WallTrace: Indoor Navigation via AR and Textual Cues

Fata Asgharzadniazi Matthew Falcone Abhi Ketan Prajapati

CSC311 Summer 2025 Course Project
University of Toronto

Abstract

Navigation in unfimiliar environments is a challenge that humans have long faced. From ancient methods that relied on the stars to the sophisticated software available today, each step has helped us find our way around more efficiently. Historically, the primary focus has been on outdoor navigation or traversing city streets. Even though there have been recent developments in smaller-scale navigation—such as walking paths in parks being integrated into applications like Apple Maps[3]—indoor navigation remains underexplored due to various inherent challenges. In this paper, we present a learning-based mobile application designed not only to train a locator model but also to assist users in navigating through a building by leveraging the textual information present in their surroundings.

0 Introduction

Navigating unfamiliar indoor spaces such as university buildings, hospitals, or malls has the potential to be significantly improved. While GPS has revolutionized outdoor navigation, it performs poorly indoors due to signal degradation and the inability to estimate vertical elevation. This gap has spurred interest in alternative solutions that leverage on-device sensors, computer vision, and machine learning for quick and highly efficient indoor navigation.

Recent advances in Augmented Reality (AR) and Optical Character Recognition (OCR) have opened new avenues for indoor navigation, allowing mobile devices to understand and interpret textual cues present in the environment. Room numbers, hallway signs, and other printed information can be captured and processed in real time, offering a low-cost and hardware-friendly method to identify the location of the user indoors.

In this work, we introduce WallTrace, a pipeline designed to provide real-time indoor navigation assistance using AR and textual cues extracted from the environment. Our pipeline is comprised of two main stages:

- **Model1**: A fine-tuned BERT-based binary classifier that filters OCR-detected text to filter out redundant text cues scanned from the environment.
- **Model2**: A classifier that uses semantic sentence embeddings from Model1's filtered text to predict the user's current location within a building.

Unlike traditional SLAM or LiDAR-based systems that require heavy computation or specialized hardware, WallTrace is designed to be lightweight and scalable, relying only on textual information and relative positioning. Our approach was tested on data collected from the Bahen Centre at the University of Toronto, and demonstrated high accuracy in localizing users based solely on textual cues. The main takeaways from this project are:

- We propose a scalable indoor localization pipeline that relies on textual cues rather than physical maps or heavy 3D scanning.
- We evaluate the performance of two classification approaches (KNN and Random Forests) for location inference, and compare their accuracy and inference speed.

The rest of this paper is structured as follows: Section 1 reviews related work in indoor navigation and

Copyright © by the paper's authors. Copying permitted for private and academic purposes.

In: W. Aigner, G. Schmiedl, K. Blumenstein, M. Zeppelzauer (eds.): Proceedings of the 9th Forum Media Technology 2016, St. Pölten, Austria, 24-11-2016, published at http://ceur-ws.org

machine learning-based localization. Section 2 describes our data collection process, models, and training methods. Section 3 presents experimental results, while Section 4 discusses limitations and potential improvements.

1 Related Work

1.1 Navigation in General

The latest research in navigation and mapping technologies have touched on the utilization of the newly emerging technologies being adopted into this domain. Such technologies include, but are not limited to, Augemented Reality (AR) Navigation, Autonomous Navigation, Machine Learning (ML), Artificial Intelligence (AI) and integration of Internet of Things (IoT) devices. Various combinations of these approaches, along with different architectural frameworks, are utilized to research and develop tools tailored for specific use cases. For example, planetary mapping and navigation[7], which undergoes infrequent changes, necessitates a distinct approach compared to mapping and navigating a crowded urban setting that changes every moment [4]. The characteristic needs of each use case are influenced by multiple factors, including scale and the frequency of environmental changes, which ultimately impact the reliability of the software or tools available to users.

1.2 Indoor Navigation

Prior to the advent of Simultaneous Localization and Mapping (SLAM) [2], indoor localization and navigation did not attract significant interest within the fields of computer science and engineering, nor was it regarded as a straightforward problem to solve. Later, the application of SLAM concepts to indoor environments garnered attention, with initial approaches employing statistical methods [1] and appearance-based techniques [6] for the localization of mobile robots. As the popularity of Convolutional Neural Networks (CNNs) surged within the research community, novel Dense Correspondence-based approaches [5] began to emerge, which reached higher accuracies than the traditional methods by leveraging the power of CNNs to capture a wider range of patterns in images.

