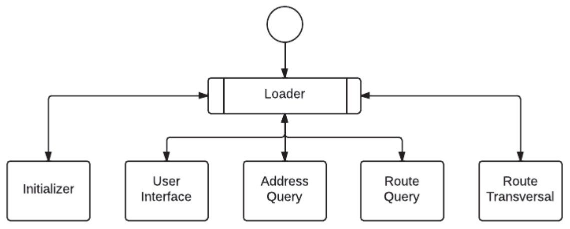


Fig. 1 : System Architecture

III. Sound Feedback

A. *Binaural Sound Recordings*

The main difference in the methodology of providing the user with navigation directions between our project and mainstream applications, is that we use audible beeps instead of voice commands. It is much easier to navigate by listening to a beep and estimating the spatial point of origin. The beeps are very short in duration, thereby minimizing the amount of time that the user is disturbed.

In order for us to create realistic sound files, the sounds were recorded using in-ear binaural microphones. We used Audacity to record the sounds [11]. The sound clips were obtained online through free open source websites. We chose the clips which we thought would be the easiest to detect when they were played. We also tried to predict the annoyance of the clip after having to listen to it for numerous instructions. We wanted to pick beats which the user would enjoy listening to repeatedly. The user can select the sound file.

After selecting the sound files, we used Audacity to clip them into shorter sections of 2 to 3.5 seconds in length. Anything less than that would be too abrupt for the user to understand the sound's origin and anything greater than that would cause discomfort. The frequency of the feedback was minimized.

B. *Binauralization Station*

We created a Binauralization Station which had a speaker mounted to it. We used VLC media player to play the sound, and we varied the amplitude of the sound (volume) to the comfort of the user. The speaker was mounted on an arm that would rotate 360° around the user's head. We rotated the speaker 24 times at 15° increments. We wanted to have the ability to provide the user with feedback for as small of a change as 15°. After testing, we realized that it would be more beneficial if we increased the angle to 30° because this decreases the possibility of getting a correction instruction due to natural body swaying while walking. Therefore, we have the sound files saved in 15° increments on the device. We select the file that we need to play, and then we play it to the user through Audio Bone Headphones.

C. *Binaural Microphones*

In order to record the sound in the same way that the person would hear it, we used in-ear binaural microphones that we placed inside the person's ears. By using the in-ear binaural microphones, we did not need to concern ourselves with factors such as the shape of the person's ears or sound reflecting off of other objects in the environment.

D. *Audio Bone Headphones*

In order to make a product that a blind person could use comfortably, we would need to ensure that the product would not impede their senses. Therefore, we decided to use Audio Bone Headphones because of their bone conduction technology. They sit outside of the user's ear which allow them to freely hear their surroundings as they normally would. This gives us the ability to play sounds when we need to give an instruction to them, or to simply allow them to use their hearing to gauge what is around.

IV. User Interface

The entire system fits inside a fanny pack that is mounted around the user's waist. The user only controls a power switch and three buttons. The system is designed to start up once the power switch is turned to "ON." Once "ON," it will welcome the user and it will ask for the desired destination address through a series of questions. The user can input the responses through the three buttons and a microphone.

Once the destination address is obtained, the system will determine the user's current position through a Bluetooth GPS receiver. It will then begin instructing the user on which direction they should start heading towards. A compass, mounted inside the fanny pack, provides the system with the user's heading. We also incorporated a safety feature in the device. Whenever a person comes within 10 meters of an intersection, the device will play a distinctive sound to let the user know that they should be aware that they are nearing an intersection.

A. *Address Input*

The system gives the user two options for inputting the destination address. The first option is to select an address that has been previously loaded into the system's memory, such as a home address, by pressing the up button.

The second option is to input a new address by pressing the down button. It will ask the user for the address. The user will have to press up to start recording and down once they have finished. It will then play back the information that it understood from the user. If the information is correct, the user can press the middle (Select) button to continue to the next step, otherwise they can press the up button and repeat the process. This procedure is repeated every time the system needs input from the user.

