KHARMA: A KML/HTML Architecture for Mobile Augmented Reality Applications

Alex Hill\textsuperscript{1}, Blair MacIntyre\textsuperscript{2}, Maribeth Gandy\textsuperscript{3}, Brian Davidson\textsuperscript{4}, Hafez Rouzati\textsuperscript{5}
GVU Center Georgia Institute of Technology

\textbf{ABSTRACT}

Widespread future adoption of augmented reality technology will rely on a broadly accessible standard for authoring and distributing content with, at a minimum, the flexibility and interactivity provided by current web authoring technologies. The growing number of augmented reality platforms for mobile devices suggests that a single browser for viewing this content may be just over the horizon. The ideal solution for fostering broader adoption is an open architecture that empowers millions of web authors by leveraging the tools and content already available to them. We introduce KHARMA, an open architecture based on KML for geospatial, marker and relative referencing combined with standard browser supported HTML5 and JavaScript technologies for content development and delivery. Our main contribution is a re-conceptualization of KML that turns HTML content formerly confined to balloons into first-class elements in the scene. We introduce a namespace extension, KARML that gives authors extensive control over the presentation of HTML content and its spatial relationship to other content. This combination lets users rapidly develop and host rich interactive mobile augmented reality content using existing HTML authoring tools, client-side JavaScript scripting, AJAX-style database communications and multi-user session-controlled HTTP web-hosting. This architecture also introduces a bridging strategy for content delivery on commodity mobile devices based on the use of surveyed geographic locations and synthetic backgrounds. A reference browser implemented for the iPhone platform is described along with a number of ongoing projects that are using the technology.

\textbf{Keywords:} Augmented Reality, World Wide Web, Authoring

\textbf{Index Terms:} I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Virtual Reality; bdavidson@gatech.edu

1 INTRODUCTION

Since it was first demonstrated by Ivan Sutherland in 1965, augmented reality research has been largely confined to the realm of research laboratories where the expensive hardware and custom software it required could be found. Recently, powerful mobile devices with GPS and orientation sensors such as the iPhone have made video-see-through augmented reality (AR) only a few clicks away for millions of people. This development looks to be an excellent match for AR technology since mobile devices are pervasive and frequently used outdoors where the overlay of content onto the physical world is likely to be of value. A number of applications like UrbanSpoon, TwitARound and NearestTube are now available to guide users to restaurants, Twitter feeds and public transportation within their midst. Although these applications have served to introduce mobile users to AR, their development has relied more on easy access to hardware than on any significant decrease in the complexity of authoring AR content. Companies dedicated to building AR applications such as Wikitude, Layar, and Accrossair have since developed SDKs and protocols that allow individuals to either develop their own applications or publish their own content as a separate channel in one of these dedicated browsers. Although it has yet to be realized, this trend suggests that a longstanding vision of a single browser through which all AR content is viewed may be just over the horizon [5,12].

This concept of authoring all content for a single viewer is analogous to how the web browser operates as a single gateway to the web. Authors develop more with a focus on content than on the idiosyncrasies of the device rendering it, all the while retaining control over its production and distribution. Unfortunately, the current state of AR browser authoring severely limits the expressivity of content developers. Authors are forced to mold their content to limited template, given minimal tools for providing client-side interactivity and must contend with restricted control over the publishing and delivery of their content. In a sense, this almost linear model of augmented reality content delivery mirrors the early days of HTML 1.0 where content remained largely static without tools such as CSS, JavaScript and HTML5 canvas. The introduction of each new media form, from film to the Internet, has demonstrated that it will only reach maturity when it is sufficiently approachable by people with different backgrounds and skill sets. The AR browser holds the promise of bringing to mobile AR authors expressive tools on a par with those being used today to build rich interactive Web 2.0 applications. Instead of merely creating tools that are analogous, appropriating and reusing as much of existing web architecture towards AR development has the potential to empower millions of new AR content authors.

Researchers in AR technology appreciate the multiplying effect of building and distributing development tools for broad adoption, and they have a rich history of building such systems [18]. Beyond the difficulty of supporting tools for general use in a research environment, the impediments to broader adoption of laboratory-developed toolsets include the specialized computers and sensors in use and the need to manually install or configure software. Marker-based toolkits such as ARToolKit and the recent Flash based incarnation, FLARToolkit, have been reasonably successful because they rely on equipment such a webcam that many users already have.

