Visualization in Engineering
a SpringerOpen Journal

RESEARCH ARTICLE Open Access

SMART: scalable and modular augmented reality template for rapid development of engineering visualization applications

Suyang Dong[1*] and Vineet R Kamat[2]

Abstract

Background: The visualization of civil infrastructure systems and processes is critical for the validation and communication of computer generated 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 create or replicate on the computer alone.

Methods: This paper presents a scalable and extensible mobile computing framework that allows users to readily create complex outdoor AR visual applications. The technical challenges of building this reusable framework from the software and hardware perspectives are described.

Results: SMART is a generic and loosely-coupled software computing framework for creating AR visual applications with accurate registration algorithms and visually credible occlusion effects. While SMART is independent of any specific hardware realization, ARMOR is built as a hardware implementation of SMART to test its algorithm correctness and its adaption to engineering applications. In particular, ARMOR is a modular mobile hardware platform designed for user position and orientation tracking, as well as augmented view display.

Conclusion: The framework has been validated in several case studies, including the visualization of underground utilities for excavator control and collision avoidance to demonstrate SMART's rapid creation of complex AR visual applications.

Introduction

Augmented Reality (AR) refers to a visualization technology that superimposes virtual objects on the real world so as to provide information beyond reality and enhance people's interaction with the environment. It possesses distinct advantages over other visualization methods in at least three aspects: 1) From the perspective of visualization, the integration of the real world can significantly mitigate the efforts to create and render contextual models for virtual scenes, and can provide a better perception of the surroundings than virtual reality alone (e.g., visualization of construction simulations) (Behzadan and Kamat 2007), and the visualization of architectural designs (Thomas et al. 1999); 2) From the perspective of information retrieval, AR supplements a user's normal vision with context-related or Georeferenced virtual objects (e.g., looking through walls to see columns (Webster et al. 1996), looking beneath the ground to inspect subsurface utilities (Roberts et al. 2002), or retrieving drawings and BIM models for on-site communication (Yeh et al. 2012); and 3) From the perspective of interaction, authentic virtual models can be deployed to evaluate the physical condition of real objects (e.g., evaluation of earthquake-induced building damage (Kamat and El-Tawil 2007), and automation of construction process monitoring (Golparvar-Fard et al. 2009)). There have been other proposed applications for AR in construction such as training platforms for heavy construction equipment operators (Wang and Dunston 2007). For a comprehensive and state-of-the-art review of applications of AR, especially in construction and built environments, the reader is referred to (Wang et al. 2013).

While the aforementioned AR engineering applications possess tangible economic and social values, some fundamental technical challenges have to be overcome before

* Correspondence: dsuyang@umich.edu
[1]Department of Civil and Environmental Engineering, University of Michigan, 2350 Hayward Street, Suite 1380, Ann Arbor, MI 48105, USA
Full list of author information is available at the end of the article

AR can be deployed in practical outdoor applications. The difficulties are associated with two requirements: 1) maintaining a constant spatial alignment between the virtual and real entities, which is also referred as registration; and 2) creating a sustained illusion that the virtual and real entities co-exist, which is also referred to as occlusion.

Since most engineering AR applications expect to encounter registration and occlusion problems, it is reasonable to solve these challenges and build the solutions into a scalable and extensible AR computing framework that is openly accessible to the community. Researchers who are interested in exploring AR for their specific application in construction or another domain can immediately have access to the core logic components without starting from scratch on developing solutions for the registration and occlusion issues. The existence of such a reusable framework can significantly shorten the lifecycle of developing AR engineering applications.

A real-time occlusion algorithm with robust depth sensing and frame buffer techniques for handling occlusion problems in ubiquitous AR applications has been presented in the paper (Dong and Kamat 2012b), and the designed software module has been built into the SMART framework. This paper primarily focuses on the fundamental challenge of achieving precise registration in AR visualizations from both the hardware and software perspectives.

Backgound: Registration challenges in unprepared environments

The fundamental challenge in Augmented Reality is the difficulty of aligning virtual objects with the real environment with the correct pose, a process which is called registration in the literature (Azuma 1997). It is difficult because the registration errors arise from both the spatial and temporal domains (Azuma 1997). In addition, different tracking technologies have their own registration error sources. This paper focuses on solving the registration challenge in an unprepared environment (i.e., outdoors, where sensor-based AR is by far the most reliable tracking method free from constraints put on the user).

Errors in the spatial domain are also referred to as static errors when neither the user nor virtual objects are in motion. The static errors of sensor-based AR include: 1) inaccuracy in the sensor measurement; 2) mechanical misalignments between sensors; and 3) an incorrect registration algorithm. The selection of high-accuracy sensors is crucial, because the errors contained in the measurement can rarely be eliminated. Rigid placement of sensors on the AR backpack and helmet is also essential; else it can further compromise the system accuracy. Some early, relatively fragile AR backpack design examples can be found in the touring machine (Feiner et al. 1997) and Tinmith-Endeavour (Piekarski et al. 2004). A more robust and ergonomic version is demonstrated by the Tinmith backpack 2006 version (Stafford et al. 2006), in which a GPS antenna and an InterSense orientation tracker are anchored on top of a helmet. However, the 50cm accuracy of its GPS receiver is not sufficient for centimeter-level-accuracy AR tasks.

