StraightGoer: A Structure From Motion Navigation Aid for People With Visual Impairments

Katie Dektar
Stanford University
dektark@stanford.edu

Abstract

Independent navigation can be a challenge for people with significant vision impairments. While there are some non-technical solutions like guide dogs and white canes, and some technical solutions involving specialized hardware or routes within known environments or turn-by-turn directions to specific destinations, there does not seem to be a solution that would help a person walk straight in a given direction until they chose to stop. Using Google's ARCore Android SDK to provide camera pose and position tracking via SLAM, an Android app was developed to fill this need. The app communicates information about how far a user must turn left or right to follow a target ray within the world, and is robust to phone z axis rotations (portrait/landscape) and x axis rotations (tilt up/down). While feedback may be provided via speech, users may choose to rely on only sonification of the direction to reach the target ray, making use of both pitch (for angle) and stereo audio (indicating direction). In experiments, users were able navigate to an average of 3 feet from a target point that was an average of 250 feet away from the start, even when taking significant detours from the target path. This functionality could be built into existing mapping and navigation solutions in order to help people with visual impairments navigate.

1. Introduction

Over 36 million people in the world are blind, about 0.49% of the global population. Conditions that give rise to blindness can occur for a variety of reasons at any age, but more often affects older adults. Thus as the world's population grows and ages this number will grow [1]. Yet now, modern technology, especially applied concepts from 3D computer vision, can be used to augment the information a blind person can gather from the world with their own senses. Providing well structured augmented visual data can help people with visual impairments with certain vision-based tasks that can otherwise be hard to perform independently. For example, using visual data to give a person relevant, timely information about their environment can be a valuable aid to independent navigation.

Generally the two main goals of navigation are obstacle avoidance, the "local problem" of not bumping into anything, and wayfinding, the "global problem of planning and following a path towards the destination" [19]. For people with full vision these can both be solved by sight: people may watch their step to avoid obstacles and use their vision to ensure they generally stay on a target trajectory.

Many people with visual impairments, including those who are legally blind and completely blind, must rely on alternative techniques to navigate their environments. For example, many people may rely on a white cane for tactile feedback or a guide dog's vision to avoid obstacles [18], and on their own memory and sense of space for wayfinding. Support from a sighted person is effective but not always be available; hiring such a person would be prohibitively expensive for most people.

Besides white canes, guide dogs, and sighted guides, there have been quite a few research projects proposing navigation aids for users with visual impairments to address the problem of independent navigation. These solutions often rely on specialized hardware, or assume the user is navigating in a 3D environment that has been previously modeled, or that they are navigating along a memorized route, or on navigating to a particular destination. None of these solutions seems to solve the problem of walking straight along an arbitrary direction in the the world and then picking a new straight direction when ready.

1.1. Walking straight

I asked a co-worker who is blind what he would like to be solved with computer vision, and he said "walking straight".

Sighted people are able to pick a target destination visually and walk towards that destination while avoiding obstacles. For example, a sighted person could ask someone else in a train station or public park, "where is the restroom?", and they can simply point "straight over there", and the first

person can set off in that direction and arrive at my destination.

People with "normal" vision may take this interaction for granted. However, for a person who is blind it can be easy to veer slightly off-course, especially when navigating around obstacles. A person may walk around a table or tree end up going in a slightly different direction than they intended, missing expected landmarks, and then getting lost. This is especially tricky for people in wide open spaces where there is no sidewalk edge or wall edge to follow, for example in large public transit stations, parks, and pedestrian malls.

Although there are many pre-existing explorations for augmenting navigation for people who are blind or have visual impairments, I was not able to find pre-existing work that allowed users to generally explore and walk straight through their environment that made use of cheap, unobtrusive, readily available hardware, and that does not need GPS or floor-plans. This means there's not a solution that allows a blind person to up to someone, indoors or outdoors, and ask them to "point me in the direction of the bathroom". Yet in real life this happens with some frequency, and it's difficult for both the sighted people to explain how to navigate without visual cues, and for the person with visual impairments to follow their directions.

In this work, I propose and implement an Android application that could be used by people who are blind to navigate along a straight path. Specifically, the app is used to figure out the user's target direction and provide feedback when they deviate from that direction, without forcing them to stay on some particular route. Users can decide themselves which direction to go and the app can tell them if they've turned or if they are going straight. The application has no need for environment maps or floor plans, and works indoors and outdoors given sufficient lighting. There is no dependency on cellular/wifi data or GPS, so this works in even more situations, like in a state park, indoors, etc.

Feedback is provided to users via non-visual real-time cues, including stereo sound and text-to-speech cues.

