

## **ROBUST MOBILE COMPUTING FRAMEWORK FOR VISUALIZATION OF SIMULATED PROCESSES IN AUGMENTED REALITY**

Suyang Dong

The University of Michigan  
1318 Hayward Street, 2340 G.G. Brown Building  
Ann Arbor, MI 48109, USA

Vineet R. Kamat

The University of Michigan  
2350 Hayward Street, 2340 G.G. Brown Building  
Ann Arbor, MI 48109, USA

### **ABSTRACT**

Visualization of engineering processes can be critical for validation and communication of simulation models to decision-makers. Augmented Reality (AR) visualization blends real-world information with graphical 3D models to create informative composite views that are difficult to replicate on the computer alone. This paper presents a robust and general-purpose mobile computing framework that allows users to readily create complex AR visual simulations. The technical challenges of building this framework from the software and hardware perspective are described. SMART is a generic and loosely-coupled software application framework for creating AR visual simulations with accurate registration and projection algorithms. ARMOR is a modular mobile hardware platform designed for user position and orientation tracking and augmented view display. Together, SMART and ARMOR allow the creation of complex AR visual simulations. The framework has been validated in several case studies, including the visualization of underground infrastructure for applications in excavation planning and control.

### **1 INTRODUCTION**

In a broad sense, Augmented Reality (AR) is a multi-sensory technology that blends virtual contents with the real environment. In particular, AR refers to a visualization technology that superimposes virtual objects on the real world. AR has distinct advantages over other forms of visualization in at least three aspects: 1) from the perspective of visualization, the real world can significantly mitigate the efforts of creating and rendering contextual models for virtual objects, and provide a better perception about the surroundings than pure virtual reality, e.g. visualization of construction simulations (Behzadan, et al., 2007), and visualization of architectural designs (Thomas, et al., 1999); 2) from the perspective of information retrieval, AR supplements user's normal experience with context-related or Georeferenced virtual objects, e.g. looking through the walls to see columns (Webster, et al., 1996), and looking beneath the ground to inspect subsurface utilities (Roberts, et al., 2002); 3) from the perspective of evaluation, authentic virtual models can be deployed to measure the physical condition of real objects, e.g. evaluation of earthquake-induced building damage (Kamat, et al., 2007), and automation of construction process monitoring (Golparvar-Fard, et al., 2009).

A typical AR system should possess the following properties concluded by (Azuma, et al., 2001): 1) real and virtual objects coexist in the augmented space; 2) run in real time; 3) register real and virtual objects with each other. Each property corresponds to a field of research challenges, e.g. the coexistence of real and virtual objects leads to occlusion and photorealism problems. This paper primarily focuses on the challenge of achieving precise registration from both the hardware and software perspectives.

## **1.1 Importance of the Research**

The fundamental problem in Augmented Reality is placing virtual objects in the augmented space with correct pose, which is called registration. The registration process is difficult because its errors arise from both spatial and temporal domain (Azuma, 1997a). Furthermore, different tracking technologies have their own error sources. This paper focuses on registration problem of AR in an unprepared environment, i.e. outdoor, where sensor-based AR is thus far the most reliable tracking method free of constraint on the user.

Errors in spatial domain are also referred as static errors when neither the user nor the virtual objects move (Azuma, 1997b). The static errors of sensor-based AR include: 1) inaccuracy in the sensor measurement; 2) mechanical misalignments between sensors; 3) incorrect registration algorithm. The selection of high accuracy sensors is crucial, because the errors contained in the measurement are often non-compensable. The accuracy of measurement can be further compromised by insecure placement of sensors on the AR backpack and helmet. Some early AR backpack design examples can be found in touring machine (Feiner, et al., 1997) and Tinmith-Endeavour (Piekarski, 2004), that are fragile and cumbersome. A more robust and ergonomics version is demonstrated by Tinmith backpack 2006 version (Piekarski, et al., 2006), where a GPS antenna and an InterSense orientation tracker are anchored on top of the helmet. However the 50cm accuracy of the GPS receiver is not qualified for centimeter level accuracy AR task.

Static errors are relatively easy to eliminate given high accuracy sensors, rigid placement and correct registration algorithm. On the other hand, dynamic errors, errors in temporal domain, are much more unpredictable, and create the “swimming” effect. Noticeable dynamic misregistration is mainly caused by the differences in latency between data streams, which is called relative latency by (Jacobs, et al., 1997). Relative latency has its source from: 1) off-host delay: Duration between the occurrence of a physical event and its arrival on the host; 2) synchronization delay: The time in which data is waiting between stages without being processed. 3) computational delay: Time elapsed for processing data in the host system. Some common mitigation methods for resolving relative latency are: 1) adopting multi-threading programming or scheduling system latency (Jacobs, et al., 1997); 2) predicting head motion using Kalman filter (Liang, et al., 1991) (Azuma, et al., 1999).

## **1.2 Main Contribution**

The mobile computing framework presented in this paper provides a complete hardware and software solution for centimeter level accuracy AR tasks in both spatial and temporal domain. The robustness of the framework has been validated on application for visualizing underground infrastructure as part of the on-going excavation planning and control project.

Augmented Reality Mobile Operation platform (ARMOR) evolves from the ARVISCOPE hardware platform (Behzadan, et al., 2008). ARMOR improves the design of ARVISCOPE from two aspects: rigidity and ergonomics: 1) introducing high accuracy and lightweight devices; 2) placing all tracking instruments rigidly with full calibration; 3) renovating the carrying harness to make it more wearable.

