AR-Based Navigation in Crowded Public Spaces

January 2019

Michael Nissen 428074
University of Southern Denmark

Paul Nylund 84930028

Thank you

Our projected is indebted to the knowledgeable friends and experts that we were so lucky to speak with. We are grateful for their invested time and continued interest.

Alexis Rappaport (Architect, Clive Wilkinson Architects), for supporting us through several long discussions

Asa Kremmer (Architect, HASSELL Studio), for testing our paper prototype and participating in a lengthy interview

Cindy Soo (Strategic Director, IKEA), for showing such strong interest in our project and referring us to Curtis Barrett

Curtis Barrett (Principal Technical Program Manager, Google), for taking the time to offer us plenty of invaluable advice

Elizabeth Churchill (Director of UX, Google), for showing strong interest in our project

Marie Ehrndal (Design Technologist, Topp), for helping us iron out a bunch of bugs in Unity3D

Michael Knopoff (Principal, Montalba Architects), for being so supportive and giving us a tour of his firm

Moina Medbøe Tamuly (Student, NTNU), for his interest in using the Lace technology as part of a public exhibition in Oslo

Morten Nissen (Apprentice, Branel), for helping out with load balancing on Lace's virtual servers

Nikolas Gundersen (Creator, Norwegian Rain), for offering his facilities for initial prototype testing

Spencer Cappiello (Co-Founder, Where Art Thou), for testing out our paper prototype, as well as endless discussions about augmented reality

---

We would like to make a special shout-out to our supervisor, Danielle Wilde (Associate Professor, SDU), for being so kind and supportive, keeping us on track whenever we inevitably drifted astray.

Table of Contents

1. Abstract

The lack of dynamism of signage displays in crowded public spaces can lead to confusion amongst pedestrians about how to safely and efficiently navigate these spaces. In this project, we focused on serving pedestrian's individual requirements about how to best navigate towards a destination through such crowded spaces. We also investigated methods of implementing real-time crowd-analysis to ensure that our solution could respond dynamically to users' behavior. Our response is an augmented reality-based navigation app that dynamically updates its route according to the calculated density of all pedestrians in a space. We spoke with a range of experts across different fields, from consumer technology to architecture, to learn about how we could implement a solution with minimal upfront cost and greater market-value. We conducted user-studies, experiments, and a survey —comparing the resulting data to research in the field of crowd control and analysis. Our response, which we call 'Lace', can help alleviate situations in which crowd control is needed without prior warning. It can consequently act as an early warning system for dangerous overcrowding. Lace could be implemented in public transportation terminals, wherein travel time is a significant concern.

All the code can be found in the following GitHub repositories:

https://github.com/qruzz/lace-web

https://github.com/paultnylund/lace-server,

https://github.com/paultnylund/lace-mobile

2. Intro— duction

Variation in signage can be confusing for traveling pedestrians, potentially causing diversions at the expense of time or energy. The lack of real-time signage augmentation is a safety issue pertaining to crisis-response and other situations in which individuals in crowds would need to be advised.

As the world becomes increasingly globalized, differing signage patterns can be difficult to translate for travelers. Real-time signage augmentations can be expensive, requiring human or structural resources to direct large groups of people. Personalized signage augmentations are rare outside of private tours.

The opportunity herein lies in the potential to create an immersive solution which assists individuals in navigating public spaces. This could amount to more effective crowd-control in said space, assuming that several of the aforementioned individuals are provided for. We aim to design a scalable system that can be integrated into existing public spaces with little to no added infrastructure.

Our response is called 'Lace' because it is designed with the ever-changing fabric of interactions between pedestrians, obstructions, and spaces in mind. Lace provides dynamic and responsive pedestrian navigation based on individual user input and collective foot traffic. Lace responds to the collective movement of pedestrians through a public space. Through an augmented reality interface, Lace responsively advises each active participant about how to navigate through that space in order to reach a specified or changing destination. In ensuring that Lace is able to inform users in real-time, it

Figure 1: Paul Nylund (left) and Michael Nissen (right)

accommodates unforeseen circumstances, such as users spontaneously navigating to points of interest.

Lace can be built upon the widespread infrastructure of existing security cameras. Current consumer-focused crowd-control solutions add physical infrastructure or require human labor. Adding such amenities is an expensive and thus difficult-to-scale practice. Hence, Lace does not consider this practice.

The team that worked on this project is comprised of Michael Nissen and Paul Nylund. The two of us undertook this project together, as we have worked together on many similar projects before. Because of this, we are familiar with each others process. We are both passionate about questioning the role of technology in our everyday lives, continually striving to build ubiquitous solutions for the betterment of society. While we share a similar perspective and common knowledge, we specialize in different areas making decision-making a simpler process. Having different areas of expertise helps us distribute the work between us in an efficient way, allowing us to achieve more in a shorter timespan.

Michael Nissen (http://www.michaelnissen.me) studied Computer Science at the University of Aarhus (AU) before studying Interaction Design Engineering at the University of Southern Denmark (SDU). During a semester abroad in Paris, France he followed a masters in Embedded Systems at École d'Ingénieurs. Living in Paris led him to take an internship at Snips.ai during his sixth semester. He has worked on numerous development projects as a self-employed software engineer for many years earning him expertise in safety-critical engineering practices and full-stack development.

Paul Nylund (https://www.paulnylund.com) studied Industrial Design Engineering at The Hague University (THUAS) in the Netherlands, before moving to Denmark to study Interaction Design Engineering at the University of Southern Denmark (SDU). He spent his summers working as a designer or front-end developer for software companies in New York,

New York and Oslo, Norway. During his fifth semester at SDU, Nylund studied Industrial Design at the Norwegian University of Science and Technology (NTNU), where he worked on projects involving machine learning. During his sixth semester, Nylund was employed as an Interaction Design Intern at 'ustwo' in Malmö, Sweden.

3. Background

As urban areas become bigger and denser, crowd-management becomes increasingly important [12]. And just as crowd behavior is unpredictable, so are crowd-related accidents [12]. This makes studying accidents decidedly difficult in the present day while strengthening the argument for persistent crowd-analysis solutions, such as Lace. Computer vision can be especially effective for analyzing especially dense crowds [12]. Feliciani describes the "intrinsic risk" of a crowd as the product of the congestion level and density in a space. Hence, making accurate density measurements could be a means to prevent dangerous overcrowding in the future. Through guiding pedestrians around crowded areas, our response, dubbed 'Lace', could offer this solution.

How pedestrians move is not at all random. In fact, pedestrian movements are often influenced by their personal motivations [18]. These motivations can be treated as forces, acting towards a goal. Providing guidance to pedestrians effectively may turn out to be quite difficult, as most people interpret environments through their own frame of reference. Specifically, it has been determined that all pedestrians have a preferred velocity —their preferred walking pace according to an environment (averaging 1.34 m/s) [22]. This preferred velocity can be interpreted as a force, exerting itself on a user's physical behavior. A user's actual velocity will change exponentially over time towards that of their preferred velocity [22].

Some simulation models are based on physical theory, treating crowds as a whole as if they were a liquid or gas. Some of these simulations have proved to be quite successful at predicting pedestrian movement [18]. Users' individual preferred velocity force, as well as external social forces and obstacle forces, act on users and change their behavior over time. For example, pedestrians will make small changes in their movements to avoid other pedestrians and obstructions around them [19]. Small changes made across crowds cumulatively affect the rate of individual approaches to each respectively preferred velocity. In an obstacle-free environment, pedestrians approach their preferred walking velocity exponentially over time [22]. Determining velocities is helpful in programming simulation models. Additionally, the ability to measure individual velocities could introduce opportunities for optimization; predictive algorithms could be run at varying rates for different individuals, depending on their walking velocity [8]. In our case, however, measuring pedestrians' velocities with the server was determined to be negligible. Because we were detecting pedestrians' positions with close to real-time frequency, predicting their individual courses was unnecessary. Nonetheless, measuring velocity on individual users' devices could still be useful when calculating the time remaining in their journeys. Pedestrians in the real world are generally quite coordinated, planning much further ahead than their simulated counterparts [8].

Pedestrians have been shown to imitate others and/or change their own fundamental behavior in response to crowd dynamics [15]. Fridman offers Social Comparison Theory as further explanation for this behavior; humans will typically mimic others if they are failing to objectively evaluate their own state. Implementing Social Comparison Theory in crowd simulations has been shown to quicken the formation of lanes. While pedestrians have a natural tendency to form separate lanes when they are walking towards each other [18], there still exists a significant amount of friction

Photo by mauro mora on Unsplash

between pedestrians —yet another potential force impacting an individual's walking velocity. These lanes could form as a result of pedestrians' personal reluctance to accommodate other pedestrians' conflicting paths [19]. Johansson describes this phenomenon as dynamic stripe formation [22].

It is possible to calculate the potential movement of single pedestrians using a social force model. A social force model is calculated based on a series of forces, including a preferred force, an interaction force, and obstacle force. The preferred force describes the willingness of a pedestrian to reach their preferred walking velocity. An interaction force describes the willingness of a pedestrian to avoid surrounding pedestrians. An obstacle force is a force which causes pedestrians to keep their distance from inaccessible space. Perhaps another force could be added, which describes the influence that signage and commercial stimuli may exert on certain users. A random force, or fluctuation variable, has previously been introduced as a means to resolve the difficulty of predicting spontaneous decisions [18].

In the context of Lace is that if a user feels uncertain about their own whereabouts, there is a good chance they might look to others —especially persons whom they perceive as being similar to themselves— as a guide for where they should walk and/or how they should behave in a crowd.

Increasing the amount of circulation in smaller spaces can help such spaces feel wider and thereby more accommodating [3]. Helbing describes the emergence of "unstable roundabouts" in pedestrian traffic [16], indicating that circular patterns emerge as a collective subconscious effort to improve the average efficiency of foot traffic. By fostering increased circulation in crowded spaces, one could create the illusion of greater personal space for all pedestrians involved. By efficiently guiding pedestrians through crowds, Lace improves users' relationships with a public space.

Pedestrians will choose routes that allow them to walk in long, straight lines, avoiding crowds and obstacles at any cost [16][8][5]. Elderly pedestrians or pedestrians carrying heavy items will generally choose to walk longer routes to avoid taking the stairs [27]. This could be due to the fact that pedestrians will prefer to make the smallest amount of decisions en route

to their destination [27]. Millonig introduces five different navigation strategies exhibited by pedestrians [27]:

- random (lack of information)
- taxonomic (following visual cues towards a goal)
- praxic (forewarning with distance or time-based measurements)
- route (using landmarks)
- locale (by mental reconstruction of an environment)

The use of landmarks, or 'salient objects', as visual cues can be an extremely effective wayfinding strategy [27][5]. While existing signs can serve as effective landmarks [27], signs vary quite a lot from destination to destination, making their usage decidedly less scalable. Lace was predominantly based on taxonomic navigation, in that its primary purpose is to display a persistent guiding path towards a user's chosen destination. The path itself is made to support praxic navigation by textually conveying the remaining distance to the destination.

4. Method

We started our journey in Copenhagen airport, waiting to board our flight to Los Angeles, California. We would spend three weeks traveling from Los Angeles to the San Francisco Bay Area, meeting with various experts within architecture as well as consumer technologies along the way. In these three weeks, we gained a lot of insights pertaining to our initial project proposal. These insights would help reframe our project going forward. Our user-centered work on the project would not truly kick off until we touched down in Melbourne, Australia later that month. Of course, that followed a layover in tropical honeymoon destination, Nadi, Fiji. Who says students can't enjoy life once in a while? We stayed in Melbourne for close to three months, conducting user research, designing prototypes, developing, and writing.

To define the overall project structure, a GANTT chart —or categorized timeline was created [Appendix]. We first determined different preliminary tasks of the project. These tasks were then grouped into larger phases, including the following project-specific subsections. For some time, we were planning on using the Agile framework for organizing our project tasks. Agile is a method of planning engineering projects in a structured fashion while allowing tasks to be periodically evaluated. We decided against using Agile for during the initial design phase when working with an unstructured process could help us work openly with several

concepts when it mattered the most. Our interest in using Agile eventually dwindled at the beginning of the development phase, as we became less certain of who would be benefitting from Agile's contributions to our parallel workflows.

Preliminary Investigation and Literature Review

This phase was comprised of interviews with experts in the fields of architecture and consumer technology. We had also set out to establish a user group, or groups, in Melbourne, Australia with whom to conduct research. Additionally, time was allotted for researching and making sense of academic papers and/or articles that could provide supporting arguments for this project.

User Studies

We were determined to adhere to a participatory design process throughout this project. Participatory design is, in essence, is an invitation for users themselves to contribute to the design process, validating or negating any learnings or decisions along the way [40]. The involvement of participatory design in our process was dependent on the readiness of the prototype and consequently our ability to test it.

Prototyping

The team considered user feedback in the design and ideation of prototypes. One or more low-fidelity prototypes would have to be created quickly with limited resources, in an effort to retain focus on feedback and iteration until sufficient data had been found. The role of low-fidelity prototypes was to gain insights from users with a minimum investment of time, energy, and funds. The high-fidelity prototype would be synthesized towards the end of the process, based on the total collected insights from the low-fidelity prototypes.

Figure 2: Waiting room at the Google Hardware Lab

5. Expert Interviews

In an effort to further clarify our concept, we conducted several expert interviews. These experts were selected from a wide set of what we determined were relevant fields to our project. They ranged from specialism in architecture to system design. We were especially interested in learning from architects, as their work often considers the movement of people through space.

The remainder of this section will detail each of the expert interviews, outlining their perspective on and position on our project proposal.

5.1. Principal Technical Program Manager, Google

Curtis Barrett is the Principal Technical Program Manager of Consumer Hardware Chipsets at Google Inc. He was previously Senior Technical Program Manager of Google Glass and Program Director at Intel, where he focused on augmented reality wearables. Given his history of managing such large and consequential projects, we felt that Barrett could offer significant insight on how we could seek to develop our response to be both scalable and future-proof.

In September 2018, we met with Barrett at Google headquarters in Mountain View, California. We presented the project definition at the time, as well as our hunches about how certain technologies might be leveraged. We hoped he would either reject or acknowledge some of these hunches, thereby helping us set further limitations to the project scope.

Coming into our meeting with Barrett, our proposed idea featured real-time anticipation of users' movements in a space. Barrett was pessimistic about this idea, assuring us that accurately predicting a single person's movement —given all the variables that arise in such a scenario— would be extremely difficult to accomplish in the timespan of the project. This could be especially true for predicting the movements of dozens of users in low-latency real-time.

Barrett urged caution about the amount of processing power required to track crowds of unique individuals. The potential expenses associated either with running servers or ensuring that users experience a minimal delay when interacting with our solution would have to be minimal in order for it to be considered as a feasible option; lowering the cost of our solution could consequently lead to wider adoption. Barrett explained that keeping these time and energy costs low could lie beyond the scope of the project's 3-month timeframe. He suggested instead that we focus on proving that we could accurately predict the movements of people and verify said prediction through a highly controlled experiment.

Barrett strongly favored the use of cameras for collecting data about user behavior. He referenced Nest, claiming that, "people are putting cameras all over their houses with 1080p imaging capabilities, so the ability to take out some reasonable space and capture really high-end video is completely feasible from an economic standpoint." This would function as an alternative to leveraging mobile devices, which could severely limit user participation, or building unique hardware, which is likely too expensive to produce and maintain. This also runs counter to our adherence to avoiding requiring additional physical infrastructure.

Following our meeting with Barrett, we were led through the dimly lit hallways of Google's secretive hardware building, past the cafeteria, and out into the parking lot. After shaking hands and parting ways, we walked back to our rental car in a quiet perplexity. The potential technological difficulty of implementing our initial idea loomed over us, as we sat in the car for several minutes in the Mountain View parking lot. The silence was interrupted only by the chewing on crunchy Japanese snacks we had purchased at a Japanese supermarket in Los Angeles the previous day. We decided to lend some space to the new insights that we had gathered, pausing the conversation until the following day.

That next afternoon, we started to narrow down the core technical requirements of our prototype, while ensuring that it was fully feasible to build by December the same year. We decided to scrap pedestrian movement anticipation in favor of a purely reactionary solution, statically capturing pedestrians' positions in real-time. Anticipating the walking patterns of any number of pedestrians would likely distract too much from design considerations. Programming the ability to anticipate the movement of even a single pedestrian would likely have occupied a significant portion of our time and energy.

The common closed-circuit television camera (CCTV), or security camera, in public spaces could provide an existing infrastructure upon which to build our prototype. This would limit Lace's infrastructure to rely solely on that which is often already existing in so many public spaces in the developed world. Given mobile phones' massive popularity, providing the service via a mobile app was determined to be the most affordable point of entry for on-boarding users. It would also be cheaper to prototype, as we could avoid acquiring expensive dedicated augmented reality headsets.

5.2. Principal, Montalba Architects

Michael Knopoff is the Principal at Montalba Architects in Santa Monica, California. He has been involved in several consequential public projects, including a Los Angeles Metro station, the Canyon Lodge in Mammoth, California, the Westfield shopping center in Palm Desert, and the Tom Bradley International Terminal at the Los Angeles Airport (LAX).

We discussed several projects that Knopoff had been working on. One of these projects comprised of the redevelopment of an older industrial building into a new mixed-use space. After donning hard-hats, he brought us to the active construction site just across from the Montalba office at Bergamot Station. While perhaps less fruitful to the distillation of our project, Knopoff spoke about the challenges associated with his project, especially how Montalba had designed the space while considering the flow of surrounding Metro traffic. The potential foot-traffic convinced them to reduce the usable office space and increase the coverage of commercial storefronts.

With regards to the Tom Bradley International Terminal, Knopoff discussed the concepts of circulation of pedestrians in public spaces. In an effort to increase the efficiency of pedestrian flow throughout the terminal, Knopoff and his colleagues planned a layout with several "roundabouts" in the form of commercial or seating areas. These "roundabouts" would simultaneously guide pedestrians through the space while providing those pedestrians with enough flexibility to gather at the peripheral commercial areas.

Knopoff expressed an especially negative opinion towards the large electronic displays installed throughout the interior of Tom Bradley International Terminal by another firm. He suggested that the displays obstructed

the architectural dignity of the space by interrupting the intended peacefulness he preferred would greet arriving passengers. This reaffirmed our adamancy against the addition of minimal physical infrastructure.

Figure 3: Michael Knopoff gives us a tour of the construction site at Bergamot Station

5.2. Landscape Architect, HASSELL Studio

Asa Kremmer works as an architect at HASSELL Studio, as well as a tutor in architecture and landscaping at the Royal Melbourne Institute of Technology's (RMIT) Faculty of Architecture and Urban Design, in Melbourne. Kremmer specializes in topics within urban planning and landscape design, as they pertain to the planning of public and private spaces. We wanted to speak to Kremmer because of his obsessive attention to detail in how spaces are orchestrated. We were especially interested in hearing his opinion on how a digital pedestrian wayfinding might integrate with such spaces. Our interview with Kremmer was conducted as a follow-up to having him test paper prototypes later in our design process [Section 10.2.2].

Kremmer expressed interest in the potential of tracking pedestrian movement throughout spaces, stating that having such information is potentially powerful to designers such as himself. Automating the creation of heat and/or flow maps of foot-traffic through spaces could save firms such as HASSELL a lot of manual effort. Such maps could support designers by indicating where pedestrians typically collect. One could then "rearrange public benches," as well as anything else in a space to adapt to users' wants or needs. Mapping pedestrian traffic, including where and when people collect, can help validate an architect's decision when presenting to clients. Integrating real-time mapping as a regular component of architects' toolsets could have significant implications —specifically in product cycle iterations and client communications.

Figure 4: Kremmer (left) and Nylund (right) conducting an experiment with the paper prototype

Kremmer also mentioned the curation of paths between public spaces, particularly with respect to the way in which the city of Melbourne shows interest in linking its outlet shops, food halls, and other points of interest. He pondered the potential of "controlling" pedestrians' journeys throughout the city. In particular, Kremmer would want to be informed of the whereabouts of the best taco truck en route to his chosen destination. We think the idea of controlling pedestrians' journeys is too harsh and would rather talk about guiding them.

6. Exploration

6.1. Venues

We visited numerous venues in an effort to identify which could be used to conduct controlled experiments. The venues needed a sufficient vantage point in which we would be able to attach a camera for our project. The venues visited include large shopping malls, transportations hubs, and university buildings. The main point of this exercise was to evaluate the suitability for doing user research, in each of the venues.

We were looking for environments that could afford a multitude of path choices. Some considerations were whether or not the room was too crowded to conduct controlled experiments, or if pedestrians had a sufficient number of passages available to them.

When working with computer vision, a good placement of the camera will make development simpler when trying to get accurate detections. Good camera placement considers both the angle of capture, as well as the lighting conditions. For the camera to accurately distinguish between different objects in the frame, venues need appropriate lighting and the distance to objects being detected cannot be too far. When evaluating the venues, understanding how the camera might be placed was therefore also considered.

We visited multiple venues which quickly proved non-optimal. These were either too busy for controlled testing, or the options of navigation were too limited. The Crown Casino, despite it being almost empty, offered a few different spots with multiple passages leading to the same destination. The high ceilings, reflective floors, and dim lighting which can be seen in [Figure 6 top] did not provide optimal conditions for the basic computer vision which we were going to employ.

We came across a foyer at the RMIT campus, with multiple revolving doors, all leading out to Swanston Street [Figure 6 bottom]. The foyer was well lit, with a single point of entry and multiple points of egress. This provides good conditions for computer vision as well as a controlled experiment. The ceilings, however, were quite low which could not provide a camera with a high enough vantage point.

Figure 5: Melbourne Central Railway Station leading into a mall with multiple points of egress

We also visited the Melbourne Central Railway Station [Figure 5]. It is simultaneously a large transportation hub and a shopping mall. Much of the signage in this venue is obscured, making it difficult to navigate through the many crowded spaces. Although this venue could benefit from Lace, the venue was simply to busy for us to conduct a controlled experiment.

Several of the venues inspired use-cases for the concept. To understand if our concept was helpful for our potential users, we needed to conduct a controlled experiment featuring a prototype.

Figure 6: (Top) Foyer at The Crown Casino with multiple similar paths. (Bottom) RMIT campus building with multiple revolving doors.

6.1. Establishing a Control

To test our prototypes function and user-friendliness, we needed a static, or unchanging, environment. This would narrow the focus of user testing to specific interactions while filtering out noise associated with complex real-world environments. We could regulate lighting, camera angles and obstacles in a venue. Our thinking was that answering as many questions as possible through controlled experiments could eliminate uncertainties when testing the final prototype in a more realistic scenario.

The control was primarily designed to test whether dynamically and autonomously altering signage over time was helping pedestrians better navigate a space. Conducting a controlled test would allow us to observe how people are moving through a crowded space without any guidance. The same users could then attempt to navigate that same space while being guided by the final prototype. Both tests would feature the same number of participants and thereby similar levels of density. After the test, we could ask open-ended questions such as how the participants felt navigating with and without the system, and if it helped them in any way. Such questions would provide qualitative data on whether the prototype helped its users navigate to a destination, considering any obstacles or densely crowded areas in a space.

6.2. Measuring Success

Because we intended for the concept to be able to exist within the constraints of a venue's existing hardware infrastructure, it had to require little to no additional hardware. Lace is supposed to respond to the collective movement of pedestrians in a space and advise each individual pedestrian about how to navigate that space.

The control was designed to test whether dynamically and autonomously altering signage over time was helping pedestrians better navigate a space. To determine if the concept is a success, we compared the result from our public user-testing with that of the control test. If a significantly greater number of pedestrians moved in the suggested direction with Lace when compared to the control, we would consider Lace to be a success.

7. Public User Testing

To converge onto our concept, we needed to engage with our potential user group. This section will show how we investigated various signage display methods, and how the investigation helped us determine which was most effective. We also investigated if there is an association between the position of signage and a passage leading to a user's desired destination.

We chose two primary display methods: digital overlays and two-dimensional interfaces. Digital overlays sit on top of the user's field of view, while two-dimensional interfaces for signage may resemble a mobile application. Because our aim was to add as little infrastructure to venues as possible, we would develop our response for use on personal digital devices, such as mobile phones.

To understand how users would react to different signage, we created an in-person experiment. This would give us the ability to engage in conversations with the users, and also allow for new insights and questions to arise.

7.1. Method

Prior to conducting our first user test, we created a number of low-fidelity prototypes for the aforementioned signage types. To allow for a quick and inexpensive workflow, our low-fidelity prototype was crafted using paper.

