Literature Survey on Combining Digital Tables and Augmented Reality for Interacting with a Model of the Human Body

Jishuo Yang
University of Calgary, Canada
jisyang@ucalgary.ca

ABSTRACT

In this paper, we present a survey of existing augmented reality implementations to determine a software and hardware approach in implementing an augmented reality API. The API seeks to combine augmented reality on a mobile device with a 3D model of the human body on a digital table. We address the advantages and shortcomings of existing augmented reality APIs and examine their compatibility with current mobile devices.

1. INTRODUCTION

In medical schools, human dissection is a fundamental part of a physician's education and an important rite of passage. However, the number of body donors frequently fall short of the huge demand of cadavers for research and teaching [1]. This causes a mounting shortage that is detrimental to physician training and is difficult to address in a social and moral context.

Modern computing advances may shed light on a technological answer to the problem, in the form of an interactive, 3D model of the human body. While it is impossible to replace the experience of a physical cadaver, it is hoped that the system can supplement physician education while reducing the need for cadavers.

The motivation for this project arises from a convergence of several new technologies: the digital tabletop, powerful and affordable handheld devices, and augmented reality platforms.

Augmented reality (AR) is the real-time integration of virtual 3D environments and the real world [2]. The real environment is analyzed through a computer camera, and the computer then renders a virtual 3D model at a designated location. Often the use of fiducial markers, in the form of non-symmetrical pictures, is used to indicate the location of a 3D object. The computer calculates the distance and location of the 3D models in relation to these markers. This technology makes it possible for us to see a projected 3D model as it would appear in the real world.

The use of AR involves heavy use of visual recognition and tracking, and is understandably computationally expensive. Many prior experiments with augmented reality were conducted with AR head-mounted-displays (HMD), where the lenses are LCD displays inches away from the viewer's eyes [2]. These displays are expensive, and must be used in conjunction with a computer, limiting its mobility. The introduction of powerful handheld technologies, such as smartphones means that it is now possible for AR calculations to be done on a mobile device.

Finally, for AR and the 3D model to be effective as a teaching device, a digital tabletop is necessary to simulate a dissection environment. The digital tabletop allows for the real-time display of markers, and offers interaction through multi-touch capabilities. This gives the system a degree of flexibility that is not possible with physical markers in traditional AR implementations.

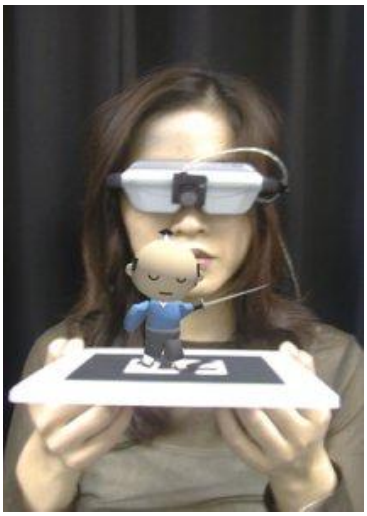

Figure 1. Augmented reality as seen through a head-mounted display, using traditional physical markers [9].

With a wide range of devices and software libraries to choose from, a literature survey is necessary to find the appropriate tools for such an implementation. This project focuses on the construction of an API that allows the display and interaction with a 3D model of a human body. In such a system, the tabletop is responsible for displaying the human body and AR markers, while the handheld device uses the markers as reference and overlays medical information on the 3D model.

This project aims to evaluate current handheld devices and AR tools to determine which is most suitable for such an API that integrates all three components.

2. EXISTING WORKS

The advent of powerful smartphones made AR applications possible on a handheld device. In contrast to the cumbersome backpack-and-HMD implementations, the portability and affordability of smartphones makes widespread adoption of AR possible. Today we are seeing some of the first AR applications released for commercial use. Examining these existing applications can give us insight into the strengths and limitations of current AR implementations and mobile hardware.

2.1 Limitations of AR on Mobile Devices

In spite of the power of modern mobile devices, we should keep in mind that traditional AR applications are implemented on the PC. Designing a mobile AR API means we have to deal with weaknesses inherent in a mobile platform.

