Navi Final Documentation

Table of Contents

1. Problem Description

Ever felt lost looking for a store item in complex, large retailer stores or a shopping mall?

First time or early visitors of such stores feel confused and overwhelmed with the multitude of locations and item selections. They face difficulty navigating such environments and lose much time figuring out how to reach their destination.

You can watch a demo of using Navi at IKEA here.

2. Brainstorming leading to concept generation

Brainstorms from the Winter Quarter construction project with DAQRI are here.

Our idea for Navi was rooted in our Winter quarter navigation idea[1] designed for construction workers. The brainstorm preceding Navi is drawn from Spring Quarter Week 2 progress doc:

[1] At the start of a construction, an architect creates a 3D model of the building. Then, a surveyor manually layouts points from the 3D model onto the construction site ground. These latitude/longitude points marked on the ground are called "control points" and are used to reference the location of other structures (walls, beams) in the building.

However, whenever the control points are lost / washed from the ground, the construction site has to pay for a re-survey of points. This is a time and cost loss.

After 1 survey, store the latitude/longitude coordinates of the control points. Then have the helmet guide our users to locate and navigate to the points in the real world. This is done through directional arrows and a mini-map of the construction site.

- The control point storage idea didn't resonate nor excite the team. Lucas re-presented one of our older ideas around indoor navigation. During the meeting we briefly brainstormed general use cases of indoor navigation like locating places in shopping centers, airports, and museums. This idea really excited the team, but we hadn't completely fleshed it out so we agreed to hold off abandoning the construction direction.

- On Thursday, we decided to abandon the construction direction altogether and stick with the indoor navigation general use case, particularly shopping. Alan, Lucas, and Catherine met with Jay to get his input on the indoor navigation. Some things Jay had us consider were:
    - The price of the Tango and the consumer market. Will shoppers have to pay for the Tango? At the end of the day, given our time constraints, we agreed that it's fine to disregard the business model and just build our app on the Tango. However, it was worth creating a compelling use case of the app.
    - Brainstormed use cases:
        - One time visits to large complex indoor spaces
            - Cruises
                - Lucas suggested that, since cruises are pricey already, the host can give Tangos to each rider
            - Travelling to foreign places
            - Museums
        - Frequent visits to indoor spaces
            - Shopping malls
                - I want to find this store or eatery
                - Filter stores with pants that are 50% off
                - I'm going to prom. Find me the dress, shoes, makeup etc.
                - Leverage store data and use AR. Maybe show someone wearing the prom dress via AR
            - Grocery shopping
                - I want to cook this meal. Show me where all the ingredients are.

  Outside the meeting, another use case we brainstormed was finding a book at a library. Someone could check out a Tango from the front desk.

- Ultimately, we decided to stick with the shopping mall case because several malls surround Stanford and it was the quickest to user test.

3. Design Criteria

N/A given situation with DAQRI

4. User Testing Documentation

You can find photos, videos, and notes from our user testing here:

- Choosing the Right Customer

    - Interview notes
- Launch Day
    - Photos and video

5. Design Rationale and Analysis

When we switched from construction workers, we had to consider how we would redesign our application and cater to a consumer audience, particularly for a shopping settings. Below is a explanation of design choices as we explored this new audience.

First, we realized that we two types of customers: environment (store) owners and visitors (shoppers). We designed our interface and functionality to cater to the use cases per customer type.

5.1 Store Owner View

Originally, we were not going to support the store owner interface because the biggest value (where the AR can be displayed) is only in the shopper view. However, we felt that the store owner use case necessitated it's own interface. How else would an indoor environment be set up?

Below our features we decided to support:

- Instructions in setting up and managing store
    - Setting up an indoor environment can be a very complex task. We decided it was worth automating this process and guiding the store owner user as much as possible. We do this in a simple step-by-step manner.
- Pre-scanned ADF and maps
    - For the reasons above, we agreed that we should have ADFs already prescanned and a 2D map of the floor plan included in the app's storage. This would avoid extra manual work from the store owner because scanning an ADF is currently performed in a separate application. We would have to rebuild the exact functionality in our app. Moreover, for the user to provide a 2D map, the 2D map might not be of the right quality. Thus, for now, we have chosen to handle these tasks for the store owner.

5.2 Visitor View

- At the core, we wanted to allow the user to navigate with a camera view (AR digital route) and a 2D map and list view to select store items.

- Design changes to the visitor view were based on the feedback that we received from lessons from the launch day. These included improving the 2D map to include names of store items and showing AR symbols at category locations (e.g. location with on-sale items show stars). We added a way to easily switch between the camera view and list view/2D map so that the user can quickly choose new items to navigate to.

- Another big piece of user feedback we received was around showing instructions. Our test users needed much guidance from us when using the app because there are several steps. Adding instructions into the app alleviated this problem.

5.2.5 Augmented Reality Component

The purpose of the Augmented Reality component of our application is to guide the user to their chosen destination, and also to flag points of interest along the way (ex. sales). To that end there were a number of decisions that had to be made regarding the the objects to be displayed, and how to display them.

