

Augmented Reality for Later Playback of Videos

Project Proposal

Jordan Stratton

Abstract: Augmented reality's popularity has recently exploded since mobile phones have had the capability to create a great user experience, while at the same time show useful information. Unfortunately, augmented reality applications that have been created today face a problem in that the user cannot later playback a video they have taken in order to easily find information about places they have been. We propose a method of allowing the user to take a video and choose whether or not to have augmented reality on while they record. Then the user can later playback the video with augmented reality. This can very helpful for people who want to record a place they have been and later want to remember everything that was in the video. Also the user can choose to send the video to anyone who is interested who then can watch the video and learn about where the user was.

Table of Contents

Abstract.....	1
Figures.....	3
Introduction	4
Previous Work.....	5
Different Uses of Hardware	5
Mobile Device	10
Image Analysis.....	11
Finding Optimal Ways to Display Information on Screen	12
Distinct iPhone Applications	13
Proposal	14
Purpose	14
Layout of application	15
Current Work	19
Demo.....	19
Future Work.....	20
Future Features to add	20
Works Cited.....	21

Figures

Figure 1: Prototype campus information system. The user wears a backpack and head worn display, and holds a handheld display and its stylus. [7]

Table 1: Operations in the render pipeline.

Figure 2: Going out sweetheart game. [13]

Figure 3: UML Diagram

Figure 4: RecordView Interface

Figure 5: Database Design

Figure 6: Playback Interface

Figure 7: MainMenu Interface

Figure 8: UMAR Framework

Introduction

Augmented Reality as defined by the book titled Telegeoinformatics “Augmented reality (AR) presents a particularly powerful user interface (UI) to context-aware computing environments. AR systems integrate virtual information into a person's physical environment so that he or she will perceive that information as existing in their surroundings.” The main idea behind augmented reality is for the user to be able to easily gather information about their surroundings. Since most of the work for this project will be done on an iPhone 4, this paper will mostly talk about augmented reality techniques for smart phones. The most common technique for applications to display information is to have a camera feed with text overlaid onto the feed. This allows the user to easily see what is in front of them, while at the same time quickly, easily, and in an unobtrusive way to find out information about their surroundings.

We propose a unique way to help make augmented reality on a mobile device more enjoyable and useful for the user. Augmented reality today only allows the user to observe or manipulate the information while the camera feed is still rolling. This technique hinders the user in two different ways. It stops the user from realistically recording video. This is because the user is interacting with the information and holding the device in a position to record video, which makes this task very difficult and tiring. The second problem with today's augmented reality applications is the simple fact that the user cannot playback a video with augmented reality information. We propose an application that will allow the user to record a video with

the option to have the augmented reality overlays. Then the user can save that video and either watch it on their iPhone or upload it to their computer, and with the desktop side of our application play the video back with all the augmented reality information.

Previous Work

Different Uses of Hardware

Since 1997, augmented reality has utilized a broad spectrum of hardware to deliver a unique experience for the user. The first projects were very bulky, and did not provide a great user experience, but as hardware has become more advanced projects have been producing a better experience. Not only has hardware become more powerful, but also has been extended to make use of additional hardware (e.g. GPS, electronic compasses, electronic accelerometers, etc...). Each of these pieces of hardware has the ability to add features that help make augmented reality easier to use, and also generate more useful information.

The GPS is a very key piece of hardware for any augmented reality application. GPS stands for Global Positioning System, and is a radio navigation system that allows land, sea, and airborne users to determine their exact location, velocity, and time 24 hours a day, in all weather conditions, anywhere in the world. The major limitation with GPS is the ability of the device to establish to a strong connection to the GPS satellites. This can lead to inaccurate location results from the satellites.