Scalable and Modular Augmented Reality Template (SMART) builds on top of the ARVISCOPE software platform (Behzadan, et al., 2008). The main motivation of ARVISCOPE is exporting some basic modules communicating with peripheral hardware as dynamic link library that can be later imported into other potential AR applications. SMART takes advantage of these modules, and constructs an AR application framework that separates the AR logic from the application-specific logic. This extension essentially creates a standard structured AR development environment.

The in-built registration algorithm of SMART guarantees high accuracy static alignment between real and virtual objects. Some preliminary efforts have also been made on reducing dynamical misregistration: 1) in order to reduce synchronization latency, multiple threads are dynamically generated for reading and processing sensor measurement immediately upon the data arrival on the host system; 2) Finite Impulse Response (FIR) filter applied on the jittering output of electronic compass leads to filter-induced latency, therefore an adaptive lag compensation algorithm is designed to eliminate the dynamic misregistration.

## 2 ARMOR HARDWARE ARCHITECTURE

As a prototype design, the ARVISCOPE hardware platform succeeded in reusability and modularity, and produced sufficient results for proof-of-concept simulation animation. However there are two primary design defects that are inadequately addressed: accuracy and ergonomics. ARMOR is a significant upgrade over the ARVISCOPE hardware platform. The improvements can be categorized into four aspects: (1) highly accurate tracking devices with rigid placement and full calibration, (2) lightweight selection of input/output and computing devices and external power source, (3) intuitive user command input, (4) load bearing vest to accommodate devices and distribute weight evenly around the body. An overview comparison between ARVISCOPE and ARMOR is listed in Table 1.

Table 1: Comparison between ARVISCOPE and ARMOR hardware configuration.

Device	ARVISCOPE	ARMOR	Comparison
Location Tracking	Trimble AgGPS 332 using OmniStar XP correction for Differential GPS method	Trimble AgGPS 332 using CMR correction broadcast by a local base station Trimble AgGPS RTK Base 450/900	OmniStar XP provides 10~20cm accuracy. RTK provides 2.5 cm horizontal accuracy, and 3.7cm vertical accuracy
Orientation Tracking	PNI TCM 5	PNI TCM XB	The same accuracy, but ARMOR places TCM XB rigidly close to camera
Video Camera	Fire-I Digital Firewire Camera	Microsoft LifeCam VX-5000	LifeCam VX-5000 is lightweight, small volume, with less wire connection
Head-mounted Display	i-Glasses SVGA Pro video see-through HMD	eMagin Z800 3DVisor	Z800 3DVisor is lightweight with stereovision
Laptop	Dell Precision M60 Notebook	ASUS N10J Netbook	ASUS N10J is lightweight, small volume, and equipped with NVIDIA graphics card
User Command Input	WristPC wearable keyboard and Cirque Smart Cat touchpad	Nintendo Wii Remote	Wii Remote is lightweight and intuitive to use
Power Source	Fedco POWERBASE	Tekkeon myPower MP3750	MP3750 is lightweight and has multiple voltage output charging both GPS receiver and HMD.
Backpack Apparatus	Kensington Contour Laptop Backpack	Load Bearing Vest	Extensible and easy to access equipment

### 2.1 Tracking Devices

#### 2.1.1 Orientation Tracking Device: Electronic Compass

TCM XB electronic compass is employed to measure the yaw, pitch, and roll that describe the relative attitude between the eye coordinate system and the world coordinate system. It measures the heading up to 360 degree full range and maintains the accuracy of  $0.3^\circ$  rms when tilt (pitch and roll) is no larger than  $65^\circ$ , the common motion range of human head.

ARVISCOPE placed the electronic compass on top of the helmet, and thus induced more physical attitude disagreement between the camera and the electronic compass. ARMOR chooses to anchor the electronic compass rigidly close to the camera on the brim of the helmet, and parallel to the line of sight, making physical discrepancy calibration much easier. The calibration approach is described in section 4.2.1.

## **2.1.2 Position Tracking Device: Real-time Kinematic (RTK) GPS**

AgGPS 332 Receiver used in ARVISCOPE is upgraded and three principles are followed: 1) The upgraded GPS must be able to produce centimeter level output; 2) The hardware upgrade should have minimum impact on the software; 3) The existing device should be fully utilized given the cost of high accuracy GPS equipment. Ultimately AgGPS RTK Base 450/900 GPS Receiver is chosen for implementing the upgrade: 1) it utilizes RTK technology to provide 2.5cm horizontal accuracy and 3.7cm vertical accuracy on a continuous real-time basis. The RTK Base 450/900 Receiver is set up as a base station placed at a known point, i.e. control points set up by the government with 1<sup>st</sup> order accuracy, and tracks the same satellites as a RTK rover. The carrier phase measurement is used to calculate the real-time differential correction that is sent as Compact Measurement Record (CMR) through a radio link to the RTK rover within 100km (depending on the radio amplifier and terrain)(Trimble, 2007).

The RTK rover applies the correction to the position it receives and generates centimeter level accuracy output; 2) despite the upgrade, the RTK rover outputs the position data in NEMA format that is used in OmniStar XP as well. No change therefore applies to the software part; 3) the original AgGPS 332 Receiver is retained as RTK rover with its differential GPS mode being geared from OmniStar XP to RTK. A SiteNet 900 radio works with the AgGPS 332 Receiver to receive the CMR from the base station. ARMOR anchors the GPS receiver with a bolt on the summit of the helmet, so that the phase center of the receiver will not shift relative to the camera center in any case. The fixed relative distance between them is measured and added as compensated value to the RTK rover measurement.