Before we could determine what our signs would look like, we needed to set up guidelines for the experiment. The goal was to learn the effectiveness of different signing

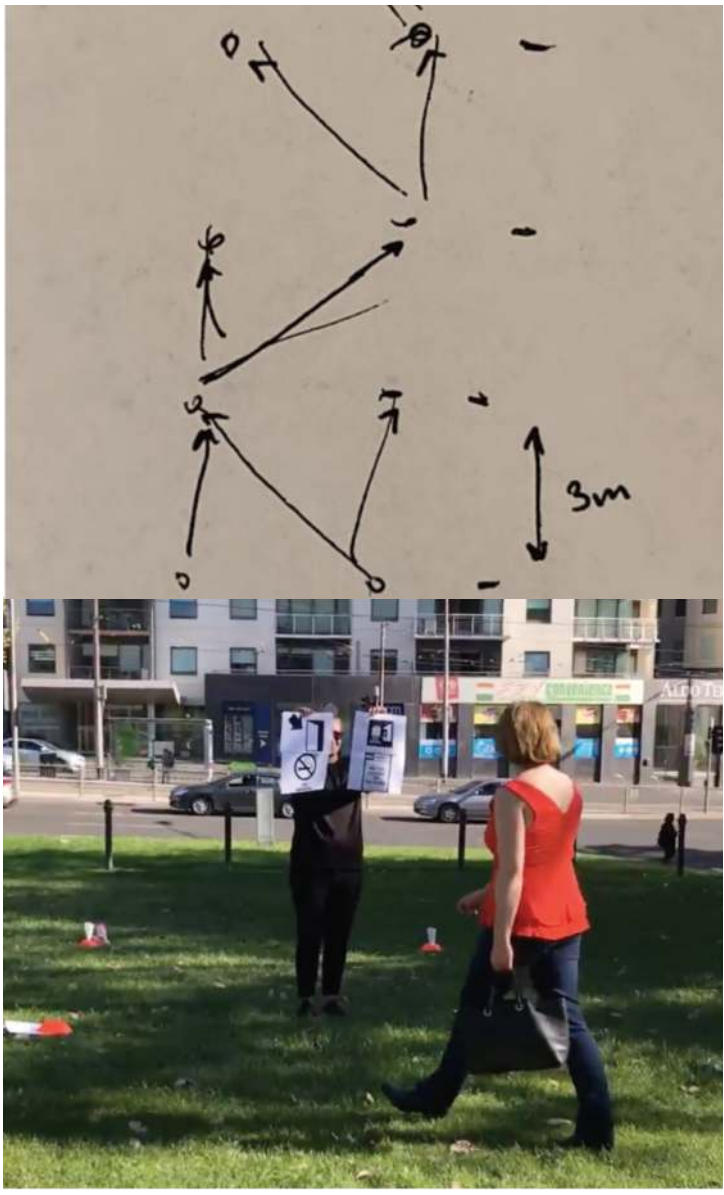


Figure 7: (Top) Design of the obstacle course. (Bottom) A participant in our experiment, walking left on a right arrow.

Figure 8: The design of the four different signage options

methods. We formulated an experiment in the form of an obstacle course. We presenting the participants with a number of signs and observed their reactions. This prompted us to design an obstacle course [Figure 7 top]. Participants were presented with a simple choice of going left or right.

The different signs presented to the participants included visual elements, textual elements, and a combination of the two. These signs were then presented as "digital" overlays and two-dimensional interfaces. The visual elements that we chose for the signage options were an arrow for giving the direction, and a door to signify the destination. The textual elements would suggest the user to either 'go right' or 'go left' to their 'destination'. [Figure 8]

The "digital" overlays were simulated by simply attaching cutouts of the above signage options to one side of a laminate pouch. This provided the intended ability of not only being able to see what is behind the sign but also being able to place it in front of the intended destination. This allowed us to test both the specific signage, as well as its position.

The signage options were printed onto A4 sheets of paper representing the two-dimensional interfaces. This part of the experiment would not only function as a way of testing the position but also act as a comparison. We wanted to understand if the medium in which the signage option is displayed could change the users' behavior about it.

We randomly selected a series of different commercial and instructional signs to display alongside the ones we had made. These surrounding "messages" could impact users' decisions about how to navigate towards their destination.v

Our experiments did not require much other than an unobstructed surface on which we could lay out our obstacle course. As our user group was rather diverse, we decided to set up and conduct the tests in a public space. This allowed us to approach random passersby and ask them to participate.

After participants reached the end of the obstacle course, we would ask a series of questions designed to start a conversation (open-ended questions) rather than a simple yes-or-no answer. This allowed us to better understand the users' desires and opinions regarding the subject.

In the end, we had conducted the experiment with ten different people over a four-hour period. This left us with plenty of data in the form of notes, videos, impression, and quotes.

7.2. Results

We had to synthesize the data collected from the experiment. We examined the answers to interview questions and reviewed the video footage from the experiments. We then noted down our observations on sticky notes and place them on the wall. [Figure 9] Sorting and coding our observations helped us discover common themes.

Figure 9: Extracting learnings from observations and interview questions gathered at the experiment

We grouped together notes addressing similar issues to establish if any compelling statements have been repeated across the board. These commonalities or themes were written onto separate sticky notes which would then form the basis for our analysis of the experiments. The themes, which can be seen in Figure [Figure 10] allowed us to spot trends in what the test users experienced.

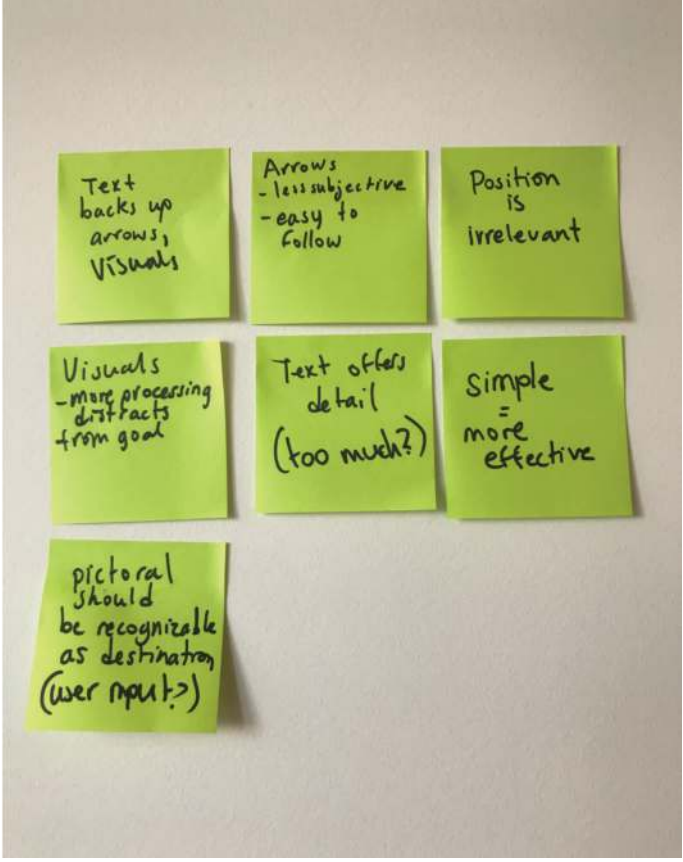


Figure 10: Themes and patterns extracted from the

7.3. Analysis

We extracted trends from users' experiences based on the results discussed in Section 7.2. The test users seemed to gravitate towards simpler methods of display, with a limited number of visuals. With multiple different visuals, the signage was distracting and require more processing. Purely textual signage seemed to cause misunderstanding between ourselves and the participants when interpreted without visual aids. This was apparent when a participant interpreted an instruction to go "right to destination" as "straight to destination"; they skipped the rest of the obstacle course and walked straight to the end. While purely textual signage might not be adequate in every situation, when phrased correctly, it can be utilized as a support for visual elements. This is especially the case if there is any uncertainty from the user or ambiguity in the language.

When asked if the physical position of signage was relevant, a majority of participants disagreed with the statement. However, when reviewing the footage of the experiment, it

became apparent that participants made a decision much more quickly when the sign was positioned closer to the destination. Some participants even started walking towards the sign before it was shown to them. This suggests that the position was relevant, even if the test users did not notice it. They subconsciously moved towards the signage which suggested less cognitive effort.

8. Survey

We had several lingering questions pertaining to the placement and perspective of arrows. In particular, discovering any correlations (or contradictions) between our experiments and the survey results was of utmost importance. We wanted to get user input on what specific types of arrow-signage combinations they might respond to. We were especially interested in scenarios in which they would be navigating with the aid of an augmented reality interface.

We determined that we needed more data backing our observations, as we felt that the data obtained from our sample size of ten participants was too inconsistent, making it difficult to draw overarching conclusions. We created a survey to gather a larger dataset, to learn from a larger sample size and compare those results to our findings from the initial experiments.

We conducted a survey with Google Forms. It garnered a total of 46 responses. The survey allowed us to compare users' qualitative opinions to their quantitative rating of different signage types. These ratings would provide us an opportunity to compare statistical analyses to opinions or suggestions from the survey —as well as from the public user test.

8.1. Method

We designed a survey to allow users to create their own combination of arrow and signage methods. Specifically, users would be able to rate each arrow position or perspective before choosing their favorite. They would then have to do the same for the signage type. Users could also volunteer an explanation of their thought process along the way.

A total of sixteen images were created for use in the survey. Users were then given the following instructions, repeating them for every step: This is the future, and you are wearing some snazzy AR glasses. Imagine you are walking around town looking for a nice cup of coffee. Please rank the following arrows by their ability to help you navigate towards a café.

Four initial images depicted different arrow types: one two-dimensional (parallel with camera plane) arrow and three perspective arrows positioned at the top, middle, and bottom of the observer's field of view. We created a set of four destination markers. These consisted of an icon depicting a cup of coffee, that same icon with associated text saying "coffee", just the text saying "coffee", and finally, a generic red location marker, or 'pin'.

We shared a link to the Google Form survey to our personal Facebook, Twitter, and LinkedIn accounts. Sharing the survey through these platforms had the potential to reach a number of different cultures, potentially providing us with different cultural interpretations. We accepted responses for three consecutive days with the goal of attaining as many responses as possible in that timeframe. By the end of the third day, the survey had gathered forty-six responses.

Figure 11: All the signage configurations used in the survey

8.2. Results and Analysis

The majority of the respondents volunteered detailed reasonings for why they chose a certain arrow or signage type. In order to make sense of the large number of responses, we generated normal distribution curves [Figure 12, 13]. We graphed curves of several parameters according to a 1-10 rating system. Parameters included the collective arrow type preference, collective signage type preference across all arrow type selections, and corresponding collective signage type preferences for each arrow type. The aim was to retain the integrity of the rating data while allowing us to qualitatively interpret that same data.

The arrow type with the highest collective preference was the bottom perspective arrow [Figure 11]. In contrast, the two-dimensional arrow had the lowest preference, with only one of the forty-six respondents having chosen it as their favorite arrow type. The middle and top arrow types were rated quite similarly —perhaps due to their common disassociation with the inferred ground plane of the image. The differences in total collective signage type preference were not pronounced, as the icon and icon with text ranked only slightly higher than the text and generic location marker. The correlation between the icon and icon with text, as well as between the text and marker should be noted. We speculate that this correlation is due to users' general preference for a representational icon —the common element between the preferred signage types.

Figure 12: Distribution curves for General Arrow Type Preference

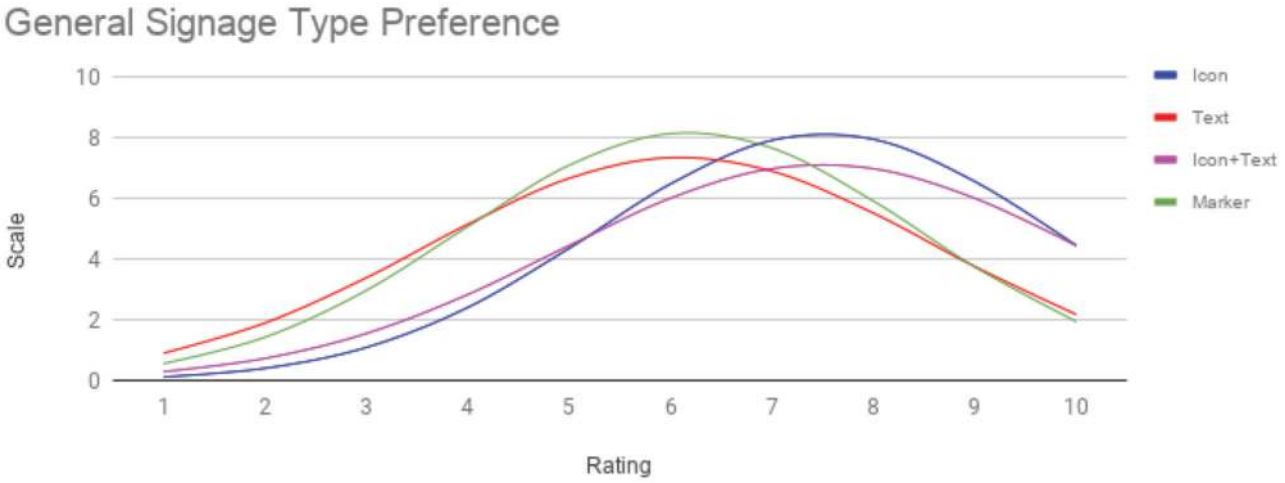


Figure 13: Distribution curves for General Signage Type Preference

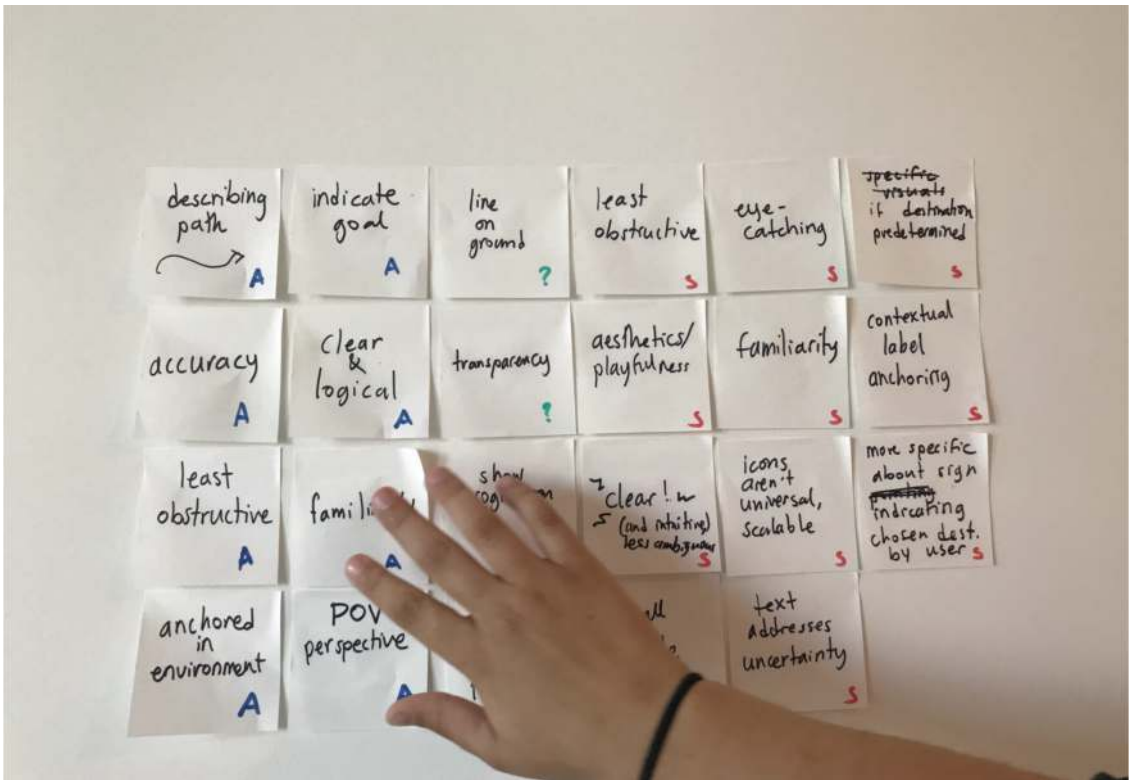


Figure 14: Feedback on arrows marked with a blue 'A', suggestions with a green '?', and destination signage with a red 'S'

Figure 15: Our learnings from the experiments side by side with the survey

9. User Personas

We selected groups of polarized opinions and began to craft personas. In this case, the personas are imaginary characters whose opinions and perspectives were made up of ones we found to be compatible when grouped together. The personas' ages and careers were based on those of our participants. The common practice of creating personas would function as a frame of support in the design and implementation of features later on in the process. In reducing the user feedback we had received into a couple of distinct characters, it would be easier to grasp and involve their concerns.

9.1. Bruno

- 55-years-old
- Process-oriented
- Seeks reassurance
- Prefers text-based signage

Bruno is a 55-year-old high school science teacher. He is data-driven and easily gets his set mind on things. This means that he prefers to see detailed information, such as specific distances or a direct visual representation to reassure him that he is navigating towards his

"I've got to get to my destination one way or another!"

desired destination. On the other hand, because Bruno gets easily distracted if unsure about the meaning of certain signage, he may ignore unclear or confusing guiding cues. Bruno will typically adapt to the environment within the limits of his preconceptions —his preconceptions being related to similar environments he has visited in the past. Providing specific text-based signage is important to earn Bruno's trust and facilitate his understanding of a space.

9.2. Victoria

- 28-years-old
- Goal-oriented
- Goes with the flow
- Prefers visual cues

Victoria is a 28-year-old paralegal at a law firm. She is career-oriented and values efficiency quite highly. This goal-focused mindset leads Victoria to care less about the process of reaching her destination. She is less actively engaged in navigation, preferring to go with the flow. Consequently, Victoria is more likely to follow simpler guidance cues. This entails signage that is less verbose and more standardized, leveraging pictograms such as arrows in conjunction with the shortened text. However, Victoria's preference for visual navigation cues is somewhat of a double-edged sword. Because she focuses more on imagery, logos and promotional messages can be especially distracting.

"I don't want to have to spend time figuring out how to get to my destination."

10. Iterating and Brainstorming

We initiated a brainstorming session that would generate ideas for four major categories. These categories were informed and defined during the research and include floating icons, picking a destination, depicting the goal, and presenting the progress.

The 'floating icons' category considered how to display visual guidance in a user's environment. This included both nudging patterns along the pathway, as well as how to represent a user's final destination. The category, 'picking a destination', considered how to enable the user to pick their desired destination. Depicting the goal referred to how our prototype could remind or describe the destination to which the user is currently navigation towards. Finally, the 'progress' category considered how we should illustrate the progress users would be making towards arriving at their desired destination.

The brainstorm was guided by the previous research and findings, allowing us at every step to compare ideas from the brainstorm with the findings and personas.

We selected groups of polarized opinions and began to craft personas. In this case, the personas are imaginary characters whose opinions and perspectives were made up of ones we found to be compatible when grouped together. The personas' ages and careers were based on those of our participants. The common practice of creating personas would function as a frame of support in the design and implementation of features later on in the process. In reducing the user feedback we had received into a couple of distinct characters, it would be easier to grasp and involve their concerns.

10.1. Conceptualizing

The brainstorming session was divided into four parts of ten minutes each. One session for each of the four categories mentioned above. The goal was to sketch out several ideas for each category, while still trying to adhere to the findings. We evaluated each of the concepts against the learnings that we had generated in Section 7.3 and 8.2 in order to determine which of them still held water. We used the "surviving" concepts in the design of the paper prototype.

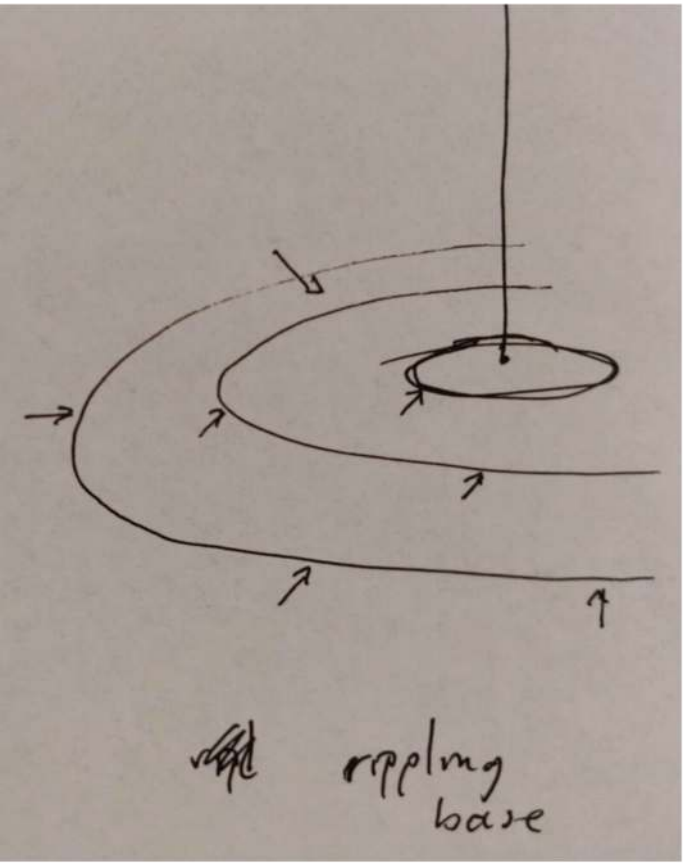


Figure 16: Rippling base to signify destination

Floating Icons

We produced an assortment of floating arrows, planted signs, rippling floors, fireworks, and cute characters. While they did not all hold up when evaluated against previous learnings, two ideas were combined in such a way that they more obviously represented the user's desired destination. Figure 16 and 17 shows the two ideas which were combined into one.

On the left is a sketch of a rippling circular or globular pattern on the ground. On the right is a sketch depicting an arrow floating above the ground and pointing downwards towards the destination. The goal of the globular pattern was to draw people in towards the destination, and also to visualize the boundaries of the destination. We felt it may be difficult and cumbersome for users to understand that they have reached their target destination if the destination is rendered as a single point in space. Rendering a larger area of completion would eliminate these issues.

The rippling circular pattern was adapted to three dimensions, taking shape as a floating spherical aura. In a crowded space, a two-dimensional aura on the ground might be obstructed. It could, therefore, be difficult for the users to find.

The purpose of the arrow was in its ability to provide a more conventional combination of

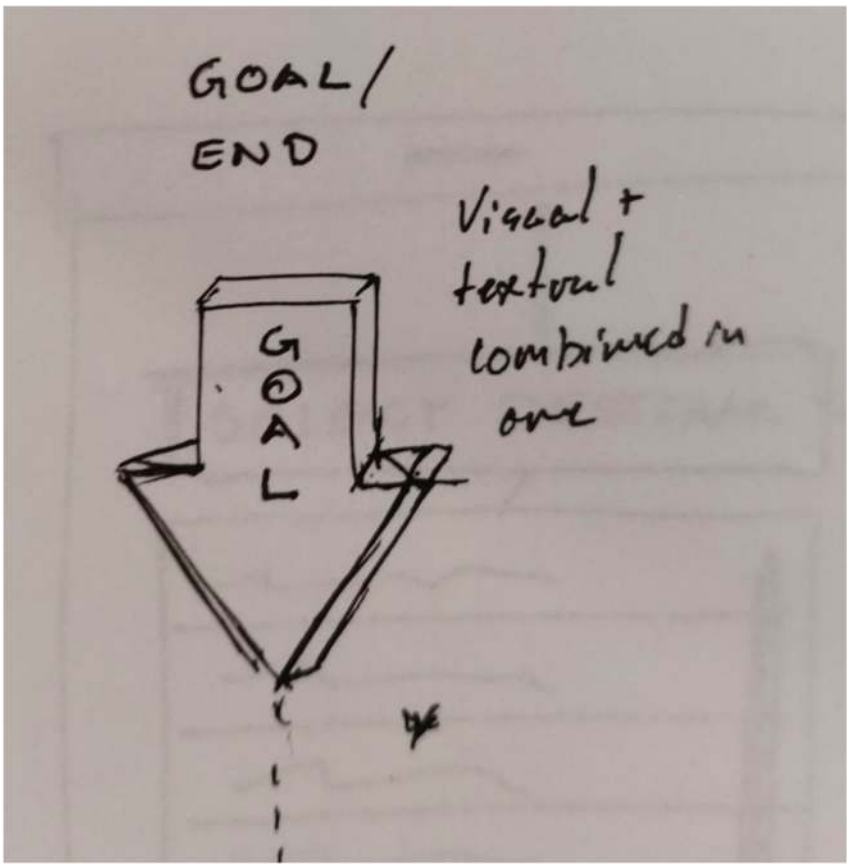


Figure 17: Floating arrow to signify destination

both visual and textual guiding cues. While floating above the aura, it could act as a label and provide some guidance from a distance.