With GPS, applications have the capability to pinpoint where exactly the user is anywhere in the world. This is useful for the sole fact that producing relevant information about the user's surroundings becomes significantly easier. Prior methods of generating relevant information about the user's surroundings were either gathering information from the camera feed or limiting the user to a small area. With the addition of an electronic compass it becomes possible to calculate which direction along the vertical the user is currently looking. Finally with the addition of the electronic accelerometer the applications can calculate which direction along the horizontal axis the user is holding the device. With these three pieces of hardware it becomes entirely possible to not only determine information that is useful for the user, but to accurately display the information.

Unfortunately these features haven't always been around, and that is what forced early projects to develop unique combinations of hardware and software. In 1997, students at Columbia University in New York created an augmented reality device. The hardware consisted of: a backpack with a laptop, a handheld PDA with a stylus, and a head unit that was a head-tracked, see-through, head worn, 3D display. The entire package weighed around forty pounds, which compared with modern devices that have the same capability is extremely bulky [7]. The system works by having most of the work happening on the laptop in the backpack. This is where all the location information and user

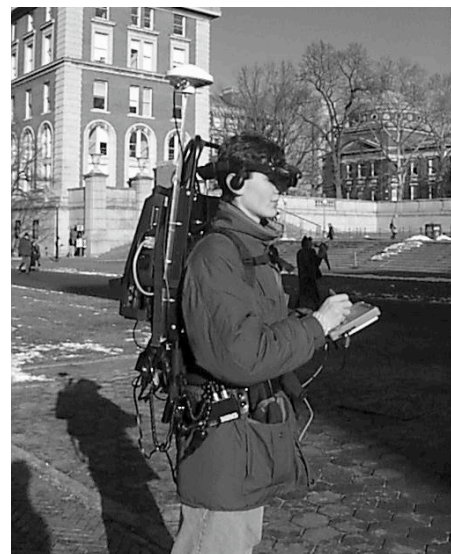


Figure 3: Prototype campus information system. The user wears a backpack and head worn display, and holds a handheld display and its stylus. [7]

interaction is used to determine what information the user should be observing. Each device communicated by sending URL requests back and forth to each other. The head unit displayed floating labels of places within its view. In order to do this the head unit would send GPS and compass information to the laptop, and the laptop would return information that the head unit should display. The PDA would also use the information that the head unit would send to the laptop and display more information on it. The laptop would send web pages to the PDA and the user could click on links to learn more about points of interest near them.

A Device-Server Model

Hardware on mobile devices is vastly inferior to that of a dedicated server. This can make some tasks for augmented reality that are computationally expensive, impossible to do on the mobile device. This problem lends itself very well to a device server model, especially since most mobile devices today have connection to the internet at all times. We will discuss two different uses of a device-server model in augmented reality: sending image data, and sending location information.

The first most used technique for a device server model is sending image data to the server for analysis. Analyzing an image can be a very computationally expensive task, and besides simple techniques, it is impossible with even today's mobile devices. Finding points of interest in an image is a very powerful tool, not only is it possible to accurately determine points of interest but how to display their information onto the screen.

At Stanford University, a research team worked on developing an augmented reality application that was capable of detecting music album covers. The phone used was a Nokia N95, and they were able to have the final product run at 30 frames per second, and have one second latency for the phone to send

image data to the server and receive the results back. The timings for the different parts of the application are represented inside Table 1 [5]. Extracting SURF

Table 1: Operations in the recognition pipeline.

Location	Operation	Latency (ms)
Phone	Align two frames	30
Phone	JPEG-encode frame	200
Network	Transmit frame	120 (WiFi), 400 (3G)
Server	Extract SURF features	100
Server	Search through k-d tree	500
Server	Check geometry	50

(Speeded Up Robust Features) features and searching through these features with k-d trees are how the application finds a matching image pair to determine what album the user is looking at. SURF is a technique of quickly identifying salient points in any given image. Then the server uses k-d trees to identify if these points are interesting. Finally the application checks these interesting points against the original image to identify if the two images are similar [5]. It is very clear that the small network latency even while on 3G is in comparable to the amount of time that it would have taken to do all the work that is on the server. Also considering that this type of task is very parallelizable, the server can easily become a cluster, thus boosting performance even more.