As AR technologies gain popularity, indoor navigation has become more accessible than ever. This transformation is largely attributed to the integration of advanced AR hardware, such as LiDAR sensors, into an increasing number of mobile devices and vehicles, which facilitates the delivery of precise AR experiences. Concurrently, the development of hardware-free indoor positioning systems has emerged as a viable alternative for navigating complex indoor environments, including hospitals, shopping malls, educational campuses, and office buildings. These systems can operate effectively on most contemporary smartphones, even in the absence of specialized hardware like LiDAR sensors. While hardware-free indoor positioning systems offer a practical solution, they tend to lack the accuracy of hardware-dependent systems. It is important to note that both categories of indoor navigation systems face inherent challenges that limit their precision. These challenges can be summarized as follows:

Figure 1: An Overview of the WallTrace pipeline

1. The absolute GPS position is not sufficiently accurate for indoor navigation.
2. Indoor environments typically feature multiple floors, complicating the ability of GPS devices to estimate elevation accurately, while readings from elevation sensors found in mobile devices is also be prone to inaccuracies.
3. Indoor environments change frequently, necessitating that ML models used in ML approaches receive continuous training to maintain their accuracy. This requirement presents significant logistical challenges, as constant retraining may not be feasible.

2 Methodology

In developing our method, we aimed not only to tackle the problems mentioned above but also to make our approach scalable and accessible. An overview of our training pipeline is illustrated in Figure 1. Each part of a building is represented by a Waypoint, which is assigned a unique ID (UUID), and a name for the convenience of data collection. Each Waypoint contains a list of Strings corresponding to the text in its vicinity. Since keeping track of visual objects and obtaining a 3D scan of the environment is computationally expensive, and neither scalable nor robust for training a learning-based localization model, our approach uses the text surrounding each Waypoint as input, which is generally less likely to change over time. Finally, each Waypoint has a list of neighbours and their corresponding relative positions.

2.1 Software Used

Our pipeline is implemented through an iOS application, primarily due to the widespread adoption of

the iOS Operating System (OS) and the advanced, highly accurate frameworks and tools provided by Apple. Our application is tested on an iPhone 15 Pro Max, which comes with a LiDAR scanner. A more detailed discussion of the specific use cases and the rationale behind each Apple framework utilized can be found in section 2.2.

2.2 The Absolute Position Problem

As discussed in an earlier section, GPS data is unsuitable for indoor navigation due to its accuracy and the lack of elevation data. The ARKit uses the initial 6D Pose of the device as the reference for each initialized AR session (ARSession), which yields significantly more reliable results compared to other methods. While this approach is decently accurate in a medium-sized room, it is prone to drift in smaller spaces, such as navigating a campus building floor. To address this issue, our method keeps track of the relative positions. Furthermore, using the initial 6D pose as the reference also means that each session will have a different reference, which is the problem that our localization model is designed to solve. In our dataset, we store the positions in ENU coordinates. When the ARSession begins, the device's heading is obtained through the Core Location framework. We then apply a conversion to the position provided by the ARKit framework, aligning them with the geographical axes to ensure consistency across sessions, and finally saving them in ENU coordinates with a fixed $z = 0$:

$$\mathbf{p}_{\text{AR}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \mathbf{p}_{\text{ENU}} = \begin{bmatrix} e \\ n \\ u \end{bmatrix} \quad \psi \text{ (radians)}$$

$$R_y(-\psi) = \begin{bmatrix} \cos\psi & 0 & \sin\psi \\ 0 & 1 & 0 \\ -\sin\psi & 0 & \cos\psi \end{bmatrix}, S = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Then the ARKit $\rightarrow$ ENU mapping is

$$\mathbf{p}_{\text{ENU}} = S\, R_y(-\psi)\, \mathbf{p}_{\text{AR}}.$$

Component-wise,

$$\begin{aligned} e &= \cos\psi\, x + \sin\psi\, z, \\ n &= \sin\psi\, x - \cos\psi\, z, \\ u &= y. \end{aligned}$$

The inverse (ENU $\rightarrow$ ARKit) is

$$\mathbf{p}_{\text{AR}} = R_y(\psi)\, S^{-1}\, \mathbf{p}_{\text{ENU}} = R_y(\psi) \begin{bmatrix} e \\ u \\ -n \end{bmatrix}.$$

2.3 Data Collection

Training data for Model2 was prepared using textual cues from the walls of UofT's Bahen building. This data can be found under WaypointInference/-data/waypoints.json in the GitHub repository. Training data collection for Model1 is explained in the next section.