## **2.2 Input/output Devices and External Power Supply**

### **2.2.1 Video Sequence Input: Camera**

The camera is responsible for capturing the continuous real-time background image. The ideal device should possess properties of high resolution, high frequency sampling rate and high speed connection, with small volume and light weight. Microsoft LifeCam VX5000 stands out from the mainstream off-the-shelf web cameras for the following reason. The size is 45cm\*45.6cm and only requires USB2.0 for both data transmission and power supply, and it doesn't compromise on resolution (640\*480) and connection speed (480Mbps). More importantly, it takes samples at 30Hz that is the same as the electronic compass.

### **2.2.2 Augmented View Output: Head-mounted Display (HMD)**

The augmented view generated by the video compositor is eventually presented by the Video See-Through HMD. eMagin Z800 3DVisor is chosen as the HMD component of ARMOR because it has remarkable performance in all primary factors including view angle, number of colors, weight, and comfort. Furthermore, stereovision is one of most important rendering effects valued by domain experts, because it helps the user to better appreciate the 3D augmented space. Unlike i-Glasses SVGA Pro used by ARVISCOPE, Z800 3DVisor provides stereovision when working with NVIDIA graphics card, that supports two perspectives in frame sequential order (Z800 3DVisor User's Manual, 2010).

### **2.2.3 External Power Supply**

External power supplies with variant voltage output are indispensable for powering all devices without integrated internal batteries. Tekkeon myPower ALL MP3750 improves over POWERBASE used by ARVISCOPE in four aspects: 1) both the volume (17cm\*8cm\*2cm) and weight (0.44kg) of MP3750 are only 1/5 of POWERBASE; 2) the main output voltage varies from 10V to 19V for powering AgGPS 332 Receiver (12V), and an extra USB output port can charge the HMD (5V) simultaneously; 3) it features automatic voltage detection with an option for manual voltage selection; 4) an extended battery pack can be added to double the battery capacity (Tekkeon, 2009).

### 2.3 User Command Input: Nintendo Wii Remote

Domain-related augmented system should be capable of obtaining users' instructions through an intuitive interaction method. For example, the user may want to use the mouse to select objects in the augmented space, query, edit, and update their attribute or spatial information. Nintendo Wii Remote (Wiimote) has proved its effective user experience not only on Wii Console but also on PC games because of its Bluetooth connection feature. ARMOR takes advantage of Wiimote's motion sensing capability that allows the user to interact and manipulate objects on screen via gesture recognition and pointing through the use of accelerometer. (WIKIPEDIA Wii Remote, 2010) A Programmable Input Emulator GlovePIE is also deployed to map commands or motion of Wiimote to PC keyboard and mouse events. (Kenner, 2010)

### 2.4 Load Bearing Vest

The optimization of all devices in aspects of volume, weight and rigidity allows the authors to compact all components into one load bearing vest. Figure 1 shows the configuration of the backpack and the allocation of hardware. The configuration of the vest has several advantages over the Kensington Contour Laptop Backpack used by ARVISCOPe: 1) the design of pouches allows even distribution of weight around the body; 2) the separation of devices allows the user to access and check the condition of certain hardware conveniently; 3) different parts of the loading vest are loosely joined so that it can fit any kind of body type, and be worn rapidly even when fully loaded. ARMOR has been tested by several users for outdoor operation over half an hour continuously without any interruption or reported discomfort.

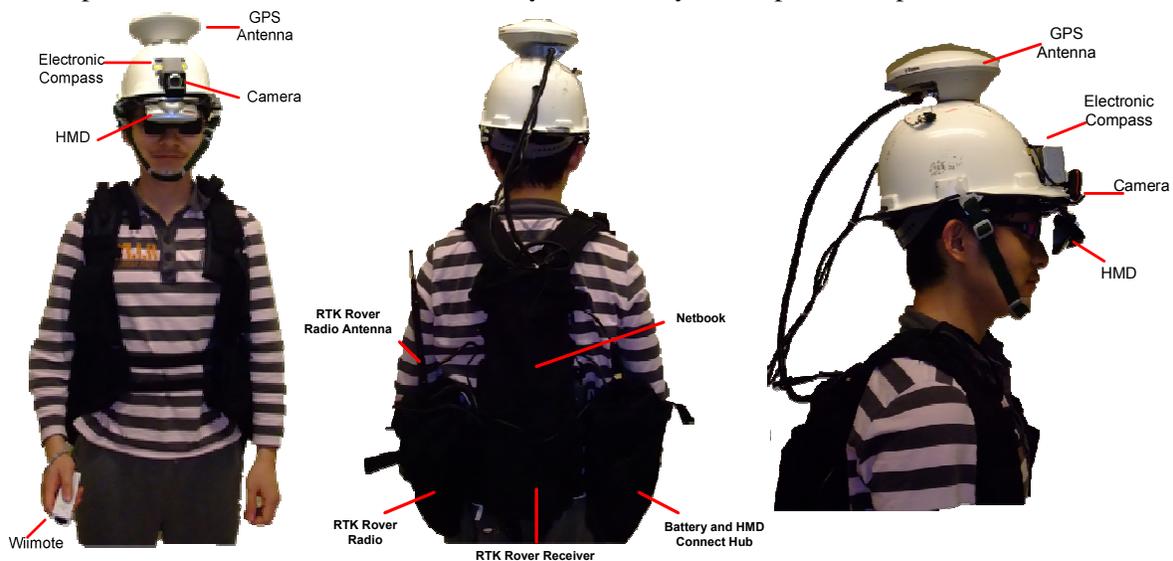


Figure 1: The profile of ARMOR from different perspectives

## 3 SMART SOFTWARE FRAMEWORK

SMART provides a default application framework for AR tasks, where most of its components are written as generic libraries and can be inherited in specific applications. The framework isolates the domain logic from AR logic, so that the domain developer only needs to focus on realizing application-specific functionalities and leaving the AR logic to the SMART framework.