Making this an Android application running on a normal smartphone greatly increases potential access and equity. Using a cell phone would avoid the stigma associated with wearable cameras and the fatigue of an augmented heavier cane. Cell phones with decent cameras are pretty ubiquitous, and are used by people with visual impairments as much as any other group. It seems worthwhile to explore whether a cell phone application could augment navigation for users with visual impairments to help reduce hazards and navigate safely in unfamiliar areas.

2. Background and Related Work

2.1. Existing navigation solutions

Most of the existing navigation solutions for people with visual impairments make use of specialized hardware, operate within known environments, or require users to go to a know destination location.

2.1.1 Using specialized hardware

Many existing navigation aids have been proposed. Several involve specialized hardware, including RGB-D cameras [18], "wearable" stereo cameras [12, 13], and 3D cameras [17, 19]. Many of these existing proposals involve mounting the technology on a cane or on the body. This can be heavy; in fact some standard white canes are already hard to use for long duration due to weight. Specialized hardware can also be awkward to use and carry stigma. In addition, specialized hardware for users with disabilities is nearly always extremely expensive due to the smaller market size that makes producing hardware not cost-effective for manufacturers. Even with insurance, getting access to assistive technology can be cost-prohibitive for many, and for those without insurance, nearly impossible.

2.1.2 Within known environments

Other prior work involves additionally navigating in pre-existing 3D environment models [11, 18, 19], and makes use of 3D computer vision to determine the user's current pose within a known environment model or floor plan. This may fail if the map becomes outdated, and more often focuses only on indoor environments.

Other works have focused on navigating along memorized routes [15], and keep the user on the target path. This may be inflexible when obstacles create detours from the known route.

2.1.3 To known destinations

Finally, a lot of prior work focuses on navigation to a particular destination (wayfinding), and works on augmenting directions to the user so that they can stay on a target path through indoor environments [4, 5, 14, 20, 21]. In this case, the user must have a known destination in mind. When indoors, GPS generally does not work well, so most research seem to make use of having a known indoor map.

When outdoors, there are solutions that make use of GPS, but even these have limitations estimating pose due to the accuracy of GPS, especially in large cities where not enough satellites are visible through large buildings [3], or in public parks where there is insufficient cellular data.

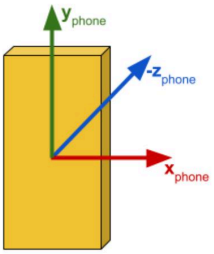


Figure 1. The phone coordinate system has negative z pointing in the direction of the camera (out the back of the phone), y along the long axis of the phone, and x along the short axis.

2.2. SLAM on Android

Google has released the free ARCore SDK, which makes it possible for developers to build augmented reality Android applications. ARCore uses structure from motion to build up 3D world information and phone pose information:

> ARCore uses a process called simultaneous localization and mapping, or SLAM, to understand where the phone is relative to the world around it. ARCore detects visually distinct features in the captured camera image called feature points and uses these points to compute its change in location. The visual information is combined with inertial measurements from the device's IMU to estimate the pose (position and orientation) of the camera relative to the world over time. [7]

3. Approach

I built an Android app which uses Google's ARCore SDK [7] to do simultaneous localization and mapping (SLAM). From ARCore I can get the current camera pose (position and rotation) in world coordinates in real time, which is sufficient input to construct the application.

3.1. Usage

A user may point their phone in a target direction and then tap the screen to indicate that is their direction of choice. Then as their phone moves in space, the app gives directions on how far left or right the phone must turn to move back towards and then along the ray defined by the initial camera direction, regardless of forwards/backwards tilt and portrait/landscape orientation. More technically, the phone's y axis rotation is used to tell the user which angle with respect to the phone's y axis brings the phone closest to the target ray, while x axis rotation and z axis rotation are ignored (see Figure 1 of the phone coordinate system).

The user is always directed towards a point that is 2 meters (Note: this was modified to 5 meters for outdoor experiments) from their current position projected back onto the target ray (see Figure 3, with the hope that this will keep the user from walking at right angles if they stray further from the ray and need to come towards it again.

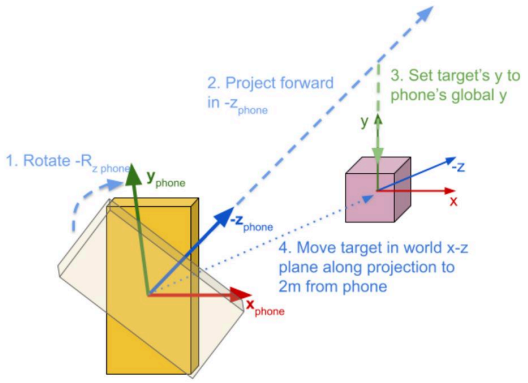


Figure 2. Diagram of steps to construct world-space target point from world-space camera pose.

3.2. Technical details

I started with Google's HelloAR tutorial [9] and example app from GitHub, which initially runs a hit test at user tap and places a target at the hit geometry intersection point. The target is rendered over top the camera image as a user moves around to create an augmented view of reality. I was able to re-use some of this functionality, namely setting up ARCore and rendering target points.