2.4 Model1: Filtering Useful Text

A general scan of a room using Apple's built-in OCR through the ARKit framework often captures extraneous text as part of the location data. While reference text can aid in accurate placement for our second model, the scanned data frequently includes irrelevant or difficult-to-parse content, such as event posters or chalkboard writings. Below is an example of an unfiltered waypoint:


```
{
    "id" : "2E43A5C4-9083-447B-84C3-3
        ↪ AD0B40C8A7E",
    "name" : "4242",
    "neighbors" : [
      "4C60881C-9CE3-4D27-A4D7-68
          ↪ ABAC61DDD5",
      [
        -3.4070642,
        0.036506787,
        0
      ]
    ],
    "texts" : [
      "A CATOTT ON TORONTO",
      "RESET OP TORONTO",
      "VACANT",
      "& ONIVERITY OF TORONTO",
      "4242",
      "A UNIVERTY OF TORONTO",
      "Boardroom",
      "OUTSITY OF TORONTO",
      "OFFICE",
      "OVERSITY OF TORONTO",
      "UNIVERSIT OP TORONTO",
      "UNDERGRADUATE"
    ]
  },
```


This waypoint corresponds to a navigation sign displaying room locations in a University of Toronto building. However, much of this data is unsuitable as reference points—for instance, the frequent appearance of "UNIVERSITY OF TORONTO" provides little discriminative value. To address this, Model 1 is a neural network designed to filter these texts, retaining only useful data for Model 2, which prioritizes room numbers. Room numbers are ideal reference points due to their low repetition rate within

a building (typically appearing only on navigation signs or lecture room doors).

To define valid room labels, we first compiled a comprehensive dataset of room numbers by scraping the University of Toronto's room directory (available at: `https://lsm.utoronto.ca/webapp/f?p=210:1:::::::`). This dataset includes room numbers both with and without appended building codes, serving as a ground truth for filtering. We kept values with the building code appended as we noticed some buildings keep the two-letter codes appended within their internal signage.

Model 1 is a binary text classifier built on the `bert-base-uncased` architecture. The model receives individual OCR-detected text snippets and predicts whether they correspond to a valid room label (positive class) or non-room text (negative class). The input text is tokenized using the BERT tokenizer, with all sequences truncated or padded to a length of 16 tokens. The classifier consists of the BERT encoder as a feature extractor, followed by a feed-forward classification head: a 64-unit fully connected layer with ReLU activation, and a final sigmoid output neuron for binary prediction.

The training data was formed by assigning each of the room titles in our database as a positive sample and featuring common text patterns we observed across the University of Toronto campus in buildings ('UNIVERSITY OF TORONTO', 'RECYCLE PLEASE', 'PROFESSOR') as non-room examples. The dataset was randomly split into training, validation, and test sets with an approximate 30/30/40 ratio to maximize evaluation robustness given the relatively small dataset size. Model parameters were optimized using `AdamW` with a learning rate of $2e^{-5}$, binary cross-entropy loss, and a batch size of 8 over 5 epochs on an Intel i5 CPU.

During inference, the model outputs a probability between 0 and 1; values above 0.5 are classified as valid room labels. Accuracy and loss metrics were computed for validation and test sets, with the model achieving high classification accuracy and demonstrating strong generalization in filtering irrelevant OCR results.

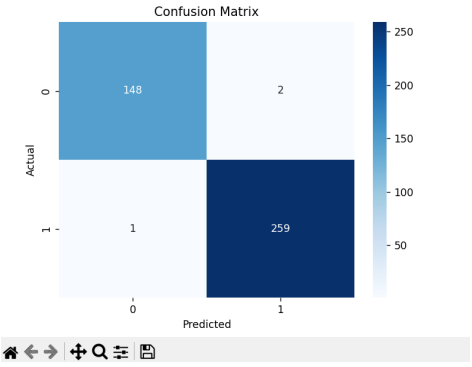
Given an input text sequence $x = \{x_1, ..., x_n\}$, the model computes the probability $P(y = 1|x)$ of being a valid room label as:

$$P(y = 1|x) = \sigma\left(\mathbf{W}_2 \text{ReLU}\left(\mathbf{W}_1 \mathbf{h}(x) + \mathbf{b}_1\right) + \mathbf{b}_2\right) \quad (1)$$

where:

- $\mathbf{h}(\mathbf{x}) \in R^{768}$ is the BERT pooled embedding
- $\mathbf{W_1} \in R^{64\times768}$, $\mathbf{b}_1 \in R^{64}$ are first-layer parameters
- $\mathbf{W_2} \in R^{1\times64}$, $\mathbf{b}_2 \in R$ are output-layer parameters
- σ is the sigmoid function: $\sigma(z) = \frac{1}{1+e^{-z}}$

On the held-out test set, Model1 achieved an accuracy of 0.993 with low loss, indicating strong generalization to unseen OCR text. The trained weights and tokenizer were integrated into our AR-based navigation pipeline, ensuring that each stored waypoint retains only location-relevant text for downstream processing in Model2. This gives us the following confusion matrix:



2.5 Model2: Inferring The Position of the user

Once we receive the filtered text snippets from Model1, they are used by Model2 to infer the user's current location within the building. We frame this as a classification problem, where each discrete position in the building, represented as a waypoint, corresponds to a class label. To perform this classification, we evaluated two machine learning techniques: k-Nearest Neighbors (KNN) and Random Forests.

2.5.1 Sentence Embeddings

To prepare the required data for Model2, we transform filtered text snippets from Model 1 into 384-dimensional dense vectors using the all-MiniLM-L6-v2 sentence transformer [1], a fine-tuned version based on the MiniLM architecture [8]. The embedding process captures semantic information from the text snippets, enabling Model2 to learn location-specific textual patterns.

The choice to use all-MiniLM-L6-v2 was mainly motivated by platform constraints. Since the system is

[1] `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`

intended to be deployed and run locally on mobile devices, it is necessary to have a model optimized for efficiency. In general, MiniLM models are well suited for resource-limited devices such as smartphones, and are ideal for real-time processing scenarios. In addition, the relatively low embedding dimensionality of 384 helps mitigate issues related to the curse of dimensionality.

2.5.2 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a non-parametric machine learning method where a query point is classified based on a majority vote among its k closest neighbors in the dataset. We implemented KNN using the `KNeighborsClassifier` class from `sklearn.metrics`, and experimented with multiple values of k. (see Section 3.3 for experimental results)

2.5.3 Random Forest

We also benchmarked the performance of a Random Forest model, which is an ensemble of decision trees where each tree is trained on a random subset of the training data and a random subset of features, in a process known as bagging. This configuration is known to be less prone to overfitting, which was an important consideration given the limited size of our dataset and the lack of publicly available alternatives. We implemented Random Forests using the `RandomForestClassifier` class from `sklearn.ensemble`, and experimented with different split criteria, number of trees, and max depth. (see Section 3.4 for experimental results)

2.6 Navigating the User to the Destination

The data structure we use to store waypoints is an undirected graph, where each node represents a waypoint and the edges between nodes represent navigable paths in the real world. To determine a route for the user, we apply a breadth-first earch (BFS) to compute the shortest path between the user's location and the destination waypoint. This path can then be displayed using augmented reality.

3 Experiments and Results

The following section provides a detailed evaluation of the models used in our pipeline. We use the sample input shown in Figure 2 repeatedly in the following subsections.

3.1 Model1 Embedding and Classification Speed

Model1 was run on `sample_cues`. The following results can be replicated by running `pipeline.py`:

```
sample_cues = [
    "ELECTRICAL ROOM",
    "8259",
    "Ra 8259",
    "65281",
    "ELECTRICAL ROO",
    "ECECTRICAN",
    "8258",
]
```

Figure 2: *Textual cues extracted using WallTrace's built-in OCR near an electrical room in Bahen*

```
Model1 embedding took 0.0026586055755615234
seconds for 7 inputs.

Model1 classification took 0.26624011993408203
seconds for 7 inputs.
```

Figure 3: *Timing results for Model1 embedding and classification methods.*

3.2 Model2 Embedding and Classification Speed

The valid sample cues returned by Model1 in the previous experiment were used to test the embedding performance of Model2. Similar to before, the following results can be replicated by running `pipeline.py`.

```
Model2 embedding took 0.010395050048828125
seconds for 4 inputs.

Model2 classification (knn) took
0.004393577575683594 seconds for 4 inputs.

Model2 classification (random_forest) took
0.017072439193725586 seconds for 4 inputs.
```