Compared with static errors, dynamic errors—errors in temporal domain—are much more unpredictable. The differences in latency between data streams create pronounced "swimming" effect of dynamic misregistration, which is called relative latency by (Jacobs et al. 1997). Given modern machine computational power, relative latency mainly results from: 1) off-host delay: the 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. Two common mitigation methods for resolving relative latency are: 1) adopting multi-threading programming or scheduling system latency (Jacobs et al. 1997); and 2) predicting head motion using a Kalman filter (Liang et al. 1991; Azuma et al. 1999).

Contributions: Scalable and modular augmented reality template

The mobile computing framework presented in this paper provides a comprehensive hardware and software solution for centimeter-level-accuracy AR applications in the outdoor environment. The robustness of the framework has been presented with a case study in visualizing underground infrastructure as part of an excavation planning and safety project.

The Scalable and Modular Augmented Reality Template (SMART) builds on top of the ARVISCOPE software platform (Behzadan and Kamat 2009). Besides its engine that interprets visual simulation scripts, another contribution of ARVISCOPE is creating some basic modules communicating with peripheral hardware that can later be 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, and make the libraries hardware independent, thus providing a flexible and structured AR development environment for the rapid creation of complex AR visual applications.

The in-built registration algorithm of SMART affords high-accuracy static alignment between real and virtual objects. Efforts have also been made to reduce dynamical misregistration, including: 1) in order to reduce synchronization latency, multiple threads are dynamically generated for reading and processing sensor measurement immediately upon the data arrival in the host system; and 2) an adaptive lag compensation algorithm is designed to reduce the latency induced by the Finite Impulse Response (FIR) filter.

In order to test the robustness and effectiveness of SMART in specific engineering application, ARMOR is constructed as an hardware realization of SMART, The Augmented Reality Mobile OpeRation platform (ARMOR) evolves from the UM-AR-GPS-ROVER hardware platform (Behzadan et al. 2008). ARMOR improves the design of the UM-AR-GPS-ROVER in two distinct ways: rigidity and ergonomics. It introduces high-accuracy and light-weight devices, rigidly places all tracking instruments with full calibration, and upgrades the carrying harness to make it more wearable.

Methods: 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. Furthermore, SMART is designed to be hardware independent, therefore developer can pick their own hardware profile, write modules that conforms to SMART interface, and make the hardware run on top of SMART.

The SMART framework follows the classical model-view-controller (MVC) pattern. Scene-Graph-Controller is the implementation of the MVC pattern in SMART: (1) the model counterpart in SMART is the CARScene that utilizes application-specific I/O engines to load virtual objects, and that maintains their spatial and attribute status. The update of a virtual object's status is reflected when it is time to refresh the associated graphs; (2) the CARGraph corresponds to the view and reflects the AR registration results for each frame update event. Given the fact that the user's head can be in constant motion, the graph always invokes callbacks to rebuild the transformation matrix based on the latest position and attitude measurement, and refreshes the background image; (3) the CARController—manages all of the UI elements, and responds to a user's commands by invoking delegates' member functions like a scene or a graph.

The framework of SMART that is based on a Scene-Graph-Controller set-up is constructed in the following way (Figure 1). The main entry of the program is CARApp, which is in charge of CARSensorForeman and CARSiteForeman. The former initializes and manages all of the tracking devices, like RTK rovers and electronic compasses. The latter one defines the relation among scene, graphs, and controller. After a CARSiteForeman object is initialized, it orchestrates the creation of CARController, CARScene, and CARGraph, and the connection of graphs to the appropriate scene. The controller keeps pointers to the scene and the graph(s).

Application for operations-level construction animation

As a study case of adapting SMART to a specific engineering application, ARVISCOPE animation functions have been re-implemented under the SMART framework as follows. In order to load the ARVISCOPE animation trace file (Behzadan and Kamat 2009), CARSiteForemanA contains CARSceneA, CARControllerA , and CARGraphA, all of which are subclasses inheriting from SMART's superclasses, and are adapted for animation function. For example, CARSceneA employs CARStatementProcessor and CARnimation classes as the I/O engine to interpret the trace file, and CARControllerA adds customized elements for controlling the animation such as play, pause, continue, and jump functions.

Static registration

Registration process

The registration process of Augmented Reality is very similar to the computer graphics transformation process: 1) positioning the viewing volume of a user's eyes in the world coordinate system; 2) positioning objects in the world coordinate system; 3) determining the shape of the viewing volume; and 4) converting objects from the world coordinate system to the eye coordinate system (Shreiner et al. 2006). However, unlike computer graphics where parameters needed for step 1 through 3 are coded or manipulated by the user, Augmented Reality rigidly fulfills these steps according to the 6 degrees of freedom measured by tracking devices and the lens parameter of the real camera. Figure 2 lists the registration process, the needed parameters, and their measuring devices.