I had to re-implement detecting taps because it was done in a way that doesn't work with the Android screen reader, TalkBack, but this was straightforward and will not be discussed here.

There are three main steps:

1 Detect and store the target ray.

2 Update as the phone moves: calculate the relationship between the phone and target ray as the phone's position and orientation changes

3 Inform the user: Provide appropriate, not-overwhelming feedback via stereo sonification of the current offsets between phone and target ray.

3.2.1 Detect and store the target ray

The approach to detect and store the target ray is outlined in Figure 2. Specifically, when the screen is tapped, the steps taken by the app are:

1 Get the current camera's pose, and remove the Z axis rotation. The camera Pose object surfaces a quater-

nion, from which the z axis Euler angle can be extracted based on math from eucliedeanspace.com [2]. I then invert the Euler angle Z, reconstruct a new quaternion based on math from Wikipedia [16], and compose it with the camera Pose to undo any rotation.

```
Pose cameraPose =
    camera.getDisplayOrientedPose();
float[] q =
    cameraPose.getRotationQuaternion();
double lock = q[0] * q[1] + q[2] * q[3];
// Code to check for gimbal lock omitted.
double theta = Math.asin(2 * lock);

double psi = 0;
double phi = 0;
theta = -1 * theta;
double cr = Math.cos(phi * 0.5);
double sr = Math.sin(phi * 0.5);
double cp = Math.cos(psi * 0.5);
double sp = Math.sin(psi * 0.5);
double cy = Math.cos(theta * 0.5);
double sy = Math.sin(theta * 0.5);
q[3] = cr * cp * cy + sr * sp * sy;
q[0] = sr * cp * cy - cr * sp * sy;
q[1] = cr * sp * cy + sr * cp * sy;
q[2] = cr * cp * sy - sr * sp * cy;

Pose newRotation = Pose.makeRotation(
    q[0], q[1], q[2], q[3]);
cameraPose = cameraPose.compose(newRotation);
```

2 Construct an initial target point by projecting forward along the camera's z axis. ARCore provides methods on Pose that allow you to make translations with regard to the current Pose's frame, so it is straightforward to make a translation that's 2 meters down the -z axis from the camera.

```
Pose targetPose = cameraPose.compose(
    Pose.makeTranslation(0, 0, -2.0f))
    .extractTranslation();
```

3 Drop the target Pose down to the same world-space x-z plane as the phone, e.g. make the world-space y coordinate of the target Pose match the world-space y coordinate of the phone's Pose. *Note: This is helpful for rendering, but assumes the ground is relatively flat and the phone is generally held at about the same height during the interaction. When calculating angles, global y translation is ignored, this assumption has no impact on usability.*

```
targetPose = targetPose.compose(
    Pose.makeTranslation(0,
        cameraTranslation.ty() -
        targetPose.ty(), 0));
```

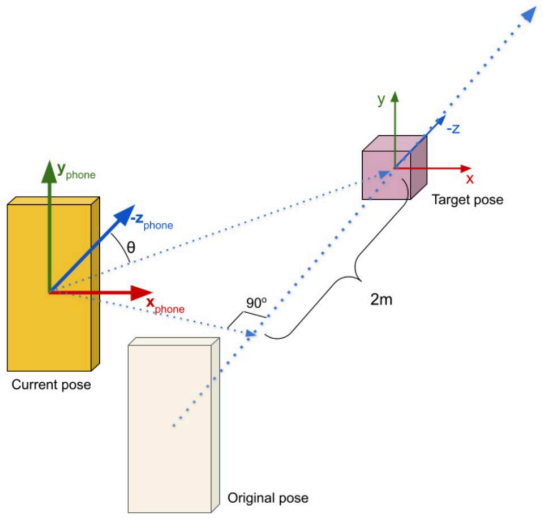


Figure 3. Diagram of geometry used to calculate target direction from current camera pose, original camera pose and target pose.

4 Finally, the target point is projected forward along the camera's z axis so that it is 2 meters ahead in the world x-z plane. This is done using simple 2D geometry and similar triangles.

```
float meters = 2.0f;
float x1 =
    targetPose.tx() - cameraPose.tx();
float z1 =
    targetPose.tz() - cameraPose.tz();
float x2 = (float) Math.sqrt(
    Math.pow(meters * x1, 2) /
    (Math.pow(z1, 2) + Math.pow(x1, 2)));
if (x1 < 0) {
  x2 *= -1;
}
float z2 = z1 * x2 / x1;
targetPose = Pose.makeTranslation(
    x2 + cameraPose.tx(),
    targetPose.ty(),
    z2 + cameraPose.tz());
```

I can store the resulting camera and target Poses as as ARCore Anchor objects, which make them trackable across multiple frames.