Figure 4: *Timing results for Model2 embedding and classification methods. Note that there are only four inputs, since Model1 filtered out three of them from `sample_cues`.*

3.3 KNN Experiments

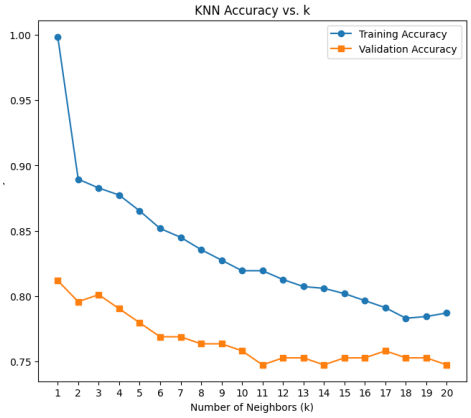


Figure 5: *KNN tested on values of k ranging from 1 to 20.*

3.4 Random Forest Experiments

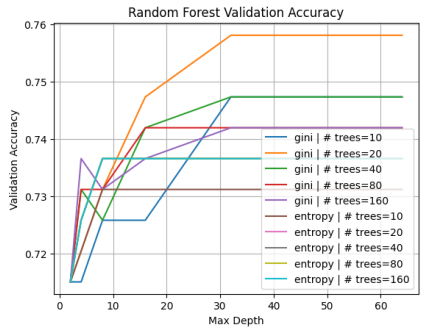


Figure 6: *Multiple Random Forests were tested, using* `gini` *and* `entropy` *as split criteria, and different values of max depth and* ***maximum*** *number of trees.*

4 Acknowledgement

While Model1 achieved a test accuracy of 0.993, the small size of our positive-class dataset (348 room labels) raises the possibility of overfitting, particularly given the strong separation between our positive and negative examples. Expanding the dataset with more diverse room label formats and harder negative samples would help mitigate this risk, especially as at different universities, room labels would be distinguished by other forms.

Another limitation lies in the reliance on OCR to extract text from the entire scene without spatial filtering. This approach increases the chance of irrelevant or noisy text entering the pipeline before classification. A potential improvement is to introduce a convolutional neural network (CNN)-based object detection stage to generate bounding boxes around signs prior to OCR, thereby reducing background noise and improving precision.

Moreover, while our binary classification scheme is effective for the room/non-room distinction, it does not account for partial matches, OCR misspellings, or numeric sequences that are not room numbers but resemble them. Future iterations could incorporate fuzzy string matching or embedding-based similarity measures to make the system more robust to imperfect OCR output.

Finally, While both the KNN and Random Forest models demonstrated promising validation accuracy, the unpredictability of text cues extracted via OCR — coupled with the limited size of our training dataset — suggests that real-world performance may fall short of these experimental results. Nonetheless, with improvements to OCR quality, the inclusion of a larger and more diverse training set, and the refinements to Model1 discussed earlier, we believe that real-world performance could reach a level sufficient for practical deployment.

References

[1] Anita Araneda, Stephen E. Fienberg, and Alvaro Soto. A statistical approach to simultaneous mapping and localization for mobile robots. *The Annals of Applied Statistics*, 1(1), June 2007.

[2] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006.

[3] Apple Inc. Apple maps, 2024. Accessed: 2024-01-01.

[4] Yuwen Liao, Xinhang Xu, Ruofei Bai, Yizhuo Yang, Muqing Cao, Shenghai Yuan, and Lihua Xie. Following is all you need: Robot crowd navigation using people as planners, 2025.

[5] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks, 2016.

[6] Jose Rivera-Rubio, Ioannis Alexiou, and Anil A. Bharath. Appearance-based indoor localization: A comparison of patch descriptor performance. *Pattern Recognition Letters*, 66:109–117, November 2015.

[7] Anja Sheppard and Katherine A. Skinner. Automatic data processing for space robotics machine learning, 2023.

[8] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

Appendix: User Interface Screenshots

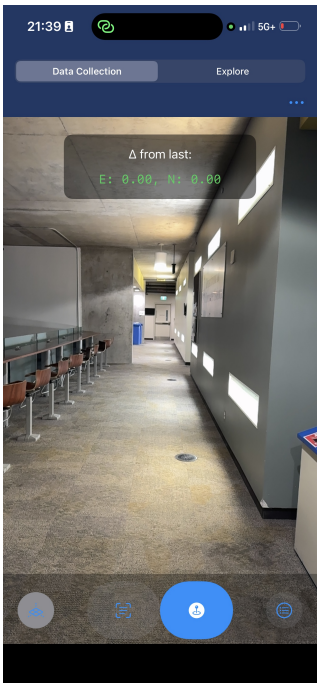


Figure 7: Main window of the WallTrace software.