Step 1 – viewing The viewing step computes the relative transformation from the world coordinate system to the eye coordinate system. The origin of the world coordinate system coincides with that of the eye coordinate system, which is the user's geographical location (Figure 3). The world coordinate system uses a right-handed system with the Y-axis pointing in the direction of the true north, the X-axis pointing to the east, and the Z-axis pointing upward. The eye coordinate system complies with the OpenSceneGraph (OSG) (Martz 2007) default coordinate system, using a right-handed system with the Z-axis as the up vector, and the Y-axis departing from the eye.

Yaw, pitch, and roll—all measured by the magnetic sensor—are used to describe the relative orientation between the world and eye coordinate systems (Figure 4). There are six possibilities of rotating sequences (i.e., xyz, xzy, zxy, zyx, yzx, and yxz), and zxy is picked to construct the transformation matrix between the two coordinate systems (Equation 1). Suppose the eye and world coordinate systems coincide at the beginning; the user's head rotates around the Z-axis by yaw angle $\Psi \in (-180, 180]$ to get the new axes, X' and Y'. Since the rotation is

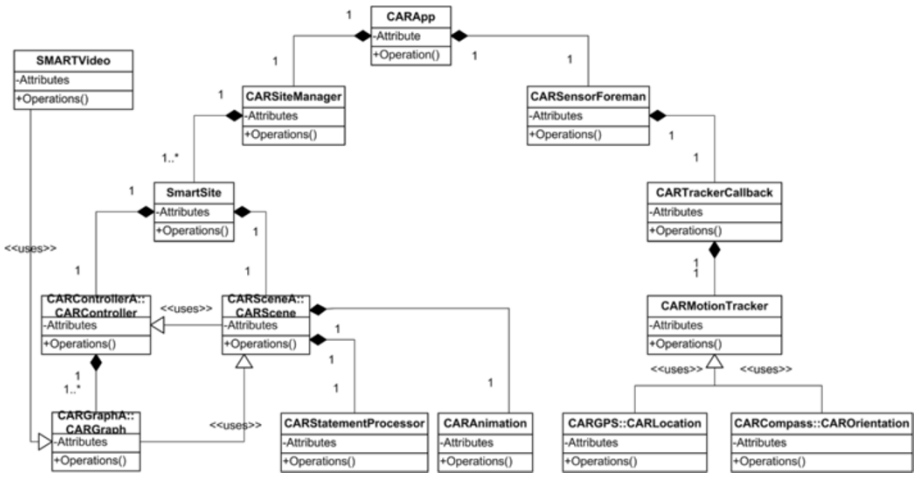


Figure 1 SMART framework architecture.

clockwise under the right-handed system, the rotation matrix is $R_z(-\Psi)$. Secondly, the head rotates around the X'-axis by pitch angle $\theta \in [-90, +90]$ to get the new axes, Y" and Z', with counter-clockwise rotation of $R_{x'}(\theta)$. Finally, the head rotates around the Y"-axis by roll angle $\varphi \in (-180, 180]$ with a counter-clockwise rotation of $R_{y''}(\varphi)$ to reach the final attitude.

Converting the virtual object from the world coordinate to the eye coordinate is an inverse process of rotating from the world coordinate system to the eye coordinate system, therefore the rotating matrix is written as: $R_z(\Psi)\ R_{x'}(-\theta)\ R_{y''}(-\varphi)$ or $R_z(\text{yaw})\ R_{x'}(-\text{pitch})\ R_{y''}(-\text{roll})$ (Equation 2). The rotation matrix can be further constructed as quaternion, a simple and robust way to express rotation, by specifying the rotation axis and angles. The procedure is explained as follows, and its associated equations are listed in sequence from Equation 3 to 5 (referenced in Figure 4): rotating around the Y"-axis by $-\varphi$ degree, then rotating around the X'-axis by $-\theta$ degree, and finally rotating around the Z-axis by Ψ degree.

$$\begin{bmatrix} Xe \\ Ye \\ Ze \end{bmatrix} = \begin{bmatrix} \cos\Psi & \sin(-\Psi) & 0 \\ \sin\Psi & \cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & \sin(-\theta) & \cos\theta \end{bmatrix} * \begin{bmatrix} \cos\varphi & 0 & \sin(-\varphi) \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} * \begin{bmatrix} Xw \\ Yw \\ Zw \end{bmatrix} \quad (1)$$

(X_w, Y_w, Z_w) is the point expressed in the world coordinate system; (X_e, Y_e, Z_e) is the same point but expressed in the eye coordinate system.