3.2.2 Update as the phone moves

The approach to get the offsets from the current camera pose to the target ray is outlined in Figure 3. Specifically, the steps taken by the app every frame are as follows:

1 Get the current camera pose and remove z axis rotation as in 3.2.1.1 above.

2 Get the original camera pose and compute the current camera pose's offset in the original camera Pose's frame.

```
// Current camera space.
float[] cameraPt = {0, 0, 0};
// World space.
cameraPt = cameraPose.transformPoint(cameraPt);
// Original camera space.
float[] newCameraPositionInOrigSpace =
    origCameraPose
        .inverse()
        .transformPoint(cameraPt);
```

3 Calculate the current target pose at 2 meters along the ray. This is done with the same math as 3.2.1.4, but witha different offset that represents 2 meters beyond the current camera pose's z coordinate in the original camera frame:

```
meters = newCameraPositionInOrigSpace[2] -
    TARGET_DISTANCE_ALONG_RAY;
```

3 Get the updated target pose in current camera coordinates. This represents how much the camera must move to reach the target, and will make angle calculation easier.

```
float[] targetDir = cameraPose.inverse()
    .transformPoint(targetPt);
```

4 Calculate the offset angles using 2D geometry in the x-z plane. This produces both an offset y-axis angle between 0 and 180, and a direction given by the sign of `tx`.

```
float tx = targetDir[0];
float tz = targetDir[2];
float dist = (float) Math.sqrt(
    Math.pow(tx, 2) + Math.pow(tz, 2));
double theta = Math.acos(-tz / dist);
```

3.2.3 Provide feedback via sonification

The target user of this app is not able to see the augmented reality rendered on the screen by ARCore. Therefore I sonified the current theta from the target. I use audio cues rather than text-to-speech so that this can be used as background sound and the user might still have a conversation or listen to other instructions (like walking instructions from a Maps app, for example).

The sonification implementation makes use of the jsyn Java audio library [10] and example code from open-source Science Journal [6]. The app creates stereo audio that changes in both stereo volume and pitch as the phone moves relative to the target. Specifically, each time the updated angle and direction was calculated, the app:

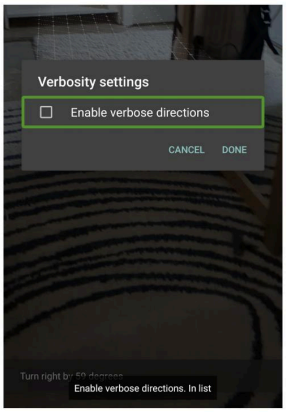


Figure 4. Users may adjust verbosity settings to get a live textual description of how to navigate or just use tone. Note that TalkBack is enabled in this screenshot and I've turned on the developer option to show speech at the bottom of the screen.

1 Determines the note to play: calculates frequency of a sine wave to play by mapping the angle theta to a pitch within a range of pitches, and then converts that pitch to frequency. If the angle is straight enough, plays a tone two pitches lower than the the next lowest tone, to make it stand out.

```
int minPitch = (int) Math.floor(AudioMath
    .frequencyToPitch(FREQ_MIN)) + 2;
int maxPitch = (int) Math.floor(AudioMath
    .frequencyToPitch(FREQ_MAX));
double freq = AudioMath.pitchToFrequency(
    (int) (theta / 180 *
        (maxPitch - minPitch) + minPitch));
if (theta < STRAIGHT_ENOUGH_THETA) {
    // Extra low note for straight enough.
    freq = FREQ_MIN;
}
```

2 Determines the amplitude at which to play the note, which is louder for the ear closer to the object and quieter for the ear further from the object.

```
double leftAmplitude =
    (Math.abs(90 - theta) / 90) / 2;
if (tx > 0) {
  // It's on the right.
  leftAmplitude = 1 - leftAmplitude;
}
double rightAmplitude = 1 - leftAmplitude;
```

The pitch and audio volumes for any degree of y axis rotation can be visualized in 5.

Finally, I made sure the app was compatible with TalkBack, the screen reader on Android. This included some

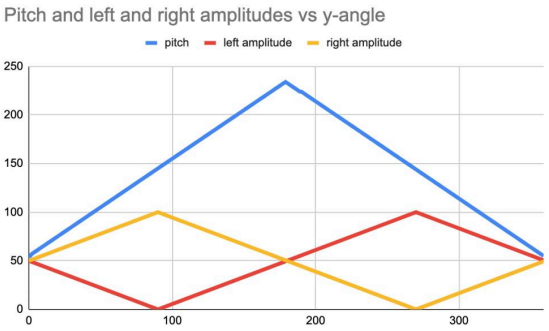


Figure 5. Generated pitch and left/right audio amplitudes across all 360 degrees of y axis rotation create a unique mapping for each degree of rotation.

refactoring to make sure that TalkBack users could still place targets (this was not possible with starter code), to provide helpful content descriptions, and to add verbosity settings for directions, which can be spoken on change or just on demand (see figure 4). In particular, I made the high verbosity navigation instructions more concise than the text displayed on the screen.

