Bilkent University
Department of Computer Engineering

Senior Design Project

Project Specifications Report

Urbscope

urbscope.com

Group Members

Burak Mandıra
Mustafa Motani
Waqaas Rahmani
Syed Sarjeel Yusuf
Abdullah Wali

Supervisor

Mustafa Özdal

Innovation Expert

Veysi İşler

Jury Members

Selim Aksoy
Mehmet Köytürk

Table of Contents

1. Introduction

The tourism industry is one of the largest industries in the world with global financial contributions of 7.6 trillion dollars in 2015 alone [1]. However, only a handful of apps exist to facilitate tourists in their travels, and this poses great issues for the industry. Once at the destination tourists and travellers have to rely on traditional methods to explore the wonders of the destination. This usually entails conventional tourist guides, pamphlets and travel magazines. Moreover, tourists usually also have to rely on conventional ways to find points of interest in a city or whatever destination they visit. This also leads to several points of interest being missed out as tourists simply do not know that they exist. It is quite common for smaller attractions to be overshadowed by larger more iconic city attractions, usually resulting in void of knowledge that could be gained by the smaller attraction. Even locals usually miss out and are not informed about all the attractions in their city. Furthermore budget restricted travellers usually avoid tour guides to show them around and tell them about the touristic attractions. This further leads to tourists missing out valuable cultural and historical information that would otherwise enhance their trip. Sometimes these tour guides do not even hold a speciality in the points of interests that they cover in their tours, and hence usually provide incorrect information.

Hence, it is quite surprising that the tourism industry, an industry that some countries base their whole economy's on, has not been revolutionised by mobile technology. There are very few technological solutions that could better the experiences of tourists and even locals experiencing their own cities. Moreover, there is also no concrete solution for travellers being informed of attractions while on the go, unless they check for points of interest from sites such as Trip Advisor [2], before setting out on their adventures. Hence, Urbscope attempts to address these issues with new forms of technology.

1.1. Description

Urbscope is aimed at the tourism industry to solve the problems mentioned above. It aims to allow travellers to identify buildings from great distances using augmented reality markers to tag the far away locations of points of interests. It also attempts to recognise points of interests, mainly touristic buildings via the camera view of the user's mobile device. Similarly it also aims at informing users of nearby attractions while he or she walks around. Additionally, the product shall incorporate a recommendation system whose objective would be to suggest other places of interests depending on similar travellers. The basic scenario that is expected to be provided by the product is as below:

> *The user is walking on İstiklal Caddesi and he/she passes the small church located on the street. The user uses his device to see the church through the camera and it identifies it as St. Anthony of Padua Church. The user also gets all the relevant information about it. After moving away from the Church he goes down İstiklal Caddesi and gets a notification that there is a point of interest near by, which is Galata tower. The tourist then passes, and he or she uses the app again and it tells him that it is Galata Tower, and some information about it*

comes up. Then on when the user is on the balcony of Galata tower, he uses his camera to scan around the horizon of Istanbul, and again little augmented markers keep coming up indicating there is Hagia Sofia. Then as he turns a bit more and it indicates there is Sultan Ahmet Mosque, and then he turns a bit more and it indicates there is Süleymaniye Mosque.

So as can be seen there are three integral parts to the product proposed.

1. Augmented reality markers indicating points of interest nearby and their locations. These markers will be color coded depending on how far away they are from the user's current location. I.e Intense red indicates they are relatively extremely far, whereas light green indicates that they are relatively close. Also by clicking on one of these markers, the user will be able to able to set a target location and respective directions to the location shall be provided.
2. Detection of points of interest with camera. This will allow the user to detect the point of interest, mainly buildings and structures. Once the building is detected the user will get options to check for all relevant, touristic related, information pertaining to the point of interest on the screen. Similarly, if other points of interests are located nearby the user will be notified of their presence.
3. Recommendation System. This will analyse the user's places of visit and suggest other points of interest depending on similar users that have visited the same place the user has visited thus far. The recommendation system may also suggest popular points amongst tourists and not only attractions. These include restaurants, cafes, bars etc.

1.2. Constraints

- Platforms: The target platforms will tentatively be Android and iOS devices. However, we might decide to focus on one platform instead of tackling both at the same time, depending on the frameworks and tools that we decide to use. We expect this to be finalised after doing the initial analysis and clarified in the Analysis report in the future.
- Devices: We expect that our application will be heavy on CPU usage and will require high end hardware to run smoothly due to the usage of Augmented Reality (AR) and possibly Image Recognition. Therefore we will be limiting the devices that we will be supporting to iPhone 6S and higher on the iOS spectrum to the latest flagship phones on Android (due to the wide range of phones running Android it is not possible for us to specify which devices we will support, but it is sufficient to say we're targeting the latest devices).
- Economic: We will probably need access to several APIs in our implementation, and some of those APIs might be expensive (most of the ones we have looked at so far charge per request after a predetermined number of initial requests). We will try to find the most economic option to use without compromising quality and response time.

- Recommender System Data: We will need to collect and store user preferences to recommend useful locations and cafes on their trip. We might need to access some APIs like tripadvisor for better and faster data.
- Manufacturability: The app targets mobile platforms as stated earlier. Therefore, it will only consist of software and, as a hardware, it requires mobile phones specified in Devices part. We can assume that most of the people have mobile phones and thereby, the application will not demand an extra hardware. This substantially increases the manufacturability of the app. In terms of software part, the development process is expected to be hard because of the complexity of the app. However, this will not be much if we consider the time given.

