1

Addressing Challenges with Augmented Reality Applications on Smartphones

J. Benjamin Gotow, Krzysztof Zienkiewicz, Jules White, and Douglas C. Schmidt
Email: {ben.gotow, krzysztof.k.zienkiewicz, j.white, d.schmidt}@vanderbilt.edu

Vanderbilt University, Nashville, TN USA

Summary. The popularity of smartphones equipped with GPS and geomagnetic sensors has spurred mobile application developer interest in augmented reality (AR), which presents highly contexualized, spatially relevant information that enhances user knowledge of their immediate surroundings. AR applications typically mesh relevant information with user views of the physical world. Prior research has focused on interfaces built with custom hardware, but a smartphone equipped with GPS, a camera, and a geomagnetic sensor is an attractive alternative to traditional solutions. These devices can be programmed to present context-sensitive information to users without needing custom hardware.

This paper examines three key challenges facing AR developers on mobile devices and presents solutions applicable to modern mobile platforms, such as Apple's iPhone and Google Android-based smartphones. First, we investigate methods of filtering raw sensor data and present an algorithm that eliminates sensor noise. Second, we explore the process of implementing a "magic lens" interaction metaphor by overlaying perspective-rendered graphics on the device's camera using OpenGL and UIKit. Third, we provide an efficient technique for fetching and caching geographically tagged points of interest from a server.

1.1 Introduction

Augmented reality (AR) overlays highly contexualized, spatially relevant information on user views of the physical world [1, 2]. In a typical mobile AR application, users point their smartphones at objects of interest and view the augmented display that is drawn on the phone's screen. The display provides additional information about their environment, *e.g.*, to make them aware of information that is not immediately visible, such as the dates of upcoming events or ratings of nearby restaurants.

AR has been used to create mixed reality video games for use in education [3, 4] and handheld tools for underground infrastructure visualization [5]. It has also made inroads in the medical domain. For example, AR has been used to give surgeons information about the position of internal organs and the adjustments needed for needle biopsy [2].

Meshing content onto users' views of their environments (*e.g.*, as shown in Figure 1.1) is a fundamental challenge of AR, requiring methods for determining user locations and estimating the area within their field of view. In applications where environments contain known identifiable markers (such as 2D barcodes) image analysis of these markers can be used to infer the camera's position and frame of reference. Markers are typically designed

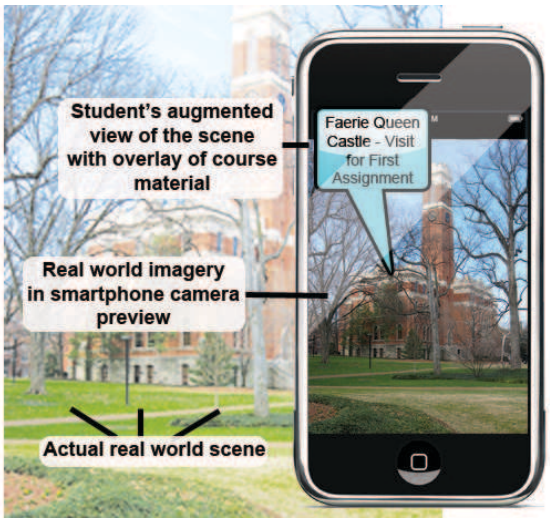

Fig. 1.1: An AR Application Overlaying Labels on Real-world Objects

for ease of recognition and planted in fixed locations within the environment. This class of solutions for pre-prepared environments has been studied extensively [6, 1, 7, 8].

In open environments that have not been previously instrumented with markers, on-board sensors or natural feature recognition can be used. Identifying naturally occurring features of an environment and inferring the user's location is computationally intensive, however, and traditional solutions have been relegated to research labs due to the high cost of the custom hardware required [1]. Today's smartphones are an attractive alternative to custom hardware since they are equipped with Internet access, cameras, and GPS and geomagnetic sensors. The prevalence of smartphones—combined with the ease with which new software can be delivered—makes them a promising platform for building AR applications and conducting future research.

Due to data caching and processing power requirements, natural feature recognition is beyond modern mobile device capabilities. Approaches that centralize data processing [9] are undesirable for consumer mobile applications due to the high cost of scaling server-based solutions and the relatively low-bandwidth networks connecting mobile devices to servers. Several applications perform detection and pose estimation of 2D barcodes and fiducial markers [10, 7], which can be used in AR applications that display special content on objects branded with 2D markers [11]. Marker tracking, however, is a specialized use-case not suited for general-purpose, open environment applications, such as providing nearby restaurant reviews or information about events in a city.