4. Experiments and Analysis

Supplemental materials: In addition to what is included here, you can see the source code on github.com/dektar/arcore-android-sdk and additional photos and videos of the app.

4.1. Functionality unit testing

Functionality testing was performed within my house and in my back yard to ensure robustness to rotations, including beginning the target at many assorted x and y rotations of the phone to ensure it was always only determined by phone y rotation, and performing many x and y rotations after tracking began to ensure the directions were only influenced by phone y rotation. See figure 6a, 6b, 6c, and 6d for examples, and short videos online.

ARCore on Android is known to degrade in darker conditions or when faced with flat, untextured surfaces like plane walls. Functionality testing indoors showed this to be the case: lights had to be on in the evening and one could not have most of the frame towards a blank surface.

Functionality testing was also performed with the stereo audio, using one headphone side at a time, to ensure audio output matched the goal (audio is louder in the ear closer to the target, pitch is lowest facing the target and highest at 90 degrees, there are no sudden jumps or clicks). This was confirmed with various rotations of the phone, and user, along all axes.

4.2. Real-world testing: Walking straight with eyes closed and headphones

Data from all real-world tests is summarized in figure 7.

The app was taken to local parks park with headphones and my family (figure 10 to make sure the user testing didn't bump into anything. My husband and I, both sighted, did several trials, outlined below. As these trials were outdoors, the 2 meters target distance described in previous sections was changed to 5 meters.

4.2.1 Outdoor control

In a control trial, the user attempted to walk straight with their eyes closed but did not use the app. In this trial they veered slightly off-course, and while it looks locally straight it globally misses the target by around 50 feet. See figure 8a.

There was no control trial with a detour as users felt they would have immediately gotten lost.

There was no control trial indoors as a user could probably feel around with their hands and feet to get to the destination, and GPS wouldn't be able to show the path difference.

4.2.2 Walking straight

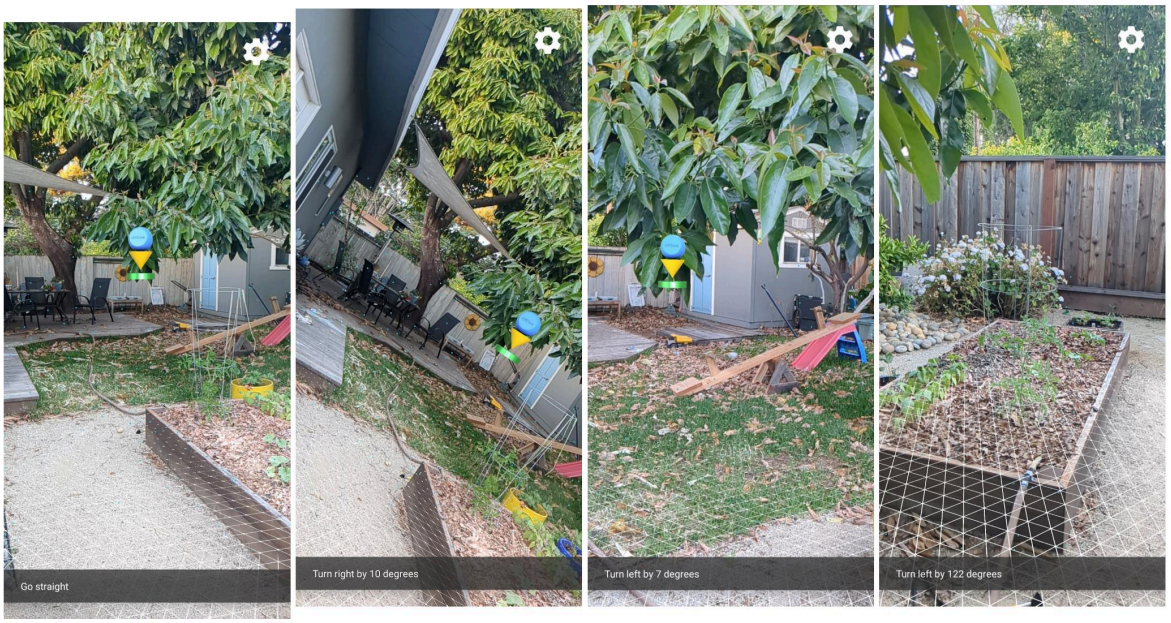
Indoor trials were performed in the author's home, and involved maybe spinning in place, then crossing the house while avoiding low counters, chairs, and wall edges. The longest straight route through the house was 40 feet and the user arrived exactly at the light switch targeted (within 1 inch).