In our own lab, we have been investigating both handheld mobile AR and other more traditional AR approaches using dedicated devices such as hybrid tracking systems and head mounted displays (HMD). Given the status of broadly accessible authoring for AR, we were motivated to develop a mobile
architecture that would allow us to easily prototype and deploy AR applications using markers, natural feature tracking, GPS/gyrosopes and dedicated trackers. We felt that an ideal ecosystem was one parallel to web technologies that lets users author content without compilation, control the behavior of client viewers at runtime and build applications that display and interact with content from any number of distributed hosts. After surveying a number of alternatives we came to the conclusion that two technologies already in widespread use, the KML markup language used by Google Earth and contemporary Web 2.0 standards, could be combined into a powerful and flexible AR authoring platform. This architecture we are calling KHARMA, KML/HTML Augmented Reality Mobile Architecture, centers around the re-conception of the KML language within the context of handheld AR. The most significant conceptual change we introduce to KML involves making HTML content a first class citizen in the scene. We are introducing an extension to the markup, KARML, which not only lets this balloon content reside undecorated in the scene but also gives the user precise control over the position, orientation and scale of that HTML content. Our architecture extends 3D support into the 2D context through user triggered animations and JavaScript events analogous to those already provided to HTML content.

Along with authoring content through a combination of KML and web standards, the KHARMA architecture addresses some of the practical issues related to mobile AR authoring. First, we introduce the concept of multiple simultaneous active. Like the tabs on a typical browser, individual channels have their own namespace and are prevented from implementing cross-site scripting. Second, we introduce a strategy for mitigating inaccurate and frequently unavailable tracking data on mobile devices through the use of surveyed locations and synthetic backgrounds. The inaccuracy of commercial-grade GPS can easily create a scenario where locations actually behind the user are indicated to be in front of the user. We allow the user to search nearby for, move within range of and indicate their presence at surveyed locations we call GeoSpots. Finally, we extend the multiple channel concept to include the sharing of GeoSpot and infrastructure resources between channels. Infrastructure includes 3D models used not for rendering but instead for determining collisions with, occlusions against and content placement relative to physical buildings and terrain in the scene. We feel an approach that uses publicly accessible repositories of GeoSpot and infrastructure content lets AR authors to focus their efforts more on content development and less on reconstructing the physical environment.

After a section covering related work, in section 3 we detail the specifics of the KHARMA architecture and the KARML extension to KML it introduces. This same section also introduces the concepts of shared infrastructure and our bridging strategy for mitigating accuracy on mobile devices. The final section details our canonical implementation of the KHARMA architecture on the iPhone platform and describes a number of ongoing projects that are taking advantage of this platform.

2 RELATED WORK

Since Vannevar Bush first described his hypothetical "memex" device [researchers have been seeking new ways to browse and create connections between all types of information [5]. From the beginning of AR research, systems were created that took data with spatial meaning and attached it to the real-world objects and locations it pertained to. From merging ultrasound imagery with the patient [2] to providing operating instructions for a printer visually registered with the physical components [6], it was clear early on that the promise was linked to information and our need to consume and explore it in a spatial context. Yet until recently, the technology, tools, and infrastructure prevented AR from becoming a ubiquitous and widely available medium for commonplace information access.

A few years after the initial AR applications outdoor systems began to emerge. Homemade hardware and large backpack systems were not ready for deployment in the marketplace, but the application ideas presented in these early systems were prescient; anticipating the need to bring information and media off the 2D page and into the physical world. The MARS system [10] allowed developers to create "situated documentaries", hypermedia narratives situated in outdoor environments. The "Touring Machine"[9], built with MARS, was a campus information system that contained many features of modern commercially available mobile AR systems including 3D and 2D augmentations registered with the location to which they pertain, and the ability to follow AR links that fed traditional web content to a handheld computer for browsing. The TINMITH system was not just for browsing 2D and 3D AR content but supported in situ creation of content [17]. This "construction at a distance" technique used line of sight techniques, head and body movements, and various carving and painting methods to allow users to author complex virtual objects. Unfortunately these applications were before their time, outpacing the progress of technologies for mobile computing, tracking, and display. Now, years later, these applications can be built and deployed, but the other challenge was and is authoring.

Over the past decade a number of researchers have tackled the issue of accessible authoring tools. Tools such as StudierStube combined software abstraction layers for AR infrastructure and technologies into a framework usable via code or GUI front-ends [18]. While the DART authoring environment added AR concepts to an existing high level media authoring tool, Adobe Director [13]. More recently the Goblin project made it possible to create AR applications in the widely used Microsoft® XNA game engine [16]. An alternate approach is to create a simple authoring environment for a specific application domain. Amire supported the creation of applications specifically for hierarchical assembly tasks such as putting furniture together [28]. The Powerspace project automatically turned Microsoft® PowerPoint presentation into 3D slides that could be placed in the physical world [9].