SMART framework follows the classical model-view-controller (MVC) pattern. Scene-Graph-Frame is the implementation of MVC pattern in SMART: (1) the counterpart of model in SMART is the scene that utilizes application-specific I/O engines to load virtual objects and maintains their spatial and attribute status. The update of virtual objects' status is reflected when it is time to refresh the associated graphs; (2) Graph corresponds to view and implements the AR registration process for each frame update event. Given the fact that the user's head can be in continuous motion, graph always rebuilds the transformation matrix based on the latest position and attitude measurement, and refreshes the background im-

age; (3) Frame plays the role of controller, manages all the UI elements, and responds to user’s commands by invoking scene’s member functions. The framework of SMART based on Scene-Graph-Frame is constructed in the following way (Figure 2).

The main entry of the program is CARApp that is in charge of CARDeviceManager and CARManager. The former initializes and manages all tracking devices, like camera, RTK rover and electronic compass. The latter maintains a list of available CARSceneTemplate. One scene template defines the relationship among scene, graphs and frame, and is only able to load one file type. If multiple file types are to be supported, AddSceneTemplate function needs to be called so that a new CARSceneTemplate is added to the list of existing scene templates. After a CARSceneTemplate object is initialized, it orchestrates the creation of CARScene, CARFrame, and CARGraph, and the connection of graphs to the appropriate scene. Applications derived from SMART are Single Document Interface (SDI), therefore there is only one open scene and one frame within a template. The open scene keeps a list of graphs and a pointer to the scene template. The frame keeps pointers to the current active graph and to the scene template.

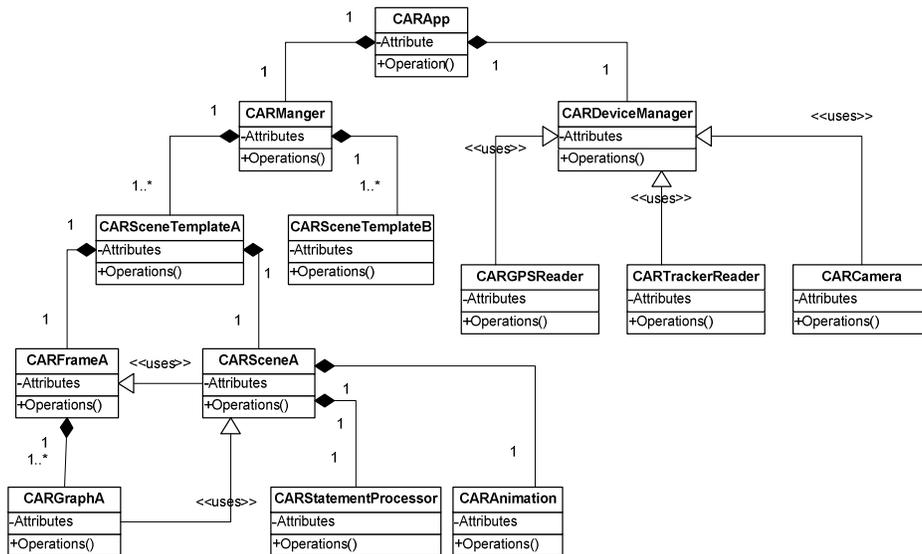


Figure 2: SMART framework architecture.

### 3.1 Application for Operation Level Construction Animation

ARVISCOPE animation function has been re-implemented under SMART framework as follows. In order to load ARVISCOPE animation trace file (Behzadan & Kamat, 2009), the pointer of CARSceneTemplateA is added to the list of scene templates maintained by CARManager. The CARSceneTemplateA contains CARSceneA, CARGraphA, and CARFrameA, all of which are subclasses inheriting from SMART’s superclasses and adapted for animation function. (1) CARSceneA employs CAASentenceProcessor and CAAnimation classes as the I/O engine to interpret the trace file. (2) CARGraphA inherits the registration routine from CARGraph. (3) CARFrameA inherits basic UI elements from CARFrame but also adds customized ones for controlling animation like play, pause, continue, jump, etc.

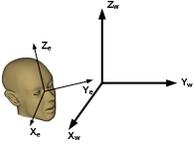
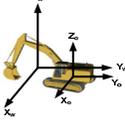
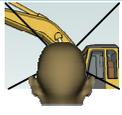
## 4 STATIC REGISTRATION

### 4.1 Registration Process

The registration process of Augmented Reality is very similar to the computer graphics transformation process: 1) positioning the viewing volume of user’s eyes in the world coordinate system; 2) positioning objects in the world coordinate system; 3) determining the shape of viewing volume; 4) converting objects from world coordinate system to the eye coordinate system (Shreiner, et al., 2006). However unlike computer graphics where parameters needed for step 1~3 are coded or manipulated by the user, Aug-

mented Reality fulfills these steps rigidly according to the 6 degrees of freedom measured by tracking devices and the lens parameter of the real camera. Table 2 lists the registration process, the needed parameters and their measuring devices.