Performance problems are a major concern in mobile devices. Visual-recognition is computationally expensive even for relatively simple markers [6]. The port for ARToolkit Plus, an AR API for the iPhone, had problems rendering more than 10 frames per second on an iPhone 3G [8]. This performance issue is further compounded with more complex algorithms used in feature-tracking, where no fiduciary markers are used. In addition, the graphics processing units (GPU) present in many smartphones are unable to render complex 3D models, either because they are not fast enough or simply do not have enough memory. Therefore any 3D models used for a mobile implementation should have its complexity reduced to achieve an acceptable memory footprint [7].

In addition to performance concerns, a handheld device also has usability problems compared to a PC. Since visual-tracking requires sight of the markers to render the models, awkward placement of markers can cause arm fatigue. Furthermore, a handheld's small screen results in a low field of view for the user. This can not only cause eye-strain after long periods of use, but might also make the users stand unacceptably far from the digital tabletop in order to see the entire table.

2.2 GPS-Tracked Mobile Implementations

Considering that visual tracking is the dominant area of AR research since its inception, perhaps it is ironic that the first commercial AR applications predominantly make use of GPS tracking. As of this writing, a cursory search of "Augmented Reality" on Apple's App Store reveals that out of a collection of 28 results, there are none that use any form of visual tracking, and 2 simply overlay graphics on top of the phone's camera feed without any tracking [4]. The remaining 26 applications use the iPhone's built-in global positioning system (GPS), accelerometer, and magnetometer to determine the user's current location and direction.

One of the most iconic applications using this tracking method is Layar, developed by the company of the same name [5]. The application determines the user's location and direction using the aforementioned method, and overlays user-desired information from the internet to the camera feed. This combined footage is rendered on the phone's screen, which may include directions to nearest attractions, real estate pricing and location, and public transit information.

One major benefit of GPS-tracking is that it is not very computationally expensive: the program needs only to read the coordinates given by the instruments to determine the device's location in space. This eliminates the heavy demand placed on the CPU for visual recognition and noise-filtering expected in visual-tracking. In addition, the use of GPS-based tracking enables these applications to track themselves and locations in a 3D environment that is global in scope, whereas visual-tracking applications are only limited to the device's immediate environment. However, these benefits come at the cost of accuracy. Visual-tracking is accurate to millimeters, depending on the size of the marker and the quality of the camera, while the error in GPS-tracking may vary as much as meters. This means that GPS-tracking, while less computationally-demanding, is not nearly accurate enough to satisfy the requirements for this project.

2.3 Visually-Tracked Implementations

Visually-tracked augmented reality applications use a variety of visual computing algorithms to search for pre-defined markers or features. These are divided into marker-tracking and feature tracking.

In marker-tracking, the marker used is usually in the form of a square monochrome symbol with highly defined borders. Feature-tracking, sometimes known as markerless-tracking, does away with these possibly intrusive symbols, and instead utilizes more sophisticated visual-recognition algorithms to look for a given image instead of a symbol. Unlike the monochrome symbol, the image may be of any shape or color.

2.3.1 Art of Defense

The Art of Defense is a tower-defense style AR game implemented for the Nokia N95 [18]. Two players must construct offensive towers to defend a central tower from attacking enemies. The towers are assigned to markers represented by arrows, and are placed on top of an assembled game board assembled from hexagonal tiles. Enemies attack the tower from predefined routes. The player must place offensive towers on optimal tiles to prevent enemies from reaching the central tower. This game uses the closed-source library Studierstube ES, an AR implementation designed for mobile devices [27].

The Nokia N95 has hardware specifications similar to that of the original iPhone. While it did not run into any limitations using the AR library, it did suffer on the graphics rendering front. The phone was unable to render sophisticated 3D models quickly, a performance ceiling likely also present in other mobile devices.

Figure 2. The Nokia N95 is unable to render complex models in Art of Defense due to hardware limitations [31].

The game is an effective proof-of-concept of running a visually-tracked AR application on a mobile device. However, there are limitations in the system arising from the lack of a tabletop component.

Each piece of the gaming area is represented by a board with a printed marker. This makes the game harder to manage as players need to find the correct board piece to play, which is distracting in an application conducted in real-time. Furthermore, adding a

board piece to the existing play area risks moving the pieces already in play, possibly upsetting the location of existing AR models. By contrast, a digital table is able to dynamically generate markers on its screen through a touch interface, and the user is not put at risk of upsetting the existing markers on screen. This can give this game the possibility of computer-generated maps, enlivening the play experience.