The seminal projects in AR required expensive equipment and high levels of arcane expertise. The ARToolkit [3] made it possible for anyone with a C++ compiler and a web cam to create AR applications. More recently the FLARToolkit, an integration of the ARToolkit into Adobe Flash, has spawned a new community of web developers creating AR experiences (http://www.libspark.org/wiki/saqoosha/FLARToolKit/en). The appeal of FLARToolkit is that even though it is limited in its 3D capabilities, its availability in web browsers and support for a wide array of webcams makes it trivial for developers to distribute their applications, something that previously has been a major hurdle. Fortunately, the next step in tracking is around the corner as researchers such as Wagner et al create 6DOF natural feature tracking algorithms that run in real-time on mobile phones [25]. Natural feature tracking is the next milestone in mobile AR that will provide the level of sophistication and registration performance that users expect. The lesson learned from the ARToolkit is that when tracking becomes cheap and ubiquitous the number of applications that can be created and, even more importantly, deployed rises drastically. All of these systems provide examples of how making the supporting technologies of AR (from tracking, to device support, to geospatial information) available to a diverse set of people is critical. The KHARMA
While outdoor AR has existed for many years, it took much longer for AR to become a reality on mobile devices. The first marker-based AR on an unmodified PDA was developed by Wagner et al in 2003 [23] and was used to create applications such as the "Invisible Train"[24] and a handheld museum guide [19] installation both of which were deployed and evaluated.

As with traditional AR, tools are needed for authoring mobile AR experiences. However, the requirements of such a tool differ from those of a traditional system as computational resources are extremely limited, relatively speaking, the display and interface are unique, and vary from device to device, and issues of cross platform support, in a landscape where hundreds of different operating systems and devices exist, are critical. The Augmented Presentation and Interaction Language (APril) toolkit was an authoring platform for MR presentation, which was independent of specific applications, or target hardware platforms. The goal of APrIL was to raise the level of abstraction on which MR content creators could operate [12]. This is also a goal of the KHarMa/Kamra browser model with the approach of small native apps that handle the differences in mobile device making them transparent to the content developer.

The concept of bringing the information from the digital world and the Internet into the physical world has long been explored. "Windows on the World" incorporated an existing 2D window system within a 3D virtual world [8]. This system took XWindows windows from the desktop and placed them into the physical world. This 2D information could be linked to the HMD, to a surrounding information sphere, and to locations and objects in the world; a precursor to the modern consumer mobile AR applications that have recently become available. WorldBoard proposed a planetary augmented reality system that would provide innovative ways of associating information with places [21]. Spohrer envisioned a system that would allow users to post content (from pictures to text) on any of the six faces of every cubic meter of space on the globe. Lastly the goals of the Real World Wide Web (RWWW) project were very similar to those of KHarMa and the issues they foresaw related to presentation and user interface are relevant to our current work [12]. The vision of RWWW was outdoor, GPS tracked, mobile applications superimposing data from the World Wide Web on the user's surroundings. Kooper et al developed a prototype "browser" that allowed them to experiment with interfaces to this 3D spatialized information space. They were interested in exploring the implications of adding context information to documents on the World Wide Web. They noted that there is a wide range of research to be done as the interface design must balance the conflicting requirements of minimizing the volume of information displayed (to avoid distracting the user and cluttering their visual field) with the need to provide rich context (to capitalize on the users ability to rapidly scan and synthesize data).

In the last three years, with the advent of mobile phones with GPS, 3D graphics capabilities, data connections, and application distribution channels, there has arisen a crop of commercially available AR platforms most of which are designed for outdoor information browsing and retrieval. Wikitude released their Wikitude World Browser (www.wikitude.com) for the Android platform, which presents location-based Wikipedia and Qype content. Layar (www.layar.com) distributes the Layar Reality Browser, which allows developers to create custom "layers" of information that can be served up to users via their custom publishing platform.

Metaio (www.metaio.com) has a markerless tracking solution as well as authoring tools. Their Unifeye Design 2.0 supports the creation of presentations and live-marketing via a GUI interface. They have also created the Juniao mobile AR browser, which has an open API. Juniao 2.0 utilizes LLA markers which contain GPS locations, encoded in their image; enabling high resolution tracking which substitutes for or enhances traditional GPS position data. They call this technique "indoor GPS".

KHARMA is not the first architecture to leverage Google Earth's Keyhole Markup Language (KML). An initiative to standardize the method of describing points of interest, the ARML (openarml.org) initiative has also extended KML and with AR specific structure. Their approach has been to add a number of language extensions to support specific browser features such as "wikitude:thumbnail" and "ar:author" while the KARML approach has been to work within the KML with the intention of avoiding additions wherever possible.

One innovation that has made these location-based applications possible is the creation of ubiquitous and freely available geospatial information. One well-known platform for such information is Google Earth (GE) (earth.google.com). While KML is a widely used language that provides a solid foundation for AR web specifications, unfortunately, GE does not provide a useful software platform for viewing AR content. Although there is the GML (an open geospatial consortium) there are no free reference viewers. It is possible to envision using GE for gaming and VR, but it is not possible to add and distribute the background video support necessary for AR. However, we have found GE as a useful authoring tool for KHARMA content. It is provides a graphical method of initially placing content in the world and generating a starting KML file for subsequent editing.