A very similar technique was used at MIT that created a vastly different product. An application was proposed that would send image data to a server, which would be analyzed and generate keywords based off of that image [17]. Then the server would use these keywords to send Google a search string and pull image results from that. Finally they would run those

results through their server program again to make sure that the images retrieved were similar to the original image. This was done by simply checking to see if the features in the received images are similar to the original image. The application would then send images it deemed correct along with the websites they were obtained from back to the phone [17]. This whole system provides a great user experience by allowing the user to quickly check to see if a result is relevant from the image, and then find useful data from the web pages that correlate with that image.

Another useful way to utilize a device-server model is to send location information to the server and have the server return back useful data about the surrounding area. Even though the server's task is not computationally expensive, it lends itself very well to being on a remote server because real world location data can easily be a very large amount. So much so that it is unrealistic to keep it all on the phone. Also the data can be constantly changing, making a server that is constantly updating a great solution to the problem.

At the IBM T. J. Watson Research Center, an application was created for a mobile phone that would read a barcode and then provide information about that product [8]. The phone captures an image of the barcode and then proceeds to analyze the barcode to determine its identification number. Then it sends that number to the server, which then returns data about that product [8]. This data can include anything from website reviews that are text, audio, or even a live audio stream with someone who has recently purchased the product. All this information is very useful in helping the customer get very detailed information about a

product very quickly allowing them to make good decisions on whether or not to buy the product.

Mobile Device

Since smart-phones have been becoming more powerful and popular, they have become a very popular target platform for augmented reality. This is due to several reasons: it is lightweight; the user carries it around all the time, it has a data connection at all times, and has all the features needed to create an augmented reality application. These features eliminate most of the problems that projects before focused to overcome.

At Linköping University, a team worked on demonstrating that augmented reality was in fact possible on mobile phones at the time (2001) [9]. This capability was demonstrated using both a SonyEricsson P800 and a Nokia 3650. A port of ARToolKit was created to run on both devices. ARToolKit is an image analysis library, which is specifically created for augmented reality [9]. This library was designed to find points of interest in the image stream. Then they took these points of interest and drew 3D virtual objects on top of the screen.

The ARToolKit is a completely open sourced library. It provides a simple interface to do fast image analysis algorithms on any of the platforms it supports. It also allows the user of the library to do things such as calibrate the camera, and single camera position tracking. Its tracking code is based off of finding black squares and following them throughout the video stream. This lends itself nicely if you want to place a sheet of paper that has a checkerboard pattern on it and place 3D objects on top of it.

The conclusion was that even though the system was not very complicated, as smart phones become more sophisticated the project could become more useful. Also at the time of the paper mobile phones data connection was expensive and not fast enough to be practical for the needs of the application.

Another team located at Cambridge University, started a project focused on creating an augmented reality



Figure 4: Going out sweetheart game. [13]

experience that worked well in an urban environment [13]. The main work focused on generating vectors to help represent the environment in a 3D model. This model was used to accurately place virtual 3D markers. As demonstrated in Figure 2 one application they demoed was a game where the user chooses a sweetheart, and then slides a virtual ladder under their window. Then a virtual note slides up the ladder and enters the window. All of this work was done on a small PDA, which shows the amount of work a small mobile device is capable of.

Image Analysis

The most common technique for analyzing images for augmented reality is finding salient points (prominent points) in an image. Then applying several different algorithms to filter out non-interesting points allows for easy tracking of these features. Finding salient points

in an image is considerably less time consuming than the later algorithms that are applied to these points, which is why this technique is so popular.

Analyzing just a single image can be a very powerful tool, but not as appropriate as finding points of interest in an entire image stream, especially for augmented reality applications. The reason for this is simple, the user is constantly moving around and by the time one image is analyzed it may no longer be relevant, whereas finding salient points in an image stream and keeping track of them is a much better solution to the problem.