Table 2: The four steps of registration process

Step	Task	Caption	Parameters and Device
1. Viewing	Position the viewing volume of user's eyes in the world		Attitude of the camera ( Electronic Compass)
2. Modeling	Position the objects in the world		Location of the world origin (RTK GPS)
3. Creating Viewing Frustum	Decide the shape of viewing volume		Lens and aspect ratio of camera (Camera)
4. Projection	Project the objects onto the image plane		Perspective Projection Matrix

## 4.2 Registration Validation Experiment

### 4.2.1 Calibration of the Mechanical Attitude Discrepancy

The mechanical attitude discrepancy between the real camera and the sensor needs to be compensated by the following calibration procedure: A real box of size 12cm\*7cm\*2cm (length\*width\*height) is placed at a known pose. A semi-transparent 3D model of the same size is created and projected onto the real scene, so that the level of alignment can be judged. The virtual box is first projected without adjustment of the attitude measurement, and discrepancy is thus present. The virtual box is then shifted to align with the real one by adding compensation value to the attitude measurement as shown in Table 3 Row 1.

### 4.2.2 Validation of the Static Registration Algorithm

A series of experiments are performed to validate the agreement between the real and virtual camera: If the static registration algorithm works correctly, the virtual box should coincide with the real box when moved together with 6 degrees of freedom. Overall the virtual box matches the real one in all tested cases very well and a selected set of experiments are shown below in Table 3 Row 2~3.

## 5 RESOLVING LATENCY PROBLEM IN ELECTRONIC COMPASS

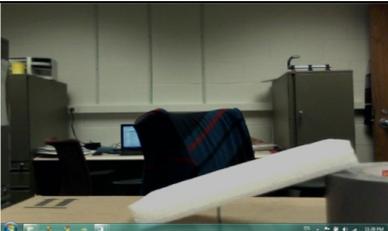
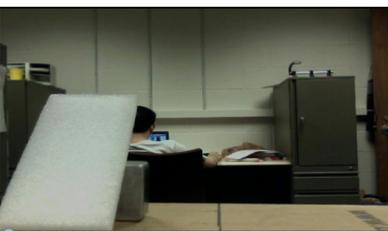
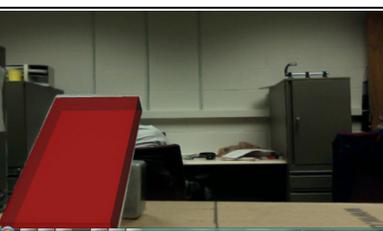
Due to the latency induced by the compass module itself, correct static registration does not guarantee that the user can see the same correct and stable augmented image when in motion. This section addresses the cause and solution for the dynamic misregistration problem.

### 5.1 Multi-threading to Reduce Synchronization Latency

There are two options for communicating with the compass module: POLL and PUSH mode. POLL is a passive output mode for the compass module, and is used by ARVISCOPE for polling data out of the module. Since ARVISCOPE does not separate I/O communication with the electronic compass as a back-

ground task, the main function has to be suspended when the program requests orientation data from the module. One loop of polling request is 70ms on average and significantly slows down program performance. Thus the maximum frames per second for ARVSCOPE is 15, causing noticeable discontinuity.

Table 3: Mechanical attitude calibration result and validation experiment of registration algorithm.

<p>Calibration Result            Yaw offset: <math>-4.5^\circ</math>            Pitch offset: <math>-7.3^\circ</math>            Roll offset: <math>-1.0^\circ</math></p>		
<p>X pos: <math>-0.15\text{m}</math>            Y pos: <math>0.30\text{m}</math>            Z pos: <math>-0.04\text{m}</math></p>		
<p>X pos: <math>-0.05\text{m}</math>            Y pos: <math>0.30\text{m}</math>            Z pos: <math>-0.09\text{m}</math>            Roll: <math>-22.21^\circ</math></p>		
<p>X pos: <math>-0.07\text{m}</math>            Y pos: <math>0.30\text{m}</math>            Z pos: <math>-0.09\text{m}</math>            Pitch: <math>46.12^\circ</math></p>		

PUSH mode is an active output mode for the compass module. SMART selects PUSH mode as its data communication method to increase the program efficiency. If PUSH mode is selected, the module outputs the data at a fixed rate set by the host system. If the fixed rate is set to 0, which is done by SMART, it means the module will flush the next data packet as soon as the previous is sent out. The sampling and flushing happens at proximately 30 to 32 Hz. The biggest advantage of choosing PUSH mode is that, once the initial communication is successfully established, and no FIR filtering is carried on in hardware, the host system can acquire the observed orientation data with only 5ms on average.

However disadvantage of choosing PUSH mode also exists: Since the data packet arrives at faster than 30Hz, if the software is not capable of handling the data queue at the same rate, it will cause rapid accumulation of data packet in the buffer. Not only will this induce latency to the view updating, but also overflow the buffer and crash the program eventually. Therefore SMART adopts event-based asynchronous pattern to tackle high frequency data packet arrival. When SMART detects that a character is received and placed in the buffer, a DataReceived event is triggered, and the data parsing function registered with this event beforehand is invoked and proceeds on a separate thread in the background without interrupting the main loop. This multi-threaded processing accelerates the main function rendering speed up to 60 fps, and also reduces the synchronization latency to the minimum.

### 5.2 Filter-induced Latency

Even though PUSH mode is free of synchronization delay, there is still significant latency if FIR filter is switched on inside the compass module. This section explains the reason for this phenomenon. Calibrating the magnetometer can compensate for local static magnetic source within the vicinity of the compass module. However dynamic magnetic distortion still has its impact on the module in motion, and the noise magnification depends on the acceleration of the module. Usually the faster the acceleration is, the higher the noise is. Among the three degrees of freedom, heading is the most sensitive to the noise.