GPS and geomagnetic sensors in modern smartphones require significantly less processing power and can work in open environments lacking custom markers needed for image analysis. The use of GPS and geomagnetic sensors in commodity smartphones are, however, accompanied by significant challenges [9], such as the limited accuracy of the GPS sensors in and the noise present in sensor data. For example, the noise in geomagnetic heading values can cause jitter in onscreen information presentation.

The paper provides the following contributions to R&D on mobile AR:

- We show how 3D interaction-enabled elements can be overlaid on smartphone camera video in real-time under the processing constraints constraints typical of modern smartphone hardware and presented on the displays readily available in a wide number of smartphones.
- We present an algorithm that filters sensor data in real time, eliminating noise and allowing for a smooth display based on GPS and geomagnetic sensor data alone. We show that limited processing speeds are not a barrier to filtering sensor data necessary to create smooth AR displays.
- We show that a large number of geographically tagged data points can be stored on a central server, retrieved, and cached using geographic ranges without the processing required to compare latitude/longitude values under the bandwidth constraints typical of modern commodity smartphone hardware.

The remainder of this paper is organized as follows: Section 1.2 examines key challenges facing developers of AR applications on modern smartphone devices; Section 1.3 presents solutions to these challenges based on our *Vanderbilt AR Toolkit* (VART), including an approach for efficiently storing and retrieving geotagged data, a filter for effectively reducing geomagnetic sensor noise, and methods for creating perspective rendered overlays in real-time; Section 1.4 evaluates the benefits of our solutions empirically by analyzing database query speed for point-of-interest retrieval and quantifing the benefits of our sensor filtering algorithm; Section 1.5 compares VART with related work; and Section 1.6 presents concluding remarks.

1.2 Challenges of Mobile AR Application Development

This section presents four key challenges facing developers of AR applications for modern smartphone platforms.

1.2.1 Challenge 1: Mobile 3D Solutions are Non-optimal and Hard to Mesh with Camera Imagery

The magic lens interaction metaphor [12, 13] (where widgets are placed above content to reveal hidden information) is common in AR applications, but is hard to produce on resource-constrained smartphones. Information displayed over the camera preview must be transformed and rendered in real-time according to information about the user's position, orientation, and heading within the environment. Rendering accuracy is important since AR applications offer a rich user experience by precisely associating overlaid information with elements in user surroundings.

Overlaying information directly on top of physical objects obviates the need for context in the information displayed and results in more intuitive data presentation. User experience thus deteriorates quickly when accuracy is lost. Incorrectly aligned overlays provide misleading information because the context assumed by the user is not accurate. Previous AR applications have achieved fast rendering using OpenGL or by moving processing to a server and streaming video to embedded devices [14].

Graphics libraries (such as OpenGL) are available on modern smartphone platforms and can render three dimensional models in real-time. On most devices, pixel fragment processing is done on dedicated graphics hardware, so rendering does not block other

CPU-intensive operations, such as the loading of points of interest (POI). The use of OpenGL on smartphone platforms introduces other challenges, however, *e.g.*, rendering content and displaying it over live camera video requires integrating low-level services provided in mobile OS APIs.

Using OpenGL to display interface elements is also undesirable on modern mobile platforms. Once perspective-rendered content is displayed onscreen, it is hard to perform hit testing because OpenGL ES 1.1 does not provide APIs for "picking mode" or "selection" used to determine the geometry at particular screen coordinates. When controls are rendered in a perspective view, it is hard to determine whether touch events lie within the control bounds. While OpenGL supports perspective 3D rendering under the processing constraints typical of modern mobile smartphones, it is not optimal. Section 1.3.1 describes how VART addresses this challenge with an alternative graphical solution employing nested view objects that display perspective distorted content while preserving user interaction with overlaid visuals.

1.2.2 Challenge 2: Real-time Estimation of Frame of Reference is Computationally Demanding

AR requires high-performance techniques for mapping a virtual environment onto the real-world coordinate space. As users move their smartphones, the virtual viewport must update quickly to reflect changes in the camera's orientation, heading, and perspective, so it is essential to gather information about the device's physical position in the environment in real-time. Traditional approaches [6, 1] to frame of reference estimation depend on identifiable tokens embedded in the environment or computationally-intensive image processing of natural markers.

Image processing techniques must be optimized extensively to fit within the hardware constraints imposed by mobile devices. Detection and frame of reference estimation of identifiable markers (such as two-dimensional barcodes) is an option for closed environments that can be instrumented with such markers. This approach, however, is less suitable for AR applications in outdoor environments since instrumenting the environment with markers prior to the applications use is unlikely.