2.3.2 ARhrrrr!

ARhrrr is an AR shooting game implemented for the Nvidia Tegra platform through the Studierstube ES library [19]. The game uses feature-tracking techniques to render a zombie-infested city block on top of a map. The player plays as a helicopter pilot tasked to rescue civilians by shooting the zombies by tapping the touchscreen on the mobile device.

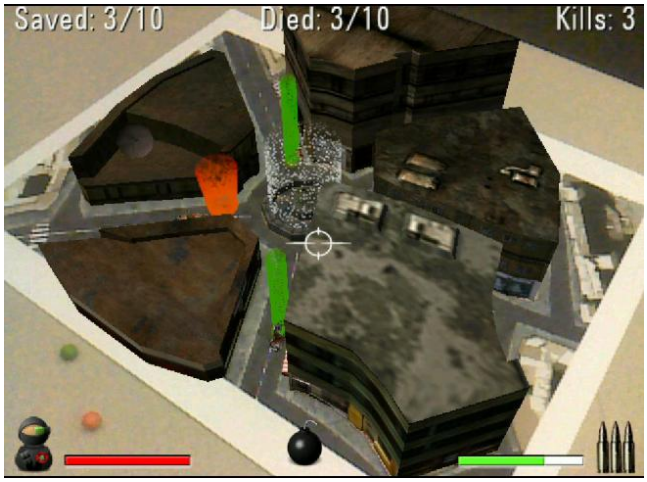

Figure 3. Feature-tracking enables ARhrrr to track and display 3D models on top of terrain features of an image [19].

This AR implementation uses noticeably more complex models than those from Art of Defense, and demonstrates the Tegra platform's superior hardware capabilities. The entire game, combined with feature tracking, runs well on the platform, showing that perhaps other open-source libraries can also run well on the Tegra.

The game could still be improved by a dynamically-rendered map, however. The integration of a digital surface has the capability to render and scroll existing defined-features, thereby simulating a changing field of view. This allows for extensionality and variety that are the main strengths of an AR application when integrated with a digital tabletop.

2.4 Existing AR Libraries and Toolkits

There exist a number of commercial and open-source AR toolkits for developers. Since short implementation time and code-reuse are a priority for the project, each major AR library is evaluated for compatibility, functionality, and efficiency.

It should be noted that since the majority of these APIs are used for PC development, there are performance concerns when run on a mobile device.

2.4.1 ARToolkit and Derivatives

ARToolkit is one of the first open-source visual-tracking libraries available [9]. First introduced in 1999, its strength is its out-of-the-box approach by integrating a simple graphics library and real-time marker tracking.

Since it is written in C, it is compatible with the iPhone. It also has a Java port in the form of NyARToolkit [22] and Android with AndAR [23]. While it is no longer being actively developed, its popularity means that it is well-documented and is used by a large community.

However, due to its age, it does have a number of limitations. The performance of its tracking algorithm is not as good as those of the more modern libraries [8]. Its tracking is also less accurate and has a higher chance of error. In particular, it is more prone to finding false positives in cases where lighting makes one marker resemble another [30].

Finally, ARToolkit only features marker-tracking, in the form of a black square with a non-symmetric symbol in the center. One of the weaknesses of this implementation is the need of the marker to be always visible. This means that any user interaction with the markers will cause the models to vanish, which is not ideal for a tabletop that is centered on touch-based interaction. A workaround might include the use of multiple markers or tracking based on the environment. In order to make use of feature-tracking, a visual-recognition algorithm such as SIFT needs to be integrated in the API, which takes additional time to implement.

2.4.2 ARTag

ARTag is a more recent library that uses digital encoding instead of binary correlation to identify markers [21]. Its improved detection algorithms enable more accurate tracking compared to ARToolkit in cases where there is lighting variation in the markers [30]. It includes a library of 2002 unique markers.

In addition, while ARToolkit uses the square marker boundary to distinguish the marker, ARTag offers a limited ability to hypothesize about broken borders so that markers remain identified even when a part of the border is obscured.

Despite its advantages, however, there does not currently exist a mobile implementation of ARTag, and any mobile implementations would have to be ported from C++ to work with a mobile platform. ARTag is also no longer in active development, and does not benefit from support of a large community like ARToolkit.