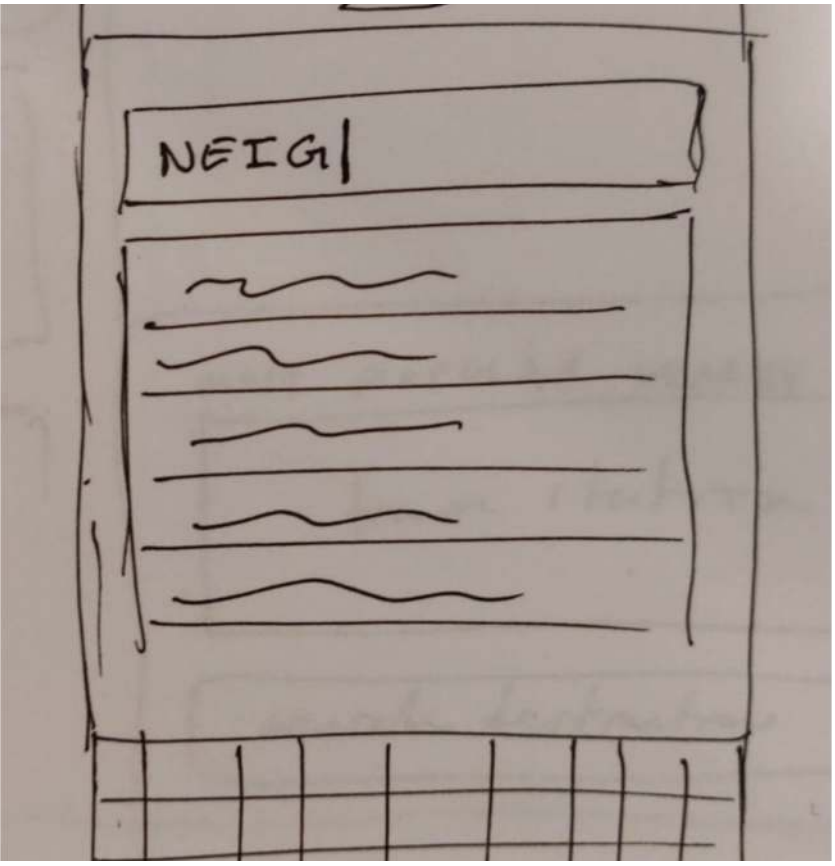


Figure 18: List view for picking destination

Picking the Destination

For this category, we ideated on different ways for users to select their desired destination. Ideas included tapping on the floor in the real world or tapping on the representational map or floor plan. Users could be a shown list of destinations in a drawer and sort that list by categories. These categories could be described either through three-dimensional bubbles or by planting locations in the direction of their real-world coordinates. Some ideas carried unnecessary complexity —not only in terms of development but also in ease of comprehension. We decided to move forward with a more traditional two-dimensional list as the focus of this project was to guide users, rather than to enable them to explore destinations. The simpler, two-dimensional list would also make it faster to develop the prototype.

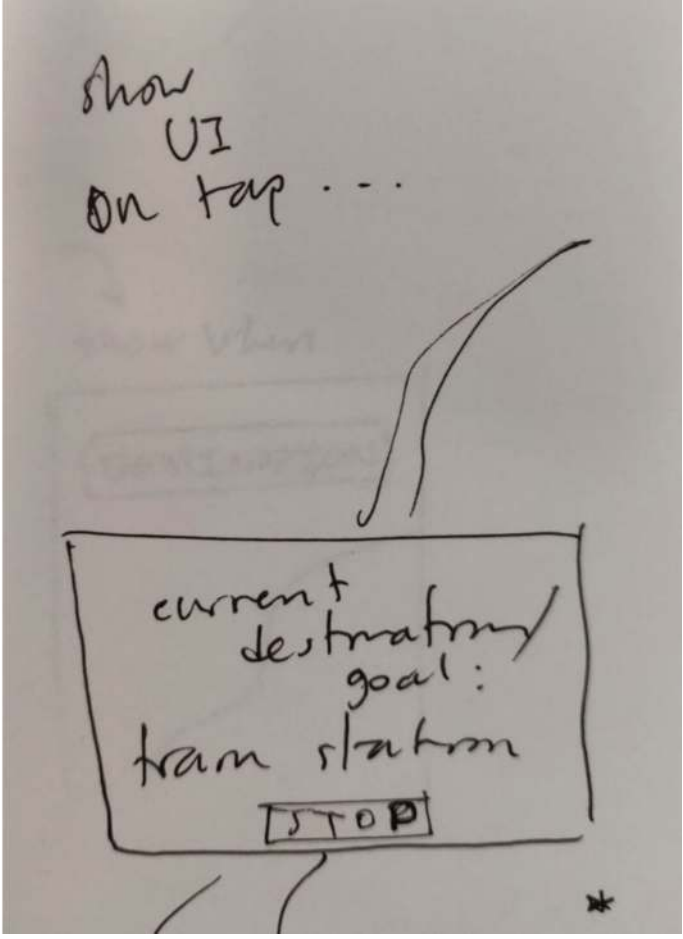


Figure 19: Depicting the goal during navigation

Depicting the Goal

Here, we generated ideas about how to display users' desired destination while they navigate. This was mostly intended for goal-oriented users. The session generated numerous ideas. Most of them were situated around describing the goal textually within a drawer, upon tapping the screen, or as a smaller label in the corner of the screen. Some of the zanier ideas included a crazed animated train conductor yelling at the user from his locomotive. In an effort to avoid cluttering users' field of view, we chose to provide textual instructions on a two-dimensional card. This card would show up when users tap the screen. This was simple enough to develop for the prototype, while still adhering to our learnings from Section 7.3 and 8.2.

Progress

This was perhaps the most ambiguous prompt, as the team had not yet determined what progress could really mean to users.

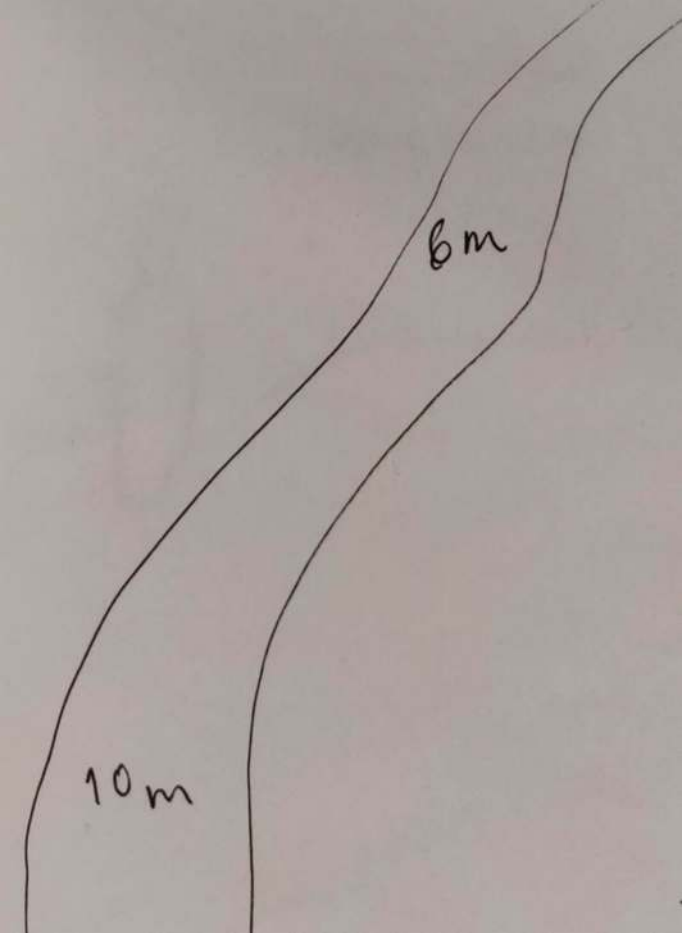


Figure 20: Signifying process during navigation on the line, using meteres to destination

Distance to the destination was referenced the most. Represented as changing colors measurements along the path or in smaller labels. In a couple of instances, the distance was depicted in its entirety as a secondary simplified graphic along an edge of the display. We also considered how crowd-density might be represented in the aesthetic of the path itself. The idea that was chosen to move forward with, can be seen in the figure on the left. Specifically, it is a three-dimensional "path" lightly fading away from the user. The distance to the goal is depicted at the base of the path using a standard textual representation.

10.2. Paper Prototyping

The early prototyping session was done with paper, as the low fidelity properties of the material allowed us to quickly iterate upon, and experiment with different concepts. The intention was to tie together ideas from the brainstorm and bring forward a single concept that the team could test with users.

10.2.1. Creation

The two sketches from the brainstorm were a simple list of possible destinations, as well as a two-dimensional floor plan of the venue. We created multiple different versions, allowing us to test which would provide the most clarity and ease-of-use.

After a user selects their desired destination, they would be shown the most efficient path which is determined in relation to other people

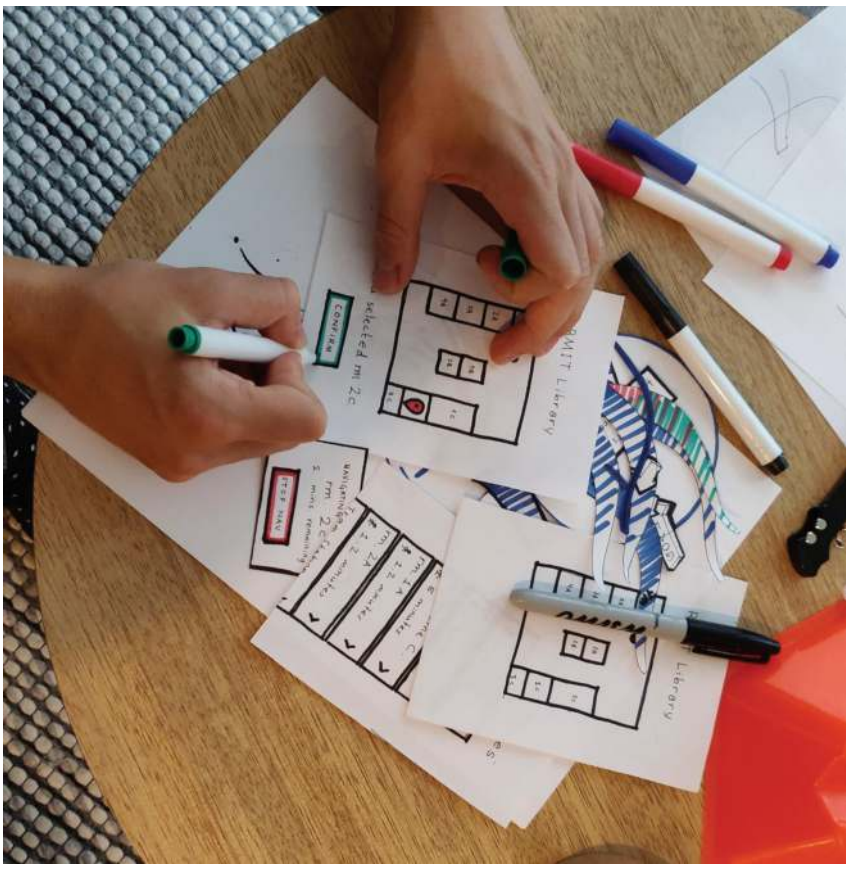


Figure 21: Crafting paper prototypes

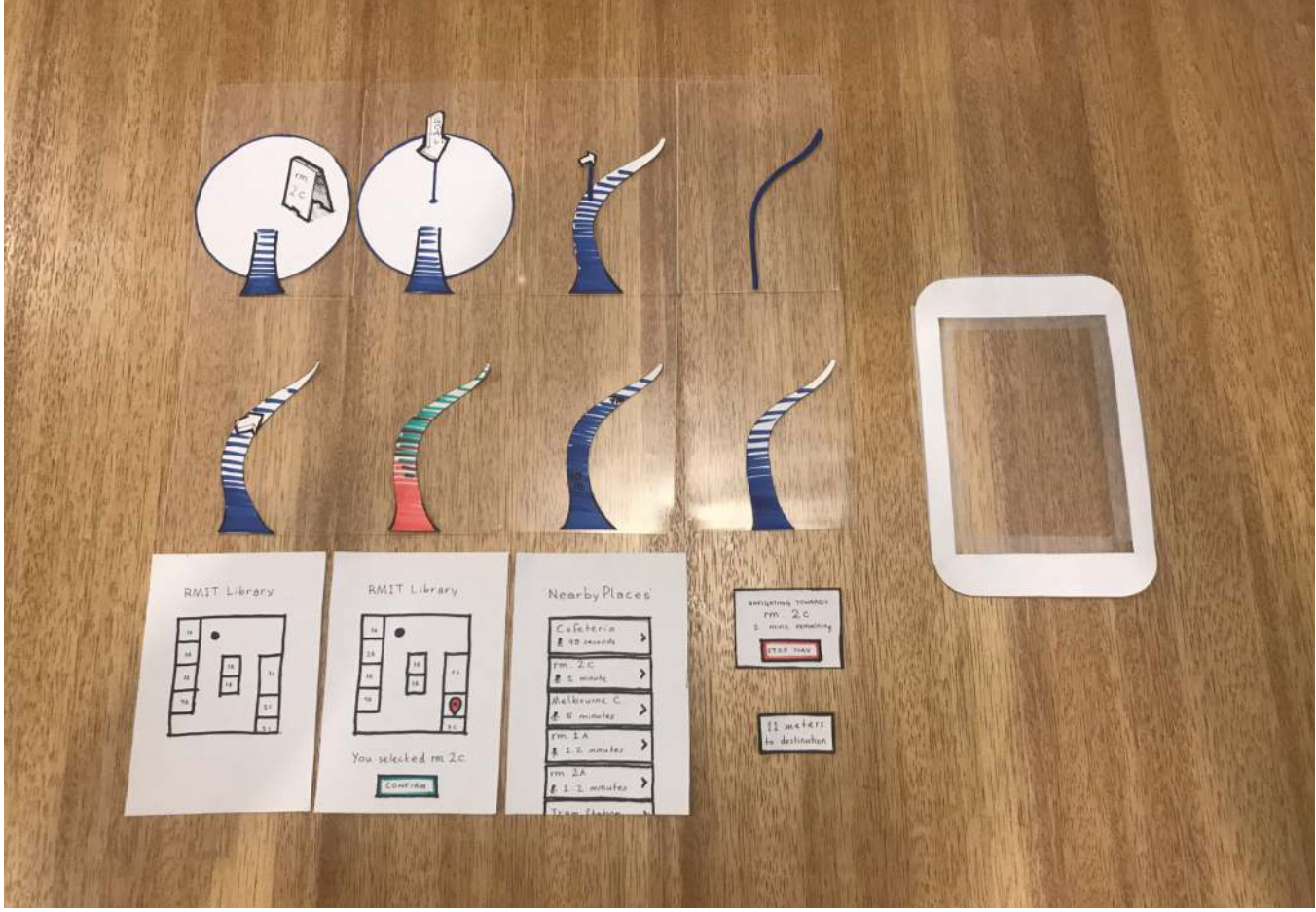


Figure 22: The individual components of the low-fidelity paper prototype

and objects in the space. This path was visualized in two different ways - either as a thin line, or a wider path tapering off in the distance. We designed two different visualizations: a color gradient fading away from the user or a textual representation with the number of meters from the user to the destination as shown in Figure 20.

When starting navigation towards a destination, the client application needed some way of depicting the goal. Depicting the goal is intended for ending navigation, changing the destination, or simply getting reminded of the chosen destination. We depicted the goal as two different two-dimensional cards which would be placed somewhere in the user's field of view. Both cards provide a textual description of their destination, as well as an option to end navigation. The main difference was that one of the cards would be permanently visible in the user's field of view, while the other would be prompted by tapping the screen.

The goal object was composed of a globular aura and a secondary visual element accompanied by text [Figure 16 and 17]. Two different options were made for the latter, one being a floating arrow with the word "GOAL" written on it and another being a physical sign that stood on the ground.

With all the components drawn and cut out in paper, we created a frame of a mobile device. The frame had a translucent material in the center, on which the different components would be placed. This would also allow the users to perceive the context of the environment behind the device, as it would be in augmented reality.

10.2.2. Testing

To test the validity of the different paper prototypes, we sat down with a landscape architect and a media technologist. The process involved a 10-15 minute discussion where the participant would explain what they understood from different concepts, as well as what they expected to happen when performing certain interactions. This was followed up by a more in-depth discussion about different use cases and other topics that might have become apparent during the testing itself. The conversation was purposefully kept open, to gather qualitative recollections rather than yes or no answers.

Landscape Architect

The landscape architect liked that the prototype featured iconography for walking distance and time estimations. The subject expected a dropdown menu when tapping on a destination, indicating that he was seeking further information about each destination. He expected to be shown directions, as well as his current location on the map, in the mentioned dropdown. Hence, the augmented reality capabilities of the prototype could have been less than clear. His first impression of the prototype was apparently confined to the destination selection view.

Upon showing a floor plan of the room, the architect struggled with locating his destination. This was primarily due to the handwriting depicting the room numbers. Locating the destination on the map took much longer than locating the destination from a list. Upon locating his destination. He did not hesitate to tap the green “Confirm” button to lock in his choice.

We suspected that a faded pathway might lead to easier depth comprehension as it would recede into the distance. This would aesthetically mimic the effect of atmospheric diffusion on the visibility of distant objects, a familiar natural phenomenon which could aid users’ depth comprehension. The subject suggested representing doorways in the space as the user moves through it. Providing enough context relative to the path could play an important role in effectively guiding users.

Upon tapping the screen, the architect said he would expect either some sort of blinking to occur or a point in his surroundings to be highlighted. This point could be represented as an icon, avatar, character, person walking through the space. When faced with a pop-up window instead, he toyed with the idea of placing a walking man or stopwatch icon on this screen to indicate remaining time in the user’s journey. Upon tapping a button on this pop-up labeled, “Stop Navigation”, he expected to be shown a floor plan of the space he was in. Alternatively, he would expect to be shown amenities or details pertaining to his destination.

In retrospect, it is unclear whether or not the architect fully understood how the pathway would be situated in an environment when interacting with the prototype. We found that it can be a challenge to test prototypes about augmented reality, as it is such a new technology. Something that could improve our paper prototype testing methods would be to draw in an imaginary three-dimensional context behind the user interface. This might offer more context to the user interface elements. Unlike paper prototyping for contained two-dimensional prototypes, three-dimensional objects, or installations, augmented reality experiences demand a significant correlation between the hardware form factor, digital two-dimensional interface, and the context of the test itself.

Media Technologist

We took what we had learned from the previous test, and came to the conclusion that we should test our paper prototype with at least one more person. This time, we met with a media technologist and augmented reality enthusiast. Because this subject was more familiar with the

relevant technology, we felt that he could express a more cohesive understanding of the prototype.

When the media technologist was asked to navigate to his destination, he immediately wondered whether that meant doing so through the app or by looking around his real environment. When navigating to a numbered room, he preferred to just seek out the numbering patterns of all the rooms. Alternatively, if the rooms did not correspond to any numbering system, he would need to ask for help on how to reach his destination. The subject preferred using the list view to choose a destination, especially if the destinations could be sorted alphabetically or numerically.

The media technologist wished to see more information about a room when selecting it in the list view, such as its availability, a photo of that room, and/or more information. However, he did not think starting navigation immediately would be too much of a commitment. Upon tapping on a destination in the room layout-view, he expected to see a blue guiding "trail", or path, akin to that in Google Maps. He cited its universality, and by extension, its familiarity, as reasons for why the path should be blue.

When launching the navigation view, the media technologist wanted to see some form of instruction explaining to the user that they could follow the path toward their destination. Since so many people are not yet familiar with using augmented reality applications, explaining to users that they are able to move their phone around could be helpful.

Upon tapping the display while in the navigation view, he expected to be shown more information or alternative routes. He specifically mentioned the ability to see how much time was left in his journey, cancel navigation, and possibly even an option to change the color of the path. The subject spoke passionately about the potential of creating scenic routes; he thought it would be cool if he could add stops to the route if he tapped on the path.

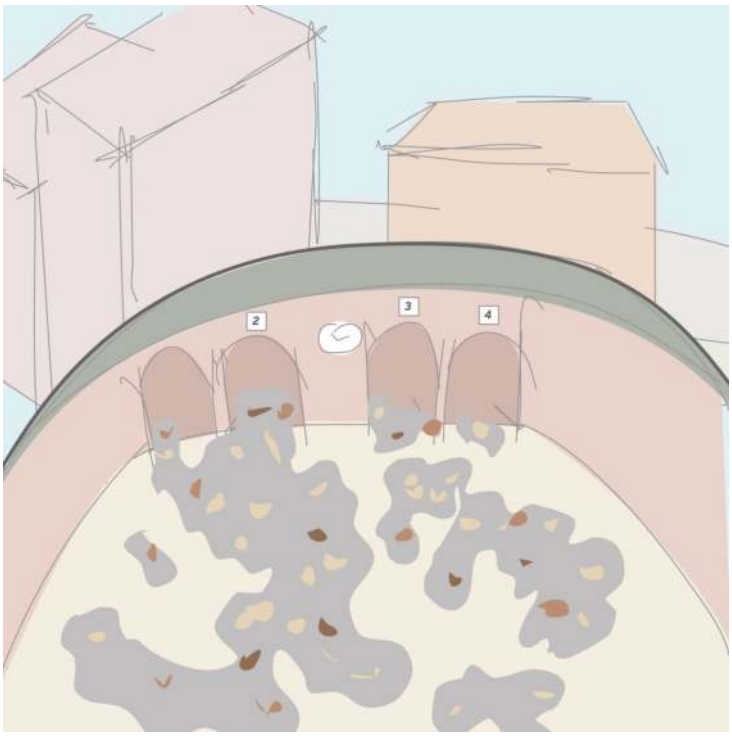
Showing the estimated time to the destination was especially important to the subject. While distance is constant, the time it takes to reach the user's destination is dynamic. He talked about the joy of trying to beat the time estimation he is given in traditional navigation apps, such as Google Maps. Apparently, it can feel like a challenge —he described using the time to indicate progress as being playful. He described the remaining time as his current task as an individual. So, he preferred the time estimation to always be present on the screen. The time estimation could be shown together with the remaining distance to the destination.

The media technologist was intrigued by the idea of having the guiding path follow him so that he would never veer off the path. He preferred a wider path to a narrower one, as a wider path would require him to spend less time searching for it. He expressed concern regarding a path of greater width: in a worst-case scenario, a wider path could obscure less prominent obstacles on the ground, posing a possible danger to users. The subject had quite a strong preference against having the path fade away into the distance. He felt that it was limiting his perspective, as it kept him from seeing the whole route. He was concerned that limiting the visibility of the route to the first few meters —despite its preferable aesthetics— would result in users mindlessly following the route ad infinitum. The infiniteness was the most bothersome to the media technologist. If the path was solid for the whole route, he suggested drawing notches on it to aid the user's sense of depth.

According to the subject, the navigation process should be made harder to stop in order to prevent that happening by accident. Rather than tapping on the screen to access the 'Stop Navigation' button, he suggested enabling users to tap and hold on the route until it wiggles —indicating one's ability to discard it. He also suggested using extreme movements, such as

having the user point their device at their feet or to the ceiling, towards an “Exit” button.

In both the layout diagram and the AR view, the media technologist expressed a preference for initial guiding arrows. In the layout diagram, the blue location indicator could show a direction in which the user should walk. In the AR view, arrows could help users find the path, indicating to them to turn their devices until the path is in view. He asserted that any assets along the path should be rendered in three dimensions in order to utilize the full ability of augmented reality.