At Carnegie Mellon University, a research team focused on a set of algorithms for detection and tracking of point features [16]. There were two main problems that had to be overcome; how to determine if a feature was worth tracking and how to follow that feature from frame to frame. For the first problem it was decided to select features that were easily tracked, and were only selected on a purely mathematical way [16]. Instead of trying to match a feature to some image in a database, an attempt was made to find features that had good texture, and didn't change drastically over time. After a feature was selected then the problem of tracking it frame to frame had to be overcome. A common technique was employed using an affine map to keep track of the feature over a 3D space. At the end of the project it was determined that it was possible to keep track of features and to do this in an efficient manner.

Finding Optimal Ways to Display Information on Screen

Since the invention of Graphical User Interfaces (GUI), lots of research has gone into how to best use the screen real estate. Since augmented reality is a completely new type of application it is tough to follow any known, GUI standards, thus more research has gone into creating an intuitive experience for the user.

At the University of Oldenburg located in Germany, a new idea for how to visualize off-screen points of interest emerged [14]. The basic problem was that current augmented reality interfaces provided a difficult to understand interface for visualizing points of interest that were not in the direct view of the user. A method was decided on of placing 3D arrows that would point in the direction of the interest point. After having many different users try this approach and common approaches that other applications used it was found that users had a significant increase in not only being able to find the point of interest faster, but remembering where and what all the interest points were.

Distinct iPhone Applications

Since the iPhone 3GS was released, augmented reality applications have exploded. The reason for this is that the iPhone 3GS was the first iPhone to have GPS, accelerometer, compass, camera, and a persistent data connection. With all of these pieces of hardware it becomes almost trivial to develop an augmented reality application. This explosion has led to many new ways to use augmented reality. For example one application helps golfers read each hole. They created a database of over 37,000 golf courses, and used this data to show the user where sand traps are located in their view, and any other features on a course [6]. Another

application tries to help color blindness. They use a series of Ishihara tests to help aid the user to determine what color any real world object is. Ishihara tests are simply a collection of images that include a spotted circle made up of two colors then numbers inside the circle that is also a different color [6]. If the person viewing the image is colorblind they will be unable to see the numbers and tell what colors are being represented. It was found that it greatly improves the user's ability to identify colors.

A major genre of augmented reality applications has become very popular on the iPhone and Android platforms. This genre is overlaying a virtual game on top of a real world scene and having the user and the game interact with real world objects. One example of this is an alien space invaders game that uses the real world environment [6].

Proposal

Purpose

This project is focused on creating a different augmented reality experience for the user. We have found that current augmented reality applications only allow the user to view the augmented overlays during the time they capture the video stream. We propose a system that will allow the user to capture a video with or without the augmented overlays and then save the video for later playback where they can also have the option to view the overlays.

The purpose of being able to watch a video later with an augmented overlay while not being in the area is the fact that sometimes the user does not care, or have the time to care,

immediately about what is in front of them, but would like to learn about points of interest at a later time. A simple case of this is a family is on vacation. They are on a guided tour, and are taking video of their trip. Instead of having to mess with all the information on the screen at that moment of time, they can go back into their hotel that night and get more in depth knowledge about where they were at.

We also plan to create a desktop side application where the user can upload their video, and watch the video from the comfort of their home and can easily remember everything they saw. Another example is the family is on vacation and would like to send the video to their friends and relatives that are not there so they can see and learn about the places they have been.

We propose to follow a similar framework as the UMAR (Ubiquitous Mobile Augmented Reality) application. The design the used can be seen in figure 8. This is the basic layout for any location based augmented reality application.