Except the high frequency vibration noise, other types of noise can be removed by FIR Gaussian filter. The compass module comes with 5 options of filtering: 32, 16, 8, 4, and 0 tap filter. The higher the number is, the more stable the output is, but longer latency is expected. Consider the case of selecting 32 tap filter (Figure 3). When it is the time to send out estimated data at moment  $A$ , what the module does is adding a new sample  $A$  to the end of the queue with the first one being dropped, and applying Gaussian filter to the queue. However the filtered result actually reflects the estimated value at moment  $(A-15)$ . Since the module samples at approximately 30 – 32 Hz, it induces 0.5 second delay if choosing 32 tap filter; 0.25 second delay for 16 tap filter and so on. This is called filter-induced latency and applies to both POLL and PUSH mode. 0 tap filter implies no filtering but with significant jittering.

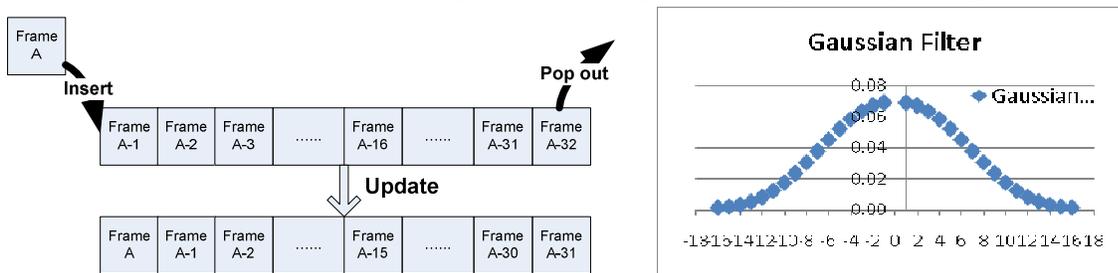


Figure 3: The Filter-induced latency when 32 tap Gaussian filter is used.

### 5.3 Half Window Gaussian Filter

In order to avoid the filter-induced latency, the Gaussian FIR filter is removed from hardware to software but with only half window size applied. For example, if complete Gaussian window is used, it is not until moment  $A+15$  that estimated value can be available for moment  $A$ . However half window replicates the past data from moment  $A-15$  to moment  $A$  as the future data from moment  $A+1$  to  $A+16$ , and generates estimated value for moment  $A$  (Figure 4). Nevertheless as it is shown in the graph chart, half window still causes 4-5 frames latency on average. Depending on the speed of module movement, the faster the speed is, the longer latency it presents. We address this kind of latency as half window induced latency.

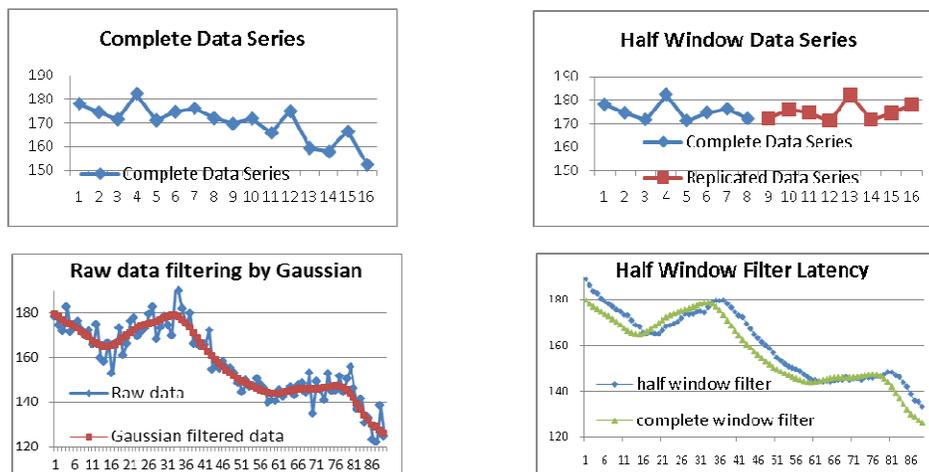


Figure 4: Half window filter latency.

Because half window Gaussian filter puts more emphasis on the current frame, it makes the estimated result more sensitive to noise contained in the current frame, and consequently more jittering than the estimated result of complete window Gaussian filter. Therefore a second half window Gaussian is applied on first filtered result for smoothing purpose but introduces extra 1-2 frames latency (Figure 5). However this additional latency can be discounted because it does not exceed the original latency, the one between half window Gaussian filter and complete window Gaussian filter. Therefore double of the additional latency is subtracted from the twice Gaussian filter result, and it makes the estimation closer to the actual data than half window Gaussian filter result. Unfortunately, this approach fails during the transition state and leads to overshooting during change of direction, and transition from dynamic to static states.

### 5.4 Adaptive Latency Compensation Algorithm

In order to resolve the overshooting problem, the estimated result needs to be forced to the observed data when the module comes to a stop. This is possible because the observed data is very stable and close to the actual value when the module is static. Large collections of observed value shows standard deviation as a good indicator of dynamic and static: when the standard deviation is larger than 6, the heading component of the module is in motion; otherwise it is in static or on the way of coming to a stop. Therefore the adaptive algorithm equals the latency compensated value to double the difference between the twice Gaussian filter and half window Gaussian filter result, when the standard deviation is no larger than 6; otherwise equals it to the difference between twice Gaussian filter result and the observed data.