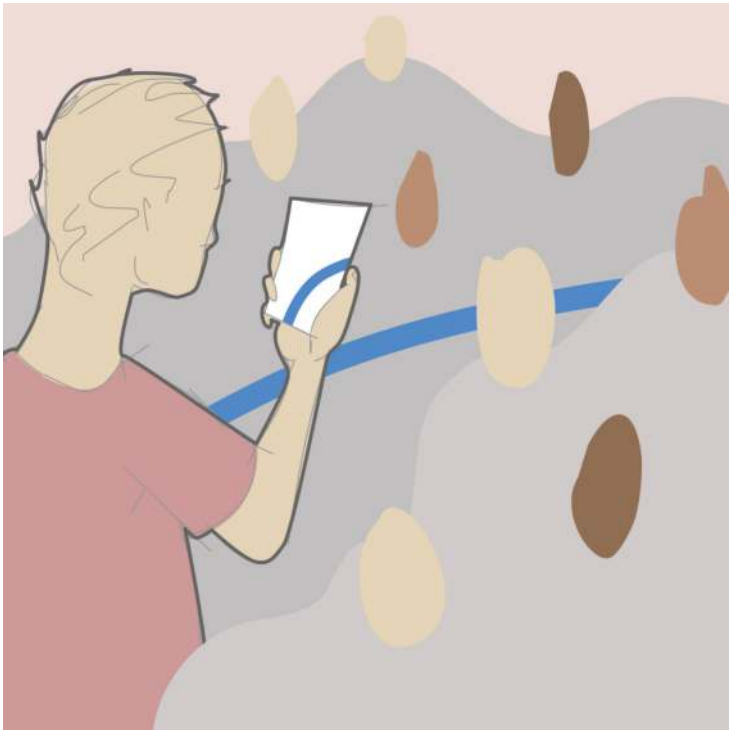


1 — A crowded train station in a foreign land

11. Final Prototype: Design

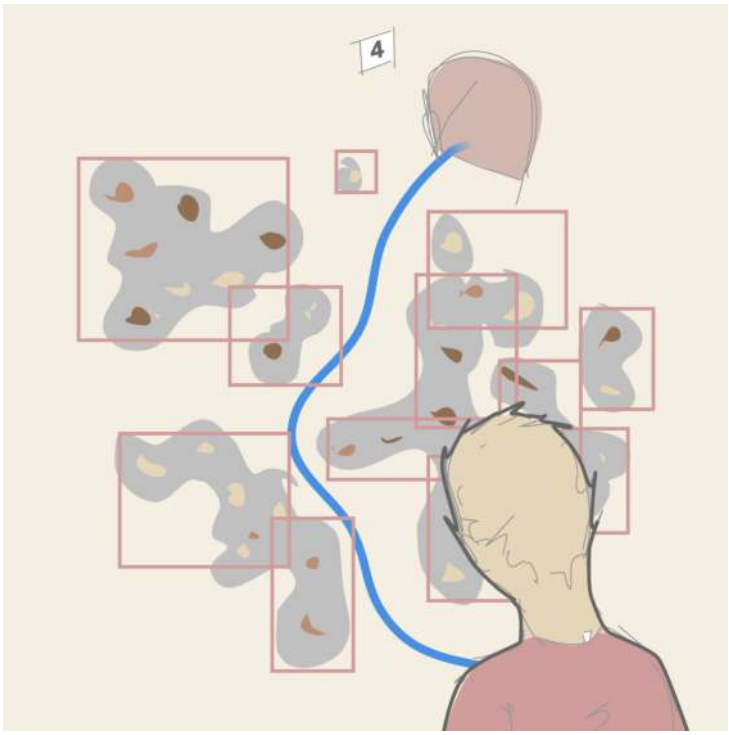
It was important to build a sense of familiarity into Lace in order to lower the barrier for consumer adoption. Although the vision for Lace is to run it on an augmented reality headset, the Lace prototype will be built and tested on a mobile device. As such, we still wanted the prototype to feel like a mobile application.

We felt Lace should incorporate a level of playfulness to facilitate an otherwise utilitarian task. Including some playful and dare-we-say flashy elements could attract users to start using the application. While a certain level of playfulness is good for gaining attention, retention rates stem from having an application that is simple and concise in communicating its use. Therefore, minimalistic concepts were weighted more heavily.

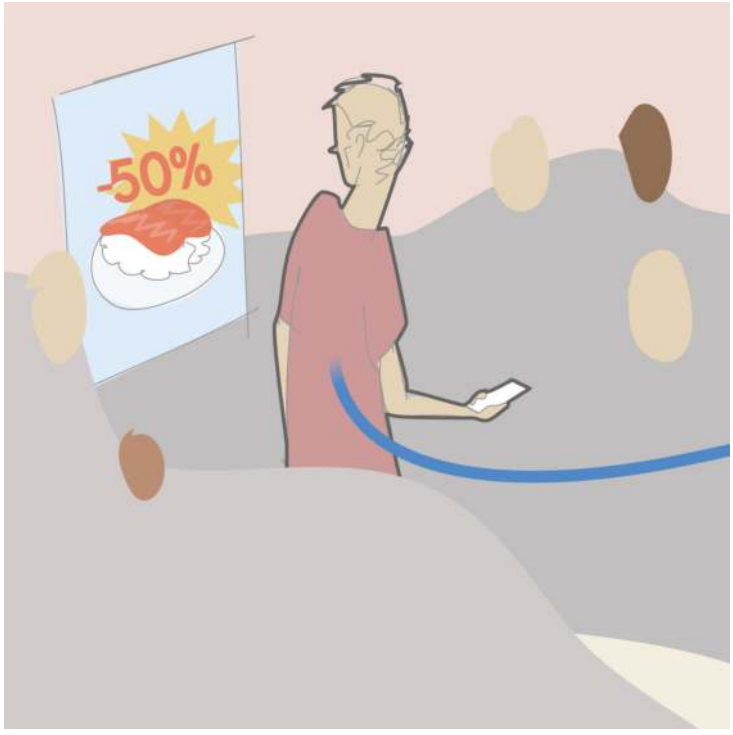


4 — Our friend uses Lace to find his platform

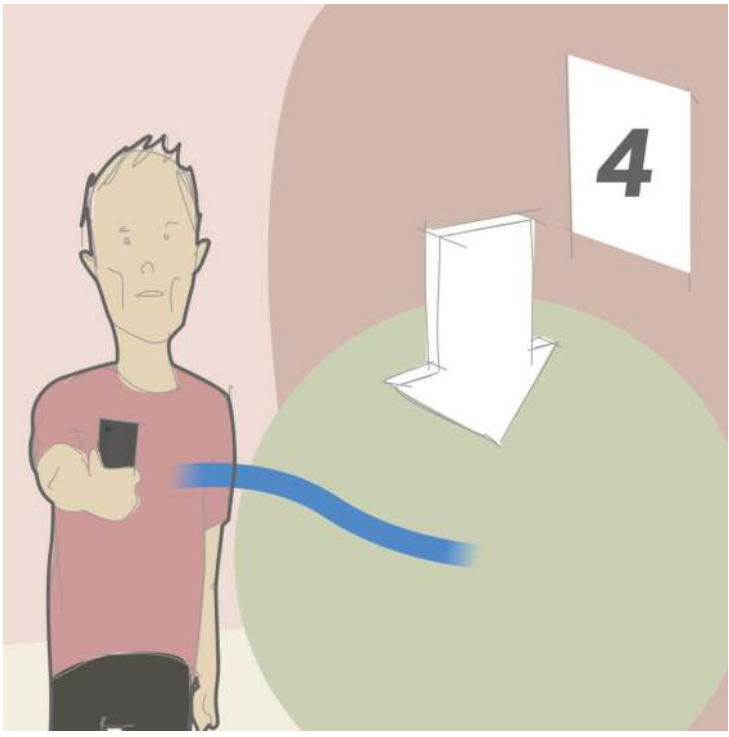
2 — Our friend looks confused. He's late for a train.



3 — Fear not! Lace has a plan!



5 — He's a tad hungry. No matter what he decides to do, Lace will pick up right where he left off



6 — Our friend finds the platform. The day is saved!

11.1. Flow and Layout

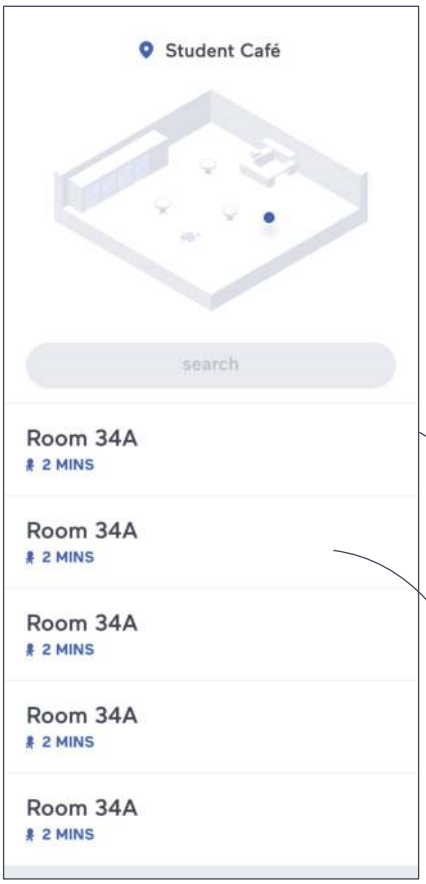


Figure 23: Destination selection from list

Our primary concerns when approaching the design of the app were affording users the ability to select a destination and guiding them towards that destination. The user interface was broken up into two primary screens: a two-dimensional interface which affords the ability to select a destination to navigate to, and the world-space (representation of what user sees) augmented reality (AR) view, which draws an optimal path from the user to that selected destination. In the final design, users are able to show and hide a card with some options on the latter screen. We reasoned that separating the destination selection screen from the navigation view would ensure that after launching the app, users' attention would primarily be occupied by a single, simple task. Users' real environments could be distracting, with real-world elements interfering with the task at hand. Making a distinction between the two-dimensional interface and the three-dimensional AR view is made possible by the fact that the Lace client is running on a mobile phone. Because displaying an interface on a 5-inch display is not as fully immersive as integrating digital objects into a real-world environment, we do not have to worry about any elements obstructing users' actual field of view.

We had previously begun to consider how to onboard users onto the app. However, we've learned from past projects that implementing an on-boarding flow can become a major inhibitor for users —especially someone like our persona, Victoria. Data is a sensitive subject, which makes it difficult to attain before users have used the app. So we discarded on-boarding altogether. The primary benefit of having an on-boarding process in an app lies in its ability to teach users how to access its basic functionality, as well as how that functionality could benefit that user. In the absence of an on-boarding flow, the immediate functionality of the Lace client would have to be inferred intuitively. While the best way to verify the effectiveness of the app's initial affordances would be to test it with users

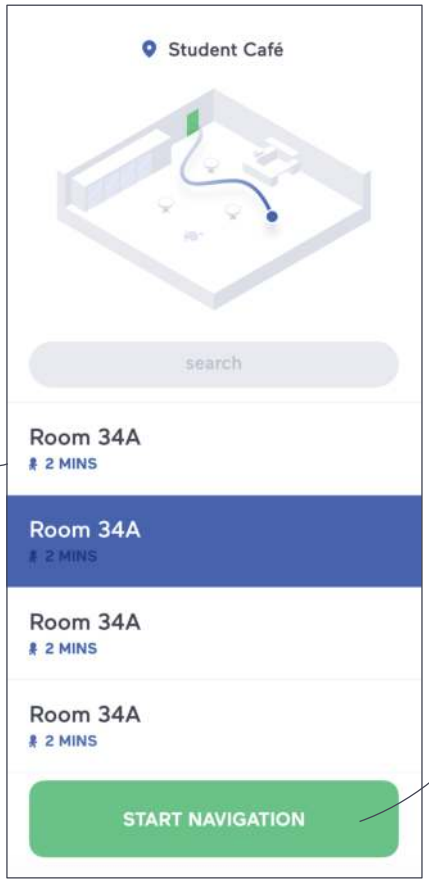


Figure 24: Starting navigation towards Room 34A

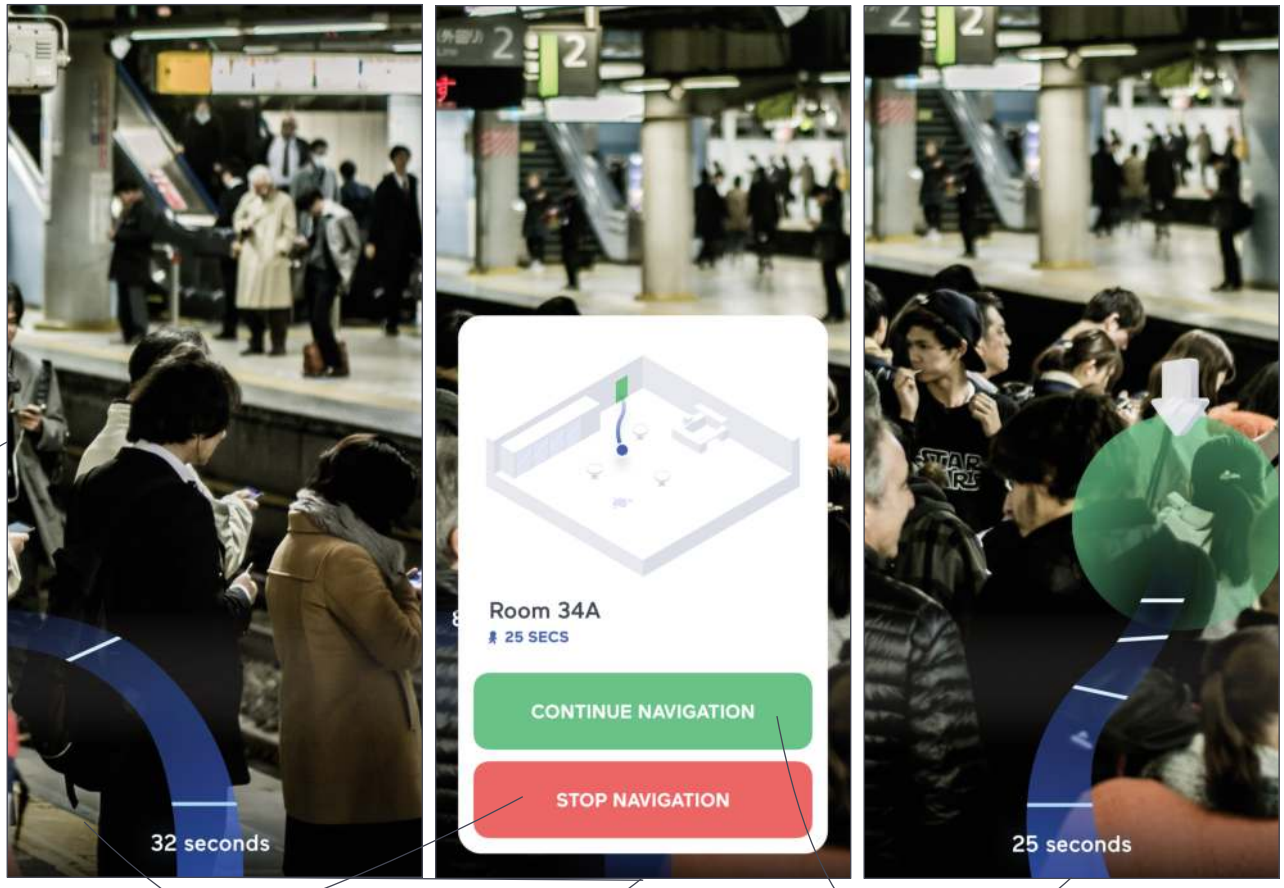


Figure 25: Navigating towards destination (left). Getting informed about the destination (center). Seeing the destination endpoint (right)

[Section 13], there were still certain precautionary considerations we could make.

Both new and old users will see a destination selection screen upon launching the Lace client. On this screen, they can see the name of the venue in which they are located, a visual layout of their location featuring their position in the venue space, and a list of destinations they could navigate to in that space. In future iterations, this list could be filtered by alphabetical order. Each destination in the list features an estimated travel time. Restricting the selectable destinations to a preset list was preferable in the context of developing the Lace client prototype, considering that we were expecting to run controlled experiments in the future. Upon tapping a destination name, the Lace client calculates and visualizes a path on the room layout according to the current conditions of that room. Users of the paper prototype expected to be shown some additional information about a destination when selecting it.

When a destination is selected, a button will appear along the bottom of the screen. This button is not visible if no destination is selected. Upon tapping this button, the destination selection screen will be hidden and the user is shown the AR view with a path guiding them towards the selected destination. When testing the paper prototype, both of our test users hesitated to tap the green button when it appeared.

The AR view is comprised of the device's camera feed, as well as a 3D-rendered path that connects the user to their destination in the world-space. The path always starts at the user's position, inspired by the concept that the user can never be wrong; the user is always on the right path. This is a mechanic that could satisfy both Bruno and Victoria: Bruno should be able to trust his own decision over Lace, and Victoria would not have to think twice about following the path. If the user is looking away from the path, Lace gives a hint to the user about which direction they should turn to face the path.

The distance to the destination is marked by notches at every meter along the path, in an effort to help users understand their journey's progress, as they move through the space. The destination is marked by a translucent "aura" accompanied by an arrow. The arrow indicates the exact position of the destination, while the aura represents the "goal-space". When the user enters the goal-space, the navigation will end.

If a user taps on the navigation screen, a card-like overlay will appear on top of their world-space. This card features the layout of the room with the user's location and a path connecting them to their destination (the same one that is rendered in the world-space). It also displays the name of the destination to which they are navigating, the estimated travel time to that destination, an option to stop navigating and show the destination selection screen, and an option to continue navigation. The latter option hides the card-overlay.

11.1. Visual Design

It was important to us to properly represent an environment with just the right amount of details for each individual; Lace could help users orient themselves with respect to the path connecting them to their destination. An isometric projection of the room layout was selected as an artistic nod to architectural drawings, as the concept of flow and circulation in architecture has had a strong influence on our response to dangerous crowding. Isometric drawings lead to aesthetically pleasing and undisrupted overviews. The main drawback of using isometric projections is that they are not effective at depicting details. Resolving height differences can also be difficult; Vertical displacement is not easy to illustrate isometrically. However, in this case, the vertical axis was not necessary to consider, as we planned to test the prototype in a single-level environment. The projected layout only serves to facilitate the user's understanding of a space. It is not interactive.

The isometric project primarily reconstructs the users' environment, providing a visual aid to help the users understand the space and their physical relationship to it on a holistic level. We considered placing various landmarks in a map view, such as the position of a café or the available exits, thereby supporting 'route' navigation. While it was determined that detecting landmarks in real-time would require significantly more processing power, some objects' positions could be hard-coded and visualized in the initial prototype.

The guiding path was colored blue as a nod to Google Maps' blue navigation path. It was determined that using a familiar color for the path could help earn users' trust in the prototype. Google Maps is a service people use every day, and so we wanted to capitalize on their market success. We also trust that Google's solution is rooted in a substantial amount of research. The path is of constant semi-transparency, to avoid obstructing users' fields of view. The path is also wide, to increase recognition and visibility.

It has been found that text is most legible when placed in the lower half of augmented reality displays [37]. Additionally, displaying white sans-serif fonts on black backgrounds is preferable [9]. Rzayev states that augmented reality interfaces inhibit reading ability and comprehension [37]. This could be due to the fact that the respective experiment took place on a mobile device, where there is a frequent shifting of focus from the display to a user's surrounding environment. We understood that this is a limitation that the Lace client prototype could have faced. Yet, we were inclined to forego testing on more expensive headsets due to budget constraints, instead focusing on designing for mobile devices with greater ubiquity —such as the smartphone.

12. Final Prototype: Development

To asses the movement of crowds in a space. This system would need to understand the space, as well as how people would move through it. Our system needed a way to detect and identify people in the space and translate their respective placements into a set of coordinates. This information would then have to be transmitted to the user's application, allowing it to determine the best path through the space.

We utilized users' mobile devices to calculate the best path through the space, and present it back to the users. The final system was divided into two separate systems, which would then communicate with each other through a simple and secure RESTful API, with a single endpoint.

The first system, the server, utilizes a video camera placed in a space to understand the movements of the crowds. The server would run on NodeJS, an open-source, cross-platform run-time environment used to execute JavaScript code. We implemented a library for developing machine learning algorithms called Tensorflow. To then detect objects and people in the video stream, we use an SSD architecture with a MobileNet feature extractor which is a state-of-the-art, object detection system. All of this would be programmed in Python. Finally, a MongoDB database is used for storing the models of the space, until they are requested by the client through the API.

The second system, or the client-side implementation of the prototype, is implemented as a mobile application. Per section 13.1, the goal of the client was to render an optimized three-dimensional path from a user to their selected destination. This path is determined by a common pathfinding algorithm known as A* ("A-Star"). The client utilizes the Google ARCore SDK in conjunction with Unity in order to render this path as a digital layer in a user's physical environment. ARCore is also used to determine an accurate reading of a user's location, as well as to ensure that three-dimensional assets are properly anchored in their environment. As the client is developed entirely within the Unity environment, all of the program scripts are written in C#.

12.1. Client

Developing the client prototype for augmented reality required a host of new platforms and tools with which neither of us had any prior experience. This presented a new set of challenges which led to quite a steep learning curve.

12.1.1. Mobile Platform

Prior to this project, most of our prior experience in developing front-end interfaces involved React Native, ReactJS, and C++. While we would have been much more comfortable developing the app using a familiar set of tools, the computational resources required to properly render augmented reality environments are heavy. This limits the compatibility of AR platforms to only a handful of new devices.

We chose to develop the prototype for the Android platform, due to budget constraints. An Apple developer account is expensive and required to register test-devices. Since both of us had Android devices, we decided to load and test on one of those devices: a Huawei Nexus 6P.

React Native was ruled out as a development

platform, due to augmented reality being such an intensive process. React Native runs processes on a single thread, hence limiting the number of processes a program can run at one. A resource-hungry augmented reality process would likely block the thread. This could cause postponement of other operations, such as making HTTP requests or running the pathfinding algorithm. We wanted to ensure that the program as a whole ran as efficiently as possible to accommodate for any sudden crowd-movement in the testing environment.

12.1.2. Augmented Reality

Achieving a sense of presence through augmented reality is important because it can aid users in connecting with their environment. Presence can be described as either eliminating mediation or the illusion of mediation [25]. The superimposition of digital layers on top of a perceived real-world reality is the introduction of a mediator between a user and that reality.

An inherent risk in guiding pedestrians is the offloading of cognitive decision-making to our system. This could cause pedestrians to become worse at navigating on their own over time. [5] The danger in this is that some pedestrians might learn to ignore stimuli in their environments [7]. If Lace is able to provide sufficient tracking and "eliminate" the mediary, users may feel more connected in their environment and consequently more in control. [7]

A potential drawback from designing Lace for use of mobile phones rather than dedicated augmented reality headsets is that users may be more distracted by constantly switching attention from a small device to their surrounding environment [7]. Indoor navigation requires a lot of resources from pedestrians [5], so it is of the utmost importance to decrease the cognitive load by focusing on avoiding any level of obstruction between users and their environments.

Tracking in augmented reality can be defined as the ability of digital artifacts to consistently anchor themselves to the real world. Good tracking is when objects stick to their environment, despite any movements of the device running the application. This is especially important if we wish to improve the sense of presence for our users. Increased sense of presence can enable users to feel a more intuitive sense of control when interacting with their environment [39]. Inaccurate tracking could lead users to distrust our prototype, to the point where they might consider using an entirely different solution [30]. Misaligned arrows could lead users to interpret the guidance differently than intended [30], possibly causing them to stray so far off a route that any potential benefit from our prototype is ultimately negated.

The importance of effective hardware-software integration cannot be understated. Because augmented reality is still so new to the market, how to achieve high-accuracy tracking is still unclear. Augmented reality technologies rely on a myriad of different sensors [39] to anchor artifacts in a user's environment, and no available solution has a significant advantage over its competitors.

Photo by David Grandmougin on Unsplash

The Lace client was intended to be run on a mobile device. While there are several advanced augmented reality headsets available on the market today, such as the Microsoft Hololens or Magic Leap One, it was difficult to conduct full-scale testing, as the vast majority of people do not currently have access to one, including ourselves. Because the majority of people carry a mobile phone with them —with most of the newer models natively supporting augmented reality— it made sense to optimize our development process for supported mobile devices. In this section, we compare a few mobile augmented reality development frameworks and SDKs available today.

ARCore

Google's ARCore is the successor to the company's Project Tango, which has since been deprecated. Project Tango was highly effective in tracking users' locations indoors; It had the ability to accurately determine users' locations using depth-sensing cameras. ARCore has since picked up the torch from Project Tango, inferring depth with software alone. Today, ARCore is less accurate than Project Tango ever was.

One of the biggest challenges in current mass-market augmented reality solutions is in preventing assets from drifting in the environment. While ARCore is accurate in determining a user's location relative to it their surroundings, in relying on two-dimensional visual input alone, environmental lighting can cause Google's algorithm to lose track of a device's surroundings.

ARKit

Apple's ARKit is ARCore's main competitor in the mass-market augmented reality space. While it is similar to ARCore, ARKit is consistently shown to be even more prone to drifting. Because crowded rooms can be such volatile and random systems, it is critical that our three-dimensional path accurately represents the environment in which it is rendered. Because of this, in addition to the high entry-costs of developing for iOS, ARKit was eliminated as a viable option.

12.1.3. Development Stack

The Lace client was developed in C#. The app is rendered and compiled for Android devices with Unity3D and the Android SDK. Its augmented reality abilities are built atop Google's ARCore SDK, which supports indoor positioning, anchoring, and tracking.