2.4.3 Studierstube ES

Studierstube ES is one of the only libraries designed ground-up with mobile devices in mind [27]. It provides support for Windows CE, Symbian, and iPhone platforms. It supports rendering through OpenGL, and boasts a small memory footprint to compensate for the poor memory bandwidth in mobile systems [7]. It is in active development at the Graz University of Technology and is object-oriented and extensible.

It uses modified SIFT and Ferns algorithms to provide feature-tracking, and claims to be twice as fast as ARToolkit on mobile platforms [29].

While this library seems to be the ideal one to use for this project, it remains closed-source and the developer is not currently licensing it. It is included for consideration as a candidate when it becomes open to licensing.

2.4.4 BazAR

BazAR is a C++ augmented reality library designed for feature-based tracking [20]. Implemented in 2007, it is the most modern of the open-source AR libraries available. BazAR uses the Ferns feature-tracking algorithm, enabling it to track features much more quickly than with SIFT.

The main problem with BazAR is that it does not support any mobile platform out-of-the-box, so it would have to be ported to run on the iPhone. In addition, since it is a newer library, the community support for this library is non-existent. It is, however, well-documented on its website.

Figure 4. BazAR utilizes the Ferns algorithm to rapidly track surfaces without the need for fiduciary markers [20].

2.4.5 ARKit

ARKit is an open-source AR library natively implemented for the iPhone [24]. It uses the GPS-tracking method and is therefore unsuitable for this project. However, since there are no other native AR libraries for the iPhone OS, this library can help with the porting of other libraries to the iPhone, by supplying the camera code necessary to get the camera feed.

2.5 Implementations for Feature Tracking

Two of the leading approaches in feature-recognition are SIFT and Ferns [17]. Both algorithms have been used to implement feature-tracking in AR libraries and applications, including BazAR and Studierstube ES.

If a marker-tracking library is chosen for development, the more efficient algorithm of the two will be used to implement feature-tracking. It is important to examine these two approaches to determine which is more suitable for a mobile platform.

SIFT (Scale Invariant Feature Tracking) uses "keypoint localization, feature description and feature matching" [17] to determine the location of the image. Without getting into the algorithm itself, SIFT does have a reputation of being a computationally expensive, comparison-heavy approach [16]. By comparison, Ferns finds interest points in the video feed to maximize the probability of finding the marker, and is much faster in PC-based tests [16].

Wagner et al. studied the performance of both algorithms on the mobile phone in 2008 [17]. It was found that while SIFT did have a higher CPU overhead, Ferns utilized much more memory to store the interest points. While high memory consumption may not have been significant to desktop machines, it is much more taxing on mobile phones, which have significantly less memory. The study concluded that despite the computational differences, SIFT and Ferns had a similar level of performance [17].

The result of this study is especially important with respect to Ferns. Since mobile phones use the same memory for both the CPU and GPU, high memory consumption for Ferns precludes the use of large textures for the AR 3D models.

2.6 Compatibility with Mobile Platforms

2.6.1 iPhone

The iPhone is one of the two possible platforms for use in the project. It has advantages in being a popular platform with a large selection of applications and community support.

However, one major hurdle is that there are no existing open-source AR libraries that work with the iPhone platform out-of-the-box. It would be necessary to port any C/C++ AR libraries for use on the iPhone.

Performance of visual-tracking and 3D model rendering on the iPhone is also a cause for concern. A sample ARToolkit implementation for the iPhone was unable to track a marker at more than 10 frames-per-second [8]. Feature-tracking on the iPhone is likely to be even slower owing to the complexity of the algorithms. In addition, the iPhone is likely unable to render the highly complex 3D human body overlays for use in AR. Therefore should iPhone be chosen for use for this project, it is developed with the knowledge that the application most likely will run with a low frame rate, and that newer generations of the iPhone are needed before acceptable performance can be achieved.

2.6.2 Android

Google's Android platform is the second possible candidate for this project. Its primary advantage over the iPhone is having two native AR libraries that include NyARToolkit and AndAR. While neither library offers feature-tracking, using them does save time in porting a preexisting API over to the iPhone OS.

Performance of the Nexus One Android phone uses a more powerful ARM processor and GPU than the one in the iPhone, but its performance in AR applications is not known.

Deserving of particular mention is Nvidia's Tegra platform, which uses a dual-core ARM processor and an Nvidia GoForce GPU [12]. With a superior CPU and GPU, it is better equipped than other platforms to handle the load of visual-tracking and 3D rendering. It is used in the aforementioned ARhrrr project and demonstrates that it is capable of smooth feature-tracking, albeit with the optimized Studierstube ES library.