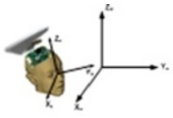
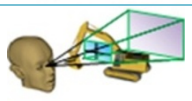
Step	Task	Illustration	Parameters and Device
Viewing	Position the viewing volume of a user's eyes in the world		Attitude of the camera (Electronic Compass)
Modeling	Position the objects in the world		Location of the world origin (RTK GPS)
Creating Viewing Frustum	Decide the shape of the viewing volume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

Figure 2 The four steps of registration process.

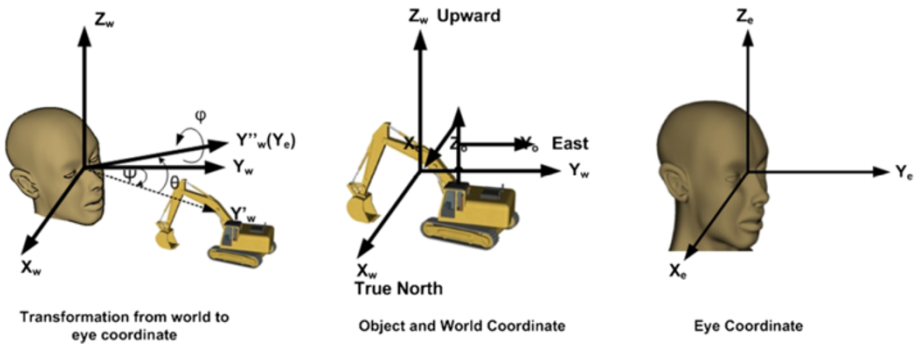


Figure 3 Definition of the object coordinate system, world coordinate system, and eye coordinate system.

$$Pe = RzΨ * Rx'(-\theta) * Ry''(-\varphi) * Pw \tag{2}$$

Equation 2 is a simplified version of Equation 1, while $R_A B$ stands for rotation matrix around axis A by degree of B.

$$Z\text{–}axis = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{3}$$

$$X' \text{–}axis = \begin{bmatrix} \cos\Psi & \sin\Psi & 0 \\ \sin(-\Psi) & \cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos\Psi \\ -\sin\Psi \\ 0 \end{bmatrix} \tag{4}$$

$$Y'' \text{–}axis = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin(-\theta) \\ 0 & \sin\theta & \cos\theta \end{bmatrix} * \begin{bmatrix} \cos\Psi & \sin\Psi & 0 \\ \sin(-\Psi) & \cos\Psi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} \sin\Psi \\ \cos\ \theta\cos\Psi \\ \sin\ \theta\cos\Psi \end{bmatrix} \tag{5}$$

Step 2 – modeling The modeling step positions the virtual objects in the world coordinate system. The definition of the object coordinate system is determined by the drawing software. The origin is fixed to a pivot point on the object with program-specified geographical location. The geographical location of the world coordinate origin is also given by the GPS measurement; the 3D vector between the object and world coordinate origins can thus be calculated. The methods to calculate the distance between geographical coordinates is originally introduced by (Vincenty 1975), and (Behzadan and Kamat 2007) proposed a reference point concept to calculate the 3D vector between two geographical locations. SMART adopts the same algorithm to place a virtual object in the world coordinate system using the calculated 3D vector. After that, any further translation, rotation, and scaling operations are applied on the object.

Steps 3 and 4 – viewing frustum and projection With the viewing frustum and projection, the virtual objects are ultimately projected on the image plane. The real world is perceived through the perspective projection by the human eye and the web camera. Four parameters are needed to construct a perspective projection matrix: horizontal angle of view, horizontal and vertical aspect ratio, and NEAR plane and FAR plane (any content is clipped if its distance falls outside the bound of NEAR and FAR place). All of them together form a viewing frustum and decide the amount of virtual content shown in

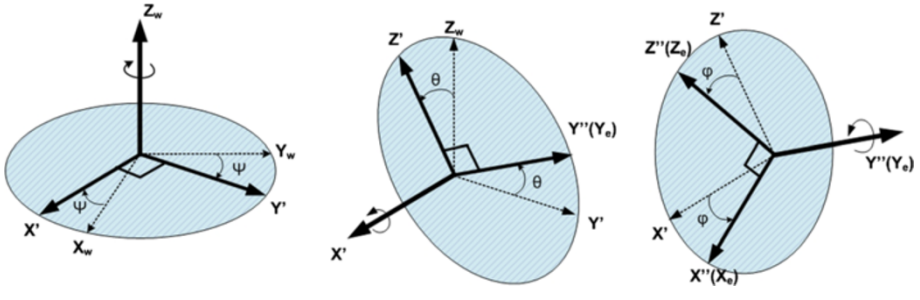


Figure 4 The relative orientation between the world and eye coordinate systems is described by yaw, pitch, and roll.