1.3. Professional and Ethical Issues

Marketing and Integrity: Veysi Isler, our innovation expert, suggested including cafes and restaurants in the recommender system in addition to touristic landmarks. However, we want to make sure that recommendations would only be given to places that the user is expected to enjoy (according to what others who visited similar places have liked) while fulfilling the promise to potential commercial partners (cafes, restaurants, etc.) by recommending them to the users.

Keeping user's information anonymous: We might need to access the user's location to detect nearby landmarks. However, we have to respect the user's privacy by not storing his/her location information on our servers and making sure any analytic reports we collect remains anonymous.

2. Requirements

This section includes the requirements of the app as well as the 3 different implementation plans. The requirements are as follows:

2.1. Functional Requirements

- The user will be able to identify landmarks by pointing his camera towards the landmark.
- The application's main interface will be superimposed on the camera view.
- The application will then show basic general information about the landmark, with the option to navigate to more detailed information outside the app.
- The application will provide navigation facilities towards the target building such as arrows, lines etc. in the AR layer.
- As a default, only the attractions within some predefined distance (i.e. this can be represented as a circle with specific diameter) will be shown. We call this distance as proximity limit.
- The popular attractions around the user will be shown by pop-ups in the AR layer.
- Users will be alerted with popups when they have passed by any popular touristic landmark.
- The application will take into account the altitudes of the nearby landmarks not displaying information about obscured landmarks even if they are within the proximity limit.
- The information layer will display the information about the attraction.
- The information layer will be triggered when the user click on AR pop-ups and the relevant information about that attraction will be displayed.
- At the end of the displayed information, there will also be some other recommended attractions which are visited by the other users who also have visited the current attraction.

2.2. Non-Functional Requirements

- Our application will be available in English.
- The application should show that a certain landmark has been recognised when the user points his camera at it.
- The user should be able to click a detected landmark to see information about that landmark.
- Users are able to change the proximity limit from the settings of the app.

- Depending on the proximity between the user and the surrounding attractions, the colours of the pop-ups will be different, e.g. green for near attractions and grey for further ones.
- The alerts for passed by attractions can be switched on/off depending on the user's preference.

2.3. Implementation Plan

- React Native is to be used as the primary platform to implement at least the front end
- TripAdvisor API is used to decide what points of interest to include, along with location
- Cloud Vision API is used to detect landmarks
- Vuforia framework for augmented reality, as suggested by innovation expert
- Android AR-Toolkit will be used if Vuforia to comprehensive as a tool of implementation
- Collaborative filtering algorithms are used for recommendation system
- Google Maps API is used for directions and other map features i.e Polygons [3]

2.4. Backup Implementation Plans

Primary Plan

Our main plan is to use image recognition through Google Vision platform to identify the touristic landmarks. However, on the unlikely scenario that our initial plan fails and we don't find an similar alternative image recognition library, we have 2 backup plans regarding the implementation of the application. These backup plans are named as Plan B and Plan C in decreasing order of their priority. These plans are as follows:

Plan B

In the context of Plan B, the app will not use image recognition techniques to identify the attractions. We thought that Vision API might not work accurately for recognition of attractions and in that case we have to implement our own recognition system for attractions. This increases our workload by many folds making it infeasible. Recognition of attractions without using computer vision APIs can be implemented in two ways. One of them is using deep learning via convolutional neural networks; however, neural networks requires a lot of data to work accurately. We don't have such a database that includes images of popular attractions around the world. Another way is applying similarity based methods for a given set of data. This might work better for less amount of data; however, it restricts the practicality of the application in terms of sheer number of recognizable attractions. Thereby, in the context of Plan B, we will use our own methods to identify the attractions which are indeed very plain. We will use the compass to determine which direction the user is pointing to; the accelerometer determines the angle of the direction (i.e. whether the user points towards upwards or downwards); and the absolute altitude of the user and the close attractions. We'll

use the altitudes while determining if there's an obstruction that does not allow user to see the actual attraction. For that, we will compare the altitude difference between the user and all the surrounding attractions that are in the range of proximity limit and decide accordingly. If there is an obstruction between the user and the target attraction, the transparent popup will be displayed. Recognition technique of the attractions is the only difference between Plan A and Plan B. So, every other features will be same as in Plan A.

Plan C

As for Plan C, it is considered as a last ditch effort indeed. Therefore, it is unlikely that we will employ this plan; however, it is always good to have simple and easy to do backups. This plan differs from previous plans again in terms of landmark recognition. If Plan B's implementation proves to be too inaccurate to be usable, Plan C will be employed. All of the attractions will be added manually, i.e. there will be a predefined data set for landmarks and the app will solely show these landmarks. On the other hand, it will allow users to add attractions in order to increase usability. It will serve as a compensation mechanism in case of having a small sized date set for landmarks.

3. References

1. Fuggle, “70 travel & tourism statistics to know about in 2016 [Trends Report],” *Tour Operator Online Booking Software*, 09-May-2016. [Online]. Available: https://www.trekksoft.com/en/blog/travel-tourism-stats-2016.

2. “Attractions,” *TripAdvisor*. [Online]. Available: https://www.tripadvisor.com/.

3. “Simple Polygon,” *Developers.Google*. [Online]. Available: https://developers.google.com/maps/documentation/javascript/examples/polygon-simple.