Attempts to perform natural feature detection in open environments on commodity mobile devices have been largely unsuccessful [9] since they use large amounts of cached data and significant processing power. Devising a strategy for determining the device's position, heading, and orientation with high accuracy is a significant challenge given the limited processing capabilities of mobile devices. Section 1.3.2 describes how VART addresses this challenge using GPS and geomagnetic sensors for frame of reference estimation.

1.2.3 Challenge 3: Geomagnetic Sensor Noise Makes Orientation Estimation Hard

Modern mobile smartphones contain a number of sensors that are applicable for AR applications. For example, cameras are ubiquitous and accelerometers and geomagnetic sensors are available in many smartphones. Geomagnetic sensors provide information about user headings, which can be combined with GPS data to estimate field of view.

The geomagnetic sensors in popular mobile devices present unique problems, however, since they do not provide highly accurate readings. To map the virtual AR environment into a real-world coordinate space, sensor data must be accurate and free of noise

that causes jitter in rendered overlays. The reduction of noise thus represents a significant challenge confronting AR software.

The Savitzky-Golay smoothing filter [15] is a natural approach to removing sensor noise. This filter leverages the fact that data from most types of rotations can be modeled by a fairly small number of standard equations. Different regression tests can thus be run iteratively on a portion of the most recent data to identify the regression with the highest coefficient of determination and use the resulting equation to adjust the incoming point. Unfortunately, the Savitzky-Golay smoothing filter is not usable in mobile AR application since running a single regression algorithm is expensive and doing it multiple times for a single incoming point at 40 Hertz is infeasible. Section 1.3.2 describes how VART addresses this challenge via an algorithm that efficiently filters sensor noise within the processing constraints of modern mobile smartphones.

1.2.4 Challenge 4: Filtering Geotagged POIs by Proximity is Computationally Intensive

Mobile AR applications focus on providing information about immediate user vicinity. In areas of high information density (such as a city) there may be a dozen POI within a few hundred feet of a user. Efficiently storing a large number of geotagged points and retrieving those most relevant to individual users is hard due to the large number of comparisons necessary to identify which item(s) are near user(s). Geotagged points change frequently, so mobile devices need to query a central database server regularly to retrieve information about nearby POI.

Unfortunately, there are several problems with this straightforward approach. Querying a database of geotagged points by specifying latitude/longitude ranges is not practical for mobile applications with many users. It is also inefficient to place bounds on two numeric columns in a large data set because comparisons must be performed on each row to compile the result. Databases index content for faster retrieval, but numerical values cannot be efficiently preprocessed for faster querying. While speed problems could be mitigated by subdividing points into separate tables based on geographic region, a popular AR application might offer thousands of POI within a small geographic area. A different approach is thus required to obviate the need for complex database queries.

Requesting and retrieving data on a mobile smartphone is also problematic for several reasons. WiFi and cell network connectivity consumes battery rapidly and users may observe rendering interruptions or a drop in frame rate as data from remote servers is received and processed. Caching data on the mobile device partially alleviates the need for network retrieval. This approach is also problematic, however, since it is hard to aggregate geotagged points and filter them in a latitude/longitude window with limited processing power. Section 1.3.3 describes how VART addresses this challenge by quantizing geotagged points into geographic blocks and fetching, caching, and filtering on the block level, which consumes less processing as users navigate their environment.

1.3 The Vandy AR Toolkit

This section describes our solutions to the challenges presented in Section 1.2 based on the *Vanderbilt AR Toolkit* (VART) for iPhone and Android smartphone platforms.[1]

[1] VART is open-source software available at `code.google.com/p/vuphone`.

1.3.1 Using Hardware Accelerated 3D APIs to Display Perspective Rendered Content

Section 1.2.1 describes that meshing perspective rendered graphics onto smartphones is hard due to limited control over their camera image. It is also hard to determine what object in 3D space users are interacting with since screen coordinates do not map directly to coordinates in the 3D environment once a projection has been applied. Hardware-accelerated rendering can be achieved using the OpenGL graphics library on some mobile platforms but fails to adequately address the challenge in Section 1.2.1. Below we present an alternative solution based on nested view objects that display perspective distorted content while preserving user interaction with overlaid visuals.

An alternate approach utilizing nested views. To easily enable user interaction with rendered content, VART employs nested view objects to which a 4x4 visual transformation matrix is applied. When the view hierarchy is rendered, the transformation matrix is applied to each view allowing for basic perspective distortion of the content rendered in each view. The benefits of this approach are that hit testing can be achieved by applying the transformation matrix to incoming touch locations and platform-standard view objects allow the display of standard graphical interface elements.