The order that the address is obtained begins with the Borough or county of New York. It is then followed by the House Number, the Street, the City, the State, and finally the zip code.

Once the destination address is successfully input into the system, the user can place the microphone and the buttons back into the fanny pack and they can start following the directions of the device.

V. Text To Speech

In order for us to give verbal instructions to the user, we need to covert our text instructions into audible speech. For this we decided to use the Festival Speech Synthesis System (FSSS) [2]. The major benefits of using FSSS are that it is open source and it allows you to output speech in many different languages. Therefore, we sent a string of data to FSSS with the instruction that we wanted to tell the user, and FSSS converted the speech to text, which was then played for the user.

VI. Speech To Text

Speech to text is used to get user input in ways that the three button system cannot. We can use the three input buttons to get simple user inputs such as up, down, or select when in a menu. However, when we require specific information such as an address, we must rely on a more versatile method for input. Speech to text allowed us to get input from the user simply by asking them to say the required piece of information.

Simply put, speech to text allows us to record the user's speech and covert it to text which can be used for the other parts of the system. We mainly used speech to text to get the user's desired destination address. For the address, we need the house number, street, city, state, and zip code. This required extensive use of the speech to text engine, PocketSphinx.

A. PocketSphinx

We used the open-source speech to text engine, PocketSphinx [1], developed by Carnegie Mellon University. We chose this particular open-source engine because it allowed us to easily add new words to the dictionary of required words and also allowed us to train the engine to better recognize the speech of a particular user [9].

To use PocketSphinx we needed to create input files for the engine. To "teach" the engine a new word, we had to generate a phonetic dictionary containing the word and a language model containing the word as well. We also needed an acoustic model, but the one that came with PocketSphinx would suffice.

The phonetic dictionary file is a simple mapping of the word to its corresponding phones. The phoneme for a word is just the distinct set of units of sound that describe that word and distinguish it from other words. Below is a word and its associated phoneme.

SAMPLE	S AE M P AH L

To generate this mapping with a large number of words, we used a modified version of Phontisaurus, a grapheme-to-phoneme framework [4].

As mentioned before, we also needed a language model to describe the "language" we needed to recognize. For address input, we needed a language models for numbers, street names, and cities (as well as the corresponding phonetic dictionaries for each of those models). The language model allowed us to restrict the word search. We used MITLM, a language modeling toolkit developed by the Massachusetts Institute of Technology, to create the language models.

Finally, we needed an acoustic model which contains all the acoustic properties of each senone (a Gaussian Mixture Model which represents an atomic acoustic unit). PocketSphinx came with a default acoustic model. Following the PocketSphinx documentation, we were able to adapt the default acoustic model to suit our speech. We can train the model to any given user's speech with ease (this is not required, but makes the recognition more accurate).

We achieved very accurate recognitions by restricting the size of the dictionaries used and extensively adapting the acoustic model.

B. Improving Recognition by Splitting Dictionaries

To get very accurate speech recognition, the dictionaries were split up. For example, we had a dictionary of words for each of the different parts of the address. There were separate phonetic dictionaries for numbers, cities, streets, and boroughs. In addition, the cities and streets were further split up by borough.

Before splitting up the dictionaries, we noticed that words that sound alike were often recognized incorrectly. After splitting the dictionaries, recognition accuracy drastically improved.

VII. Route Query

After the destination address has been obtained from the user, it must be translated to a geographic point. The destination address will be geocoded using the Geo-Coder-US [10] module and passed to the MoNav [5] module to generate a pedestrian route.

A. Geocoding

The Geo-Coder-US module is an open source Perl based geocoding library that is able parse addresses to geospatial coordinates. The module uses a spatial database of the features of the US to perform its operation. The spatial database is built from TIGER shapefiles [12] provided by the US Census Bureau. These shapefiles contain boundaries, water features, and roads within the US. Geo-Coder-US performs preprocessing on the shapefiles, so that when an address is queried, only a table lookup is needed, and thus the operation time is minimized.