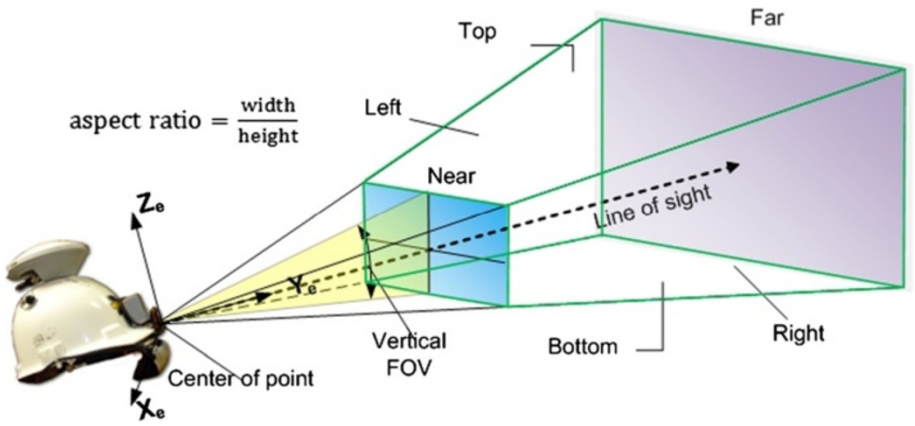


Figure 5 The viewing frustum defines the virtual content that can be seen.

the augmented space (Figure 5). Virtual objects outside of the viewing frustum are either cropped or clipped.

The NEAR and FAR planes do not affect how the virtual object appears on the projection plane. However, to maintain a high precision z-buffer, the principle is to keep the NEAR plane as far as possible, and the FAR plane as close as possible. The horizontal and vertical angle of view directly influence the magnification of the projected image and are affected by the focal length and aspect ratio of the camera. In order to ensure consistent perspective projection between the real and virtual camera, both of them need to share the same angle of view. This is achieved by calibrating the real camera and conforming the virtual camera's angle to view to that of the real camera.

Registration validation experiment

Calibration of the mechanical attitude discrepancy

The mechanical attitude discrepancy between the real camera and the electronic compass needs to be compensated by the following calibration procedure. The mechanical position discrepancy between the real camera and the GPS receiver can be calibrated in the similar way in the outdoor environment. A real box of size 12cm × 7 cm × 2 cm (length × width × height) is placed at a known pose. A semi-transparent 3D model

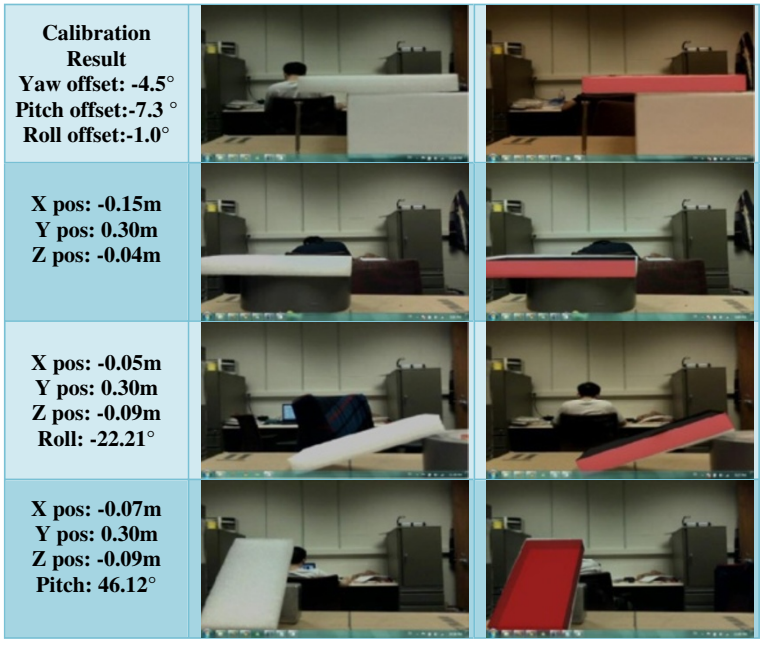


Figure 6 Mechanical attitude calibration result and validation experiment of registration algorithm.

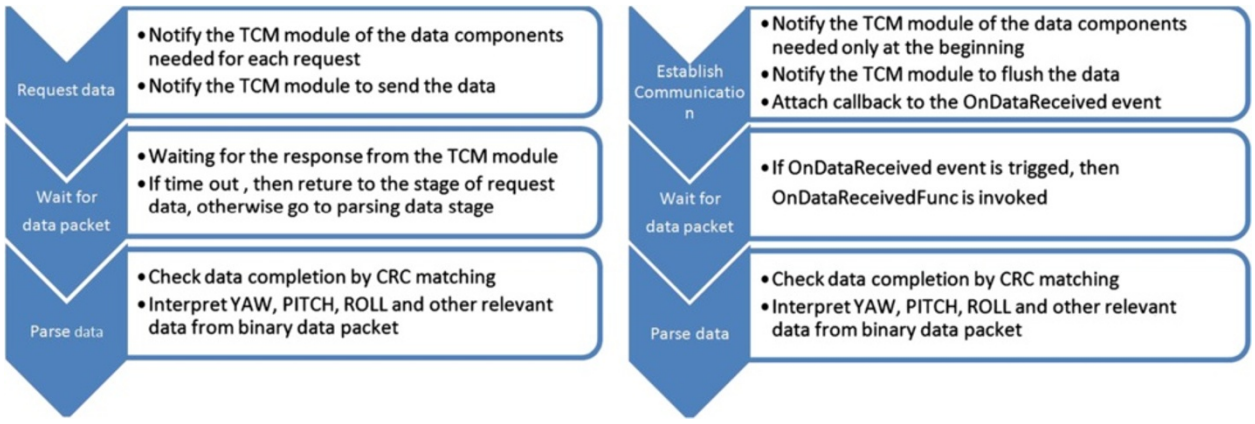


Figure 7 Communication stages in the PULL Mode (left) and the PUSH Mode (right).