2.7 AR on Digital Tabletops

Finally, existing AR implementations on digital tabletops are rare. Most implementations are completed for a physical table, such as ARTHUR for urban development [10]. These make use of paper markers, and therefore lack the flexibility compared to a digital marker rendered in real-time. The closest approximation of a digital tabletop comes from Dr. Chow Lam's implementation of

AR into a tabletop trading card game [11]. However, this implementation used a plasma TV laid on its side instead of a tabletop, and AR is used to simulate a touch screen. This is not necessary in digital tabletops, which have multi-touch features.

One of the reasons for the lack of digital tabletop AR might be the inconvenient location of the tabletop's cameras. The use of AR on a tabletop generally requires a mobile overlooking camera, which is not found on traditional tabletop surfaces. This means that this project is one of the first in the field to deal with the use of AR on a digital table, as well as its use in tandem with a handheld device.

3. PROPOSED SOLUTION

This project envisions an API that relies on the digital tabletop to render the 3D model and markers, and a handheld device to display additional medical information in AR. The medical information includes 3D overlays such as muscle and bone structure. In addition, the user should be able to manipulate and request information for areas of the model viewed under the handheld, which will appear in the form of pop-up windows. Additional interaction methods are available on the table, such as zooming, panning, and rotating. Any manipulation to the 3D human body on the digital table is communicated to the handheld device. The device then executes the corresponding transformations to the augmented 3D information on its own field of view.

The goal of the API is to be able to have it adapted to use any other 3D model. Possible extensions might include AR display of topographical maps, or 3D displays of medical CT scans. To this end, there are three major design goals for this API:

1. Code reuse: to reduce implementation time, existing tools and libraries should be used in favor of coding new implementations.
2. Support and documentation: the API should be well documented and all its components well-supported by their developers. This means avoiding libraries that are obscure or no longer supported.
3. The API should be general and extensible. It should have an interface for user extensions, support new models, and modular to allow for swapping of its components.

3.1 Hardware Selection

The 3D model used is the LINDSAY virtual human project developed by the University of Calgary Evolutionary and Swarm Design Research Group. The majority of the project is programmed in Cocoa and Objective-C. There is already a limited implementation of LINDSAY on the iPhone. In addition, the Agile Software Engineering Group has LINDSAY running on the Smart Table with a Mac Mini.

In keeping with design goal #1, the iPhone and the Smart Table should be the platforms on which to implement the project. While there are concerns whether iPhone's hardware is up to the task, the more powerful Nvidia Tegra is discounted because it would involve rewriting much of LINDSAY's code. Furthermore, the more powerful iPad also fails to meet the requirements for this project due to the lack of a rear-facing camera.

3.2 Software Toolkit and Libraries

Since this project does not aim to recreate an AR environment from scratch, it will make use of existing AR libraries. However, there are no native implementations of any existing AR library for the iPhone, barring the closed-source Studierstube ES.

The alternative is to choose the best C/C++ implementations of an AR library and port it to the iPhone, which is between ARToolkit and BazAR. BazAR is a superior choice in this case since it is more modern, and offers native feature-tracking. The only downside with this library is the lack of community support.

One of the main hurdles of this move is getting access to the code to access iPhone's camera API. Fortunately, there is a guide on porting the C++ implementation of NyARToolkit to the iPhone, complete with camera code [25]. In addition, the open-source ARKit can also be used as reference to access the camera.

4. IMPLEMENTATION PLANNING

4.1 Implementation Schedule

The first stage of the implementation of the API is to be able to render a part of the LINDSAY human model on the iPhone, and have it oriented to a static paper marker. This stage involves modifying LINDSAY code so that it can be rendered on the iPhone, and porting BazAR to the iPhone. This stage is done independently of the digital tabletop, and focuses mainly on enabling basic AR functionality on the iPhone.

Since the LINDSAY model on the iPhone is the AR overlay to the 3D model on the digital tabletop, the next stage enables the iPhone to track the zoom and orientation of the tabletop 3D model. This is accomplished by embedding markers on the tabletop model, tracked from the iPhone through marker-recognition. This phase completes the multi-marker tracking of the tabletop model, and overlays the iPhone model on top of it.