B. MoNav

The MoNav module is an open source desktop application, based on the diploma thesis of Christian Vetter, that provides navigation and routing. MoNav provides fast routing with very few computations by performing preprocessing on map data procured from OpenStreetMaps. The preprocessing converts the OpenStreetMaps data into MoNav's own raw data format for quick lookups during path planning. MoNav is able to reduce computation time by using a heuristic approach when determining paths.

The source coordinate used to determine the path is obtained from the GPS receiver, while the destination address is obtained from the geocoding module. The pedestrian path that MoNav generates is the geometry of the route. For example, the path can be an array of coordinates, where each coordinate is corner of a street

VIII. Compass

An electronic compass is needed to determine the heading of the user at any given time. This task is achievable by using a three axial Magnetometer. The magnetic north is recognized on the three axis of the sensor, which allows us to compute the heading of the sensor in comparison to magnetic north. The heading of the user become inaccurate once the sensor is tilted because the axial magnetic components are not perpendicular to earth's magnetic field.

A. Madgwick Algorithm

Madgwick algorithm [8] is used to fuse sensor data to achieve orientation in reference to the gravity vector and magnetic north. This algorithm requires every X, Y, and Z component from the accelerometer, gyroscope, and magnetometer. Once the sensor readings are input into the algorithm the sensor's orientation (pitch (□), roll (θ), yaw (ψ)) is the filtered output.

B. Sensor Requirements

Calculating the heading of a horizontal sensor is given in the equation below. [6]

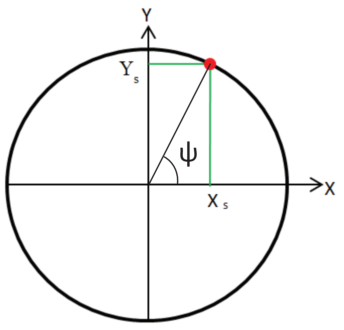


Fig. 2: Horizontal plane showing how the magnetometer's X and Y readings are used to calculate the yaw angle.

$$\psi = \tan^{-1}\left(\frac{Y_S}{X_S}\right) \qquad (1)$$

Where:

ψ is the angle between X axis and magnetic north

Y_s is the component of the magnetic field detected on the magnetometer's Y axis

X_s is the component of the magnetic field detected on the magnetometer's X axis

To solve for the accurate yaw angle, the proper quadrant is to be determined. As noted before, the heading of the user is determined by the magnetometer, yet we need to compensate for tilt because the sensors will not always be horizontal. A three axial accelerometer provides enough information to calculate the tilt of the inertial measurement unit. The accelerometer is a sensitive device which needs to be filtered for this application. The gyroscope is fused with the accelerometer to provide us with stable and accurate roll and pitch readings.

C. Sensor Calibration

The magnetometer sensor must be calibrated to adjust for manufacturing faults and inconsistency. Calibration needs to be completed for the offset values on each axis to be found. Once the offset values are calculated for that particular hardware, it is not required for the procedure to be conducted again. The procedure consists of recording the raw values of the magnetometer while the sensor is being turned 360 degrees about all three axis. Similar graphs should be seen.

It can be seen in the Fig. 3 that the ovals that have been plotted are not centered at the origin. The next step in the procedure is to center the circles. The offsets are achieved using the formulas:

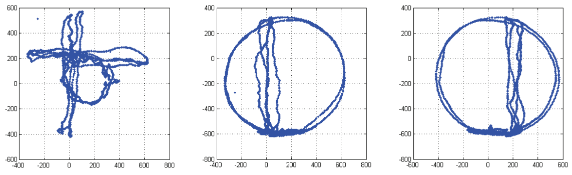
$$X_{offset} = \frac{\left((abs(X_{\max})) - (abs(X_{\min}))\right)}{2}$$

$$Y_{offset} = \frac{\left((abs(Y_{\max})) - (abs(Y_{\min}))\right)}{2}$$

$$Z_{offset} = \frac{\left((abs(Z_{\max})) - (abs(Z_{\min}))\right)}{2}$$

In order to locate the center of the circle, we took the maximum and minimum point on each axis and found the average. We then used this average to translate the entire circle by the same number of offset units so that the new center of the circle would lay on the (0,0,0) coordinate of the X, Y, Z plane.