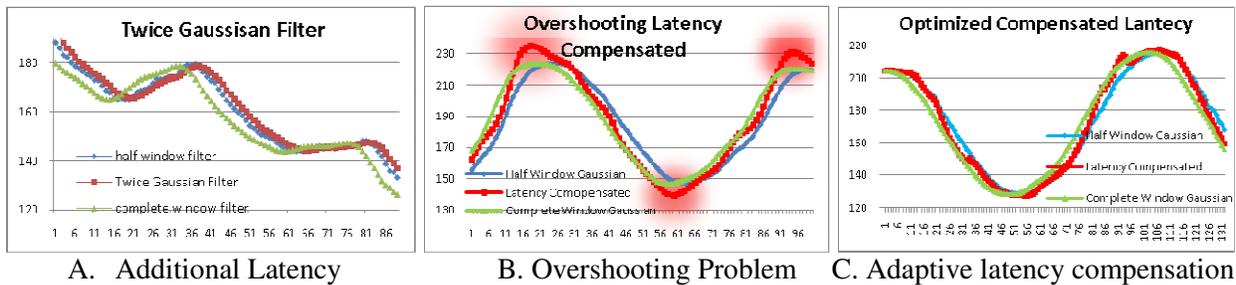


Figure 5: Adaptive latency compensation algorithm.

## 6 VALIDATION

The robustness of ARMOR and SMART framework has been tested on an ongoing excavation collision avoidance project. Electricity conduits in the vicinity of the G.G. Brown Building at the University of Michigan were exported as KML files from a Geodatabase provided by the DTE Energy Company. The following procedure interprets KML files and builds conduit models: (1) extract spatial and attribute information of conduits from a KML file using libkml, a library for parsing, generating, and operating on KML (Google libkml, 2010); (2) convert consecutive vertices within one “LineString” (Google KML, 2010) from geographical coordinate to local coordinate; (3) a unit cylinder is shared by all conduit segments as primitive geometry upon which the transformation matrix is built; (4) scale, rotate, and translate the cylinder to the correct size, attitude and position.

## CONCLUSION AND FUTURE WORK

This paper has demonstrated a robust mobile computing platform composed of rigid hardware platform ARMOR and application framework SMART. Targeting at centimeter level accuracy AR tasks, algorithms for both static and dynamic registration have been introduced. So far, dynamic misregistration is still under investigation by the authors. Several efforts are being made: 1) synchronizing the captured image and sensor measurements; and 2) optimizing the adaptive latency compensation algorithm with image processing techniques e.g. optical flow can afford a better clue about the moving speed.

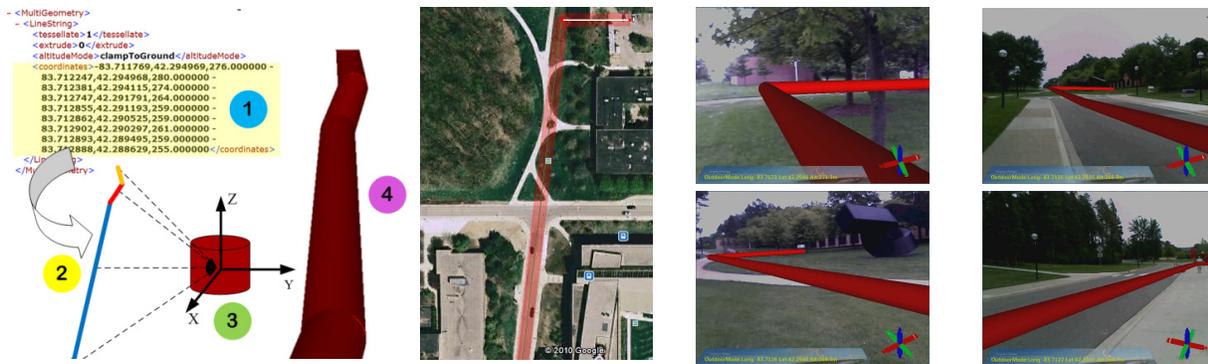


Figure 6: Conduit loading procedure, conduits overlaid on Google Earth and field experiment results

## ACKNOWLEDGMENTS

The presented work has been supported by the United States National Science Foundation (NSF) through Grants CMMI-0448762 and CMMI-0825818. The authors gratefully acknowledge NSF's support. The authors thank the DTE Energy Company for providing geodatabases of their underground assets and for their enthusiastic ongoing collaboration in this project. The authors are also grateful to Ph.D. student Mr. Sanat Talmaki for helping prepare the conduit datasets. Any opinions, findings, conclusions, and recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF, DTE, or the individual mentioned here.

## REFERENCES

- Azuma, R. T. 1997a. A Survey of Augmented Reality. In *Proceedings of the 1997 Teleoperators and Virtual Environments*, 1~38.
- Azuma, R. T. 1997b. Making Direct Manipulation Work in Virtual Reality. In *Proceedings of the 1997 ACM SIGGRAPH*.
- Azuma, R.T., and B. Hoff, H. Neely, and R. Sarfaty. 1999. A Motion-Stabilized Outdoor Augmented Reality System. In *Proceedings of the 1999 IEEE Virtual Reality*. Houston, Texas: IEEE Computer Society Washington, DC, USA.
- Azuma, R.T., Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre. 2001. Recent Advances in Augmented Reality, *IEEE Computer Graphics and Applications*, 21(6): 34-47.
- Behzadan, A. H., and V. R. Kamat. 2007. Georeferenced Registration of Construction Graphics in Mobile Outdoor Augmented Reality. *Journal of Computing in Civil Engineering*, 21(4): 247-258, Reston, VA: American Society of Civil Engineers.
- Behzadan, A. H., B. W. Timm, and V. R. Kamat. 2008. General Purpose Modular Hardware and Software Framework for Mobile Outdoor Augmented Reality. *Advanced Engineering Informatics*, 22 (2008): 90-105, New York, NY: Elsevier Science.
- Behzadan, A. H., and V. R. Kamat. 2009. Automated Generation of Operations Level Construction Animations in Outdoor Augmented Reality. *Journal of Computing in Civil Engineering*, 23(6): 405-417, Reston, VA: American Society of Civil Engineers.
- Feiner, S., B. MacIntyre, T. Hollerer, and A. Webster. 1997. A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. In *Proceedings of the 1997 ISWC*, 208-217, Cambridge, MA.
- Golparvar-Fard, M., F. Pena-Mora, and S. Savarese. 2009. D4AR- A 4-Dimensional augmented reality model for automating construction progress data collection, processing and communication. *Journal of Information Technology in Construction*, 14(2009), 129-153.
- Google KML 2010. *KML Reference*. Available via <http://code.google.com/apis/kml/documentation/kmlreference.html> [accessed June 6, 2010].