In the third stage, we will stop using symbols for fiduciary markers and fully take advantage of the feature-tracking algorithms in BazAR. Instead of using markers, the iPhone will track the tabletop model through textures on the model itself.

Finally, in the last stage we will add the ability to display information pop-ups. These windows display information about the area of the body under the iPhone camera, such as medical charts and CT scans.

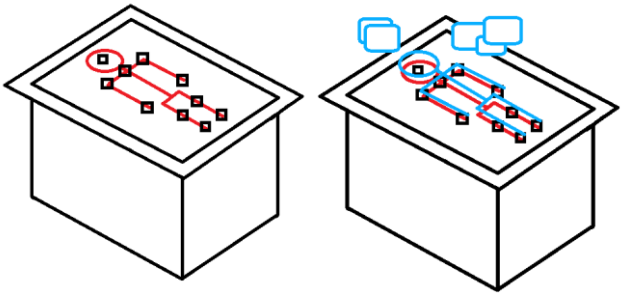

Figure 5. Implementation concept drawing. The left image depicts the digital table as seen through the naked eye, the red figure being the 3D human body, black squares representing embedded markers. The right image shows the augmented scene as seen through the iPhone, complete with 3D overlay of the human body and pop-ups.

4.2 System Design

The API is broken down into several main modules to allow for greater extensibility and component-swapping. This makes it easier to switch to different AR libraries to support other platforms.

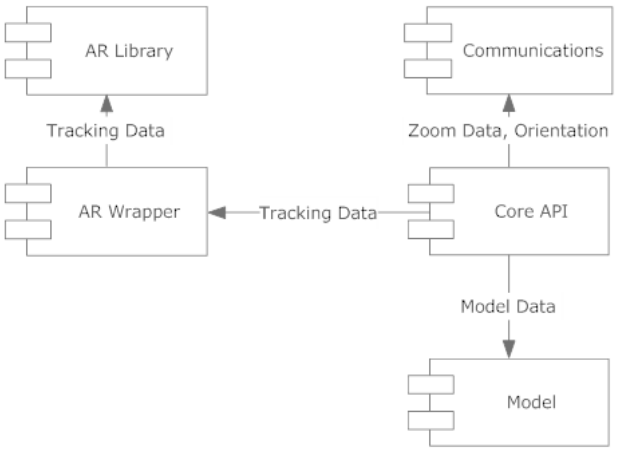

Figure 6. System Component Diagram

The AR library component is responsible for visual tracking of the 3D human model. BazAR with ported iPhone camera code is the chosen candidate for this module. Should Studierstube ES ever be released as open-source, it would most likely replace BazAR due to performance concerns.

The AR wrapper is used to facilitate the swapping of the AR component. It is an interface between BazAR functions and those called by the core API.

The core API handles most of the graphics functionality. This includes transformations on the 3D model and the iPhone AR overlay. It also allows for the use of alternate 3D models, not just the human body.

The model component is the 3D model data, including both the tabletop model and the overlay models. These are displayed by the core API and are replaceable with other models.

The communication component allows the iPhone to connect to the tabletop server in order to receive information on the degree of zoom, camera panning, and rotation on the tabletop model. This data is used by the core API to transform the iPhone overlay. This component will likely be using either REST or SOAP protocols.

4.3 Alternative Design

In the case where it becomes impossible for BazAR to perform well on the iPhone, or it becomes too difficult for BazAR to be ported, an alternative is to do the AR processing on the digital tabletop's computer instead of the mobile.

The PC outputting to the digital tabletop is responsible for rendering the 3D LINDSAY model like in the original design. In addition, the PC also runs the unaltered implementation of BazAR. Since BazAR is known to perform well on the PC [15], there should be no performance issues like with the original design.

The iPhone then sends its own video feed to the PC via a wireless connection. The PC uses BazAR to calculate the location of the iPhone, overlays the 3D model, and sends the augmented video stream back be displayed on the iPhone.

This design completely circumvents the performance issues with the mobile device. However, the issue is whether a wireless connection has sufficient bandwidth to transfer two video streams at the same time. In addition, this model also scales poorly when multiple mobiles are used, since the network load increases significantly whenever a new phone is added.

5. CONCLUSION

We have presented a survey of existing augmented reality libraries and mobile devices. We also explored the appropriate software-hardware configuration necessary to implement a human body AR API on a digital tabletop in conjunction with a mobile device.