Once the offset values are determined, they are to be subtracted from every axis on the magnetometer. This type of calibration provides us with a stable heading.



A)X-Y plane B)Y-Z plane C) Z-X plane

Fig. 3 : The circles before the offsets were applied.

IX. Route Transversal

The route transversal module determines a user's position within the route and uses their orientation to instruct them on the direction to walk. The user is provided the direction in the form of a 3D sound, i.e. the user will hear sound originating from the direction in which they need to walk. The user is provided the sound within an accuracy of 30 degrees; however, this accuracy is configurable by increments of 15 degrees. This means that if the user needs to walk at either 25 degrees or 35 degrees, they will hear a sound at 30 degrees.

A. Transversal Algorithm

During the route transversal, the GPS receiver's primary role will be to determine user's position within the route, whereas the compass's role will be to determine the direction needed to turn or walk. Both devices will be used to determine when a user is straying off the path. Notation to keep in mind is that a route refers to the transversal from the source to destination, whereas path refers to the transversal of usually one street.

Let be the distance of the user to the end of the current path. Let $_c$ be the minimum distance of the user from the current path and $_n$ be the minimum distance of the user from the next path, so is the ratio of the user's distance to the current path to the next path. is the angle formed by the current path's starting point, the user's position, and the next path's ending point. is the angle between the current path and the next path. is the probability that the user is transition from the current path to the next. and the difference between and is inversely proportional to . is proportional to . is the probability that the user is straying from the current path. $_c$ and the difference between and is proportional to .

The route transversal algorithm is defined as follows:

```
Determine  .
A GPS coordinate is received:
    Compute  ,  ,  , and  .
    If   is above a certain threshold:
        Increment the path (i.e. current path is set to next path).
            If current path does not exist:
                The user has reached their destination, finished.
            Else if the next path does not exist:
                The user is on the final path,   and   have no meaning.
            Else:
                    Compute  .
            Inform the user of the angle they need to turn.
A compass reading is received:
    Compute  c,  , and  .
    If   is above a certain threshold:
        Inform the user that they are straying from the current path.
    Else:
        Play sound in the direction the user needs to walk.
```

X. Experiments

After testing out the device in New York City around The City College of New York (CCNY), the results are very promising. We began in front of the Steinman Hall building of CCNY. We input an address that was several blocks away and we began testing the device. In the first test, we tried to see if each component of the device worked properly. Therefore, we turned it on and we heard the welcome message. We were then prompted to enter the destination address. We verbally entered a new destination and we began getting directions from the headset.

We followed each direction that the device gave us. When we came to an intersection and it told us to turn, we purposely kept going straight to see if it would correct us, and sure enough it soon caught that we were going off route and it corrected our path. We repeated this same procedure numerous times to make sure that it worked flawlessly. We were able to conclude that the navigation on the Raspberry Pi worked successfully. The only issue that we keep having with the navigation devices is the accuracy of the GPS receiver. For our experiment we selected a Bluetooth GPS receiver which costs about $100 (the

Dual Electronics XGPS150 receiver). We know that if the user were to use a more expensive receiver, then they could obtain better results in the city. This receiver works very well in suburban areas but it can become problematic in areas where there are high-rise buildings. We do not see this as a major problem because we designed the device to be able to connect to any GPS receiver that works using Bluetooth. Fig. 4 shows the different components of the navigation device. Fig. 5 shows the GPS accuracy next to a tall building with a flat metal façade, and Fig. 6 shows the accuracy in a residential area. The desired path is mapped out with circles representing every time a change of direction needs to occur. From the Figures we can tell that the device works very well.

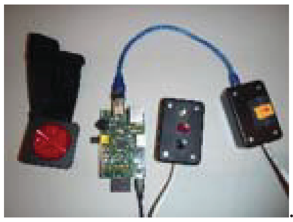


Fig. 4 : Components of the Device from left to right, Bluetooth GPS Receiver, Raspberry Pi, Buttons for UI.