Selecting a front-end development stack was largely dependent on choosing which programming language we wanted to learn. Preferably, this programming language would have the shallowest learning curve. Ensuring that this programming language would be easier to learn than others would help us to expedite development in the long run.

The C# language has wider community support than its rivals, such as C++, and is considered by many to be easier. Choosing C# also had a lot to do with personal preference. While we attended a course at SDU in which we learned C++, it was not immediately intuitive. Because of this, as well as the respective languages' associate stacks, we decided to try our hand at C#.

SDKs (Software Development Kits) such as ARCore or ARKit offer means of interfacing with APIs on popular platforms like iOS and Android. While each platform has its respective APIs, such as Sceneform, developing a prototype for both operating systems has become a problem in itself. Luckily, there are several widely supported solutions that introduce opportunities for cross-platform development.

When programming three-dimensional applications, one will have to choose a rendering platform. Video game engines have been the solution of choice for many years now, energized by the massive success of the gaming industry. Today, however, there are some new kids on the block that are making augmented reality development easier than ever before. We considered the following options:

Unity3D

To date, Unity3D remains the most universally supported development platform. Scripts in Unity3D are written in C#. In this case, C# is more widely supported in game development. This is due to the fact that Unity3D has been on the market for much longer. Unity3D is the most popular game development platform at the time of writing. This implied an increased ability to reach out to friends and ask for help with debugging if we decided that it was needed.

Unity3D is better for lightweight mobile development. This is particularly dependent on whether it is used for game development or something else. As Unity contains fewer built-in game frameworks, it has a tendency to use more resources during runtime than Unreal Engine. As our prototype would not feature any complex game mechanics, a lighter weight platform like Unity3D perfectly served our purposes [32].

Unreal Engine

Programming with Epic Games' Unreal Engine (www.unrealengine.com) is done using C++. Its primary focus is on blueprint-programming to make working with C++ easier and less prone to mistakes. The defining feature of blueprint-programming is visually drawing connections between nodes. This was a major consideration when choosing a 3D-development engine, as we were less accustomed to blueprint-programming than scripting. Unreal includes a lot of pre-programmed modules that offer specific game-related logic. This makes Unreal ideal for designing games such as high-performance first-person shooters and is not as wieldy for less traditional applications.

Still, there is little difference overall between Unreal Engine and Unity3D; the differences can often be entirely subjective. We plan to keep Unreal Engine on the backburner, as its more visual approach is enticing. It could be worthy of more time and effort later on when timing is not such a huge constraint.

React Native

The major drawback of React Native-based solutions is that React Native cannot run multi-threaded programs. This was especially important considering Lace was to run our pathfinding algorithm while making an HTTP request to the server. Because Lace was to be implemented in a crowded and dynamic environment, reducing latency was extremely important. Communicating with the server and generating the optimal path would have to run simultaneously in order to avoid any long delays. Such delays could occur, while the client waited for other methods to complete.

Viro Media's Viro AR (https://viromedia.com/viroar/) offers a promising cross-platform development solution for developing augmented reality apps. Viro AR integrates directly with the native process of both ARCore and ARKit and compiles for both platforms. This would have been the easiest point of entry for us, as apps using Viro AR are developed with the React Native platform. In addition to Viro AR, there are several other React Native solutions that make it easier for JavaScript developers to create apps for iOS and Android.

WebAR

WebAR likely represents the future of augmented reality development. To develop for WebAR, one could use Google's WebARonARCore or WebARonARKit frameworks, which integrate tightly with ARCore or ARKit, respectively. To develop for these frameworks, one uses the Three.js JavaScript framework. While developing the Lace client with JavaScript would have been a much more familiar and comfortable choice than with C#, WebAR is still widely untested and unsupported, requiring the user to install a dedicated browser. Google Chrome has already added in-browser ARCore support via experimental flags as part of their WebXR Device API [28]. However, as support is yet to be made official, WebARonARCore is highly susceptible to potential syntax changes and unforeseen bugs.

12.1.4. Indoor Positioning

The Lace client uses ARCore's built-in tracking ability to determine user positions. It was chosen, because of unworkable inaccuracies in alternative methods. In complex, rapidly changing environments, high-accuracy position tracking is needed to guide users. Anything could change in an instant, and every step counts. Indoor positioning was not a primary focus of the Lace client. This was because there exist several pre-packaged solutions that were good enough for our purposes. The considered solutions are listed below:

ARCore

ARCore was found to be the most consistently accurate indoor positioning solution that we tested. It tracks a user's position by anchoring a multitude of points to an environment. These points are anchored visually, meaning ARCore recognizes features —using a device's camera— in the user's environment and tracks the position of those features relative to the device's sensor data.

ARCore provides acceptable position tracking performance in low-density spaces, as it is consequently easier to "see" more features of the static environment. Crowds could potentially obscure ARCore's ability to perform ground plane detection, causing positioning with ARCore to be more unreliable.

While ARCore's tracking technique is primarily based on its ability to process visual data, it is limited in that it does not perceive depth. This makes it difficult for ARCore to determine the distance from devices to static objects in the environment, occasionally resulting in loss of tracking and/or orientation.

IndoorAtlas

IndoorAtlas (https://www.indooratlas.com) was our first choice for indoor positioning for a significant duration of the project. Unfortunately, none of the multiple tests that we ran with IndoorAtlas were successful. This was due to an apparent position-accuracy of roughly the same area of the room in which we were testing. This could be due to the construction of the building in which we ran the calibration tests. Because IndoorAtlas relies partially on magnetic readings in order to locate a mobile device, the steel beams of the building could have had a significant effect on its accuracy.

The main drawback of using a third party API such as IndoorAtlas is having to exchange data between Lace and that third party. Because preservation of user privacy is so important to this project, the consideration of IndoorAtlas was admittedly credited to a "fake it 'til you make it" mentality. Maybe it was for the best that it did not work so well.

FIND3

FIND3 [14] is the premium open source option for indoor positioning. Similar to IndoorAtlas, FIND3 boasts support for triangulation based on a multitude of data sources, such as Bluetooth, Wi-Fi, and magnetic fields. It also supports built-in passive scanning —something which previously required a separate server.

Despite allowing us to use a private server, calibrating a device running FIND3 was determined to be more time-consuming, with a more manual set-up process compared to IndoorAtlas. This extra hassle was something we did not want to focus on too much in the creation of the Lace client. After all, there were plenty of other things that could go wrong.

12.1.5. Pathfinding

The main purpose of the Lace client is to find and show a path to its users. The Lace client utilizes a weighted graph of crowd-density provided by the server. The path that the Lace client generates is determined through a pathfinding algorithm called A*, or "A-Star" [45]. The Lace client is effective only when the dimensions of a room are certain.

A* is based on Dijkstra's algorithm [2], which was purpose-designed for finding the shortest path between two nodes by creating and evaluating trees of possible connections to the end node. Dijkstra's algorithm does not depend on an organized grid of nodes in order to determine the shortest route. It is also much slower than A* because it evaluates so many paths before determining the shortest one. A* simplifies that process by pathfinding on a square grid of nodes, making it easier to keep track of which nodes it has already evaluated. Additionally, Lace server determines the density across a room according to a square grid.

This is not to say A* is the only option (nor is it the most efficient), especially considering the JPS+ pathfinding algorithm. We chose A* because it was efficient enough and supported weighting nodes —the most critical component. JPS+, for instance, has been shown to be over one hundred times faster than A* at finding the shortest path [35]. Unfortunately, JPS+ is so efficient, because it only draws a path at every decision point; its efficiency is entirely credited to its ability to skip evaluating as many nodes as possible. In a dynamic and complex environment with crowds of pedestrians, skipping any amount of nodes was certainly not an option.

The Algorithm

The Lace client runs a modified version of the A* algorithm which factors in the given crowd-density throughout the room. Instead of generating the shortest possible path from the

start position to a chosen target position, Lace attempts to generate the most efficient path. Specifically, the crowd-density in the room is made to affect the generated path by guiding pedestrians to avoid the most crowded areas. Whereas the standard A* algorithm generates scores for defined cells in a room based on their individual distance from the start point and the end point, the Lace client weights this score with the crowd-density values received from the server. This version of the A* algorithm is based on the excellent example from Sebastian Lague [23].

An x—y coordinate grid of nodes is drawn to represent a room which the Lace server is analyzing, with the origin being the upper-left node. Each node represents the corner of a cell. A cell is the shape of a perfect square with a width and height equal to the distance variable supplied by the server.

The A* algorithm relies on the creation of two lists of nodes to keep track of which of those nodes it should consider at every iteration. It runs until the current node is found to be equal to the target node, or destination. The open list contains all of the nodes that should be considered as part of the most efficient path for a node. The closed list contains all of the nodes which have been determined not to be part of the most efficient path.

The A* algorithm is contained within the 'AStar.cs' C# script. Firstly, classes are imported and local variables defined. This includes a two-dimensional array of integers. This array will come to represent the weights of all the nodes in the room. An object of type Coordinate is instantiated, representing the route's start position. Finally, we define a second coordinate which represents the route's end, or target, position.

When AStar.cs is enabled from Main.cs, the Start method will run. The Start method sets the local graph array equal to the graph array in the global response object. It initiates the global grid object and sets the start coordinates of the start node equal to those in startPos. Then, it sets the start coordinates of the start node equal to those in startPos.

A List of nodes is instantiated as 'openSet'. This set will include all of the nodes that the algorithm has yet to evaluate. A HashSet of nodes, 'closedSet', will include all of the nodes that the algorithm has successfully evaluated. The start node is added to the open set.

```
1 List<Node> openSet = new List<Node>();
2 HashSet<Node> closedSet = new HashSet<Node>();
3
4 openSet.Add(startNode);
```

The 'Node' class is used to describe a node in the coordinate space. First, we define some class variables: the node object's parent, its position in a GridObj, a boolean determining whether or not the node is traversable by foot, the node's g-cost, h-cost, and crowd-density weight.

All nodes are assigned a g-cost and an h-cost. The g-cost is the Pythagorean distance of the current node at coordinates *xc* and *yc*, from the user's starting position at coordinates *xg* and *yg*, which will always be equal to node's coordinate-position on the grid object. The h-cost is the Pythagorean distance from the current node to the target node at coordinates *xh* and *yh*. The real distance between can be disregarded in both cases because it remains a constant value.

$$gCost = \sqrt{(x_g - x_c)^2 + (y_g - y_c)^2}$$

$$hCost = \sqrt{(x_h - x_c)^2 + (y_h - y_c)^2}$$

In this class, we automatically create a property of the node's f-cost. The f-cost of a node is equal to the sum of the g-cost, the h-cost, and the

crowd-density weight at that node's coordinates. The f-cost is what is ultimately compared with other nodes' f-costs in order to help the algorithm choose the optimal route.

$$fCost = gCost + hCost + \omega_c$$

Also included in the Node class is an optional method to be run upon its instantiation. In this method, we set the node's walkability and position, as well as the crowd-density weight of the node equal to that of the corresponding position in the graph array of the API response.

```
public Node (bool _isWalkable, Coordinate _position) {

    isWalkable = _isWalkable;
    position = _position;
    density = (int)graph[position.X][position.Y];
}
```

Figure 26: The initialisation method of the Node

The algorithm loops through the following steps while there are still nodes in the open set to be evaluated:

1. Set the current node equal to the first node in the open set. For each node in the open set: Check if the f-cost of node 'i' in the open set is less than or equal to the f-cost of the current node. If the h-cost of node 'i' in the open set is less than the h-cost of the current node, sets the current node equal to the node 'i' in the open set

2. Remove the current node from the open set and add the current node to the closed set.

3. If the current node is equal to the target node, run the RetracePath method with arguments startNode and targetNode and sets the global boolean value, pathFound, to true.

4. Iterate through each neighboring node of the current node. If the neighboring node is either not walkable or has been added to the closed set before, continue to the next iteration of the loop. Create an updated cost and sets it equal to the g-cost of the current node plus the distance from the current node to that neighboring node. If this cost is less than the g-cost of the current neighboring node, or if the open set does not contain the current neighboring node, set the g-cost of the current neighboring node equal to the updated cost. Then set the h-cost of the current neighboring node equal to its distance from the target node, and the parent node of the current neighboring node equal to the current node. If the open set does not include the current neighboring node, add the current neighboring node to the open set.

Figure 27: Code snippet for the RetracePath method

```
static void RetracePath(Node startNode, Node targetNode) {

    List<Node> p = new List<Node>();
    Node currentNode = targetNode;

    while (currentNode != startNode) {

        p.Add(currentNode);
        currentNode = currentNode.parentNode;
    }

    p.Add(startNode);
    p.Reverse();
    Global.Instance.grid = new GridObj() {path = p};
}
```

The 'RetracePath' method retraces the path backward from the target node via its parent nodes, their respective parent nodes, and so forth. Firstly, this method creates and instantiates a new list of nodes. This will come to represent the final optimized path from the start node to the end node. Then, the current node is made equal to the target node.

While the current node is not equal to the start node, the current node is added to the optimized path, and the current node equal is made equal to its parent node.

```
public static double GetDistance(Node nodeA, Node nodeB) {

    return Math.Sqrt(
      Math.Pow(
        (nodeB.position.X - nodeA.position.X), 2) +
      Math.Pow(
        (nodeB.position.Y - nodeA.position.Y), 2)
    );
}
```

Figure 28: Method for calculating the distance between two Nodes

The start node is added to the optimized path. The order of the optimized path is reversed so that the first node is the start node. The path object of the global grid object is then made equal to the optimized path.

The 'GetDistance' method utilizes the Pythagorean theorem to calculate the distance between two node objects and returns the calculated distance.

The 'Coordinate' class consists of two int objects, X and Y. These represent a position on the node grid.

The 'GridObj' class represents a grid object containing all of the nodes in a space. Within this class, we first define the following variables: a two-dimensional grid of objects of type Node, a 2-dimensional array of integers, a list of Node objects to be populated with the final optimized path, the maximum number of nodes in the x and y-axes (width and height), and a boolean representing walkability, which defaults to true. The GridObj class automatically runs a method upon instantiation, which sets the data array equal to the server's graph array, the width, and height equal to the number of values in each column and row of the mentioned data array, and runs the CreateGrid method.

The CreateGrid method populates the grid object. Firstly, this method instantiates the grid object with the calculated width and height. Then, for each node in this grid object, we check if its crowd-density weight is equal to the predetermined highest possible density value. If so, the grid position's walkability is set to "false", indicating that its position is not traversable by foot. Then, we instantiate a new object of class Node at the same coordinates with its corresponding walkability.

The 'GetNeighbors' method determines the

Figure 29: Code snippet for the GetNeighbors method

```
public List<Node> GetNeighbors(Node node) {

    List<Node> neighbors = new List<Node>();

    for (int x = -1; x <= 1; x++) {
        for (int y = -1; y <= 1; y++) {
            if (x == 0 && y == 0) {
                continue;
            }
            int checkX = node.position.X + x;
            int checkY = node.position.Y + y;
            if (checkX >= 0 &&
                checkX < width &&
                checkY >= 0 &&
                checkY < height) {
                neighbors.Add(grid[checkX,checkY]);
            }
        }
    }

    return neighbors;
}
```

valid neighbors of a specified node. Specifically, we want to make sure that all neighboring nodes exist within the boundaries of the space. In this method, we create and instantiate a new list of nodes. Then, we iterate through each node in the specified node's adjacent nodes, also known as the Moore neighborhood with range, r = 1 [44]. Then we check if the current adjacent node is equal to the specified node before continuing onto the next iteration of the loop.

We set temporary x and y values equal to the sum of the specified node's x-y position and the current x and y values of the current adjacent node. If the temporary x or y values are greater than or equal to 0, or if the temporary x or y value is less than the maximum size of the grid, that object of class, Node, is added to the list of neighboring nodes.

After iterating through all of the nodes in the specified node's Moore neighborhood, the GetNeighbors method returns a list of valid neighboring nodes.

12.1.6. Project Structure

The client app was developed entirely within Unity3D. Several scripts were written in C# and attached to objects in the Unity3D editor. Attaching scripts to objects allows those scripts to be run at a defined time.

Main.cs

This script manages some of the most important tasks in the app. Main.cs is dedicated to running the A* algorithm, and calculating the remaining distance from the user to their destination at every frame.

Global.cs

To avoid confusion of passing variables between functions, we used global variables. Global variables that can be updated and accessed by any program are important when running multiple threads at the same time, as values would need to be updated in no particular order. After toiling with the task of implementing global variables across C# scripts, given the limitations and nature of Unity3D —attaching scripts to objects, enabling/disabling scripts, came across a novel solution called Singleton [41].

```
1 public class Global : Singleton<Global> {
2
3   protected Global () {}
4   public bool isGlobal = true;
5
6 }
```

Figure 30: Implementation of a Global Singleton

Implementing a Singleton in Unity3D enabled the prototype to have persistent access to data across all scenes. This was especially important when switching between two-dimensional and three-dimensional interfaces, which could be allocated to different scenes yet retain the need to access and/or modify data. An example would be allowing users to choose a destination.

Singletons have several caveats, however. Lacking polymorphism, for instance. Despite this, it worked perfectly for us novice C# programmers. Global.cs creates a class called Global of type Singleton. This allowed us to set variables within a global singleton object, consequently making them accessible to any other script in the project.

API.cs

API.cs sends a POST HTTPS request to the server and assigns the server's response to its respective global variable during the execution of a coroutine. A coroutine is an asynchronous

method that mimics multi-threading, in that it resumes a method with every frame update until the method is completed. That way, the rest of the program will be able to continue as API.cs waits to receive a response from the server.

```
public class API {

    public void Post(MonoBehaviour instance) {

      instance.StartCoroutine(PostRequest());
    }

    IEnumerator PostRequest() {
      // This is the coroutine
    }
}
```

Figure 31: Code snippet for making POST requests

In the POST request, the client sends a package containing request headers. These headers include the content type which the client is expected to receive, as well as an authentication token. This token ensures that the client posting the request is valid and thereby has permission to receive a response from the server.

As the data received is of type string, it has to be deserialized. What this means is that the string is converted into a JSON object. This allows other scripts to loop through the received data and perform actions on it accordingly. The received response [Section 14.2.4] includes a unique ID, the constant distance between each pair of collinear nodes (in meters), and a two-dimensional array describing the crowd-density weights at each node in the room.

DestinationSelector.cs

This script generates a list of GUI toggles in a two-dimensional canvas element. For the proof-of-concept, available destinations are hard-coded into the script. When a toggle is selected, the "Start Navigation" button becomes available. If the user taps this button, the GUI is hidden, and the user is shown the navigation view.

Populate.cs

This script serves to draw the path from the start coordinates to the end coordinates. It plots a waypoint for every valid path node calculated by the A* algorithm. These waypoints are grouped within a shared parent. In this case, we use a free script called CurvedLineRenderer.cs [10] to magically draw a smooth line between the instantiated waypoint objects.

Figure 32: Code snippet for the Awake method

```
void Awake () {

    destinations.Add(new Destination("Swing Set", 3, 7));
    destinations.Add(new Destination("Apartment", 12, 19));
    destinations.Add(new Destination("Bike Rack", 16, 11));
    destinations.Add(new Destination("Entryway", 15, 0));
    destinations.Add(new Destination("Garden", 0, 13));

    int numDestinations = destinations.Count;
    int yOffset = -120;

    foreach (Destination n in destinations) {

        Toggle newDestinationCard = Instantiate(destinationCard, Vector2.up * yOffset, Quaternion.identity);
        newDestinationCard.name = n.name;
        newDestinationCard.group = destinationList;

        Text newDestinationName = newDestinationCard.GetComponentInChildren(typeof(Text)) as Text;
        newDestinationName.text = n.name;

        newDestinationCard.transform.SetParent(destinationList.transform, false);

        yOffset -= 240;
    };
}
```

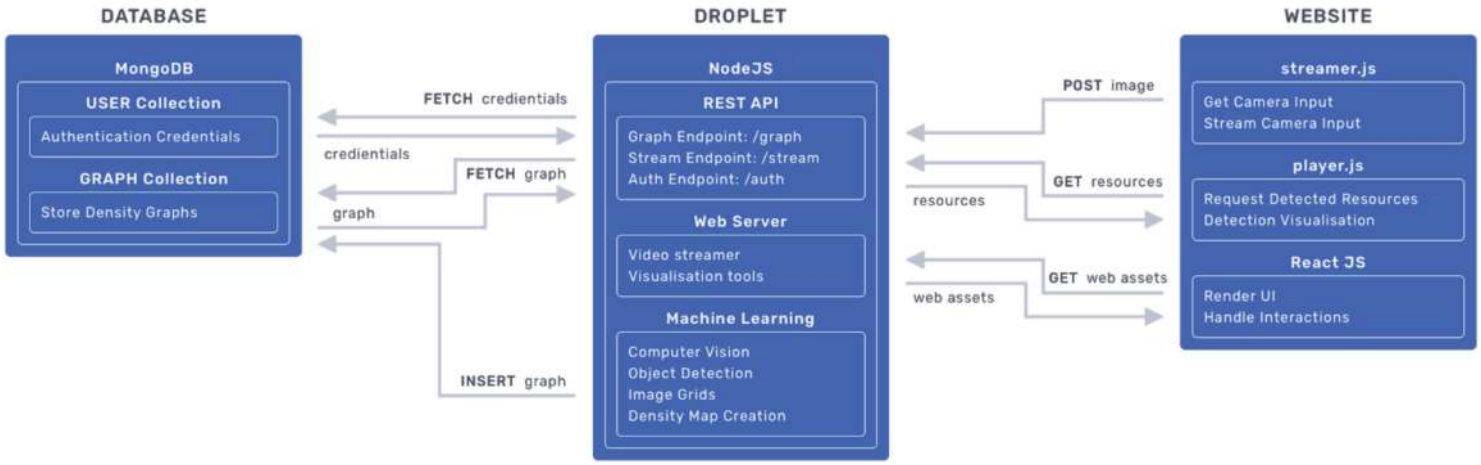


Figure 33: System architecture drawing for the "Server" system

12.2. Server

As mentioned in the introduction of Section 13.2, the prototype needed to understand the space in which it is operating as well as how people move through the space. This functionality would be handled by the system referred to as 'server'. The server system can essentially be split into three separate parts with the droplet as the intermediary between all the elements. A visual representation of this can be seen in Figure 33.

In this Section, we will dive deeper into the different parts of the server system, what they do, and how they were developed.

12.2.1. Server Droplet

The server droplet is the brain behind the server system. This is referring to a virtual system running a base form of Linux with access to virtual CPUs. Everything that needs to be processed will go through the droplet, and the droplet will direct the input to the relevant process. This results in a simple and elegant one-way communication stream between the client and the server. This is done to limit the number of entries into the server system which in turns adds to the performance and security of said system.

As the central processing is happening on the droplet, it is vital that this is kept as secure as possible. Intrusions into the droplet could have devastating consequences for the overall system. The droplet is handling communication with the user's devices. Gaining access to the droplet could result in access to the user's private information. This is why the communication stream has been fully encrypted using SSL [46] certificates which have a key pair: a public and a private key. These keys work together to establish an encrypted connection which ensures that the connection is secure. All the information being communicated is encrypted guarding the system against potential man-in-the-middle attacks which could reveal confidential user information.

To further ensure the safety of the system, each

client trying to request assets from the server will need a 512 bits unique, random string of symbols as part of the request. This random string of symbols is referred to as an auth token. The auth token will be generated using a custom built, hidden function that only the server and client application have access to. In making a request, the server will first check if an auth token has been sent through and if it is matching the pattern. The request will be allowed only if the auth token is considered valid. This ensures that only the client application will be able to request assets from the server, eliminating intrusive 3rd party access.

To ensure that the system does not fall to No-SQL Injections, extensive checks were developed for the header parameters in the communication stream. No-SQL injections are similar in nature to the more "well-known" SQL injection, which is a technique used to inject malicious code into a data-driven application which can tamper with the stored data. This will usually happen by sending executable code snippets through some sort of text input on the client side.

The verification checks look at every header parameter and determine if the data matches what was expected. If it does not match, the request will be denied.

Finally, to ensure that only authorized personnel can access and make changes to the droplet itself, the SSH keys of the authorized personnel have been added to the droplet, allowing only these keys access. SSH [41] (Secure Shell) is a cryptographic network protocol used for operating network services, matching a public key added to the droplet with a private key on the user's development machine.

A NodeJS development environment is run on top of the droplet. Because we had more experience working with Node, the environment allowed for rapid prototyping and development. Node is a stable and extensible development environment. It allowed us to quickly set up and develop the web and API server. It also provided plenty of flexibility in running separate operations on multiple CPU threads, known as child processes [6].