OpenLayers (openlayers.org) is entirely open source JavaScript library for displaying map data in web browsers. The goal was to eliminate server-side dependencies; separating the map tools from map data avoiding the proprietary "silos" often created by GIS data services.

A tradition of abstraction and open tools define many technology advances in the field of 3D, AR, and the web. It is clear that technologies must be made accessible to be adopted. Components that are typically hard to work with or understand must be made easy. The Web3D standards of Virtual Reality Modeling Language VRML 97 (www.web3D.org) were an attempt to make 3D content ubiquitous on the web. While later the Virtual-Reality Peripheral Network (VRPN) provides a device-independent and network-transparent interface to virtual-reality peripherals [22].

One feature of KHarMa is the ability to use panoramic images in the background instead of live video. Commercial systems such as Google StreetView as well as Microsoft's Photosynth and Bing Maps support the creation and navigation of 3D panoramic scenes augmented with geospatial data [20]. We have integrated this concept into KHarMa to support the authoring of mixed reality experiences that leverage the live channel data in various ways both at the physical site and for remote viewing. We have developed a simple web service that allows users to submit panoramas to the system that can be utilized by channel authors via an open API. Our plan is to eventually leverage the panorama service for both display and tracking. Wagner et al developed a method for the real-time creation and tracking of panoramic maps on mobile phones. They note that this method can also be used in the creation of panoramic images for offline browsing, for visual enhancements through environment mapping as well as standard tracking [25].
Another feature of KHARMA is the concept of a geospot, which lets a channel author define content relative to a known physical location and to utilize panoramas created at that location. Avery et al utilized this same concept to develop an AR "Moon Lander" game that was compelling despite the use of low cost high error location tracking [1]. This "GPS snap-to" technique used the GPS to calculate which game position a player was in from a set known locations. The user’s movement determined which game “square” they were currently standing in and was otherwise ignored.

3 KML/HTML AUGMENTED REALITY MOBILE ARCHITECTURE

There has been an ongoing discussion within the AR research community about an open standard for AR content that leverages existing protocols and content pipelines. A frequent consideration in this discussion is that AR applications often rely on 3D content and employ specialized hardware and computer vision techniques both for tracking and scene reconstruction. The KHARMA architecture tries to seek a balance between these more traditional AR contexts and what has come to be known as mobile AR browsing. The architecture put forward here first acknowledges that mobile AR browsing does not require that 3D content be the primary means of authoring and provides a method for HTML content to be authored, positioned in the surroundings and manipulated freely as it is in modern web browsers. Second, the architecture seeks to decouple resources such as tracking information and 3D infrastructure models of the environment from the AR content authoring process. The physical context of both indoor and outdoor AR applications is a relatively static and consistent resource that should not only be separated from the authoring assets but also shared between multiple channels of AR content simultaneously. And, as has been the goal of frameworks such as VRPN, the implementation and data associated with location tracking should be decoupled from the authoring process when possible [22].

This separation of authored content from tracking and infrastructure results in an architecture with four main components: channel servers delivering individual channels of AR content, tracking servers providing content related to location tracking, infrastructure servers delivering information about the physical environment, and mobile clients for generating the resulting augmentations (Figure 1). Just like current web authoring, AR content written in standard KML or with our KARML extension is hosted on the web by HTTP servers. Clients open any number of these channels and view the composited results on a client such as the KAMRA browser we have implemented for the iPhone. Infrastructure, consisting not of authored 3D content but of building models and terrain data, is delivered by the same KML markup over a separate channel. A number of use cases supply the motivation for making infrastructure a separate resource that can be shared between multiple channels of AR content. Infrastructure information allows user interaction with the physical environment (i.e. annotations on buildings) is used by the client to determine occlusions between the physical and augmented content and plays an important part in the authoring pipeline. Unlike the almost endless addressable space on the Internet, there are a finite number of structures in the physical world that need to be represented. This unique attribute is what sets infrastructure apart from sources of tracking data and has motivated the development of standards to assign physical assets unique identifiers. In section X we detail our strategy for including these unique identifiers into the authoring and content delivery pipeline.

Although it would appear to be tightly coupled to infrastructure, we distinguish tracking information as any information that influences the ability of the client to determine its location in the world. Like infrastructure, information about the location of the client is a resource that should be shared across different channels of AR content. Tracking information can range from fixed way-points with known coordinates to preprocessed features in the surroundings to aid natural feature tracking algorithms. We have developed and deployed a dedicated channel server that delivers surveyed locations, or GeoSpots in a range about the user. Each GeoSpot, delivered using KARML extended KML, provides a description and a photograph to help users find them. Once at the GeoSpot, users can effectively lock down their coordinates and improve the resulting augmentations by indicating this increased accuracy to the client browser. We also leverage panoramas taken at these GeoSpots to modify the current orientation accuracy and detail a roadmap between these techniques and full NFT tracking in section 3.2.