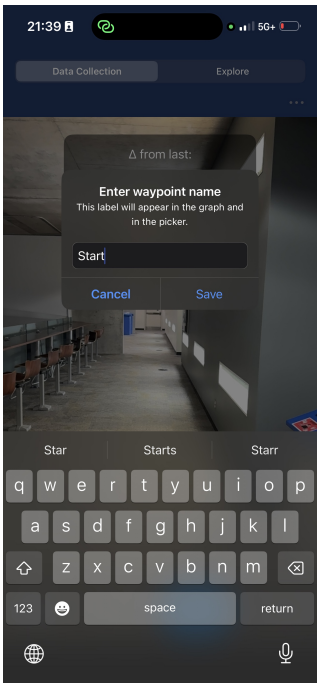


Figure 8: Saving a waypoint in the WallTrace software.

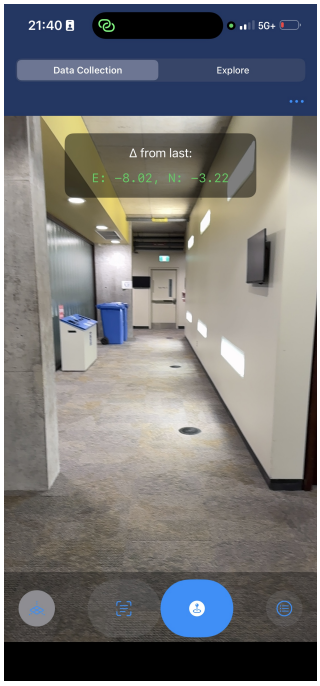


Figure 9: Moving to another position in the WallTrace software.

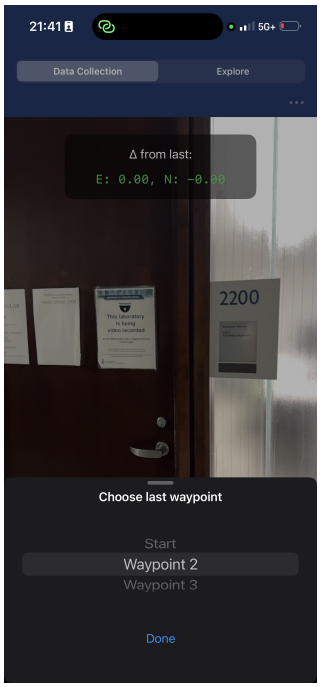


Figure 10: Linking the Waypoints by selecting a custom last waypoint in WallTrace software.

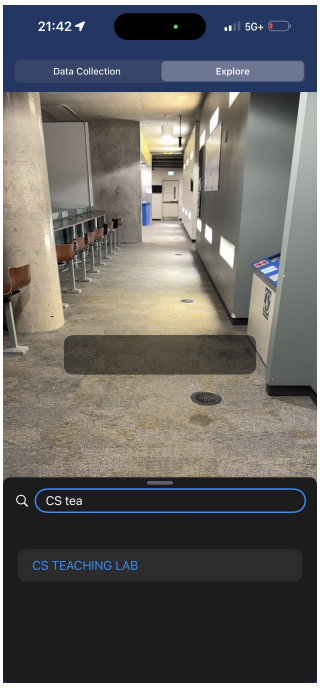


Figure 11: User searching for a location in WallTrace software.

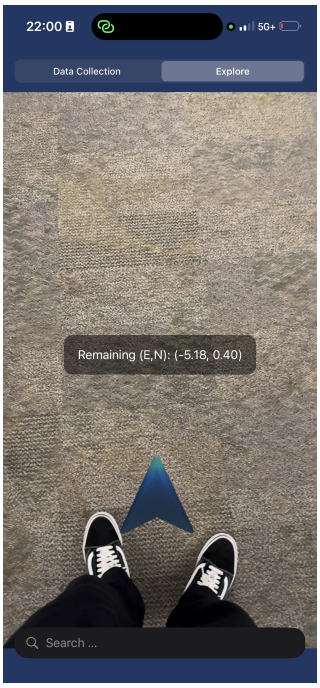


Figure 12: Navigation window showing route guidance.