For all outdoor trials, the phone was pointed at a target (like a tree, piece of fence, object on the ground) across the park, the screen was tapped to set the target ray, then the user closed their eyes and walked with headphones on until a sighted support person told them they had arrived or could stop.

1 The target was a tree, 150 feet away, at the same elevation. The user closed their eyes, and walked to within 3 feet of the trunk using only stereo audio (stopped when they could feel the grass change to dirt near the roots).

2 The target was tree about 6 feet above (y coordinate) the start position and 235 feet away (z coordinate). The user navigated to within a few feet of it using only stereo audio and TalkBack. A full video of this trial, which took about 2 minutes and 30 seconds, is in the supplemental materials.

3 Trials were also also attempted walking straight through a house, which was fairly small and fairly easy get accurate results. These trials are not reported in the results.



(a) Screenshot of the app just after tapping the screen to place a point that slides along the target ray. A message at the bottom reads, "go straight".

(b) Screenshot taken after stepping slightly to the side and rotating the phone in z at about 45 degrees. The message at the bottom reads, "turn right by 10 degrees".

(c) Screenshot taken several more steps towards the target, phone vertical. The message at the bottom reads, "turn left by 7 degrees". The target has slid backwards along the ray.

(d) Screenshot of the app, having turned nearly fully around. The target point is no longer in the frame, the message reads, "turn right by 122 degrees".

Figure 6. Functionality testing: screenshots of the app in my back yard, from placing the target pose and walking towards it, with some phone rotations.

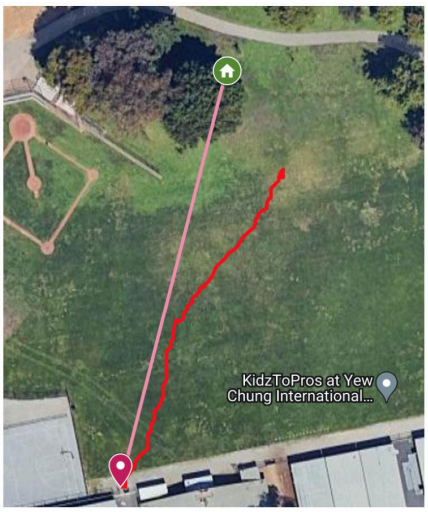
Initial target distance	Detour	Final target distance	Outdoors	App used
15 ft	none	1/2 ft	no	yes
40 ft	spin	1/12 ft	no	yes
20 ft	spin	1 ft	no	yes
300 ft	none	50 ft	yes	**no**
150 ft	none	3 ft	yes	yes
235 ft	none	3 ft	yes	yes
300 ft	15 ft at 30^o	3 ft	yes	yes
235 ft	20 ft at 90^o	1 ft	yes	yes
304 ft	20 ft at 80^o	2 ft	yes	yes
322 ft	100 ft at 90^o	6 ft	yes	yes

Figure 7. Summary of results of indoor and outdoor trials completed by sighted users with their eyes closed. Trials are described in detail in 4.2.

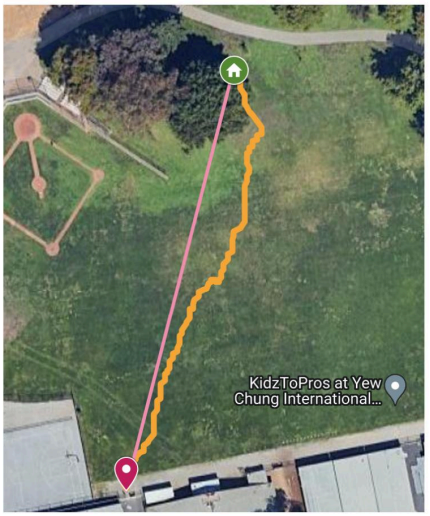
4.2.3 Trials with detours

In the following trials, the user went intentionally of-course to check how well they could get back to the target ray. There was no occlusion (e.g. no buildings or walls between the user and target) but the target was not always anywhere near the camera's frustrum.

The setup was the same as above: The phone was pointed at a target (like a tree, piece of fence, object on the ground) across the park, the screen was tapped to set the target ray, then the user closed their eyes and walked with headphones on until a sighted support person told them they had arrived.

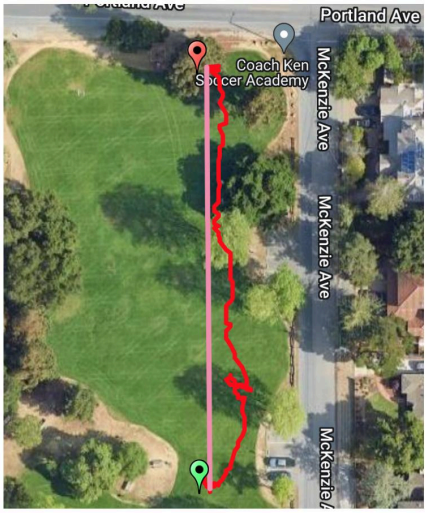


(a) GPS track from control trial. In the red track, I tried to walk straight, with my eyes closed, from the red marker at the bottom targeting the green marker at the top (along the pink path), without using StraightGoer. Although the path was locally straight, globally I ended up fairly off-target.