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 adjustments to the attitude measurement, and discrepancy is thus present. The virtual box is then shifted to align with the real one by adding a compensation value to the attitude measurement, as shown in Figure 6 Row 1.

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 accurately matches the real one in all tested cases, and a selected set of experiments are shown below in Figure 6 Rows 2~3.

Resolving the latency problem in the 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.

Multi-threading to reduce synchronization latency

There are two options for communicating with the compass module: PULL and PUSH mode. PULL is a passive output mode for the compass module, and is used by

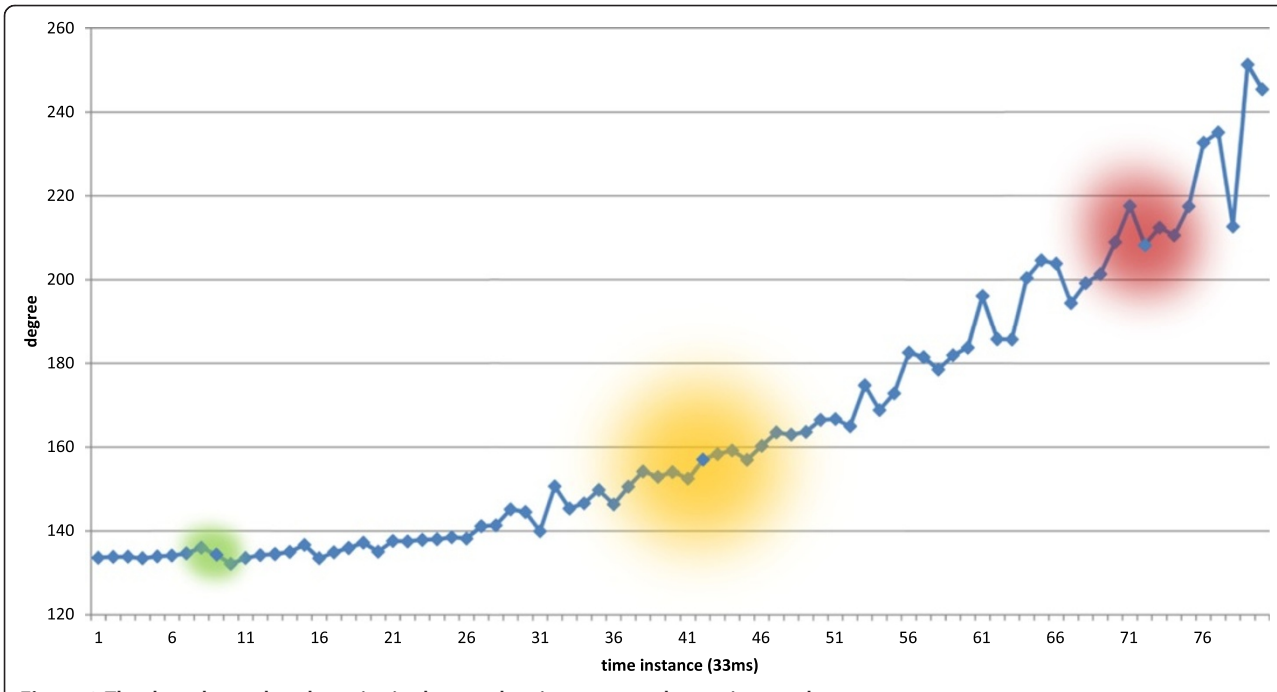


Figure 8 The data shows that the noise in the raw data increases as the motion accelerates.

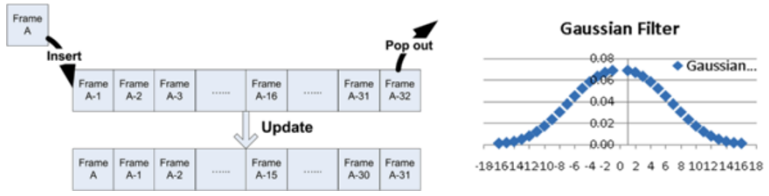


Figure 9 The Filter-induced latency when a 32 tap Gaussian filter is used.

the UM-AR-GPS-ROVER to pull data out of the module. Without separating I/O communication with the electronic compass as a background task, one loop of the pulling request costs 70ms on average, and significantly slows down program performance.