Infrastructure and tracking sources also play a role in the authoring process of AR content. Appreciating how AR content appears in the physical environment is a necessary part of the authoring process. And, knowing the accuracy limitations of tracking data in those environments is arguably just as important to understanding how augmentations will actually appear. The KHARMA architecture envisions an integrated authoring pipeline where the same infrastructure and tracking server information is available to the development environment and can be applied in real-time to the authoring task. We describe how infrastructure and tracking interplay with the authoring pipeline in their respective sections and provide several examples of current projects that help to illustrate this dynamic. First, in the next section, we describe in detail the combined KML/HTML markup and extensions.

3.1 KARML Extension to KML

The fundamental properties of most AR applications can be summarized by the familiar refrain, “What?, Where? and How?”. The What? refers to the content being delivered, the Where refers to its position or location in the world, and the How? refers to the protocols and tools available for interpreting and manipulating the former two. In reviewing the various standards being proposed today for establishing Where?, we found many that are more comprehensive and geospatially accurate than KML. However, given our focus on developing tools for general consumption, we felt that the significant penetration of KML into everyday applications such as Google Maps (GM), Yahoo Maps and
applications throughout various domains made it a difficult choice to avoid. The KML language can be lossless converted to languages such as the open Geography Markup Language (GML) and benefits from the freely distributes the GoogleEarth (GE) reference viewer and Sketchup modeling software. Given this and the large number of web services that already process and deliver KML content, we felt that utilizing it would give the architecture the best chance at immediate adoption. There are some obstacles to using a language developed primarily for geospatial visualization in the service of augmented reality authoring. One drawback to using KML is the lack of a notion of relative positioning; all points and even the vertices of geometry elements in KML are defined in terms of longitude, latitude and altitude. Relative frames of reference are an integral part of computer graphics and are an invaluable tool when positioning and animating graphical content. This shortcoming of KML can be overcome in principle by moving all such references into JavaScript and manipulating content there, however this would seem to defeat the purpose of using KML as a language for establishing where objects are located in physical space.

In considering the question of What? and How? to author and manipulate content, an obvious choice for 2D content is modern web standards for content delivery and client-side interactivity. The KML specification already incorporates full HTML content into feature point descriptions, which are displayed by the GE application in callouts called balloons, rendering a subset of HTML, CSS style and JavaScript elements using the WebKit renderer (www.webkit.org). With respect to 3D content, the KML standard already accommodates the delivery and fixed positioning of 3D models using a combination of the COLLADA format and common compression schemes. In approaching potential alterations to KML in support of AR, we attempted to re-conceive the language in the context of AR browsers and avoid the introduction of elements whose function can already be accommodated by existing elements. Some of the enhancements we propose here, such as active HTML, spatialized sound, screen overlays, relative coordinates and enhanced JavaScript events, benefit the language as a whole. Other KML concepts, such as tour elements, which move the camera viewpoint between abstract viewpoints, suggested a reformulation for the context of user-initiated movement. The most significant change we made was to promote HTML content from its role inside balloons to that of a first-class object within the scene.

To minimize visual clutter in the scene, it is now common practice in both mapping applications and mobile AR browsers to present geographic features as an icon and text label that brings up contextual content when selected. In both the GE and GM applications these balloon elements appear with a leader-line to the feature point and are surrounded by decoration in a size and location determined by the application. The KARML extension adds a modifier to KML style elements indicating that the HTML content stored in a feature description should be rendered without decoration. This allows the authored HTML content to be surrounded only by scene content. Feature points, and hence their description HTML, do not include a means for their orientation or scaling, so we added optional elements, KMLBalloon and KMLLabel, modeled after the existing KMLModel element for controlling the size, location and scaling of feature balloons and labels. To these balloon, label and model elements we added additional location, orientation and scale mode elements. A locationMode element establishes the associated content in either the default fixed geospatial coordinate system or a coordinate system relative to another KML feature. The associated orientation element follows a similar convention, with an orientationMode indicating if the orientation is fixed, relative to another feature or billboarded (default). The targetHref attribute of each mode establishes the reference feature using established fragment URL notation and includes globals for positioning content relative to the #device or to the #user (default). Another convention typically implemented by mapping and AR browsers is pseudo-depth scaling. When viewed at different distances labels and icons are scaled relatively to give a sense of depth while preserving readability. The relative scale mode implements this pseudo-scaling and is the default for labels, icons and balloon content. These extensions, along with some additional elements for controlling the visibility of balloons, labels and icons, give AR authors fine-grain control over the positioning of both HTML and 3D content in the scene.