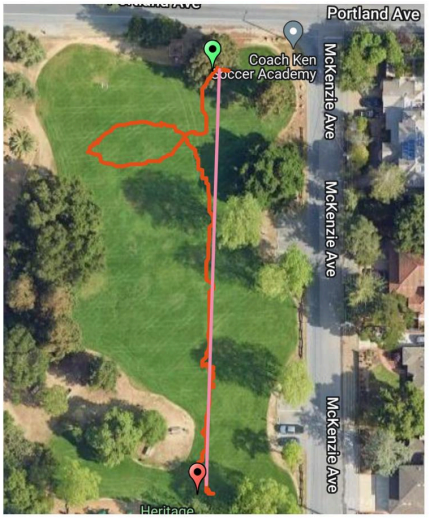


(b) GPS tracks from trial #3. In the orange track, I set up StraightGoer at the green marker oriented to the red marker (along the red path), closed my eyes, then intentionally traveled in the wrong direction (to my left) for a few meters before using the app to guide me to the red marker.

Figure 8. GPS tracks from a control trial (no app) and a trial that began with a small detour.



(a) GPS track of path taken from green to red marker, with an intentional detour in the first third of the track.



(b) GPS tack of a 320 foot path taken from the green to the red marker, with an intentional 100 foot detour after 50 feet. The curve approaching the target direction after the detour shows the value of having the target point always 5 meters ahead on the target ray.

Figure 9. GPS tracks from longer routes with larger detours

4 The user pointed the phone at a marked section of fence 300 feet from my current location. They closed their eyes, began walking based on app sonification, then spent a few seconds going off 40 degrees to the left to pretend they were going around something. Afterwords, they resumed going towards the target with

stereo audio sonification and TalkBack. The user arrived within 3 feet of the destination despite the detour. See figure 8b.

5 A second sighted user tried to navigate with their eyes closed along the same path as item 2, with an intentional large 90 degree detour of about 20 feet taken about 2/3 of the way through. They were guided back to the path using only stereo audio sonification and ended up within 1 foot of their target.

6 Similar to the above, a user attempted a 304 feet path that went over a small hill (5 feet) near the end. The user took a small detour to the right and then were directed back along the ray. They arrived within 2 feet of the target. See figure 9a.

7 Similar to the above, a user attempted a 322 feet path with a 100 foot detour at approximately 90 degrees that began 50 feet into the trial (see figure 9b). Here we can really see how the target path curves as the target point moves along the ray at 5 meters ahead of the user. We also see that the user ended the detour heading slightly backwards because the right earbud was malfunctioning slightly, not producing noise, and so they'd lost some left/right perception. In this trial, which was the longest distance attempted, the user missed the target by about 6 feet.

4.3. User feedback

Qualitative feedback was gathered from the sighed users who performed above trials (with their eyes closed).

Users felt the app was intuitive, easy to use and helpful:

- The stereo audio "makes it easy" to perceive when one is on track
- It was "surprisingly accurate".
- Quick learning curve.
- Easy to get back on track even if distracted or lost focus

But there can be improvements to the direction $\rightarrow$ sonification mapping to make it easier:

- It's hard to perceive tell if one is pointed slightly left or right without sweeping because stereo audio is quite similar at small angles.
- Similarly, it's harder to perceive direction using only one ear.
- The straight tone isn't sufficiently different from surrounding tones to stand out.

Figure 10. My family helps me test stereo sonification and placing target rays.

5. Conclusion

In this work an Android app was built on ARCore that supports independent navigation along a target path in the world. The app was shown to be fairly accurate in ranges from 15 to 300 feet indoors and outdoors, when used during well-lit conditions and in spaces with enough surface texture for ARCore to detect. For a person who is not using vision to navigate, using the app to navigate led to significantly more accurate navigation than attempting navigating without it. However, the app does have several notable drawbacks due to the SLAM technology used by ARCore, and many areas that could be improved if it were to be released to users.

The notable drawbacks are due to the fact that the app relies on textured surfaces and loses track of world position when faced with poor lighting condition or flat surfaces. If a user comes face to face with a wall, for example, ARCore may not be able to track world positions accurately. This means it works well for navigating in open spaces or long hallways but would not do well for navigating across a maze, for example, where many walls are encountered.

There are many areas where the app could be improved, given sufficient developer time.