The PUSH mode is an active output mode for the compass module. SMART selects the PUSH mode as its data communication method to increase the program efficiency. If the PUSH mode is selected, the module outputs the data at a fixed rate set by the host system (Figure 7). If the fixed rate is set to 0, which is done by SMART, the module will flush the next data packet as soon as the previous one is sent out. The sampling and flushing happens at proximately 30 to 32 Hz. The biggest advantage of choosing the PUSH mode is that, once the initial communication is successfully established, and no FIR filtering is carried on the hardware board, the host system can acquire the observed orientation data in only 5 ms on average. Accordingly, SMART adopts an event-based asynchronous pattern to handle high frequency data packet arrival and process the data on a separate thread in the background without interrupting the main loop. To be specific, a dedicated listener is set up on a separate thread. It is active in listening to the data package posted from the module device. Once data package arrives, the listener triggers its attached delegate that computes the registration result. This multi-threaded processing accelerates the main function and also reduces the synchronization latency to the minimum possible value.

Filter-induced latency

Even though the PUSH mode is free of synchronization delay, there is still significant latency if the FIR filter is switched on inside of the compass module. This section explains the reason for this phenomenon. The calibration of the magnetometer can compensate for a local static magnetic source within the vicinity of the compass module. However, dynamic magnetic distortion still impacts the module in motion, and the noise magnification depends on the acceleration of the module.

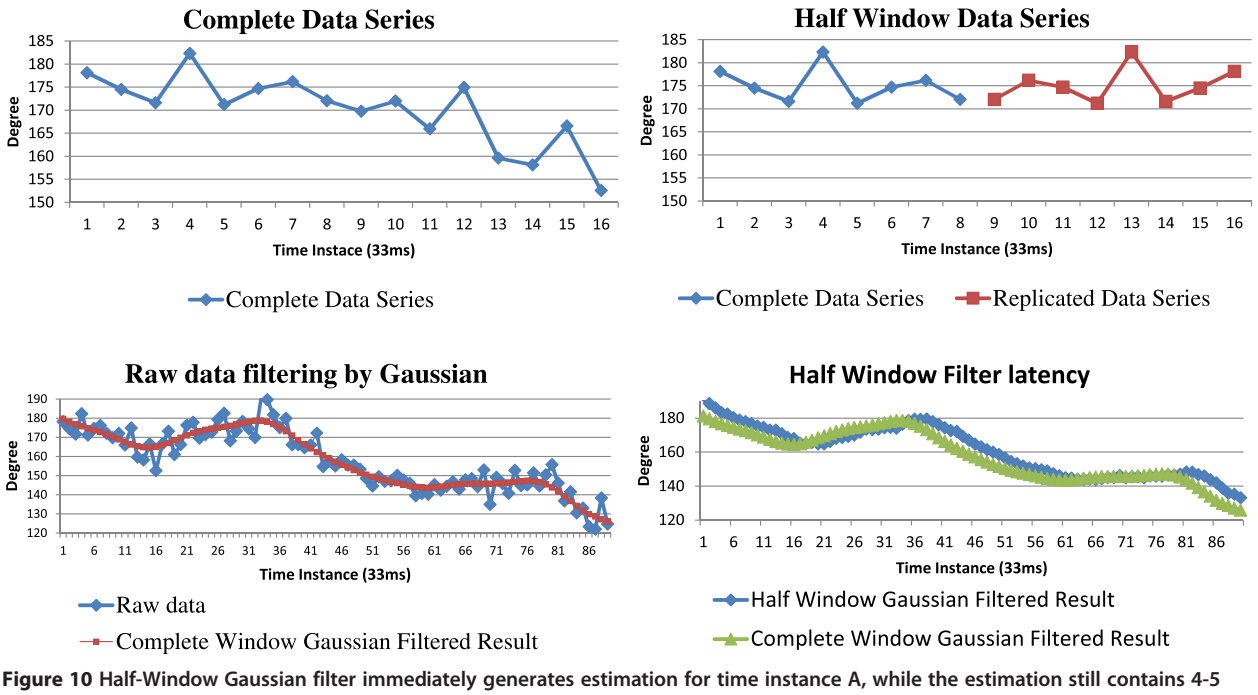


Figure 10 Half-Window Gaussian filter immediately generates estimation for time instance A, while the estimation still contains 4-5 frames of latency on average.

Usually the faster the acceleration is, the higher the noise is (Figure 8). Among the three degrees of freedom, heading is the most sensitive to the noise.