- Google libkml 2010. *Libkml*. Available via <<http://code.google.com/p/libkml>> [accessed June 6, 2010].
- Jacobs, M. C., M. A. Livingston, and A. State. 1997. Managing Latency in Complex Augmented Reality Systems. In *Proceedings of the 1997 Symposium on Interactives 3D Graphics*.
- Kamat, V. R., and S. El-Tawil. 2007. Evaluation of Augmented Reality for Rapid Assessment of Earthquake-Induced Building Damage. *Journal of computing in civil engineering* 21(5): 303-310, Reston, VA: American Society of Civil Engineers.
- Kenner, C. 2010. *GlovePIE Home Page*. Available via <<http://glovepie.org/>> [accessed June 2, 2010].
- Liang, J., C. Shaw, and M. Green. 1991. On temporal-spatial realism in the virtual reality environment. In *Proceeding of 1991 Symposium on User Interface Software and Technology*. Hilton Head, South Carolina : ACM New York.
- Piekarski, W. 2004. *Interactive 3D Modelling in Outdoor Augmented Reality Worlds*. Ph.D. thesis, Department of Computer Systems Engineering, University of South Australia, Adelaide, Australia. Available via <<http://www.tinmith.net/wayne/thesis/piekarski-thesis.htm>> [accessed June 14, 2010].
- Piekarski, W., R. Smith, and B. Avery 2006. *Tinmith mobile AR backpacks*. Available via <<http://www.tinmith.net/backpack.htm>> [accessed June 4, 2010].
- Roberts, G., A. Evans, A. Dodson, B. Denby, S. Cooper, and R. Hollands. 2002. The use of augmented reality, GPS, and INS for subsurface data visualization. In *Proceedings of the 2002 FIG XIII International Congress*. Washington, D.C.
- Shreiner, D., M. Woo, J. Neider, and T. Davis. 2006. *OpenGL Programming Guide*. 5th ed. Upper Saddle River, New Jersey: Prentice-Hall, Inc.
- Tekkeon 2009. *MP3450i/MP3450/MP3750 datasheets*. Available via <[http://www.tekkeon.com/downloads/dtasht\\_MP3750.pdf](http://www.tekkeon.com/downloads/dtasht_MP3750.pdf)> [accessed June 1, 2010].
- Thomas, B., W. Piekarski, and B. Gunther. 1999. Using Augmented Reality to Visualize Architecture Designs in an Outdoor Environment. In *Proceedings of the 1999 Design Computing on the Net*. Sydney.
- Trimble 2007. *AgGPS RTK Base 900 and 450 receivers*. Available via <[http://trl.trimble.com/docushare/dsweb/Get/Document-392922/AgGPSRTKBase\\_3.30A\\_UserGuide\\_ENG.pdf](http://trl.trimble.com/docushare/dsweb/Get/Document-392922/AgGPSRTKBase_3.30A_UserGuide_ENG.pdf)> [accessed April 3, 2010].
- Webster, A, S. Feiner, B. MacIntyre, W. Massie, and T. Krueger. 1996. Augmented Reality in Architectural Construction, Inspection and Renovation. In *Proceedings of the 1996 3rd Congress on Computing in Civil Engineering* . 913-919. Reston, VA.
- WIKIPEDIA 2010. *Wii Remote*. Available via <[http://en.wikipedia.org/wiki/Wii\\_Remote](http://en.wikipedia.org/wiki/Wii_Remote)> [accessed June 3, 2010].
- Z800 3DVisor 2010. *Z800 3DVisor User's Manual*. Available via <<http://www.3dvisor.com/downloads/UserGuide10.7.pdf>> [accessed June 1, 2010].

## AUTHOR BIOGRAPHIES

**SUYANG DONG** is a Ph.D. student of Construction Engineering and Management at the University of Michigan. He got his MEng in Construction Engineering and Management from the same university in 2010, and a B.Sc degree in Geographical Information Systems from Wuhan University in 2008. His current research interests are in building robust outdoor Augmented Reality systems and solving incorrect occlusion in dynamic Augmented Reality. His e-mail address is <[dsuyang@umich.edu](mailto:dsuyang@umich.edu)> .

**VINEET R. KAMAT** is an Associate Professor in the Department of Civil and Environmental Engineering at the University of Michigan. He received a Ph.D. in Civil Engineering at Virginia Tech in 2003; a M.S. in Civil Engineering at Virginia Tech in 2000; and a B.E. degree in Civil Engineering from Goa University (Goa, India) in 1998. His primary research interests include virtual and augmented reality, simulation, information technology, and their applications in Civil Engineering. His email and web addresses are <[vkamat@umich.edu](mailto:vkamat@umich.edu)> and <<http://pathfinder.engin.umich.edu>> .