First, ARCore provides a depth array and hit testing, which could be used to support obstacle detection when walking. The app could detect obstacles ahead of the user that come closer than a meters, and aren't the ground plane, and warn the user through vibration, sonification or speech of these obstacles, to avoid bumping into them. In fact, ARCore provides segmentation and "scene semantics", so a user could potentially even be informed of what is near them.

ARCore's depth could also be used to set a reasonable target point distance; this work used 2 meters indoors and 5 meters outdoors, but a better distance could be set dynamically based on the furthest detected distance in a given direction.

Finally, there could be many improvements to sonifica-

tion. Although outside the scope of this class, the sonification is crucial for making it easy to navigate without requiring too much focus / attention. For example, instead of a pure tone, beeping or blipping when on-target could make it more clear than the 2-pitch change currently implemented. In addition, the requirement and sensitivity to working stereo audio is not great for real-world scenarios where people might want to keep one ear free for listening to the world (see route where I traveled backwards because of malfunctioning earbuds), or for users who have some deafness. If this was taken forward, it would be best to do user studies to see what is most helpful for actual users.

As a next step, this author is hoping to reach out to the team at Lookout [8] by Google to see if they want to include this functionality in their app.

References

[1] Ackland, Peter and Resnikoff, Serge and Bourne, Rupert. World blindness and visual impairment: despite many successes, the problem is growing. *Community Eye Health*, 30(100):71–73, 2017. 1

[2] Martin John Baker. Maths - conversion quaternion to euler, 2023. 4

[3] W. Balachandran, F. Cecelja, and P. Ptasinski. A gps based navigation aid for the blind. In *17th International Conference on Applied Electromagnetics and Communications, 2003. ICECom 2003.*, pages 34–36, 2003. 2

[4] Yuki Endo, Kei Sato, Akihiro Yamashita, and Katsushi Matsubayashi. Indoor positioning and obstacle detection for visually impaired navigation system based on lsd-slam. In *2017 International Conference on Biometrics and Kansei Engineering (ICBAKE)*, pages 158–162, 2017. 2

[5] Giovanni Fusco and James M. Coughlan. Indoor localization for visually impaired travelers using computer vision on a smartphone. In *Proceedings of the 17th International Web for All Conference*, W4A '20, New York, NY, USA, 2020. Association for Computing Machinery. 2

[6] Google. Google science journal (github), 2020. 5

[7] Google. Arcore: Fundamental concepts, 2023. 3

[8] Google. Lookout app on google play, 2024. 10

[9] Google. Quickstart for android: Helloar, 2024. 3

[10] Mobileer Inc. Jsyn - audio synthesis api for java, 2015. 5

[11] Bing Li, Juan Pablo Muñoz, Xuejian Rong, Qingtian Chen, Jizhong Xiao, Yingli Tian, Aries Arditi, and Mohammed Yousuf. Vision-based mobile indoor assistive navigation aid for blind people. *IEEE Transactions on Mobile Computing*, 18(3):702–714, 2019. 2

[12] J. Manuel Saez, F. Escolano, and A. Penalver. First steps towards stereo-based 6dof slam for the visually impaired. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Workshops*, pages 23–23, 2005. 2

[13] Vivek Pradeep, Gerard Medioni, and James Weiland. Robot vision for the visually impaired. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, pages 15–22, 2010. 2

[14] Kruthika Ramesh, S. N. Nagananda, Hariharan Ramasangu, and Rohini Deshpande. Real-time localization and navigation in an indoor environment using monocular camera for visually impaired. In *2018 5th International Conference on Industrial Engineering and Applications (ICIEA)*, pages 122–128, 2018. 2

[15] Sylvie Treuillet, Eric Royer, Thierry Chateau, Michel Dhome, and Jean-Marc Lavest. Body mounted vision system for visually impaired outdoor and indoor wayfinding assistance. 2007. 2

[16] Wikipedia. Conversion between quaternions and euler angles. 4

[17] Cang Ye, Soonhac Hong, Xiangfei Qian, and Wei Wu. Co-robotic cane: A new robotic navigation aid for the visually impaired. *IEEE Systems, Man, and Cybernetics Magazine*, 2(2):33–42, 2016. 2

[18] He Zhang, Lingqiu Jin, and Cang Ye. An rgb-d camera based visual positioning system for assistive navigation by a robotic navigation aid. *IEEE/CAA Journal of Automatica Sinica*, 8(8):1389–1400, 2021. 1, 2

[19] He Zhang and Cang Ye. An indoor navigation aid for the visually impaired. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 467–472, 2016. 1, 2

[20] He Zhang and Cang Ye. An indoor wayfinding system based on geometric features aided graph slam for the visually impaired. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(9):1592–1604, 2017. 2

[21] Xiaochen Zhang, Xiaoyu Yao, Yi Zhu, and Fei Hu. An arcore based user centric assistive navigation system for visually impaired people. *Applied Sciences*, 9(5), 2019. 2