We use Apple's UIKit framework to implement this solution on the iPhone. UIKit provides sophisticated APIs for building graphical user interfaces composed of nested views. Each view has bounds declared relative to its parent and draws itself. All views may contain subviews; interaction events proceed down a call chain to the lowest view capable of handling an event of that type.

UIKit also allows an AR application to specify a 4x4 visual transformation matrix for each view, which supports basic perspective graphics. The transformation matrix is applied to graphics output when each view draws its respective content and is also applied to user interaction events as they are passed into the view stack. Since the transforms are applied to events, hit testing is handled transparently regardless of the transformation matrix.

We created a transformation matrix approximating distortion from a camera lens and used it to render buttons and other controls with a perspective projection applied. This solution obviates the need for other graphics libraries, such as OpenGL. It also enables user interaction with rendered content, which is important for mobile AR applications.

Meshing content with the camera image. Meshing rendered content with imagery from the smartphone camera required overcoming platform-specific issues on the iPhone and Android platforms. For example, restrictions built into Apple's 3.0 iPhone OS prevent camera image data from being used in the graphics pipeline. Since direct access to the camera image in memory is not provided, however, it is not possible to use OpenGL or another graphics library to display the camera image and the rendered POI elements.

Although Apple provides an API to take individual frames from the phone's camera, this approach yields low frame rates unsuitable for an AR application. Using a single graphics pipeline to draw the camera image and the overlaid content seemed attractive since images can be distorted and adjusted prior to their display, but our inability to pull frames from the camera rapidly made this option unappealing. Instead, a generic camera preview can be used to display the camera image on the screen separately, which provides little flexibility since image data cannot be manipulated and its display is beyond the application's control. Transparent content can be displayed in a layer on top of the camera preview, however, achieving the desired effect.

1.3.2 Using GPS and Geomagnetic Sensors to Estimate of Device Position and Orientation

Section 1.2.2 identified problems using traditional image recognition techniques for device position and orientation estimation on mobile devices. We now present an alternative made possible by the sophisticated sensors in commodity smartphone hardware. Prior work has focused on using specifically designed markers and token detection for location and frame of reference estimation on mobile phones [2, 6, 10], which provide accurate position and pose estimation of markers placed in user environments.

Our intended use of AR to display nearby POI does not require highly accurate pose estimation or position information. Instead, geographic location information within a few meters and heading data within a few degrees is acceptable. To avoid computationally expensive image processing and the need for environmental markers, therefore, VART uses onboard GPS and geomagnetic sensors sensors available in modern commodity smartphones to pinpoint the user on a latitude/longitude grid and compute the POI within their field of view. Points are then rendered over a camera image, allowing use of the phone as a lens to view an augmented version of the world, as shown in Figure 1.2. iPhone and

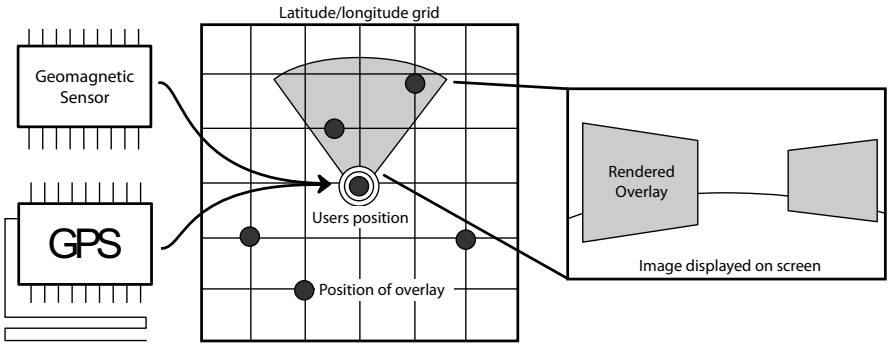

Fig. 1.2: Sensors Identify Device Frame of Reference and Update Screen.

Android devices feature GPS hardware and geomagnetic sensors, and both operating systems provide APIs for accessing data from this hardware within third-party applications.

Prevailing issues: sensor noise and accuracy. We tested Android and iPhone smartphones (such as the iPhone 3G and the Android G1) that are representative of modern mobile phone technology. While these devices offer an impressive range of features, their reliance on commodity sensor hardware is problematic. For example, these smartphones incur a great deal of input noise and have less accurate hardware than traditional mobile AR systems [14]. The geomagnetic sensors of these phones were noisy, even when the phone was lying flat on a table. This variance in heading information yielded visual jitter and degraded the presentation of POI on-screen. The GPS sensors in these smartphones also provided less accurate readings than dedicated AR devices, which often utilize GNSS surveying equipment with accuracy to one centimeter.