The question of How? also extends to the client side tools the AR author has for manipulating content. In the current implementation of GE, each content balloon has a separate namespace un-addressable by other balloons, even those created by the same source. Removing this restriction and making all HTML content generated from the same source able to use the same JavaScript namespace significantly increases the interactivity that can be achieved between different feature points in the same channel. While the same cross-site scripting restrictions in common use are also appropriate in this context, interactivity between different channels can be accomplished using the same web tools such as authentication, sessions and AJAX that let content in one browser window affect content in another. The KML language already includes several conventions such as Level-of-Detail (LOD), LOD regions and JavaScript events that let authors control the display of content, trigger network updates and respond to user events. These facilities, however are restrictive in several ways. Level of detail nodes do not emit events in the GE API when their state changes nor do regions emit events when they are entered or exited. We specify the generation of these events in order to lets AR authors program functionality in ways that are well established in gaming and VR authoring. The standard KML specification, being mainly targeted at static infrastructure models, neither provides events indicating selection of sub-sections of models nor does it incorporate a means for triggering animations within those models. Several browser-based initiatives such as WebGL are underway to incorporate 3D content and manipulation into webpages but most are targeted at a lower level of detail than full 3D models (www.khronos.org/webgl). The addition to KML click events on sub-parts of models, the firing of animations and cursor 3D position/normal helps to smooth the space between 2D and 3D without imposing a significant level of detail onto the user. Another existing extension to KML, the GX extension, adds the ability to predefine tours between any number of feature points. We found the KML soundCue elements defined within these tours inadequate for AR applications and extended their use by allowing their attachment to features and the specification of their location and orientation.

A final consideration for developing AR content is determining the tracked location of the device and of objects in the physical environment. While marker-based tracking has been very popular, it is difficult to say what algorithms or devices will be providing this information in the future. What can be said is that objects in the scene will have a tracked location and that it must be incorporated into the application in a generic manner. To this end, we have added a KMLTracker element also modeled after the existing KMLModel element. The position and orientation of this element can be updated by some external source such as marker, NFT tracking or other means. The behavior of the locationMode and orientationMode elements is somewhat different for this element. A relative location or orientation mode indicates that position of the feature is relative to the device or other feature referenced by its targetHref attribute. This mode of operation is
used to generate the type typical marker-based of AR applications that merely attach content to markers without a notion of their location in the world. Changing the mode from relative to fixed during runtime populates the location and/or orientation with its location in the physical world. Indicating that the tracker is fixed presents the opportunity to begin using that tracker as a location source for the client. As with the GeoSpots mentioned above, a browser client choose use any KML feature as its location source. Any trackable source can be placed in a known location in the physical world and become the location source for the client. It may not be correct to call this a “solution to indoor GPS” as some have, but this approach does allow any visible marker to become a tracking source for the client, even if only to stabilize the orientation while markers remain visible. We leave the sub-elements of the KMLTracker element to only a trackerDevice, trackerDescription and an associated link element in an attempt to keep it as generic as possible. Link elements in KML can have a number of refresh criteria and may serve as input to any number of tracker plugins described by trackerDevice. Typical markers can utilize the trackerDescription element for ID numbers while multimarker tracking and todays NFT trackers would likely reference a configuration file or initialization images respectively. Neither KML nor the GE viewer supports any notion of plugins, but this has become an invaluable tool for web development. As with 3D models and regions, we generate events indicating when visibility changes and provide typical metrics (i.e. confidence, area) to facilitate decisions about whether to “go to” them. We appreciate that there is a growing appreciation for the utility of fusing multiple tracking sources and that we will likely accommodate the notion of “being at” multiple tracking sources at the same time in the future.

3.2 A Bridging Strategy Toward Global AR Tracking

The notion of fusing multiple data sources to generate the best possible location estimate is analogous to the techniques currently in use for natural feature tracking. In this context, the multiple trackers are represented by the individual feature points recognized in the visual scene and the sensor fusion is the homography estimation that fuses those feature points into a single tracking estimate. These feature detection and tracking methods remain too computationally intensive for the consumer devices available today. But, if such devices could do this processing on a routine basis, the problem of registering the device mapping of the environment with the actual physical location has yet to be unaddressed. Scene reconstruction and tracking methods such as PTAM can recognize and augment objects in the physical environment, but a methodology for the storage, identification and retrieval of more general infrastructure remains to be developed [11]. In an effort to approach this problem we have developed a bridging strategy that acknowledges the limitations of current tracking technologies while anticipating a future where NFT tracking and object recognition will be routine.

The GPS sensors in consumer devices now being used for mobile AR are heavily filtered and frequently only accurate to within 10 meters. This low accuracy means that objects depicted on the phone in front of the user can easily be actually behind them, effectively limiting the distance at which augmentations can be delivered. The estimated GPS accuracy is available on most platforms, so AR authors should to be able to control how their content appears by responding to this information. Some of the mobile AR architectures do support responding to these changes on the server side, but this approach is less than ideal when accuracy is highly variable. Our strategy begins with letting users manually override the reported tracking of the device by moving to GeoSpots within their midst (Figure 2). Once at these locations, the client can report an improved accuracy range to the browser and content can respond in kind. Beyond restricting the range of objects within view, another response to increased accuracy may be to change the visuals from labels and icons to a more detailed representation.
ability to use synthetic backgrounds to view higher fidelity augmentations.