Models

- Arrows
    - Arrows were chosen as the main object to guide the user, because they are simple and intuitive.
- Path
    - In an early iteration, the arrows were linked by visible straight lines to form a path. However, we chose to do away with this feature for two reasons: 1) it made the view cluttered, and 2) the inability for augmented reality objects to be occluded made this unintuitive and unattractive.
- Destination
    - We modeled the icon to put at the destination after familiar icons from Google Maps and other map applications to leverage user's previous experiences.
- Filter Icons
    - These are models to place in the world at points of interest according to the user's filters. Currently, this consists of a rotating star at items on sale (in our demo in IKEA), and in sections that have CS210 projects (in the software fair.)

Colors

- Arrows and Destination
    - The arrows and destination are a pink/fuschia, both to match the general color scheme of our app and also to make them obvious in the augmented reality. User feedback showed us that it was easy for users to overlook augmented reality arguments if they were not bright enough and did not pop out at them enough. This resulted in a lot of confusion, as users did not know what was meant to be navigating them.
- Filter Icons
    - The star is a deep gold because it's a color users are familiar with associating with stars and prizes and excitement, and because it does not resemble the color of the navigational models (so as to avoid confusion).

Display

- Arrows and Destination
    - Issue: The biggest issue we ran into is the fact that augmented reality objects cannot be occluded. If we display a path that involves points on the opposite side of a wall, those points would still be visible (as though they were in front of the wall or the tango had X-ray vision.) This caused users a lot of confusion.
    - Solution: We decided to only display icons that it made sense for the user to be able to see. Besides, points along the path indicate places where the user must turn. So we decided to only display the point at which the user was at, and the next point that they needed to reach (each with arrows pointing to the point) at a time. We also decided the display the arrows closer to the ground so they would not obstruct view of the world.
- Filter Icons
    - We display the point of interest models a little above eye level of the user, to get their attention and also keep them out of the way of the view of the world.

- We rotate the models slowly so that it is easy to tell what the models are, and so they are eye grabbing.

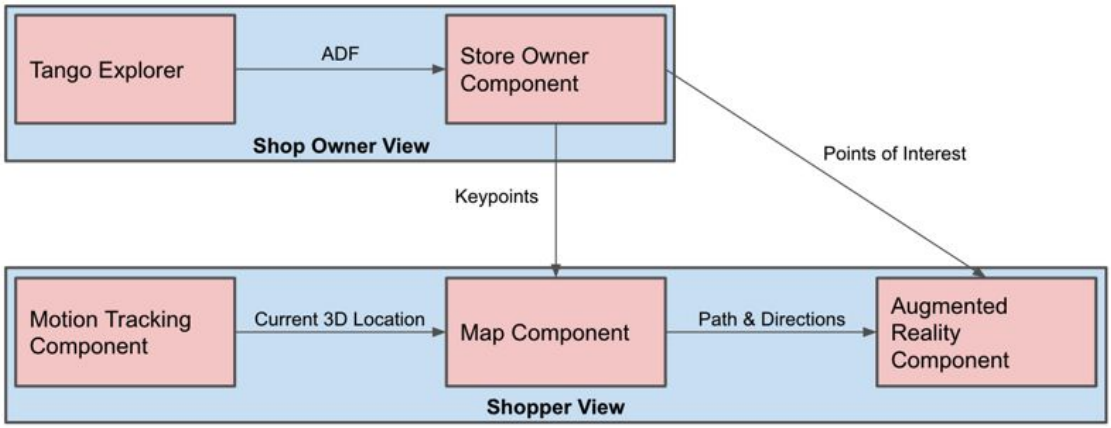
6. Future Extensions

Our sequence of extensions that we would like to add to Navi in the future:

1. Increased Capabilities of the 2D Map
   - Being able to pan and zoom on the 2D Map could be helpful for users, as well as being able to choose a destination by touching a place on the map rather than choosing a named destination from a drop down menu beside it.
2. Explore More Ways to Facilitate Navigation
   - Currently we use Augmented Reality to navigate the user via arrows at checkpoints along the path. In the future, we would like to explore more methods. For example, perhaps having an arrow or other object hover in front of the user and the user need only follow the object, instead of locating checkpoints.
3. Social Component
   - With more time, it might be interesting to add a social component to the app that could include: storing and sharing previously taken paths, following the published paths of other users, sharing your location with a friend such that they might navigate to you, and more.