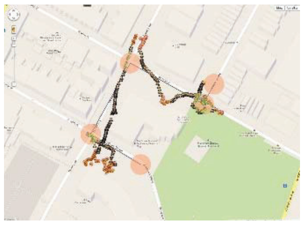


Fig. 5 : The results of the route in an urban environment with tall buildings.

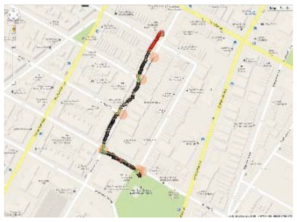


Fig. 6 : Results in residential area - Perfect Accuracy.

XI. Conclusion And Future Work

There are other research ideas out there that have the same goal as our project, however their total cost often surpasses $1,000. The entire production cost of our device is only $138 (Raspberry Pi - $25, USB hub - $7, IMU - $60, Arduino - $20, Packaging - $20). Therefore this product has a very high cost benefit advantage. Additionally, it has been proven that the device is very effective in successfully and safely navigating people.

In the future, we would like to improve on the capabilities of this system by incorporating landmarks as saved destinations. We would like to find a more accurate and cost efficient GPS receiver, as well as a faster portable Linux computer. We would also like to develop an algorithm for position and velocity so that other methods of navigation such as dead reckoning can be implemented accurately.

ACKNOWLEDGEMENT

This work is supported in part by the USA Federal Highway Administration (FHWA) under grant # DTFH61-12-H-00002 and PSC-CUNY grant # 65789-00-43.

REFERENCES

[1]"CMU Sphinx," Carnegie Mellon University, 26 August 2012. [Online]. Available: http://cmusphinx.sourceforge.net/. [Accessed 9 October 2012].
[2]"Festvox," Carnegie Mellon University, 8 November 2010. [Online]. Available: http://festvox.org/festival/index.html. [Accessed 15 October 2012].
[3]"Florida Division of Blind Services," 28 February 2012. [Online]. Available: http://dbs.myflorida.com/resources/guide-dogs.php. [Accessed 17 October 2012].
[4]"Google Code," Phonetisaurus, 23 July 2012. [Online]. Available: http://code.google.com/p/phonetisaurus/. [Accessed 11 October 2012].
[5]"Google Code," monav, 23 April 2011. [Online]. Available: http://code.google.com/p/monav/. [Accessed 13 October 2012].
[6]Laulainen, E.; Koskinen, L.; Kosunen, M.; Halonen, K.; , "Compass tilt compensation algorithm using CORDIC," *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on* , vol., no., pp.1188-1191, 18-21 May 2008 .
[7]L. M. Brown, "Tactons: Structured Vibrotactile Messages for Non-Visual Information Display," *Department of Computing Science, University of Glasgow,* p. 213, April, 2007.
[8]Madgwick, S.O.H.; Harrison, A.J.L.; Vaidyanathan, R.; , "Estimation of IMU and MARG orientation using a gradient descent algorithm," *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on* , vol., no., pp.1-7, June 29 2011-July 1 2011.

[9]"MIT Language Modeling Toolkit," T-Party Project, 27 July 2012. [Online]. Available: http://code.google.com/p/mitlm/. [Accessed 10 October 2012].
[10]S. Erle, "CSPAN," 17 May 2005. [Online]. Available: http://search.cpan.org/~sderle/Geo-Coder-US/. [Accessed 15 October 2012].
[11]"Sourceforge," Audacity, 3 March 2012. [Online]. Available: http://audacity.sourceforge.net/. [Accessed 19 October 2012].

[12]U. C. Bureau, "Census," 6 9 2012. [Online]. Available: http://www.census.gov/geo/maps-data/data/tiger.html. [Accessed 1 10 2012].
[13]Wilson, Jeff, et al. "Swan: System for wearable audio navigation." *Wearable Computers, 2007 11th IEEE International Symposium on*. IEEE, 2007.