Layout of application

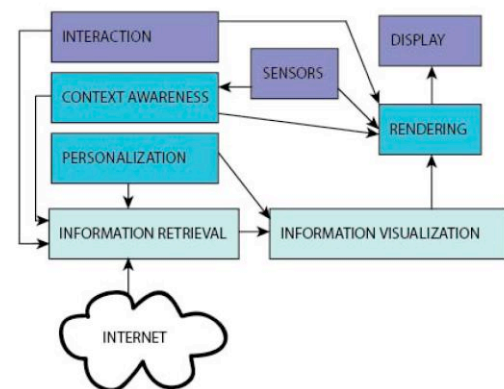


Figure 8: Overview of the UMAR Framework

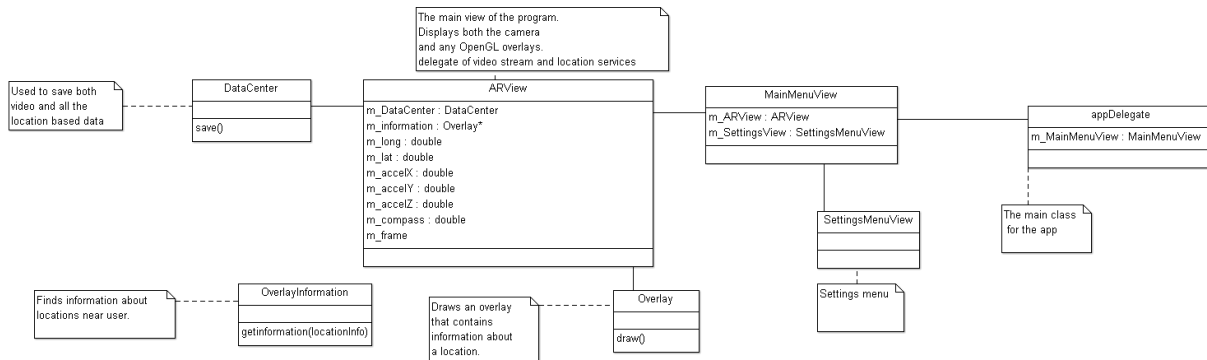


Figure 5: UML diagram

The iPhone application will be broken into three different parts. The first section of the application will be where the user records the actual video. They will have many different options inside this section. They can choose to just have a camera feed running with augmentation turned on or off. The feed will not actually be saved. This will be useful if the user does not care right that second if they save the actual video, but just about learning about points of interest. Also it gives the user time to setup what they want to capture.

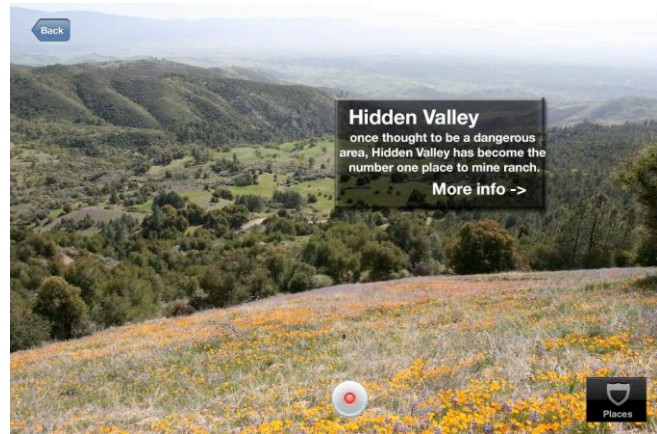


Figure 4: RecordView interface

Another option is to actually record with or without augmented reality turned on. The final option will be to look at a list of places located near them.

The actual implementation of this section of code can be found in Figure 3. The main class will be the RecordView class. This class is responsible for becoming the delegate for

gathering GPS, accelerometer, and compass information, rendering the camera feed, sending the InformationServices class the location information so that it can gather points of interest, and sending all this data to the

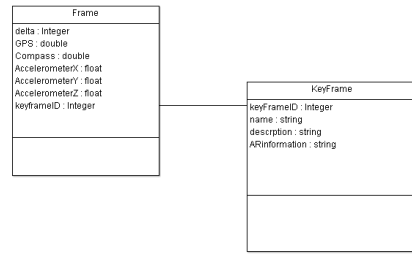


Figure 5: Database Design

DataCenter class to save it all. All the RecordView class has to do to become the delegate for the location information is declare it is a delegate and then implement the methods they need. The InformationServices class will gather all of its information from Google and Wikipedia APIs.