Final Software Design (+ flowchart of components)

5.2.1 Overview: Modify here



5.2.2 Technical Terms

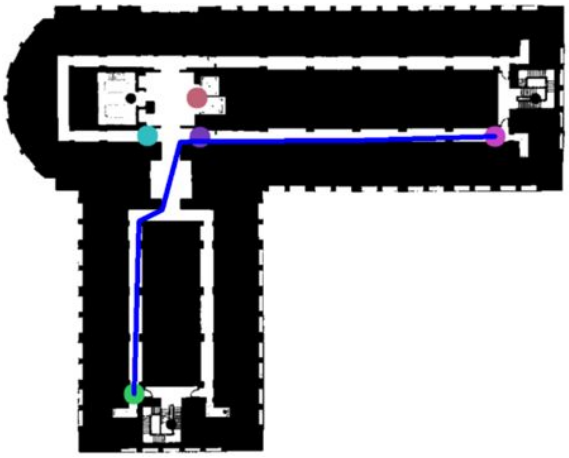
- ADF: Area Description File.
- Start-of-service coordinate frame: The coordinate system defined by the starting point of current service.
- Area-description coordinate frame: The coordinate system defined by the ADF origin.
- Device coordinate frame: The coordinate system defined by the current location of the device.
- Pose: The position and orientation of the user's device.
- DOF: Degrees of freedom.

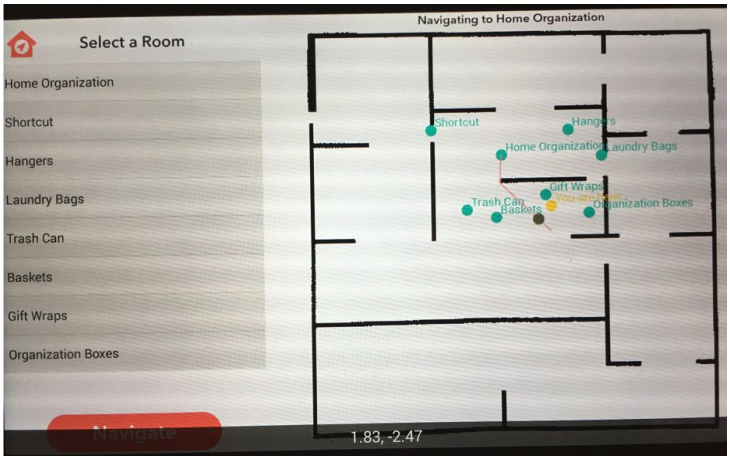
5.2.3 Motion Tracking Component

- IMU (Inertial Measurement Unit): Calculate relative coordinates (device w.r.t. start-of-service) using sensors.
- VIO (Visual-Inertial Odometry): Remember the key visual features of a physical space using Area Learning.
- Localization: Orient and position itself within a previously learned area (start-of-service coordinate w.r.t area-description coordinate) using the ADF.
- Drift correction: Improve the accuracy of the trajectory created by IMU.
- We use Project Tango Explorer to scan the building and save the information into an ADF file.
- The file is then loaded by our program and the absolute coordinates are retrieved (in the Area-description coordinate frame).

5.2.4 Map Component

- Preprocess the map (floor plan) taken from a camera to a format usable by our algorithm.
    - Algorithm: Given a 2D floor plan, process the image with noise reduction and binary thresholding (Otsu's method). The output is a binary image.
- Calibrate the map by finding the mapping between the 2D floor plan and the 3D world.
    - 2D: 3 DOF (x, y, theta); 3D: 6 DOF (x, y, z, pitch, yaw, roll).
    - Algorithm: Assume that the mapping follows an affine transform. The mapping is calculated using linear least squares.
- A path finding algorithm that calculates the optimal path and avoids obstacles.
    - This problem can be modeled as an Any-Angle Path Planning problem.
    - Algorithm: Using a graph size of around 500x500, the calculation can be done within 500 milliseconds using Lazy Theta*.
- Display the map, current location, all the points of interest, and the optimal path.
- Recalculate the path if the user is too far away from the existing path.
    - Note that the user can almost never follow the actual path, so it is necessary to project the current location to the path (lots of math).
    - At every time step, calculate the distance from the current location to the corresponding point (perpendicular projection) on the path.
    - If the distance exceeds a certain threshold, recalculate the path.
    - A prompt saying “recalculating the path” will be shown in the screen.





7. Quick Start

7.1 Prerequisites

- Device: Project Tango
- Required apps and libraries on Tango: Explorer, OpenCV Manager, ADF Inspector
- Software Repo: https://github.com/cs210/navi-code
- IDE: Android Studio
- Have Git installed on your machine (a github account is helpful too)

7.2 Setup

1.) Git pull the repo
2.) Open the project in Android studio (daqri-code/spring2016/Navi/navi_app)
3.) Plug in tango to laptop /computer
    a.) Enable USB debugging on the Tango device
4.) Click the green Run arrow in Android Studio
5.) Choose either “Visitor” or “Store Owner” depending on who you identify with
6.) NOTE: To successfully use the Navi app on the Tango device, you must to be in the physical location of the indoor environment displayed or selected

If you’d like to use an indoor environment not included in Navi’s package, you may add your own:

- Scan an ADF of the building using Explorer.
- Run the software on Tango using Android Studio. The user has to be in the physical location of one of the ADFs.

In the first page, there should be two options: Owner and Shopper.

- First click “Owner” and set up all the shop owner information (sales & deals, keypoints for calibration).
- After all the shop owner information is saved, click “Shopper” and start the navigation. The user is able to select the destination from the left sidebar, and switch between the map view and the AR view.