We originally assumed that modern smartphones have advanced to the point where visual-tracked AR can be done in real-time. However, it turned out that since most of the existing AR libraries are developed for the PC, they do not include optimizations that would have made real-time AR possible on a mobile.

The survey does reveal that, if real-time performance is not a requirement, marker and feature AR tracking on a mobile device is achievable. For the purposes of this project, the combination of BazAR and the iPhone 3GS provides our API with the highest chance for code reuse, support, and extensibility. While this project is implemented with the knowledge that it is unlikely to run in real-time on current mobile hardware, with the rapid advancement of mobile technology, we are optimistic that sufficiently powerful devices will arrive in the near future.

6. REFERENCES

[1] Boulware, L.E., Ratner, L.E., Cooper, L.A., LaVeist, T.A. and Powe, N.R. 2004. Whole Body Donation for Medical Science: A Population-Based Study. In Clinical Anatomy. Wiley-Liss Inc., Wilmington, DE, 17, 570-577.

[2] Azuma, R.T. 1997. A Survey of Augmented Reality. In Presence: Teleoperators and Virtual Environments. MIT Press, Cambridge, MA, 6, 355-385.

[3] Schmalstieg, D. and Wagner, D. 2007. Experiences with Handheld Augmented Reality. In Proceedings of the 2007 6th IEEE and ACM international Symposium on Mixed and Augmented Reality (November 13 - 16, 2007). Symposium on Mixed and Augmented Reality. IEEE Computer Society, Washington, DC, 1-13. DOI= http://dx.doi.org/10.1109/ISMAR.2007.4538819

[4] Apple App Store. Retrieved March 24, 2010 from Apple: http://app-store.appspot.com/?url=viewGrouping%3Fid%3D25204%26mt%3D8%26ign-mscache%3D1

[5] Augmented Reality Browser – Layar. Retrieved January 20, 2009 from Layar: http://layar.com/

[6] Wagner, D.; Schmalstieg, D.; , "Making Augmented Reality Practical on Mobile Phones, Part 1," *Computer Graphics and Applications, IEEE* , vol.29, no.3, pp.12-15, May-June 2009 doi: 10.1109/MCG.2009.46

[7] Wagner, D.; Schmalstieg, D.; , "Making Augmented Reality Practical on Mobile Phones, Part 2," *Computer Graphics and Applications, IEEE* , vol.29, no.4, pp.6-9, July-Aug. 2009 doi: 10.1109/MCG.2009.67

[8] ARToolKit v4.4 iPhone. Retrieved March 24, 2010 from ARToolworks, Inc.: http://www.artoolworks.com/ARToolKit_iPhone.html

[9] ARToolkit. Retrieved January 22, 2010 from University of Washington: http://www.hitl.washington.edu/artoolkit/

[10] Penn, A. 2004. Augmented Round Table for Architecture and Urban Planning. Retrieved January 22, 2010 from University College London: http://www.vr.ucl.ac.uk/projects/arthur/

[11] Lam, A.H.T., Chow, K.C.H., Yau, E.H.H., and Lyu, M.R. 2006. ART: Augmented Reality Table for Interactive Trading Card Game. Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications, June 14-April 17, 2006, Hong Kong, China. DOI=http://doi.acm.org/10.1145/1128923.1128987

[12] Nvidia Developer Zone: Tegra. Retrieved January 26, 2010 from Nvidia Corporation: http://tegradeveloper.nvidia.com/tegra/

[13] Gillet, A., Sanner, M., Olson, A., Weghorst, S. and Winn, W. 2004. Computer-Linked Autofabricated 3D Models for Teaching Structural Biology. Retrieved January 16, 2010 from University of Washington: http://www.hitl.washington.edu/publications//r-2004-45/r-2004-45.pdf

[14] ARISER NET: Augmented Reality in Surgery. Retrieved January 20, 2009 from University of Oslo: http://www.ariser.info/index.php

[15] Scherrer, C., Pilet, J., Fua, P., and Lepetit, V. 2008. The haunted book. In *Proceedings of the 7th IEEE/ACM international Symposium on Mixed and Augmented Reality* (September 15 - 18, 2008). Symposium on Mixed and Augmented Reality. IEEE Computer Society, Washington, DC, 163-164. DOI= http://dx.doi.org/10.1109/ISMAR.2008.4637347