These APIs are very simple to utilize. They both work by the device sending a url request that has the location information in the header. Then they both return a JSON or XML array of the locations

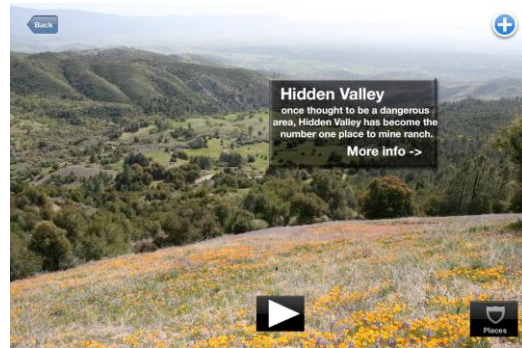


Figure 6: PlaybackView interface

near that location. The DataCenter class will be in charge of saving all the location

information and key frames information for any given video. The DataCenter class will be using CoreData as an interface to a sqlite3 database. The database schema can be seen in Figure 5.

The user interface is relatively simple for the RecordView. The screen will be filled with the camera feed. In the bottom of the screen will be a button to start or stop the recording. In the bottom right of the screen will be a button that will change to a list view of all the places near the user. Overlaid on the screen will also be the augmented overlays. On the top left will be a button to return to the main menu.

The second section of the application will be the playback of the actual video. Once again there is a lot the user will be capable of doing to the video during playback. In this section the user can choose to playback a video with the option to have the augmented overlays on. They can also during playback choose to add a key frame to the video. This will save an entry in the database that will let the user add a name and a description, also any data that is in view of that key frame. The benefit of this is to add the capability of searching the database for useful information and bringing up key frames based off that information.

The final section of the iPhone application will be a settings view that allows the user to customize their user interface and various other options. The main menu will allow the user to delete and upload videos; Figure 7 is a mock image of what it will look like.

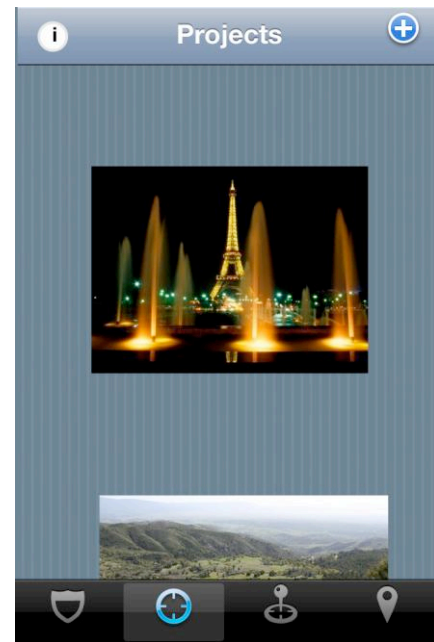


Figure7: MainMenu interface

The desktop application will allow the user to have the same capability as the playback feature on the iPhone. This includes viewing the video with or without augmentation and adding key frames. The only major difference between the iPhone version of playback and the desktop version is the desktop will have to be written in either OpenGL, or written only for Mac OSX, in which case the same library the iPhone uses (CoreGraphics) can be used there as well.