Node drastically improves the scalability as it changed the traditional request-response paradigm that is still governing the web. Instead of created a non-blocking, event-driven input/output (I/O) stream to run real-time applications across devices. By employing event-driven development, Node gathers all the requests on a single thread, allowing it a theoretical max of over 1 million concurrent connections. [1] As we have a constant stream of images coming in from the cameras, and a constant request of information from the client, this will be crucial to ensure not overloading the server droplet.

As can be seen in Figure 13.2.1 the Node environment encapsulates everything that is running on the droplet, the first thing being the REST API. Technically, API stands for Application Programming Interface and is there for the server droplet to serve the client. When thinking about the web, one can imagine a large network of interconnected, remote server droplets. Every website or asset is stored on a remote server droplet somewhere, and the way you gain access to said assets, are by interacting with the droplets API [36].

There are many different ways of designing an API. When choosing an architectural style, scalability was once again the predominant decision maker. We chose to implement a RESTful API. REST is an abbreviation of Representational State Transfer, which is essentially an "architectural style for distributed hypermedia systems" [13] defined in Roy T. Fielding's dissertation on Architectural Styles and the Design of Network-based Software Architectures [13].

REST proposes a set of guiding constraints which must be satisfied in order for an interface to be RESTful. The first constraint concerns the

client-server architectural style. When we delegate the concerns of the user interface to the client, we are freeing up the server to focus on data storage concerns, improving the scalability of the system [13]. The second constraint is to the interaction between the client and server. Each request from the client must contain all of the relevant information that the server needs to understand said request [13]. No state or context is stored on the server. This improves the scalability of the system by not having to store the state between each request, allowing the server to quickly free resources. Mr. Fielding's claims in his dissertation that this is also the biggest design trade-off [13]. This tradeoff is rooted in the increase in repetitive data sent in each request, which could decrease the network performance. 5G connections are expected in the first half of 2019 [34], with 4G being already widespread. The downside could be negated when weighing the benefits of scalability and latency against the potential decrease in network performance.

The goal of the REST API is to act as the intermediary between the clients requesting assets and the droplet executing that functionality. The droplet has no knowledge about surrounding clients [Figure 34] and interfaces only with the API. In turn, the API handles all client requests and directs the verified data to the droplet for processing. Once finished with its operations on the data, the droplet will return documentation to the API. This documentation will be returned to the client. This process occurs when the client makes a request for a density graph of the environment that they are currently in. Contacting a specific endpoint of the API will command the droplet to fetch the density graph from storage and return it. A more detailed explanation on how said density graph is constructed will be provided in Section 12.2.4.

The Lace website (www.lace.guide) can turn any capable device into our camera. It posts an image stream to the server by connecting to a different endpoint on the API. The API will then forward the image to the droplet, telling it to perform the operations needed to create a density graph from the image. This, like the requests from the client, happens concurrently and in close to real-time if negating the latency. Per Figure 33, there is a streamer and a player

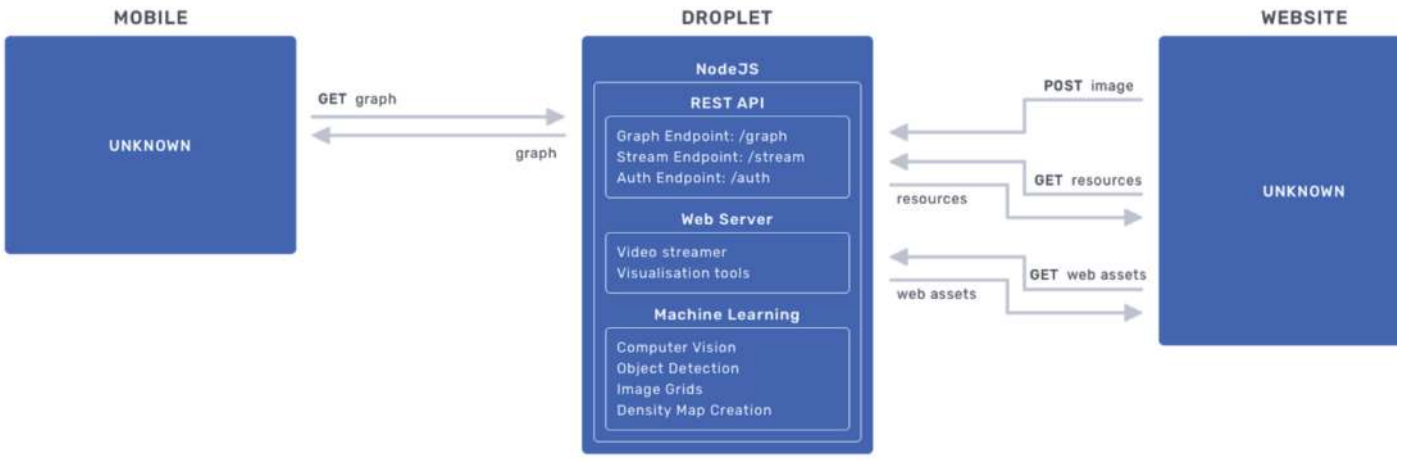


Figure 34: Illustration of the API connections and its knowledge about clients

component. The streamer component harbors the camera functionality. To keep prototyping costs down, the camera functionality was developed into a website which could then be accessed from any camera phone. The streamer requests access to use the camera of the phone and then streams images of what it sees to the droplet by hitting a dedicated API endpoint.

The player component of the website can be accessed to view what the camera is seeing. It can also draw a graphical representation of object detection and creation of the density graph on the droplet for demonstration purposes. Because the droplet is purely a command line interface with no graphical user interface, it is difficult to verify the correctness of the operations being performed.

The API takes care of storing generated resources, specifically the density graph, in a database so that it is ready when the client requests it. When it comes to databases, there are two main philosophies; relational and non-relational database structures. Relational databases are how things were traditionally handled using technologies such as MySQL and PostgreSQL. Relational databases have reliably served data applications for many years, and offer a lot of features that remain critical today. This include features like an expressive query language, known as SQL, and secondary indexes. This allows users to access and manipulate data in ways that support operational and analytical applications. As the technology is mature, applications written in a relational way becomes increasingly complex when building around an evolving data structure, known as a model. This imposes significant work if the data structure will ever have to change.

Modern applications impose new requirements, which are not handled by relational databases. This is where non-relational database come into play. Non-relational databases offer a flexible data model which is an invaluable feature when working with data-driven applications. Whether your data consists of documents, data-graphs, etc. non-relational databases offers a way to combine data of any structure without modifications of the schema. This leads to increased scalability and performance, as databases can easily scale out enabling lower latencies than relational databases. Common non-relational databases, like MongoDB and Redis, are developed for these exact purposes. Specifically, we chose MongoDB due to our prior experience dealing with this way of defining database models and schemas for operating on it. This allowed for rapid prototyping and iteration.

We wanted to rapidly iterate on the data structure during the prototyping and development phase, while simultaneously providing increased performance in production environments. Because of this, non-relational databases were a clear choice for the purpose of this project.

Storing the density graph in a database allows the droplet to free up the resources held by that graph and direct those resources to other operations. As the system is running object detection, which is a pretty expensive operation, we wanted to free up as many resources for this operation.

Generating the density graph is an expensive operation in terms of resource utilization, as it is running machine learning, computer vision, object detection, and many more algebraic operations under the hood. All of these operations could end up blocking the thread running on the central processing unit (CPU). This results in the client not be able to contact the system through the API before all the operations are complete. Although the duration of operations could be as low as a second or two, the generation of density graphs occurs in close sequence, leaving little time for other operations. This is where NodeJS Child Processes [6] comes into play.

Single-threaded performance in NodeJS is great for single processes, though running our

system, one process in one CPU is not going to handle the combined workload of the operations. While Node runs on a single thread, we could still take advantage of multi processes using the child processes module. The child processes module allows Node to spawn a child process which enables access to the operating system (OS) functionalities by executing system commands inside a child process. Node and the child process then communicate using the standard input/output streams of the OS. This allows the system to spawn a child process which runs in parallel with that same system. This parallel process will handle the expensive operations involved in creating the density graph, while the main process is completely free to handle other tasks. Once the child process is done with its operations, it will send the output back to the main process and store it in the database, ready for the client to use.

12.2.2. Machine Learning

Machine Learning essentially gives computers the ability to "learn"; they can progressively improve their performance on specific tasks based on data input. They are also not explicitly programmed to perform a specific task. Machine learning is a skill largely comprised of basic statistics, probability theory, and calculus. Implementing effective machine learning generally requires a thorough understanding of these mathematical concepts, rather than strong algorithmic skill. In Lace, machine learning is implemented to achieve computer vision and object recognition. This approach helps Lace understand the crowd density in a space. Tom Mitchell [28] asserts that a machine learning system will be able to use data to perform some task. After having computed said task, the system will have gained some experience on how to perform it. Next time the system receives data, it will use the experience gained from previous tasks to guide its process. This allows us to not specifically program a solution for every single case, but rather let the system create an inference on the provided dataset.

The child process being executed by Node is programmed using Python. [33] Python has been gaining a lot of attraction with developers after technologies which rely heavily on advanced mathematical operations have become part of mainstream products. The reason for this is not that python is better at performing such mathematical operations, but because Python has been adopted as the go-to language by scientist and researches. This made Python the clear favorite, as plenty of Python-based frameworks on which we could build, had already been developed.

Since we are working with computer vision for object detection and recognition, Python has a clear advantage in the number of supporting packages that exist for the system. An especially essential package is for operating on and manipulating images. One of the downsides, however, is that with the number of advanced and encompassing packages, a large amount of overhead is added. This means that it requires more processing power than most other languages. While this would be quite a substantial downside had this computation happened on-device, handling all of these processes on a server proves less of an issue. Since cloud servers have increasingly high processing power for increasingly lower prices, Python provides a strong foundation to which one can build their technology.

With the increase in demand for machine learning technologies, Google developed their first edition of a highly specialized processing unit for exactly this purpose. The Tensor Processing Unit [38] (TPU) is a processing unit developed specifically for neural networks and machine learning using Google's TensorFlow framework. What sets Google's Solution apart from the standard Graphical Processing Units (GPU) is that they lack any sort of hardware for rasterization and texture mapping which would be required from a general-purpose GPU. While they are not yet commercially available, it will most likely become the standard as technology

is moving forward.

We are using TensorFlow is because it is open-source. This allows us to customize it and use it however we please by compiling from source with our own modifications and optimizations.

neuron to learn from previous experience, each incoming signal will have a weight which will programmatically be adapted over time. A visualization of this can be seen in Figure 35 below.

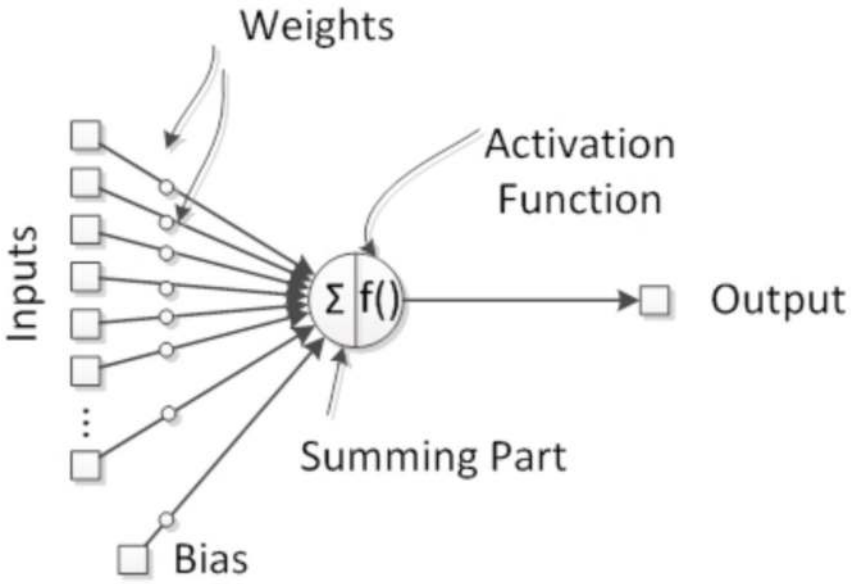


Figure 35: Visualization of an Artificial Neuron

TensorFlow is built upon the concepts of tensors, which are essentially multi-dimensional arrays. By representing images as arrays of bits, we can utilize TensorFlow to increase the performance of detecting objects in these image arrays. TensorFlow uses an artificial neural network to achieve machine learning.

An artificial neural network is an intelligent system modeled after the human brain. It emulates an artificial neuron based on the neuroscience foundations with some mathematical operations. A natural neuron structure is known to have a nucleus which receives information from dendrites which are receiving incoming signals from neurons. Finally, an axon activates a signal to other neurons. Modeling this structure as an artificial neuron, one can create an artificial neuron unit, which sums all the incoming signals, and activates a new signal to other neurons. For the

Each of these neurons is then placed in an interconnected network which is comprised of layers. Each network contains an input layer which receives signals from outside the network, as well as an output layer firing some action that will have an influence on functions outside the network. It can then have an arbitrary number of hidden layers, between the input- and output layers. The more layers a neural network has, the more complex knowledge it can represent. It essentially allows us to model complex relationships between all the inputs and outputs to find patterns in data which comes in handy for the object recognition.

It is possible to get the pre-compiled binary for TensorFlow, making getting started with the tool quick and easy. The pre-compiled binary is compiled to support as wide a number of CPUs as possible, resulting in specialized instruction sets being excluded. The specialized instruction

```
with detection_graph.as_default():
    with tf.Session(graph=detection_graph) as sess:
        # Definite input and output Tensors for detection_graph
        image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
        # Each box represents a part of the image where a particular object was detec
        detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
        # Each score represent the level of confidence for each object
        detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
        # The classification of the detected object
        detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
        num_detections = detection_graph.get_tensor_by_name('num_detections:0')

        # Actual detection.
        (boxes, scores, classes, num) = sess.run(
            [detection_boxes, detection_scores, detection_classes, num_detections],
            feed_dict={image_tensor: numpy_image_array_expanded}
        )
```

Figure 36: Code snippet of the Python routine handling object detection

sets available to you depends on the specific hardware. In the case of Lace's droplet, the hardware running it has access to the AVX2, AVX512F, and FMA extended instructions sets. These instruction sets contain optimized instructions for processing matrix or vector operations. Making a custom compilation of TensorFlow, which secures access to these instructions sets, could give between 35-50% general speed increase on this particular setup. This made it worth investing our time.

12.2.3. Object Recognition

The density graph is intended as a guide for the client to determine which path to take through a space. It is based on the crowd density. Hence, the system had to understand when something is an animate or inanimate object. Areas crowded by pedestrians is be represented as high density, whereas areas containing static objects, such as tables, will be represented as lower density. This is where the system needs to understand the object in the space. Suggesting the client move through a table might not work out so well.

Object recognition is a subset of the research and technology field of computer vision. Recognizing a visual object is a rather trivial task for humans to perform, while computers are not so good at it. Object recognition uses what is known as an image classifier to identify if an object belongs to a certain category. The image classifier's task is to assign an image one label from a predefined set of categories. It does so by taking a single image in which it detects and infers the type of visual objects present using a floating confidence rating between 0 and 1. Depending on how the classifier is built, it can assign different labels to the image, each including a rating of confidence that the label corresponds to the object in the image.

Current detection systems use image classifiers to perform detection on an image. To detect an object, image classifiers utilize an image classifier for said object and evaluate it against various locations in a test image. These image classifiers then utilize what is known as the 'sliding window' approach where the classifier is run at evenly spaced locations over the entire image. For a computer to make any sort of operations on an image, the image has to be represented by a set of values that the computer can work with. Converting an image into a large three-dimensional matrix of values. The values in the matrix are the red, green, and blue channels in an image used to represent the color of each pixel. If we are running detection on a 480 by 480-pixel image, with each pixel having three channels, we get an array

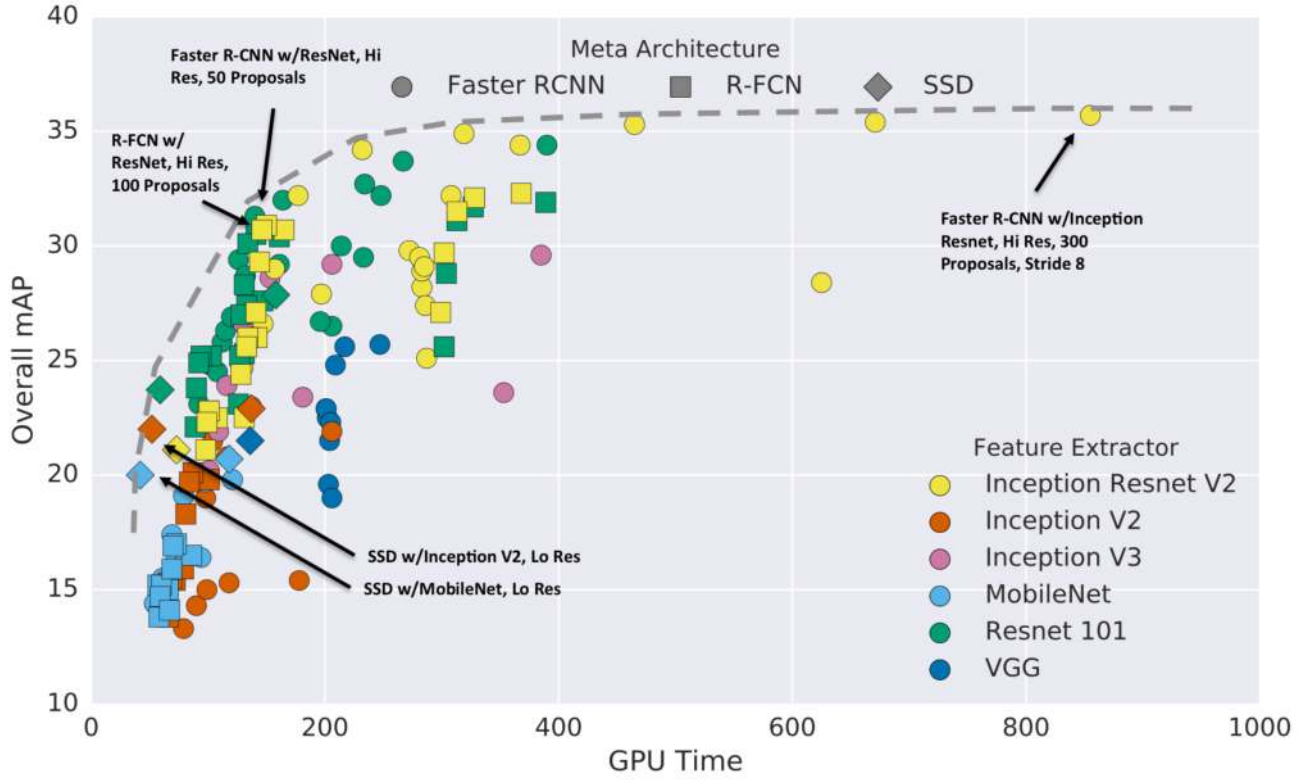


Figure 37: Time vs. Accuracy evaluation of the meta-architectures [22]

representation with a total of 691.200 values. Each specific value ranges between 0 (black) and 255 (white). The system has to turn over half a million of these values into a classification label. This is done using a data-driven approach, where we provide the computer with many examples of each class of object it is supposed to detect. This is where machine learning comes into play. Based on the many different examples of objects, the system will be able to learn numerical representations of visual objects in images, which it could use to classify further objects.

There are different types of architectures used to build these object detection systems. Although a common factor is that they all implement a convolutional neural network, such as the ones explained in earlier. Some of the most popular architectures, also referred to as the meta-architectures, include SSD (Single Shot MultiBox Detector) [24], R-CNN (Region-based Convolutional Network), Faster R-CNN, and R-FCN (Region-based Fully Convolutional Network). While the Region-based architectures are similar in structure by incorporating both a feature extractor, a proposal generator, and a box classifier, the SSD architecture combines the work done by the proposal generator and box classifier into a single step, lowering the number of times the system has to look at the image to one, increasing speed of detection. As the system needs to keep up with walking pedestrians, we need to get as close as possible to real-time detections.

There is a dramatic tradeoff when one detection system is required to run quickly. The accuracy of the detections. When wanting to increase the frame rate at which the system can run, one must be willing to sacrifice the mean average precision (mAP). The speed depends not only on the architecture but also on the feature extractor used with it. When deciding on which architecture and feature extractor to use, we can refer to Figure 37, which compares time vs accuracy on different combinations of architecture and feature extractors.

The fastest feature extractor is MobileNet [20], placing under a 100 GPU time. The overall fastest architecture is the SSD which all comes in under 180 GPU time. One radically different

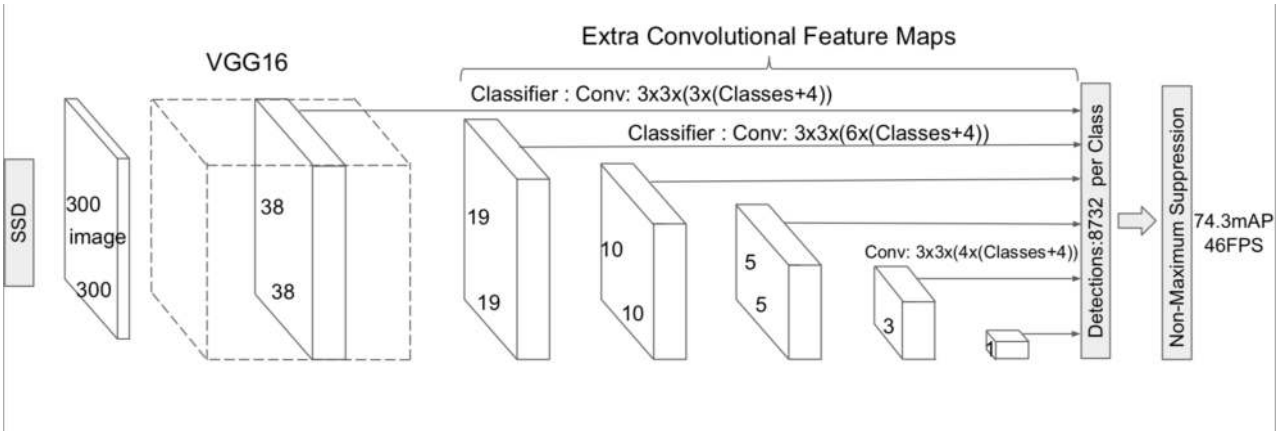


Figure 38: The SSD Model Architecture using the VGG16 Convolutional Neural Network [24]

approach, which is not considered part of the meta architectures is the YOLO (You Only Look Once) object detection architecture. While it is not mentioned in Figure 37 we tested it out. It proved less accurate than the SSD and MobileNet combination.

The SSD architecture, "it is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes". [24] The exact SSD architecture can be seen in Figure 38 above. Rather than using the VGG16 CNN for feature extraction, we would exchange this part of the SSD Model with the MobileNet CNN which has been proved to achieve VGG16 level accuracy using only 1/30th of the computational power. This makes the object recognition system incredibly fast, and still accurate enough for our purpose. The layers seen under the "Extra Convolutional Feature Maps" are the neural network layers implementing the machine learning and evaluating the image classifiers data.

The job of feature extractors is to reduce the amount of computational power needed to process an image without losing important or relevant information. Running through an image using a pool layer to reduce the dimensionality of the input image. A pool layer is used to create a summarization of a particular image patch. The summarization operation could be using the average, minimum, median or any other statistics summarization operation. From investigation, the max operation seems to be the most common. Employing a 2x2 max pooling layer [Figure 39] would take a 300 by 300-pixel image down to a 150 by 150-pixel image lowering the number of pixels to run detection on by half.