As the distance of content from the user moves outside the range of GPS accuracy, the requirement that the user is actually at the proper location decreases. With sufficient network throughput, the nearest synthetic background to the GPS location can be delivered ahead of time to the device and used when desired (Figure 3). Currently the number of panoramas available is quite small with access to the largest databases of street level panoramas remaining proprietary (i.e. Google StreetView and Bing Maps). Image-based methods, like those now being employed for Microsoft Bing Maps, can utilize publicly available photography of locations to reconstruct their geometry and synthesize arbitrary viewpoints [20]. It is not difficult to imagine a future where posting a number of photographs an area and being able to augmentations against it is analogous to putting up a webpage and being able to find it via Google. This process of creating a synthetic background need not end with replacing live video. Natural feature tracking algorithms such as PTAM operate in two distinct phases, tracking key generation and feature detection. The key generation phase involves building a three dimensional map of tracking features in the surrounding environment. The feature detection phase involves finding those features in the video stream and determining relative camera position. By leveraging server generated synthetic backgrounds with appropriate lighting conditions, the key generation phase can be processed offline and delivered to the mobile device. While the exact protocol for delivery of feature key databases remains open to research, the potential benefits are readily apparent. A public database of such tracking keys, combined with authenticated access to databases of private environments, could potentially let mobile devices do real-time tracking on a global scale while providing a solution to the mapping registration problem.

![Figure 3. An incremental bridging strategy towards global tracking can begin with surveyed panoramas and end with the delivery of tracking keys form synthetic backgrounds](image)

**3.3 The Role of Infrastructure Services in Authoring**

The separation between tracking information and infrastructure is not likely to end with thorough image-based modeled tracking databases. There are a few reasons why a truly authorable AR architecture cannot rely on image-based models of infrastructure. Private stakeholders, commercial interests and public institutions may all provide information useful for outdoor and indoor tracking, but there remains only a single entity from an authoring standpoint. During content authoring, developers need to utilize the models of the objects their augmentations are likely to interact with. By accessing the same infrastructure services that AR users will access, authors can create predictable results. When authors decide to use their own model of a building, vehicle or statue, they need a global unique identifier (GUID) for indicating the source from which to retrieve it along with a method for graceful fallback to more public sources when that source is not available.

Even if image-based models could be assigned a GUID, there remains a need for more detailed information about the interior of the building and the constituent parts of that building. By using common graph relationships, a vocabulary for common parts of buildings such as “roof”, “main entrance” or “announcements” can be built into the models of buildings. This common vocabulary of object parts would also allow content developers to create augmentations that are generalizable across multiple locations and/or renovations of a building (i.e. “on every stop” sign”, “for ever Staples store”, “on each building roof”). We propose using the existing COLLADA model node-referencing scheme that allows authors to reference subparts of models using combined a combination of GUID and fragment referencing (i.e. ABCDE12345.kml#front_door). This technique would also let authors reuse subparts of whole models as part of their augmentations (i.e. a falling tower, etc.).

4 KAMRA BROWSER REFERENCE IMPLEMENTATION

We developed a reference implementation of the KHARMA architecture for the iPhone platform. This AR browser client renders the KARML extended KML with fully realized HTML support, panoramic photo overlays, sound, and standard KML network updates. In order to ensure that CSS, HTML and JavaScript was accurately implemented we took the novel approach of leveraging the existing COCADA model node-referencing scheme that allows authors to reference subparts of models using DOM element support. This approach attractive property of channels and webpages sharing an analogous security context, with typical cross-site scripting restrictions within channels and different channels communicating through session controlled servers. The choice of using Cocoa UIWebView objects was made much easier because of their solid implementation of the WebKit 3D standard. This recently introduced standard allows any HTML element to have a 3D transformation within the browser and was invaluable in rendering HTML content into space around the user.

The reason we did not base our overall architecture on WebKit 3D in the first place is that it provides only an abstract representation 3D graphics. Limited control over the projection matrix, clipping and other aspects of 3D rendering make the standard more appropriate for moving HTML around in the browser window than as a building block for a general 3D renderer. This being said, our implementation of HTML content in the KAMRA browser may be best understood as a KML to HTML/WebKit 3D translator. Although there are now an increasing number of JavaScript libraries that support the parsing of KML within JavaScript, we take the approach of parsing KML content in Objective-C in order to take advantage of the Cocoa CoreData framework for the serialization and restoration of application content. Once parsed, HTML content is injected into the running browser via JavaScript calls. Cocoa UIWebView objects do not provide a direct means of updating outside code,
but frameworks have such as PhoneGap (phonegap.com) been developed take advantage of delegate functions to capture and interpret each new URL call made within the browser as function calls in Objective C.