Although 100% accuracy is not required, reasonable accuracy is helpful in AR applications that overlay 3D geometry to match real-world features. These applications often

render geometry on top of a view of the phone's camera. When user location and heading cannot established with high accuracy, overlaid geometry may be misaligned.

Possible approaches to filtering sensor noise. We identified several algorithms to optimally filter incoming heading data. Ideally, an algorithm should work in the following conditions: (1) device is held steady, (2) device is rotated at a uniform speed, and (3) device is moved semi-randomly. Pattern recognition algorithms run on the most recent data would select the correct filter. Unfortunately, such algorithms require computing power beyond the capacity of modern smartphones.

If we eliminate pattern recognition (which is the most processor intensive part of the above algorithm) another approach emerges: the Savitzky-Golay smoothing filter [15]. Unfortunately, this technique is not usable in an AR applications on smartphones for the reasons discussed in 1.2.3.

A lightweight and portable solution. The compass filtering algorithm shown in Figure 1.3 extends Finite Impulse Response filters [16], with added statistical analysis for data exclusion and outlier analysis. It can be customized for different noise levels by a small list of parameters. The filter structure shown in Figure 1.3 contains two ring buffers

Variables/Functions:

R = *Ring Buffer of Received Data*
O = *Ring Buffer of Outlier Data*
$|R| = |O|$ = *Maximum Allowable Size of Buffer*
$size(buffer)$ = *ReturnsCurrentSizeofBuffer*
p_i = *A compass reading as a Single Precision Float*
$Z(p_i) = (p_i - mean(R))/stdDev(R)$
Z_{range} = *Maximum Allowable Deviation*
$outlierDirection(p_i) = pi > mean(R)?1 : -1$
$enqueue(buffer, p_i)$ = *Adds p_i to the Buffer*

Algorithm:

```
filtered(p_i) =
        if size(R) < |R|: enqueue(R, p_i)
        else:
            z_i = Z(p_i)
            if abs(z_i) ≤ Z_range:
                enqueue(R, p_i)
                clear(O)
            else: enqueue(O, p_i)
        if size(O) = |O| :
            side = outlierCluster()
            ∀p_j ∈ O
                if outlierDirection(p_j) = side:
                    enqueue(R, p_j)
                    clear(O)
        return mean(R)
outlierCluster() =
            int sum = 0
            ∀p_j ∈ O
                sum+ = p_j - mean(R)
            return signum(sum)
```

Fig. 1.3: The Compass Filtering Algorithm

of set capacity, one for the recent data and one for outlier data. The filter starts in an uninitialized state, accepting all incoming points and enqueuing them into the data buffer. After we reach capacity, each new point's z-score (which is a statistic for measuring the deviation of a point from the mean of the sample) is calculated. If the z-score is within an acceptable range, we enqueue the corresponding reading into the data buffer and clear the outlier buffer. Otherwise, we enqueue the reading into the outlier buffer.

If the outlier buffer reaches its capacity, we determine the direction of the outliers by computing on which side of the mean the majority of the outliers lie. We then enqueue all of the outliers in this majority to the data buffer, thus flushing it, and clear the outliers buffer. We repeat this process each time a new sensor reading is available. When asked for the filtered value, we return the mean of the data buffer.

Calculating mean and standard deviation are the most computationally expensive operations in our compass filtering algorithm. After the initialization stage, however, these calculations can be optimized to constant time operations by keeping track of the current sum and variation. When a new point comes in, therefore, we only have to remove the old point from the current sum/variation and add in the new one. Our approach lends itself to extension via subclassing so that the filter parameters can vary dynamically.

1.3.3 A Grid-based Approach to Data Storage and Retrieval

Section 1.2.4 described the problems surrounding the storage and retrieval of many POI. Below we present a highly scalable solution to the problem of data retrieval, caching, and filtering on both the server and mobile device using a grid-based approach (shown in Figure 1.4) that progressively loads content from web sources based on GPS coordinates. A mapping function places each point denoted by its latitude/longitude into an

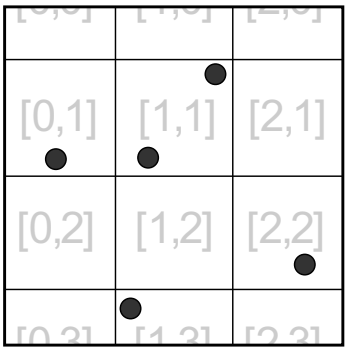

Fig. 1.4: Geotagged POI are Discretized into a Grid of Blocks.

indexed two-dimensional grid. Each square in the grid contains all points within a specific geographical area, and may be loaded by querying the database for the indexed coordinate values. Indexing the contents of the database using discretized latitude and longitude values obviates the need for numeric comparison and queries bounded by latitude and longitude values. Queries may specify an exact block index and retrieve a group of points within a predefined geographic area.