After reducing the number of dimensionalities in the input image to a satisfactory level where the number of actual pixels remaining is as small as it gets without losing any of the features from the original input image, the system will evaluate the image classifier against the remaining pixels. This is what can be seen as the 'Conv' layers under "Extra Convolutional Feature Maps" in Figure 38 above. The image classifier will use what is called a k-Nearest Neighbor Classifier algorithm to find the top k closest images in a training set and use them to evaluate which is the most similar to the input image. One of the most common ways the algorithm determines similarity is by using a pixel-wise difference to compare the distance between the images, essentially comparing the images pixel by pixel and adding up the differences. This can be done using the *L1* distance formula:

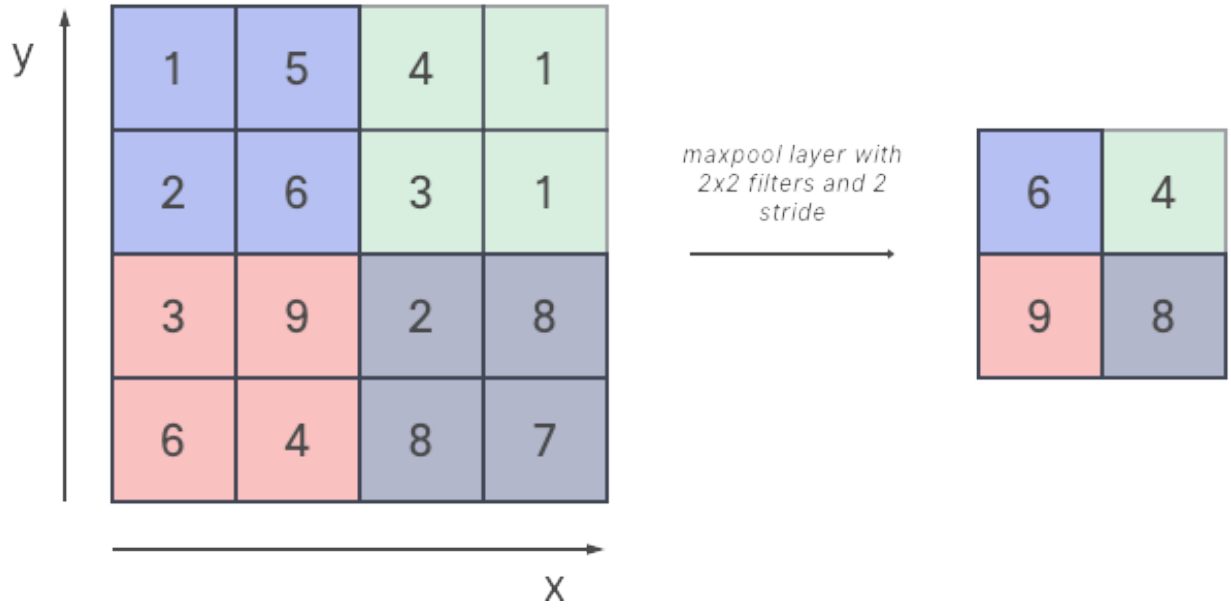


Figure 39: Visual representation of input and output from a 2x2 maxpool

$$d(I_1 I_2) = \sum_p |I_1^p - I_2^p|$$

where *I1*, *I2* are the images represented as vectors.

If the summarized distance between the two image vectors is 0, the two images are identical. The higher the value of *d(I1I2)*, the more different the image are and the lower the confidence rating will be. The lower the value, the higher the confidence. A visualization of how this would look can be seen in Figure 40 below.

The two images are subtracted at each element, [Figure 40] and all of the differences are added up to a single number. The example is not representing an entire image, rather a single color channel, remembering that the entire image is represented by a 3-dimensional matrix. The distance between the two vectors is high, d(I1Id)=1039 which means that the similarity between the two is low, giving a low confidence rating.

The system has now understood what object is present in the image. It now needed to draw what is referred to as a bounding box around the object. This bounding box is what will be used to evaluate the density against each point

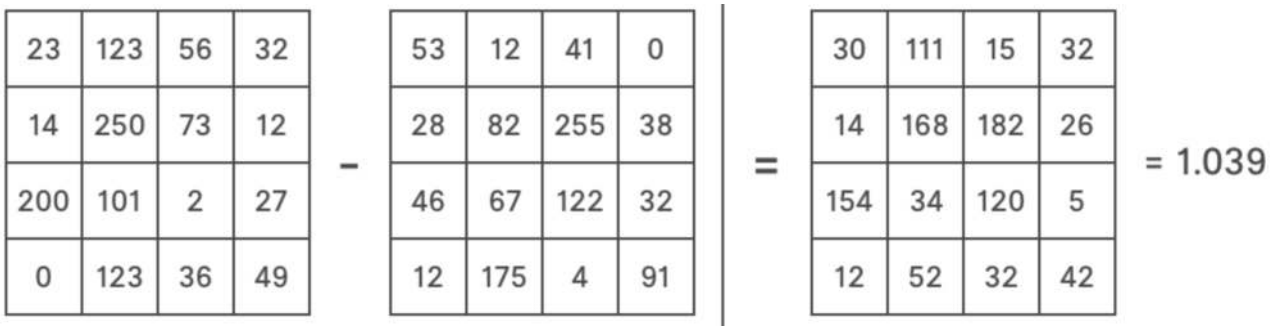


Figure 40: Calculating similarities between two vectorized images. Here done on a single color channel

in the observed space. Determining the position and size of the bounding box is what is know as object localization. Object localization will start by finding the center point of the object. Based on the center point, it can create an outline for which a bounding box will be created around.

The system now knows what object is present in the space, and how much room it takes up, which we used to construct the density grid.

12.2.4. Density Grid

The density grid is the final data structure that will be served to the client. It is a grid representing the room with an integer value representing how dense each grid box is.

The system starts by creating a grid with a range depending on the distance from the camera to the floor of the space. The longer the distance from the camera to the floor, the finer the grid will be. It is important that the grid is of a fine enough resolution to be able to distinguish different elements within them, as well as a not being so fine that the number of data points becomes large. The finer the grid, the more data point, which in turns requires increased computational power. The more computational power required, the bigger the chance for bottlenecks on the client.

There is some debate about how to define the cell size, or the constant space between each collinear pair of nodes, in order to make the "best" crowd observations. While Davidlich suggests spacing nodes every 0.53m (to suit a single Caucasian male) [8], Feliciani suggests using a spacing of 0.2m [12]. Of course, the size of the cells affects the optimal sampling frequency; The sampling frequency depends on how many cells a pedestrian is able to cross in a short amount of time, as well as our need to provide accurate and current readings to users. Feliciani found that the respective optimal sampling frequency was once every 2.5 seconds, although their specific reasoning is unclear. The typically preferred velocity of individual pedestrians has been determined to be 1.34m/s [22]. This places a strict limitation on the amount of time it takes to detect pedestrians' locations and calculate paths for Lace users. This time limitation underlines the importance of implementing prediction algorithms in the future.

The crowd-density grid of square cells, versus triangular or hexagonal alternatives, was chosen because we determined the format to be useful for developing a proof of concept. Using square cells is the most common practice in pathfinding algorithms (Davidich determined the diameter of cells —in this case, hexagonal— to be 0.53m [8]).

The specific grid size has been decided based on a number of tests and a formula was drafted for the system to automatically create the grid based on its distance to the floor.

$$G_{size} = \lfloor 2^{(\frac{d}{100})} * log\,(d) \rfloor$$

where G is the graph and d is the distance from the camera to floor in centimeters.

We modeled that an exponential function to the base 2 multiplying a logarithmic exponent of the camera, to make it grow a little more rapidly with larger values of d. The system has determined how fine the grid should be, it will normalize the pixels of the image into a [0;1] Cartesian coordinate system. This allows the system to perform calculations independent of the pixel ratio of the image which could vary depending on the distance from the camera to the floor.

Now that the grid size has been determined, the system can create grid boxes according to the calculated size. Firstly, the system will create Gsize evenly spaced grid lines within the normalized coordinate interval. The number of grid boxes corresponds to the formula (Gsize-1)2. This operation will happen on both the x- and y-axis.

From these grid lines, structured as two lists, the system can create a new list of coordinate points for each grid box. As a simple example, let us imagine Gsize=3. This results in there being 3 grid lines in the normalised coordinate interval: [0, 0.5, 1]. Because the image stream is of equal width and height, this is representative of both axis. With these 3 grid lines we will get a grid with 4 grid boxes. These boxes would have the following coordinates starting from the top left and going clockwise, (0, 0) being in the top-left quadrant:

$$box_{bottom\ left} : [(0,0), (0.5, 0), (0.5, 0.5), (0, 0.5)]$$
$$box_{top\ right} : [(0.5,0), (1, 0), (1, 0.5), (0.5, 0.5)]$$
$$box_{bottom\ right} : [(0,0.5), (0.5, 0.5), (0.5, 1), (0, 1)]$$
$$box_{bottom\ left} : [(0.5,0.5), (1, 0.5), (1, 1), (0.5, 1)]$$

We were actually able to craft these boxes by just knowing the top-right and bottom-left coordinate which would translate to the following:

$$box_{bottom\ left} : [(0.5, 0), (0, 0.5)]$$
$$box_{top\ right} : [(1, 0), (0.5, 0.5)]$$
$$box_{bottom\ right} : [(0.5, 0.5), (0, 1)]$$
$$box_{bottom\ left} : [(1, 0.5), (0.5, 1)]$$

This allowed us to halve the number of data points which each algorithm have to run through. Imagining that an operation takes 0.01 time-units for each data input. In a system where the camera is 4 meters from the floor, Gsize= ⌊2(4)* log(400)⌋ = 95 we would have (95 -1) * 2 = 8.836 grid boxes. As each grid box is a list of 4 floating point values, performing the imagined operation on each of them would take a total of (8836 *4) * 0.01 = 353 time-units. When only using two coordinates sets, the total time to complete the operation would be (8836 *2) * 0.01= 178 time-units.

With coordinates for the grid boxes and the coordinates from the bounding boxes, the system can use these coordinate sets, to calculate the area of overlap between a bounding box and a grid box. This area value is what will be transformed into the density. The higher the area of overlap between bounding boxes and grid boxes, the more objects have been detected there and greater density. This means the density values represent how dense the area is between a set of grid box coordinates.

Before the system calculates the area of overlap, it checks the classification of the bounding box, eg. is it a person or a table? If it is not a person or another animate object, it will add a null value to the density. The null value refers to the certain grid being occupied by some object, impossible to pass through. If the object is a person, it will calculate the area of overlap.

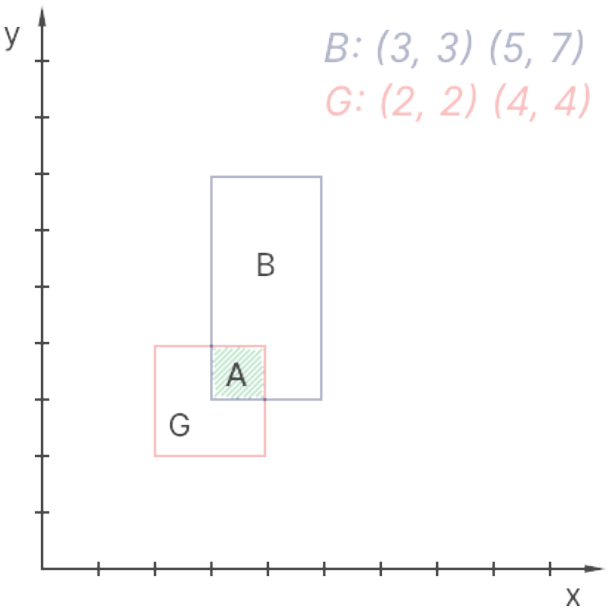


Figure 41: Visualization of the Area problem given two opposing coordinate sets for each rectangle

Visualizing the problem of calculating the area of overlap between two rectangles, while only knowing the bottom-left and top-right coordinate set will help solve it. [Figure 41]

It is important to note that the solution would have to quickly determine if the boxes do not overlap at all and just return, to not create an unnecessary number of computations or worse,

throw an exception causing the execution to halt. Firstly, it will evaluate the distance of overlap on the horizontal axis. If no overlap is present, it will return from the function, allowing the next cycle to start. Looking at the visualization in [Figure 41] the distance of

$$max(B_{1.x}, G_{1.x}) = max(3, 2) = 3$$

overlap on the horizontal x-axis is the distance between B1.x and G2.x.

$$min(B_{2.x}, G_{2.x}) = min(5, 4) = 4$$

Lace first finds the leftmost point which can then be excluded. As the leftmost point is G1 we want to exclude it, and use B1as there is no

$$x_{distance} = min(B_{2.x}, G_{2.x}) - max(B_{1.x}, G_{1.x})$$
$$x_{distance} = min(5, 4) - max(3, 2) = 1$$

overlap between the two:

$$y_{distance} = min(B_{2.y}, G_{2.y}) - max(B_{1.y}, G_{1.y})$$
$$y_{distance} = min(7, 4) - max(3, 2) = 1$$

To find the rightmost point on the horizontal axis and excluding it, one gets the minimum value of the two like so:

Having found these two points, we can find the distance by subtracting the leftmost point from

$$A = (min(B_{2.y}, G_{2.y}) - max(B_{1.y}, G_{1.y})) * (min(B_{2.x}, G_{2.x}) - max(B_{1.x}, G_{1.x}))$$

the rightmost one, making the formula:

$$A = (min(7, 4) - max(3, 2)) * (min(5, 4) - max(3, 2))$$
$$A = 1$$

Finding the distance on the vertical y-axis; by finding the top and bottom most points:

Combining the formula for x- and y-distance and remembering the formula for the area of a rectangle, A = width*height one can calculate the area of overlap simply by multiplying the two. The final overlap formula then looks like so:

This gives us an area of overlap:

This was translated into a Python function [Figure 42]

The system needed to understand what those values represented. As the area of overlap could fluctuate quite a lot in range, we transformed the values into a more understandable and workable range of density values. To allow the client receiving a density map of a space, it must have some context attached. If we just used the area of overlap, the range of values could be different each time the client gets it. By representing them with a consistent range, it makes it easier for the client to perform the necessary operations, specifically the A* algorithm mentioned in Section 12.1.5.

The density range was defined as a range of 5 different integer values. The first value, 0, describes a grid box which is completely empty, as in there is no overlap between it and the bounding boxes of detected objects. This allows the client to quickly understand where it can go with least resistance. The next three values represent three different levels of density within the grid boxes: 1 is light density, 2 is medium density, and 3 is high density. The actual number of people said densities refer to will however vary based on the size of the grid boxes and how much overlap there is in other grid boxes. As the range of area values are transformed into the density range, what the numbers represent are relative to the highest and lowest area of overlap. This was done to make sure that the density value of a single grid is always relative to the rest of the space. The last value, 4, is what was earlier in this section referred to as the null value. This value, as earlier described, represents a grid box, for which it is impossible to pass through as it is overlapping with an inanimate object.

These ranges contain two extremes, 0 which is entirely free of obstacles, while 4 is impossible to pass through. This makes sense when the client is looking for a path through the densities, as it can be taught to immediately stay

```python
def calculate_overlay_areas(grid_boxes, bounding_boxes, classes):
    '''Calculate the overlay areas between all grid- and bounding-boxes

    Args:
        grid_boxes: A [size*[size]] tuple generated by the create_grid_boxes_array
        bounding_boxes: A [n*[n*[4]]] tuple generated by the object_detection library

    Returns:
        Returns a [n] tuple of overlap areas
    '''
    # Instasiate the overlay areas graph
    overlay_areas_graph = []

    # Calculate overlay area of all grid boxes and store in new array (loop)
    for grid_box in grid_boxes:
        temp_overlay_area = 0.0
        for index, bounding_box in enumerate(bounding_boxes):
            overlay_area =
                find_grid_box_and_bounding_box_overlay(bounding_box, grid_box)

            # If the is no overlay area append a zero, else append overlay area
            if overlay_area == 'no-overlap':
                temp_overlay_area += 0.0
            elif overlay_area == 'null':
                temp_overlay_area = -1
            else:
                temp_overlay_area += overlay_area

        overlay_areas_graph.append(temp_overlay_area)

    # Return new array with overlay areas
    return overlay_areas_graph
```

Figure 42: Code implementation of calculating the area of overlap between grid- and bounding boxes

away from grid boxes with a density value of 4, while it can easily move through grid boxes with a value of 0. The three actual density values were chosen to make sure there was enough variation in distinguishing between the different densities, while not blending together.

To preserve the operations of addition and scalar multiplication in the different overlap area value, we employed a linear transformation formula to transform the range of overlap area values into the range of density values. This allowed us to transform one linear subspace onto another of lower dimension. The mathematical formula for this would be:

$$y = (x - a)\frac{d-c}{b-a} + c$$

where x is a number in a range [a; b] that one wants to transform to the number y in a range [c; d]. Let's say the lowest recorded overlap area value is 2, with the highest being 23, giving the

range [2;23]. The system would transform this range of numbers into the density range being [1;3]. The reason the system excludes the extremes, 0 and 4, is that these are constant, and should not be relative to the rest of the space. Either the grid box is free, or it is not. Likewise, either it is impossible to walk through or it is not. If the system was looking at a grid box with an area value of 5, the transformed density value would be:

$$y = (5-2)\frac{3-1}{23-2} + 1$$
$$y = 1$$

The value of y was rounded to the nearest integer that first in the range.

As the system now has a single array with all the density values, it has to adapt them to the data structure expected by the client. The number of values in each row array will be the same as the total number of rows, creating an n-by-n matrix. If there are a total of 16 values in the density array, the number of rows will be equal to the square root of 16, equalling 4-row arrays, each with 4 density values. This will require two concurrent loops, and the specific Python implementation can be seen below. After the system has adapted the density grid to the expected data structure, it serializes into a JSON (JavaScript Object Notation) object which will be passed back to the calling code - the Node API method.

The API method will then store the JSON object in the non-relational database, which is the end of the operation. The density grid can now be retrieved from the database by the client when needed.

```
const pythonProcess = spawn('python', ['/var/lace-server/exec.py']);

pythonProcess.stdout.on('data', function(data) {
    GRAPH.deleteOne({}, function(deleteError, deleteResult) {
        if (deleteError) {
            console.log(deleteError);
            return (res.send({error: CONST.DELETE_ERROR}));
        }

        parsedData = JSON.parse(data);
        GRAPH.create({
            graph:      parsedData[0].graph,
            distance:   parsedData[1].distance,
        }, function(insertError, insertResult) {
            if (insertError) {
                console.log(insertError);
                return (res.send({ error: CONST.INSERT_ERROR }));
            }

            console.log(insertResult);

            return (res.send(true));
        });
    });
});
```

Figure 43: JavaScript code for storing the data returned from the Python child

Figure 44: The blue path guiding a user towards a destination

13. Final Prototype: Reactions

We tested an early version of the Lace client with a couple of test users. The test was set up in a courtyard, with a webcam on a fourth-floor balcony being used to capture the scene. We started by measuring the dimensions of the courtyard using a household measuring tape and subsequently calculated the approximate distance between nodes in our capture.

We gave the test users a phone and asked them to describe what they saw as they used the app, in a similar manner to our paper prototype tests. While the path was not aligning correctly in the space, the users still provided useful feedback.

When selecting destinations from the list, one of the users referred to destinations and areas as separate categories —the courtyard being the area and the apartment entrance the destination.

The users found it easy to understand and select the correct destination. Despite this, one of the users theorized about not knowing what certain destinations are called if visiting an unknown place. They wondered why they were provided with a list as a method to select destinations. In most cases, they would prefer to search for a destination.

They mentioned there were too few obstacles to justify using the Lace client —at least in this specific scenario. The small size of the courtyard had a significant effect on the usefulness of the client. Users virtually ignored other participants standing in the courtyard, likely due to the fact that their selected destination was too easy to identify from where they were standing. The dimensional limitations led us to conclude that this particular experiment was not able to communicate the complete usefulness of the app.

Users found it helpful to be able to see that the line led towards the destination (when it was working properly, at least). Seeing the full length of the line was not too important in this courtyard scenario. Considering a larger area, users would still prefer to see the whole line for context —perhaps within a map view.

The blue line was interpreted as looking a bit too wide. A test user theorized about themselves perceiving it that way because of the limited surrounding area. When standing so close to the goal, they felt that it was unnecessary to be shown a blue line. Users found that the line was much too high in their field of view and would have preferred it to be situated closer to the ground plane. Because the line was situated at eye-level, the users would hold the phone up directly in front of their face. When asked, one user expressed a preference for holding the phone at a more discreet angle, such as by their waist.

Upon reaching their destination, users wanted to be shown some form of confirmation. One user in particular wanted to be offered an option either to close the app or to choose a new destination.

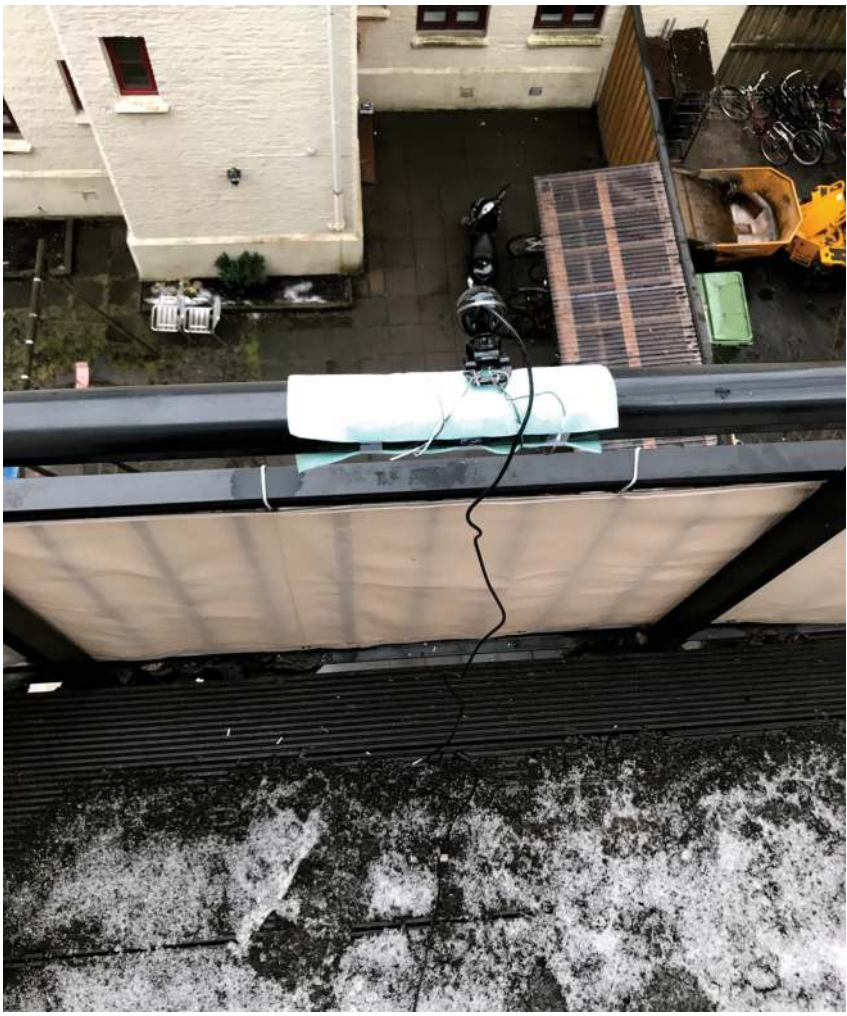


Figure 45: Webcam mounted onto balcony to capture as much of the scene as possible

14. Discussion

Destinations can have different properties which influence how they are interpreted by users. Because they can represent either finite points or entire areas, the way in which they are represented by the client could be more carefully considered.

Some users have a very specific idea about where they would like to navigate to in a public space. This can influence the way in which they wish to interact with the client. Destination selection or input methods are important to the perceived usefulness of the app; Our list view may not be useful to everyone.

Lace is primarily useful in very large spaces. Its perceived utility is more delicate in smaller spaces, where there is little activity. If users are able to see their destination from where they are standing, they may not consider interacting with the client at all, regardless of the perceived size or complexity of the space they are standing in. Users may simply choose to forego using the app in some situations, opting to follow their intuition instead.

The social impact of using the Lace client in public is also of concern. Some users may not be comfortable holding their phone up in front of them, as they navigate a crowd. The blue line may have to be more comprehensible, if users' phones are pointed straight towards the ground.

Lace has evolved into providing navigation guidance towards a user-defined destination, as opposed to improving the design of digital signage over time. While we do not see this as a disadvantage, it varies from the original problem formulation. Lace can respond to the collective movement of pedestrians through public space, advising pedestrians on how to navigate towards their destination. At the time of writing, Lace takes roughly three seconds to analyze a space. By lowering the time spent detecting objects, we can increase how responsive the navigation is towards the user's surrounding environment.

We strayed from the original project formulation by dropping support for personalized navigation. At the time of writing, the prototype does not accommodate individual users' preferences for the design of digital signage. The lack of this personalization made the development of the prototype more feasible within our timeline.

In guiding pedestrians, we have to consider how to preserve users' agency while they interact with the prototype —by extension, ensuring that their interactions are ethical. There were plenty of ethical concerns pertaining to personal data and privacy that had to be addressed.

Congestion can form when too many pedestrians try to approach the same target, in a sort-of "bottleneck" [19]. This can cause them to rub shoulders —possibly even slowing their movement to a standstill. This led us to question whether staggering pedestrians could be a viable solution. Urging pedestrians to wait before proceeding through a dense crowd could rob them of their agency; Instructing users to exhibit certain behavior can have side effects not limited to suppressing users' own decision-making abilities. We also avoided punishing users for any decision or behavior they made or exhibited for the same reasons.