[16] Ozuysal, M.; Fua, P.; Lepetit, V.; , "Fast Keypoint Recognition in Ten Lines of Code," *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on* , vol., no., pp.1-8, 17-22 June 2007 doi: 10.1109/CVPR.2007.383123 URL: http://ieeexplore.ieee.org.ezproxy.lib.ucalgary.ca/stamp/stamp.jsp?tp=&arnumber=4270148&isnumber=4269956

[17] Wagner, D.; Reitmayr, G.; Mulloni, A.; Drummond, T.; Schmalstieg, D.; , "Pose tracking from natural features on mobile phones," *Mixed and Augmented Reality, 2008. ISMAR 2008. 7th IEEE/ACM International Symposium on* , vol., no., pp.125-134, 15-18 Sept. 2008 doi: 10.1109/ISMAR.2008.4637338

[18] Huynh, D. T., Raveendran, K., Xu, Y., Spreen, K., and MacIntyre, B. 2009. Art of defense: a collaborative handheld augmented reality board game. In *Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games* (New Orleans, Louisiana, August 04 - 06, 2009). S. N. Spencer, Ed. Sandbox '09. ACM, New York, NY, 135-142. DOI= http://doi.acm.org/10.1145/1581073.1581095

[19] ARhrrr! Retrieved March 24, 2010 from Georgia Institute of Technology: http://www.augmentedenvironments.org/lab/research/handheld-ar/arhrrrr/

[20] BazAR: A vision based fast detection library. Retrieved March 24, 2010 from École Polytechnique Fédérale de Lausanne: http://cvlab.epfl.ch/software/bazar/

[21] ARTag. Retrieved March 25, 2010 from Columbia University: http://www.artag.net/index.html

[22] NyARToolkit. Retrieved March 26, 2010 from NyARToolkit Project: http://nyatla.jp/nyartoolkit/wiki/index.php?FrontPage.en

[23] AndAR – Android Augmented Reality. Retrieved March 26, 2010 from AndAR Project: http://code.google.com/p/andar/

[24] iPhone ARKit. Retrieved March 26, 2010 from ARKit Project: http://www.iphonear.org/

[25] Augmented Reality on the iPhone using NyARToolkit. Retrieved March 26, 2010 from More Than Mechanical: http://www.morethantechnical.com/2009/07/01/augmented-reality-on-the-iphone-using-nyartoolkit-w-code/

[26] Wei-Chao Chen; Yingen Xiong; Jiang Gao; Gelfand, N.; Grzeszczuk, R.; , "Efficient Extraction of Robust Image Features on Mobile Devices," *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on* , vol., no., pp.287-288, 13-16 Nov. 2007 doi: 10.1109/ISMAR.2007.4538870

[27] Langlotz, T. 2010. Studierstube ES. Retrieved March 14, 2010 from Graz University of Technology: http://studierstube.icg.tu-graz.ac.at/handheld_ar/stbes.php

[28] Langlotz, T. 2010. Studierstube ES Videos. Retrieved March 14, 2010 from Graz University of Technology: http://studierstube.icg.tu-graz.ac.at/handheld_ar/videos.php

[29] Wagner, Daniel; Reitmayr, Gerhard; Mulloni, Alessandro; Drummond, Tom; Schmalstieg, Dieter; , "Real-Time Detection and Tracking for Augmented Reality on Mobile Phones," *Visualization and Computer Graphics, IEEE Transactions on* , vol.16, no.3, pp.355-368, May-June 2010 doi: 10.1109/TVCG.2009.99

[30] Fiala, M.; , "Comparing ARTag and ARToolkit Plus fiducial marker systems," *Haptic Audio Visual Environments and their Applications, 2005. IEEE International Workshop on* , vol., no., pp. 6 pp., 1-2 Oct. 2005 doi: 10.1109/HAVE.2005.1545669

URL: http://ieeexplore.ieee.org.ezproxy.lib.ucalgary.ca/stamp/stamp.jsp?tp=&arnumber=1545669&isnumber=32980

[31] Art of Defense: a Mobile AR Game with Sketch-Based Interaction and Dynamic Multi-Marker Building Techniques. Retrieved March 28, 2010 from Georgia Institute of Technology: http://www.augmentedenvironments.org/lab/research/handheld-ar/artofdefense/