Dividing available content into a latitude/longitude grid and fetching it in discrete blocks has several advantages. Information can be requested by specifying an index to a particular block within the grid and stored based on grid coordinates, alleviating complex retrieval queries on a central server. Caching retrieved data is also straightforward since data can be stored and retrieved on the device based on the block index. Purging cached data based on its distance from the user's current location does not require iterating through each cached point. Instead, entire blocks can be quickly purged based on their discretized latitude and longitude values.

Dividing content into geographic blocks maps well onto the presentation space, where POI must be displayed/hidden as the user moves toward/away from an area. Blocks may

be partitioned into a small geographic size so that a fixed number of blocks are displayed at a time, corresponding to a few miles in each direction. Filtering blocks of points is much more efficient than processing each point and also requires constant evaluation time, regardless of the number of points present in the area. Hiding and showing POI one-by-one can yield poor application performance in high data density areas.

1.4 Empirical Results

This section presents empirical data that evaluates our techniques and algorithms described in Section 1.3.

1.4.1 Evaluating the Compass Filtering Algorithm

Below we present an experiment assessing the efficacy of our VART compass filtering algorithm described in Section 1.3.2. We sample a typical geomagnetic sensor and demonstrate favorable results produced in real-time.

Experimental setup. We used an Android Dev Phone 1 running Android 1.5 to collect measurements. The sensor was sampled at highest possible rate (roughly 36 Hz). The data was stored to an SD card via a Java application.

Raw sensory data was saved while the device was rotated at a uniform angular velocity on a magnetically insulated rotating mechanism. An adapter was then used to feed this time-stamped data into the filtering application in real-time. The resulting filtered measurements were then recorded and plotted on a time versus angle graph from which noise reduction was then calculated.

Hypothesis. Plotting the raw sensory output provides a general idea of the noise levels to reduce. The noise is most visible when the device is held steady. When it is rotated at a uniform angular velocity, noise becomes almost non-existent. An effective filter must therefore eliminate the corrupted data while preserving accurate measurements.

Analysis of results. Analysis of our filter on real-life data suggests that we are doing just that. Figures 1.5, 1.6, and 1.7 depict a graphical representation of our filter's performance. In a worst-case scenario (*i.e.*, when the device is held steady) we achieved a

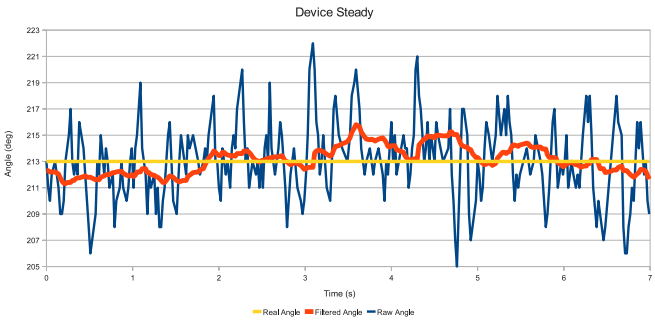

Fig. 1.5: Noise Observed When the Device is Held Steady

60% noise reduction. When the data is most accurate (*i.e.*, rotation at uniform angular velocity) we still eliminate over half the noise.

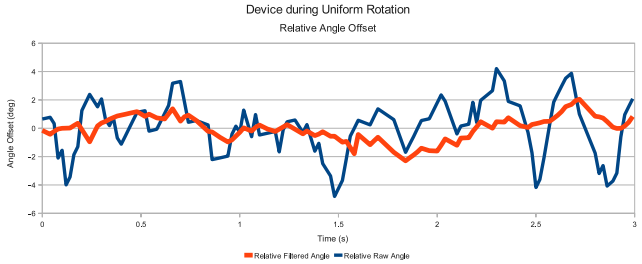

Fig. 1.6: Relative Angle Offset When the Device is Under Uniform Rotation

The results in Figures 1.5, 1.6, and 1.7 show a significant reduction of sensor noise when our algorithm is employed, even in the worst case scenario of the device lying stationary. Prior to filtering, the geomagnetic sensor was shown to produce values ranging $\pm 4.8^\circ$ at rest and we reduced the margin of error to $\pm 2^\circ$. This reduction is a significant improvement, confirming that our filtering algorithm is effective and reduces overlay jitter observed by the end user of a mobile AR application.