The argument for preserving individual privacy lies in preserving the absence of user-specific data. Users remain anonymous —if any data is ever leaked or otherwise obtained, their privacy cannot be violated.

15. Reflection

In order to unpack the value of our project, in this section, we reflect on the impact on the different choices we made over the course of the project.

We consistently spoke about the project with other people throughout the process in an effort to widen our perspectives. As part of our preliminary investigation, we spoke with experts in a wide array of different fields. Their willingness to help us out suggests an interest in our project. People we spoke with were generally intrigued and wanting to learn more about Lace and/or discuss related subjects.

We were quick to transition ideas to action. Scouting out venues early on reframed the scope of testing to a more controlled and less chaotic environment. Understanding that we were designing and developing for testing in a controlled environment ultimately focused our prototype.

The project, as a whole, was data-driven, as we used data to educate our every decision. Hitting the ground running meant we had plenty of good data to start with. This data came from conducting experiments with passers-by, from survey responses, and several qualitative in-person interviews. Each set of responses informed new research. We made sure to string together every step in the process, referencing previous findings in order to move forward; we did not move forward without basing our decisions on previous findings.

We admittedly could have done more low-fidelity prototyping. This might have allowed us to resolve any lack of understanding on part of those who tested our prototype. This is particularly true in the landscape architect's [Section 11.2.2] case. It was not until after starting development, that we understood that he might not have fully understood the role of augmented reality in the context of his environment. To make up for lack of spatial understanding, we could have conducted further experiments, such as using a string to guide users across a room. The reason we had chosen not to do so at the time was that we felt that our paper prototype was so grounded in the findings from earlier user studies. In this case, our faith in the process was tested too little too late. This led us to eventually conduct a second test, which proved to be more fruitful.

The amount of difficulty and scale of the final prototype forced our process to be more linear than we would have otherwise preferred. We could not build and test with users during the entire development phase, because it simply would not have been functional enough until it was completed. The final prototype would depend on both the server and client being fully functional. We knew because we were only two people, the time spent developing the prototype would run close to our final deadline. This forced linearity was the primary reason why we decided against using Agile. While we had set out to use Agile in our process, the framework felt increasingly redundant, as we were each developing our own part of the stack. It would have meant working with Agile just for ourselves. We felt that it was, in fact, inhibiting our processes. We think Agile could still be a great tool for larger teams. We did not feel that the time investment would provide enough benefits.

Throughout the entire design process, we collaborated closely with each other. Experiments, interviews, a survey, analysis, and low-fidelity prototyping were all made in strong collaboration, which resulted in a strong foundation for the development of the high-fidelity prototype. When it came time to start development on the high-fidelity prototype, a larger separation of responsibilities was created. Work was delegated due to the estimated scope of work which had to be done

in order to reach our goals pertaining to the minimum viable prototype. The reason for the separation of responsibilities and delegation was that the two systems, the server, and the client, were building upon each other, ergo one could not stand without the other. To reach the deadline, development would have to happen in parallel. The responsibilities and work were delegated in correspondence with each team member's personal preference and interest.

The strengths of this team mainly come from the member's familiarity with the process and working with each other. Having shared goals meant that we never spent too much time agreeing on a certain direction. Additionally, a balanced set of overlapping expertise accelerated the decision-making process. Because we had worked together on numerous previous projects, we were proactive in our process. This is both apparent in our outreach to experts in different parts of the world, as well as our conducting of experiments during the user studies.

Some of our communication suffered throughout the development process. This could have been due to a myriad of reasons, one of which was simply being overwhelmed with the amount of work in each of our delegated tasks. We never established or discussed our individual expectations on the frequency of communicating one's progress with each other, which could have been done early on. However, in writing this conclusion, it was found that our expectations for frequency of communication were not aligned. Perhaps because we were spending a lot of time together, we did not make an effort to schedule regular, more structured meetings.

The argument for communicating with agreed-upon regularity is rooted in boosting team morale. Everyone has a different personal threshold in this respect, which contributes to their trust in the process. It is not just for the sake up being updated, as every time one is prompted to describe a complex system, they can reflect and improve on their understanding of that system, as well as their presentation skills. While we may not have needed to update each other every single day, it could have benefitted us to establish regular, more structured meetings.

As the development of the final prototype was so intensive and complex, it did not leave much room for testing during development. This is due to a fault in our time management. The systems were dependent on each other, and a large part of the development happened behind the scenes. While testing an interaction or a user interface is plausible, testing the server structure and API endpoints with users would not provide much feedback. The two parts of the prototype were therefore developed in parallel, and testing could not happen until later in the development. In the future, we would prefer to have a better user-involvement during the development, establishing a co-design process. However, with only two members in the team, there simply was not enough time to develop the front- and back-end separately for testing.

Figure 46: A busy street in Bangkok

16. Conclusion

The potential of Lace lies in its ability to open up crowd-analysis to the average consumer. While there are plenty of indoor navigation solutions on the market already —several implemented using augmented reality— none of them are factoring in the potential of current crowd-analysis practices. The static analysis makes finer density maps less critical. In case of being able to predict individual persons' movements more accurately, possible future iterations of the maps could take the shape of probabilistic heat map-based velocities or even users' intentions. Future iterations of Lace should consider group behavior and urge pedestrians into circulation patterns, keeping traffic flowing at a consistent speed [17]. Cappiello suggested incorporating an indicator of commotion into the client interface. The idea of a single score to describe the state of commotion in a space could be useful in the development of early warning systems.

Collecting real-world data on pedestrian movements can help improve simulation models, which rely on a myriad of different variables. Researchers and developers could compare the results of their models to those of the real space, potentially leading to the identification of new variables and increase their realism. Automating the process of crowd-analysis could provide a huge boost to pedestrian movement-related research, enabling researchers and developers to focus on how to use the data rather than how to collect it. Using the security cameras available to us today, we can learn the specificities of how and where pedestrians get "stuck", or which specific routes they take in response to site-specific stimuli. These site-specific stimuli could take the shape of particularly distracting branding elements or unintentionally confusing wayfinding signage. The vast majority of computer vision applications only serve to predict pedestrian positions and trajectories using pre-recorded video. This is because these solutions are intended to inform spaces during the planning process rather than the spaces they are analyzing. This introduces the potential of dialogue between pedestrians, objects, and the spaces which they occupy. The opportunity herein lies in the treatment of architecture or urban areas as a living organism, continuously affected by its occupancy.

17. Next Steps

The Lace server could potentially factor in the probability of pedestrians entering and exiting at specific portals, such as doors, staircases, or queues, in a space [8]. That way, the guiding path could accommodate for unapparent traffic flow surrounding these portals. Another possible improvement could be made by implementing a dynamic blur filter to the grid matrix that responds to changing room density. Breaking up rooms into larger cell clusters with their own respective density weights could affect how Lace guides users around other pedestrians and obstacles, such as tables or walls. If Lace understands wherein a space pedestrians are likely to accumulate, it could assume certain points of interest and potentially predict the attraction certain pedestrians feel towards those points of interest.

Machine learning models could be used to assign temporary anonymous identities to identified pedestrians in a space. As our model improves over time, we could implement prediction models for each individual pedestrian. This theoretical prediction model could be run at different rates depending on how quickly certain pedestrians are moving through a space. The algorithm could be further optimized by identifying pedestrian groups and running prediction models on their movement as if they were a single person. Lace could identify groups based on their formation pattern respective to the density of the space around them. For instance, if a cluster of pedestrians starts to form a "V" pattern in a low-density space, we could assume that the pedestrians are associated with each other in some way. As roughly 70 percent of pedestrians in crowds are moving in groups [29], this could dramatically increase the efficiency of our algorithm, by reducing the number of actions taking place. The shape itself of a group could also indicate which direction they are walking in.

Roughly 70% of pedestrians in a crowded environment journey in groups [29]. Groups could be identified by their formations respective to their local density; groups of pedestrians have a tendency to shift from walking side-by-side to a V-like formation as the overall density in a room increases [29]. In identifying groups lies the potential for sweeping optimizations to Lace's algorithms. As the movement of groups typically mimics that of single individuals [16], the number of evaluations a theoretical prediction model would make could be drastically reduced. And, consequently, the Lace server could return density graph objects much more often.

When using lace, a security camera may not capture the entire room, due to either a limited field of view or overly distorted edges. While Lace may then be limited from identifying the positions of pedestrians outside of the scope of its cameras, it is still important to offer guidance to pedestrians outside of that scope. SemaFORR is an alternative solution to A* that allows for navigating ambiguous environments. This can be useful when the layout of the rest of a room is unknown. SemaFORR works by anchoring points with respect to detected walls in their proximity, reconstructing the environment and superimposing a path on top of it. While SemaFORR was intended for use in robotics, similar principles could be applied for augmented reality navigation. Frameworks such as ARCore already feature built-in plane detection. Future iterations of the Lace client could leverage collective environmental reconstruction in order to learn and navigate the spaces which the server cannot detect —albeit consequently lacking detected crowd heatmaps [11].

Sometime during our development phase, we were contacted by a fellow product designer/engineer, Moina Medbøe Tamuly, regarding a public installation in Oslo at a

future date in 2019. He proposed using the Lace server in conjunction with some digital visuals that would represent the locations of people in that space. The subject of the exhibition is the "intertwining of man and machine". Several Norwegian retailers have also shown interest in using parts of the technology to analyze customer behavior in their stores. This hints at a much wider range of potential use cases for this technology, and we could not be more excited about what the future holds for Lace.

Figure 47: Paul Nylund (left) and Michal Nissen (right) during a previous project

18. Bibliography

[1] Abernethy, M. (n.d.). What is node NodeJS. Retrieved January 2, 2019, from https://www.scribd.com/document/54504131/What-is-node-JS-developerWorks

[2] Abidy, T., Pang, H., & Williams, C. (n.d.). Dijkstra's Shortest Path Algorithm [Blog post]. Retrieved January 2, 2019, from https://brilliant.org/wiki/dijkstras-short-path-finder/

[3] Alfirević, Đ., & Alfirević-Simonović, S. (2018). "Circular connection" concept in housing architecture. Arhitektura I Urbanizam, (46), 26–38. http://doi.org/10.5937/a-u0-16252

[4] Anagnostopoulos, T., Anagnostopoulos, C., & Hadjiefthymiades, S. (2011). An Adaptive Machine Learning Algorithm for Location Prediction. International Journal of Wireless Information Networks, 18(2), 88–99. http://doi.org/10.1007/s10776-011-0142-4

[5] Brunyé, T. T., Gardony, A. L., Holmes, A., & Taylor, H. A. (2018). Spatial decision dynamics during wayfinding: intersections prompt the decision-making process, 1–19. http://doi.org/10.1186/s41235-018-0098-3

[6] Child Process | Node.js v11.6.0 Documentation. (n.d.). Retrieved January 2, 2019, from https://nodejs.org/api/child_process.html

[7] Chung, J., Pagnini, F., & Langer, E. (2016). Mindful navigation for pedestrians: Improving engagement with augmented reality. Technology in Society, 45(C), 29–33. http://doi.org/10.1016/j.techsoc.2016.02.006

[8] Davidich, M., & Köster, G. (2013). Predicting Pedestrian Flow: A Methodology and a Proof of Concept Based on Real-Life Data. PLoS ONE, 8(12), e83355–11. http://doi.org/10.1371/journal.pone.0083355

[9] Dingler, T., Kunze, K., & Outram, B. (2018). VR Reading UIs (pp. 1–6). Presented at the 2018 CHI Conference, New York, New York, USA: ACM Press. http://doi.org/10.1145/3170427.3188695

[10] Easy Curved Line Renderer [Forum post]. (n.d.). Retrieved January 2, 2019, from https://forum.unity.com/threads/easy-curved-line-renderer-free-utility.391219/

[11] Epstein, Susan & Aroor, Anoop & Evanusa, Matthew & I Sklar, Elizabeth & Parsons, Simon. (2015). Navigation with Learned Spatial Affordances.

[12] Feliciani, C., & Nishinari, K. (2018). Measurement of congestion and intrinsic risk in pedestrian crowds. Transportation Research Part C, 91, 124–155. http://doi.org/10.1016/j.trc.2018.03.027

[13] Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Pp 76-97 Retrieved from https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

[14] FIND3 [GitHub Repository]. (n.d.). Retrieved January 2, 2019, from https://github.com/schollz/find3

[15] Fridman, N., & Kaminka, G. A. (2007). Towards a Cognitive Model of Crowd Behavior Based on Social Comparison Theory, 1–7.

[16] Helbing, D. (1998). A Fluid Dynamic Model for the Movement of Pedestrians. Retrieved from arXiv:cond-mat/9805213

[17] Helbing, D. (2012). Self-organization in Pedestrian Crowds. In Social Self-Organization (pp. 71–99). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-24004-1_3

[18] Helbing, D., & Molnár, P. (1995). Social force model for pedestrian dynamics. Physical Review E, 51(5), 4282–4286. http://doi.org/10.1103/PhysRevE.51.4282

[19] Hoogendoorn, SP, & Daamen, W (2003). Controlled experiments to derive walking behaviour, European Journal of Transport and Infrastructure Research 3 (1), pp. 39-59.

[20] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (in press). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861. Retrieved from https://arxiv.org/pdf/1704.04861.pdf

[21] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., . . . Murphy, K. (in press). Speed/accuracy trade-offs for modern convolutional object detectors. arXiv:1611.10012. Retrieved from https://arxiv.org/pdf/1611.10012.pdf

[22] Johansson, F. (2013) Microscopic Modeling and Simulation of Pedestrian Traffic. Microscopic Modeling and Simulation of Pedestrian Traffic, 1–127.

[23] Lague, S. (n.d.). Pathfinding [GitHub Repository]. Retrieved January 2, 2019, from https://github.com/SebLague/Pathfinding

[24] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. C. (n.d.). SSD: Single Shot MultiBox Detector [Slide Show]. Retrieved January 2, 2019, from http://www.cs.unc.edu/~wliu/papers/ssd_eccv2016_slide.pdf

[25] Lombard, M., & Ditton, T. (1997). At the heart of it all: The concept of presence. Journal of Computer-Mediated Communication, 3(2).

[26] Medley, J. (2018, August 29). Augmented reality for the web [Blog post]. Retrieved January 2, 2019, from https://developers.google.com/web/updates/2018/06/ar-for-the-web

[27] Millonig, A., & Schechtner, K. (2006). Developing Landmark-Based Pedestrian-Navigation Systems. IEEE Transactions on Intelligent Transportation Systems, 8(1), 43–49. http://doi.org/10.1109/TITS.2006.889439

[28] Mitchell, T. M. (1997). Machine Learning. New York, NY: McGraw-Hill Education.

[29] Moussaïd, M., Perozo, N., Garnier, S., Helbing, D., & Theraulaz, G. (2010). The Walking Behaviour of Pedestrian Social Groups and Its Impact on Crowd Dynamics. PLoS ONE, 5(4), e10047–7. http://doi.org/10.1371/journal.pone.0010047

[30] Mulloni, A. (2018). Experiences with the Impact of Tracking Technology in Mobile Augmented Reality Evaluations, 1–4.

[31] Node.js v9.11.2 Documentation. (n.d.). Retrieved January 2, 2019, from https://nodejs.org/docs/latest-v9.x/api/

[32] Prinke, M. (2917, July 14). Why do people say that Unity is better than Unreal for mobile game development when both engines support mobile deployment? [Forum post]. Retrieved January 2, 2019, from https://www.quora.com/Why-do-people-say-that-Unity-is-better-than-Unreal-for-mobile-game-development-when-both-engines-support-mobile-deployment

[33] Python 2.7.15 documentation. (n.d.). Retrieved January 2, 2019, from https://docs.python.org/2/

[34] Qualcomm. (2018, February 25). Qualcomm Network Simulation Shows Significant 5G User Experience Gains. Retrieved January 2, 2019, from https://www.qualcomm.com/news/releases/2018/02/25/qualcomm-network-simulation-shows-significant-5g-user-experience-gains

[35] Rabin, S. (n.d.). JPS+: Over 100x Faster than

A* [Video file]. Retrieved January 2, 2019, from https://www.gdcvault.com/play/1022094/JPS-Over-100x-Faster-than

[36] Rudrakshi, C., Varshney, A., Yadla, B., Kanneganti, R., & Somalwar, K. (2014, August). API-Fication. Core Building Block of the Digital Enterprise. HLC Tech, 1(1), 4–7. Retrieved from http://www.hcltech.com/sites/default/files/apis_for_dsi.pdf

[37] Rzayev, R., Wozniak, P. W., Dingler, T., & Henze, N. (2018). Reading on Smart Glasses (pp. 1–9). Presented at the 2018 CHI Conference, New York, New York, USA: ACM Press. http://doi.org/10.1145/3173574.3173619

[38] Sato, K., Young, C., & Patterson, D. (2017, May 12). An in-depth look at Google's first Tensor Processing Unit (TPU) [Blog post]. Retrieved January 2, 2019, from https://cloud.google.com/blog/products/gcp/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

[39] Sekhavat, Y. A., & Parsons, J. (2017). The effect of tracking technique on the quality of user experience for augmented reality mobile navigation, 1–34. http://doi.org/10.1007/s11042-017-4810-y

[40] Schuler, D., & Namioka, A. (1993). Participatory Design: Principles and Practices. Didcot, United Kingdom: Taylor & Francis.

[41] Singleton - Unify Community Wiki [Forum post]. (n.d.). Retrieved January 2, 2019, from http://wiki.unity3d.com/index.php/Singleton

[42] SSH (Secure Shell). (2018, November 1). Retrieved January 2, 2019, from https://www.ssh.com/ssh/

[43] TensorFlow.js Documentation. (n.d.). Retrieved January 2, 2019, from https://js.tensorflow.org/api/0.13.0/

[44] Weisstein, E. W. (n.d.). Moore Neighborhood [Blog post]. Retrieved January 2, 2019, from http://mathworld.wolfram.com/MooreNeighborhood.html

[45] Wenderlich, R. (2011, September 29). Introduction to A* Pathfinding [Blog post]. Retrieved January 2, 2019, from https://www.raywenderlich.com/3016-introduction-to-a-pathfinding

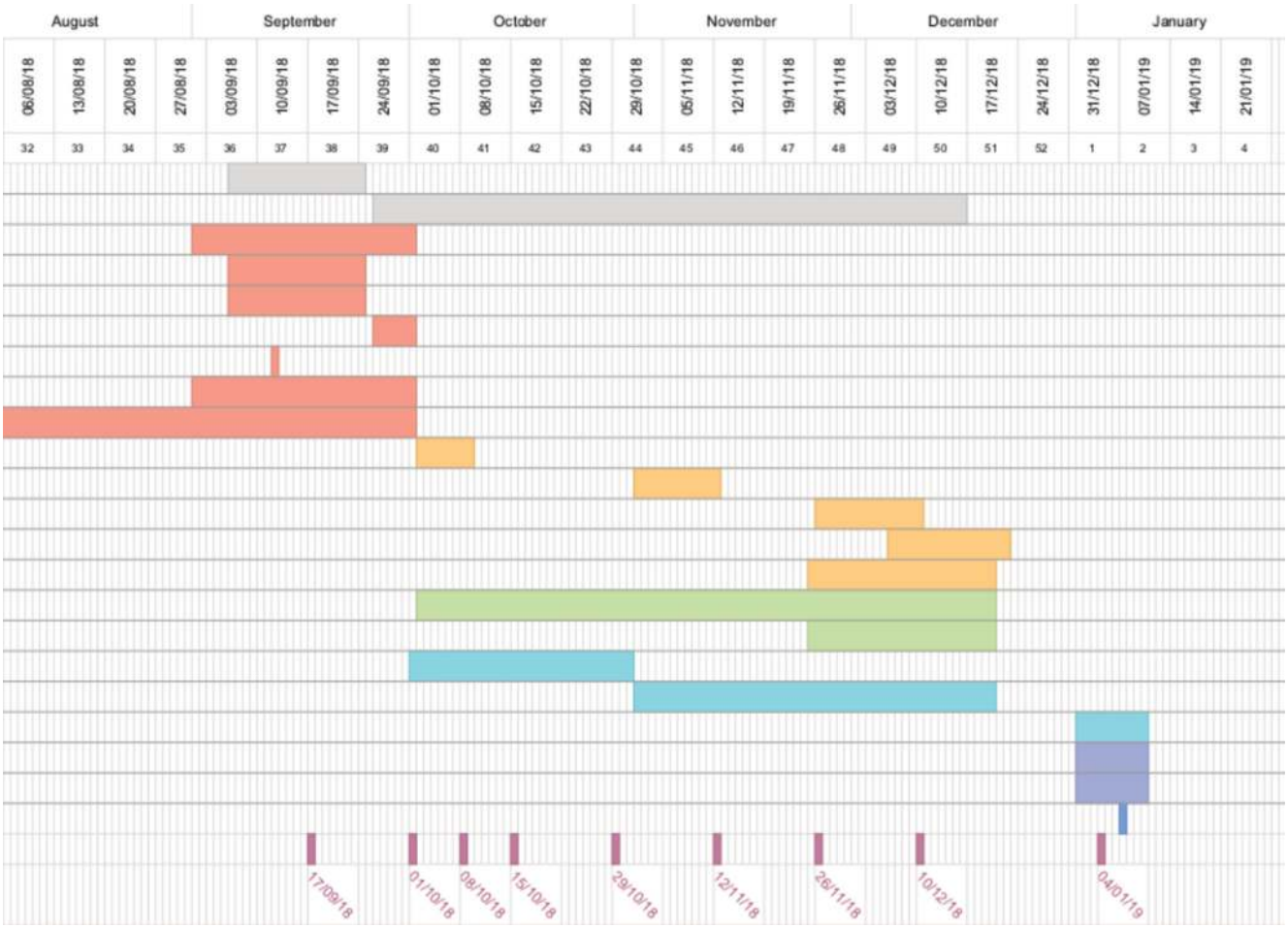
[46] What is SSL, TLS and HTTPS? (n.d.). Retrieved January 2, 2019, from https://www.websecurity.symantec.com/security-topics/what-is-ssl-tls-https

[47] Yi, S., Li, H., & Wang, X. (2015). Pedestrian Travel Time Estimation in Crowded Scenes. 2015 IEEE International Conference on Computer Vision (ICCV), . https://doi.org/10.1109/iccv.2015.359

19. Appendix

19.1. GANTT Chart

				June				July				
				04/06/18	11/06/18	18/06/18	25/06/18	02/07/18	09/07/18	16/07/18	23/07/18	30/07/18
Phase	**Task**	**Start Date**	**End Date**	23	24	25	26	27	28	29	30	31
California		06/09/2018	24/09/2018									
Melbourne		26/09/2018	16/12/2018									
	Problem Formulation	01/09/2018	01/10/2018									
	Meeting - Montalba Architects	06/09/2018	24/09/2018									
	Meeting - Clive Wilkinson Architects	06/09/2018	24/09/2018									
	Meeting - HASSELL Studio	26/09/2018	01/10/2018									
	Meeting - Google	12/09/2018	12/09/2018									
	Establish a user group in Melbourne	01/09/2018	01/10/2018									
Preliminary	Readings	01/06/2018	01/10/2018									
	Writings about the problem	02/10/2018	09/10/2018									
	Writeup on initial user research	01/11/2018	12/11/2018									
	Promotional film	26/11/2018	10/12/2018									
	Proofreading	16/12/2018	22/12/2018									
Thesis Writing	Generating learnings	25/11/2018	20/12/2018									
	Co-designing with users	02/10/2018	20/12/2018									
User Studies	Generating learnings	25/11/2018	20/12/2018									
	Low fidelity prototyping	01/10/2018	31/10/2018									
	High fidelity prototyping	01/11/2018	20/12/2018									
Prototyping	Fine-tune the prototype	01/01/2019	10/01/2019									
	Create Keynote	01/01/2019	10/01/2019									
Thesis Defence	Practice Keynote	01/01/2019	10/01/2019									
Hand-in	Delivery	03/01/2019	03/01/2019									
Supervision	Bi-Weekly Meeting (1 hour)	17/09/18	04/01/2019									


August
September
October
November
December
January
06/08/18
13/08/18
20/08/18
27/08/18
03/09/18
10/09/18
17/09/18
24/09/18
01/10/18
08/10/18
15/10/18
22/10/18
29/10/18
05/11/18
12/11/18
19/11/18
26/11/18
03/12/18
10/12/18
17/12/18
24/12/18
31/12/18
07/01/19
14/01/19
21/01/19
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
1
2
3
4
17/09/18
01/10/18
08/10/18
15/10/18
29/10/18
12/11/18
26/11/18
10/12/18
04/01/19