Current Work

Demo

The demo includes several features that were prototyped in order to demonstrate the plausibility of the final project. The first feature is capturing a video stream and displaying it. We utilized the AVFoundation framework. This framework allows for capture and saving of streams from multiple hardware devices, including camera and microphone. The next feature of the demo is gathering of GPS, Accelerometer, and Compass info. This is trivially done by having the main class become a delegate for each of the services and implementing their methods. They continually call the methods they need and send the updated information through them. The next feature that needed to be added was the gathering of points of interest. We decided to use Google and Wikipedia as the primary sources for generating that data, since they have easy to use interfaces, and a vast collection of data. The final feature added in the demo was displaying the information over the camera feed. We decided to use CoreGraphics for this implementation rather than OpenGL ES. There are several reasons for this. The first reason is that CoreGraphics allows for easy drawing of simple objects like translucent boxes, whereas this could be an extensive process in OpenGL. Another reason is that CoreGraphics allows for easy font rendering, where OpenGL has no known way of drawing a font, so we would have to implement that ourselves. Finally CoreGraphics makes it easy to take views from any of the apple libraries and create CoreGraphics objects from them.

Future Work

Future Features to add

We would like to add many features to this demo that will create the application that we described in this proposal. The first feature we would like to add saving the location information into the database along with actually saving the video to a file. After adding that feature the next feature is the actual playing back a video with augmented reality overlays. Then adding key frames feature, where the user can add a key frame and add name and description for that frame. Other nice features to add would be caching points of interest to help make the application run faster. Adding the places near me button to both the RecordView and the PlaybackView. The final feature to add would be the ability to export a video to a desktop machine.

Works Cited

- [1] Azuma, Ronald, et al. "A Motion-Stablized Outdoor Augmented Reality System." IEEE Virtual Reality (1999): 252-259.
- [2] Bay, Herbert, Tinne Tuytelaars and Luc Van Gool. "SURF: Speeded Up Robust Features." IEEE (2010): 1-14.
- [3] Bergman, Ruth, Hila Nachlieli and Gitit Ruckenstein. "Detection of Textured Areas in Images Using a Disorganization Indicator Based on Component Counts." Hp Invent (2007): 1-15.
- [4] Brown, Matthew, Richard Szeliski and Simon Winder. "Multi-Image Matching using Multi-Scale Oriented Patches." IEEE (2010): 1-8.
- [5] Chen, David M., et al. "Streaming Mobile Augmented Reality on Mobile Phones." Stanford (2009): 1-2.
- [6] Ci. 37 Best Augmented Reality iPhone Applications. 06 September 2011. 20 January 2011 <www.iphoneness.com/iphone-apps/best-augmented-reality-iphone-applications/>.
- [7] Feiner, Steven, Blair MacIntyre and Tobias Hollerer. "A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Enviroment." ISWC (1997): 74-81.
- [8] Guven, Sinem, et al. "Social Mobile Augmented Reality for Retail." IBM (2010): 1-3.
- [9] Henrysson, Anders and Mark Ollila. "Augmented Reality on Smartphones." IEEE (2010): 1-2.
- [10] Hollerer, Tobias, et al. "Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System." Computers & Graphics 23.6 (1999): 779-785.
- [11] Kamath, Chandrika, et al. "Salient Points for Tracking Moving Objects in Video." LLNL (2004): 1-14.
- [12] Kato, Dr. Hirokazu. Hitl. 1 January 2010. 20 January 2011 <<http://www.hitl.washington.edu/artoolkit/>>.
- [13] Reitmayr, Gerhard and Tom W. Drummond. "Going out: Robust Model-based Tracking for Outdoor Augmented Reality." Cambridge (2010): 1-10.
- [14] Schinke, Torben, Niels Henze and Susanne Boll. "Visualization of Off-Screen Objects in Mobile Augmented Reality." ACM (2010): 1-4.
- [15] Shi, Jianbo and Carlo Tomasi. "Good Features to Track." IEEE (1994): 1-8.
- [16] Tomasi, Carlo and Takeo Kanade. "Dectection of Tracking of Point Features." Shape and Motion from Image Streams (1991): 1-22.

[17] Yeh, Tom, Konrad Tollmar and Trevor Darrell. "Searching the Web with Mobile Images for Location Recognition." MIT (2010): 1-6.