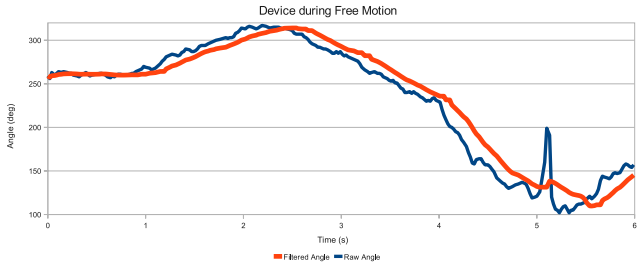

Fig. 1.7: Noise Observed When the Device is Experiencing Freehand Motion

Using data from the geomagnetic sensor of a commodity Android smartphone, we confirm that sensor noise is problematic. Our experiment also demonstrates that this algorithm is within the processing capabilities of modern smartphones. AR applications can therefore be developed to leverage GPS and geomagnetic sensors for frame of reference estimation without significant levels of jitter due to noise in orientation information.

1.4.2 Evaluating Database Retrieval of Quantized Data Points

Section 1.3.3 presented an approach to efficiently storing and retrieving data points using a grid based on latitude and longitude values. To test that avoiding numeric comparisons improves retrieval speed, we designed an experiment for measuring the speed of various database queries on a large set of geocoded points.

Experimental setup. All experiments were conducted on an Apple Powerbook with a 2.53 GHz Intel Core 2 Duo processor, 4 gigabyes of 1067MHz DDR3 RAM running OS X version 10.6.2 and mySQL 5. Queries were executed and timed using PHP 5.3.0. The results of each query were not processed, so recorded times indicate time spent performing queries only.

A single table containing 1,000,000 rows was created in a mySQL database on the machine executing the queries. Each row in the table consisted of an integer id, double latitude value, and an integer bucket. Buckets were assigned based on the latitude values so that 1000 buckets were evenly filled with 1000 rows each. A standard mySQL index was created on the bucket column.

Hypothesis. Storing geotagged points in discrete blocks within the database and retrieving them based on block index is much faster than performing numerical queries that specify upper and lower bounds on latitude, longitude values.

Analysis of results. We ran three types of queries on the database and noted average response time in microseconds for 200 queries, each returning 1,000 matching rows from 1,000,000 rows, as shown in Table 1.1. Our results confirmed that querying a large data set

Table 1.1: Average Query Response Time

Query Type	**Response Time**
Latitude range (latitude column indexed)	581700 µs
Latitude range (latitude column not indexed)	284600 µs
Specific latitude bucket	5209 µs

of geocoded points based on latitude/longitude values is a performance issue and that our solution presented in Section 1.3.3 offers dramatic performance benefits by organizing data points into discretized numerical buckets. Retrieving records based on discretized numerical buckets was exponentially faster than querying for the equivalent numerical range of latitude values. Since AR applications generally load a number of records at a time, the loss of granularity in queries for discrete buckets is not an issue and this approach will dramatically decrease server load. The poor performance observed when querying points without our approach suggests that mobile smartphones would not be capable of caching a large number of points and filtering them for display without the optimization described in Section 1.3.3.

1.5 Related Work

This section compares our work on smartphone-based AR applications with related work. The techniques employed in this paper are inspired by earlier work in mobile AR, data filtering, and magic lenses. Location and frame of reference estimation is a fundamental issue in AR and has been addressed in two primary ways in recent literature. In applications where environments can be instrumented with easily identifiable markers or contain a limited number of known natural features, image analysis techniques are optimal and provide highly accurate results. This class of solutions has been studied in great detail [6, 1, 7, 8].

Techniques utilizing sensors present a viable alternative in open, unprepared environments and have been presented in [5]. Likewise, [9] presents a hybrid approach using image recognition to refine frame of reference information derived from onboard inertial sensors. Although this approach helps increase accuracy in open uncontrolled environments, future research is needed to reduce the requirement for large amounts of pre-prepared environment data. It is likely that methods for filtering sensor data (such as our solution in 1.3.2) partially obviate the need to refine sensor data using image analysis.

To cache and retrieve points of interest rapidly, we employ a coordinate quantization technique similar to "loxels" [17], which organized image descriptors used for pose estimation by natural feature recognition into a location-based grid that could be loaded incrementally and provided the inspiration for our method of POI storage. We adapt this approach to store and retrieve geotagged data points and quantify the benefit of querying a database based on discrete blocks instead of numeric latitude and longitude ranges.

Our approach to data smoothing leverages qualities of the Savitzky-Golay filter [15]. It takes advantage of the fact that data from most types of rotations can be modeled by a fairly small number of standard equations. This insight provided the inspiration for our filter for applications where the regression required by Savitzky-Golay filter is not tractable due to constraints on processing power.