A goal of the KAMRA browser development has been to leave as much of the control over the browser channel behavior in the hands of AR authors. In addition to providing full access to the KML DOM and functions to manipulate it, we also working on JavaScript interfaces that allow authors to override default behaviors such as how to handle the backgrounding of channels or the overlapping of multiple features within the display. To date, our KAMRA browser implementation is limited to rendering HTML content against either video or panoramic backdrops. We are currently developing a 3D renderer based on the KHRONOS reference COLLADA viewer that will let us load, render, animate and do collisions against models downloaded in kml and compressed kmz files. Because the current implementation of the KAMRA browser is so tightly integrated with Cocoa UIWebView objects our access to the internal rendering engine is restricted and we will be using a number of multi-pass rendering methods to blend 3D content into the 2D content it generates.

4.1 Projects in Development Using KAMRA

Our primary research focus is not only on the technologies for widespread AR deployment but also on the ways that AR is employed by, influences and is appropriated by its users. In an effort to both foster its broader use and to understand the affordances of the KHARMA architecture, we have sought out and actively worked with a number of constituents whose focus ranges from urban planning and cultural heritage and to information search and retrieval. What follows is a brief description of several of those projects.

4.1.1 Yahoo Pipes Flickr Search

This project, developed by masters students in the School of Interactive Computing, demonstrates how easily current web services already generating KML content can be appropriated and customized into AR channels. A KML ScreenOverlay element presents the user with a search form and submit button that returns standard KML of Flickr images in their area. The Yahoo Pipes service hosted and served entirely by Yahoo servers lets users construct complex services that define query parameters, apply regular expression filters and generate mappable KML results from a number of sources such as Flickr, Google and any other existing web service.

4.1.2 Centennial Park Visitor Guide

We developed this project in collaboration with the Interactive Media Technology Center here at Georgia Tech to highlight the ability of the KAMRA channels to dynamically change content in response to changing accuracy (Figure 2). When users are first connected to this channel they are presented with labels and icons representing notable of buildings and around the park such as the CNN center. Users can bring up the map view and select from a number of sources such as Flickr, Google and any other existing web service.

4.1.3 Oakland Cemetery Experience

This local/remote tour of historic Oakland Cemetery in Atlanta is being developed in collaboration with members of the School of Literature, Communications and Culture leverages previous AR applications we have evaluated at the site [7]. Upon activating the channel, users are encouraged to move to a location outside the front gate where the narrator, a well-known local Atlanta historian, Frank Miller Garrett, greets them. Users can tour three graves and hear the voices of those buried in Oakland as they describe their lives and contributions to Atlanta. These graves are situated at GeoSpots inside the cemetery. This project demonstrates how tours, both local and remote tours can be authored through JavaScript calls that move the client between GeoSpots and control the display of their associated panorama. From each GeoSpot, users see icons representing the other available GeoSpots and can travel directly to them by tapping the screen. Our intuition is that users will naturally align the orientation of the display with their head. As a result, this project will be a testbed for experimentation with the use of spatialized sound in mobile AR environments.

4.1.4 NextBus AR

This collaborative project between ourselves and the Research and Networking Operations Center [here at Georgia Tech] demonstrates browser support for standard KML networking by superimposing information about Georgia Tech student trolleys onto the buses themselves. A session controlled Java Servlet delivers GPS data sent to the existing NextBus system using the standard KML protocols for NetworkLinks and updates via the NetworkLinkControl element. Users can travel between multiple GeoSpots on campus and, when view against a panorama, see an HTML rendered bus moving through the street.

4.1.5 Clough Undergraduate Learning Center

This project demonstrates how AR browsers can be used to enhance public knowledge about urban planning and development projects is a collaboration involving the School of Architecture and volunteer photographers and urban planners in the Atlanta area. Users connected to this channel will receive alerts when within range of several GeoSpots near the new undergraduate center under construction. Once at a GeoSpot, users can view a rendering of the new building superimposed over the existing construction site. One goal of this project is to explore the lengths to which existing web assets related to a location or project can be re-appropriated into a more compelling and interactive AR experience.

5 CONCLUSION AND FUTURE WORK

We have presented our KHARMA architecture for mobile augmented reality authoring using a combination of HTML and an extension to the KML language called KARML. Our main contribution is the promotion of HTML content to first class objects in the scene and a number of additions to KML that help taylor its use for mobile augmented reality. We developed an iPhone reference browser called KAMRA that we are currently using in a number of ongoing projects.

Having developed the HTML rendering part of the browser, we are now working on adding 3D support. And, having developed a web service for the storage and deliver of GeoSpots and their panorama, our next effort will be to develop a service for the storage and delivery of infrastructure. We are also beginning to work on the authoring pipeline by building a desktop development tool that aggregates these services during the authoring process.
REFERENCES


http://www.theatlantic.com/doc/19407/bush