1.6 Concluding Remarks

Today's smartphones are promising platforms for AR applications since they are portable, ubiquitous, and provide the processing power and sensor capabilities necessary for AR applications. This paper identified several challenges in developing AR applications for the iPhone and Android platforms and showed how our *Vanderbilt AR Toolkit* (VART) provided acceptable solutions to these challenges. Our work on VART has yielded the following lessons learned:

- **POI retrieval based on numeric geographic ranges is infeasible.** Retrieving geotagged points from a database table within a specific numeric geographic range is costly. Quantizing points into a grid of discrete blocks is a more efficient solution.
- **Discretizing POI locations eliminates costly comparisons.** Discretizing latitude and longitude values allows geotagged points to be indexed and retrieved rapidly from a database.
- **Hit testing in OpenGL is laborious.** Hit testing three dimensional content rendered in OpenGL is hard due to the lack of selection and picking modes in OpenGL ES.
- **Sensor data requires processing to remove noise.** Raw data from smartphone geomagnetic sensors contains significant noise that results in jitter in rendered overlays unless corrected.
- **Existing smartphone platforms are capable of delivering magic lens AR**, but additional work is needed to identify other forms of AR that can be supported, *e.g.*, a hybrid form of AR utilizing onboard sensor data and flucidial marker detection could allow for impressive massively-multiplayer AR games.

References

1. D. Wagner, T. Pintaric, F. Ledermann, and D. Schmalstieg, "Towards massively multi-user augmented reality on handheld devices," in *In Third International Conference on Pervasive Computing*, 2005.
2. R. Azuma, "A survey of augmented reality," *Presence*, vol. 6, pp. 355–385, 1995.
3. S. Kirkley, "Creating next generation blended learning environments using mixed reality, video games and simulations," *TechTrends*, vol. 49, no. 3, pp. 42–53, May 2004.
4. J.-Y. Huang, M.-C. Tung, H.-C. Keh, J.-J. Wu, K.-H. Lee, and C.-H. Tsai, "A 3d campus on the internet — a networked mixed reality environment," pp. 282–298, 2009.
5. G. Schall, E. Mendez, E. Kruijff, E. Veas, S. Junghanns, B. Reitinger, and D. Schmalstieg, "Handheld augmented reality for underground infrastructure visualization," *Personal Ubiquitous Comput.*, vol. 13, no. 4, pp. 281–291, 2009.
6. D. Wagner, "Multiple target detection and tracking with guaranteed framerates on mobile phones," in *ISMAR'09*, 2009.
7. M. Mohring, C. Lessig, and O. Bimber, "Video see-through ar on consumer cell-phones," in *ISMAR '04: Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 252–253.
8. D. Schmalstieg and D. Wagner, "Experiences with handheld augmented reality," in *ISMAR '07: Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 1–13.
9. Z. Zhou, J. Karlekar, D. Hii, M. Schneider, W. Lu, and S. Wittkopf, "Robust pose estimation for outdoor mixed reality with sensor fusion," in *UAHCI '09: Proceedings of the 5th International Conference on Universal Access in Human-Computer Interaction. Part III*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 281–289.
10. D. Schmalstieg and D. Wagner, "Mobile phones as a platform for augmented reality," in *IEEE VR 2008 Workshop on Software Engineering and Architectures for Realtime Interactive Systems*. Shaker Publishing, 2009, pp. 43–44. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.144.6950
11. M. Billinghurst, R. Grasset, H. Seichter, and A. Dünser, "Towards ambient augmented reality with tangible interfaces," in *Proceedings of the 13th International Conference on Human-Computer Interaction. Part III*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 387–396.
12. E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. Derose, "Toolglass and magic lenses: The see-through interface." ACM Press, 1993, pp. 73–80.
13. J. Looser, "Ar magic lenses: Addressing the challenge of focus and context in augmented reality," Master's thesis, University of Canterbury, 2007.
14. D. Perritaz, C. Salzmann, D. Gillet, O. Naef, J. Bapst, F. Barras, E. Mugellini, and O. Abou Khaled, "6th sense— toward a generic framework for end-to-end adaptive wearable augmented reality," pp. 280–310, 2009.
15. A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, July 1964. [Online]. Available: http://dx.doi.org/10.1021/ac60214a047
16. L. R. Rabiner, *Theory and Application of Digital Signal Processing*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1975.
17. G. Takacs, V. Chandrasekhar, N. Gelfand, Y. Xiong, W.-C. Chen, T. Bismpigiannis, R. Grzeszczuk, K. Pulli, and B. Girod, "Outdoors augmented reality on mobile phone using loxel-based visual feature organization," in *MIR '08: Proceeding of the 1st ACM international conference on Multimedia information retrieval*. New York, NY, USA: ACM, 2008, pp. 427–434.