Except the high frequency noise that can be caused by abrupt vibration and can significantly distort FIR filtered results, other types of noise, especially jittering noise, can be removed/smoothed by a FIR Gaussian filter. The compass module comes with 5 options for filtering: 32, 16, 8, 4 and 0 tap filter. The higher the number is, the more stable the output, but the longer the expected latency. The reason can be illustrated by the case of selecting a 32 tap filter (Figure 9). When it is time to send out estimated data at time instant A, the module adds a new sample A to the end of the queue with the first one being dropped, and applies a Gaussian filter to the queue. However, the filtered result actually reflects

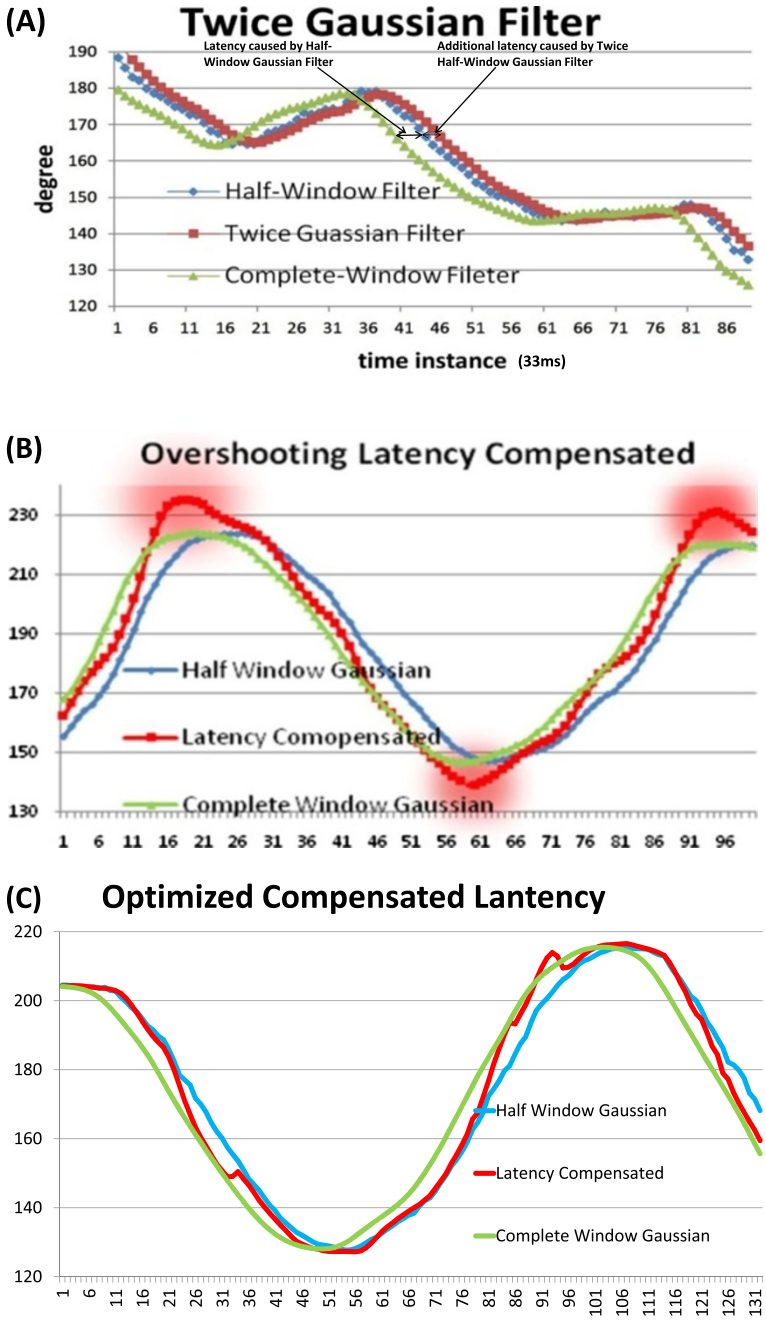


Figure 11 Twice Half-Window Gaussian Filter causes additional latency, however it is also a hint for making the Half-Window Gaussian filtered results more close to the 'true' value. (**A**) Shows the latency caused by Twice Gaussian Filter; (**B**) shows the compensation results with overshooting problem at the transitions; (**C**) shows the optimized compensation algorithm to alleviate the overshooting problem.

the estimated value at time instant (A−15). Since the module samples at approximately 30–32 Hz, it induces a 0.5 second delay for a 32 tap filter; a 0.25 second delay for 16 tap filter, and so on. This is called filter-induced latency, and it applies to both PULL and PUSH mode. A 0 tap filter implies no filtering, but significant jittering.

Half-window Gaussian filter

In order to avoid the filter-induced latency, the Gaussian FIR filter is moved from the hardware to the software, but with only half window size applied. For example, if a complete Gaussian window is used, it is not until time instant A+15 that the estimated value can be available for time instant A. However, Half-Window replicates the past data from time instant A-15 to time instant A as the future data from time instant A+1 to A+16, and immediately generates an estimated value for time instant A without any delay (Figure 10). Nevertheless, as is shown in the graph chart, half window still causes 4–5 frames of latency on average. Depending on the speed of module movement, the faster the speed, the longer latency it presents. We address this kind of latency as Half-Window induced latency.

Because the 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 the estimated result is more jittering than the one of the Complete-Window Gaussian filter. Therefore, a second Half-Window Gaussian is applied on the first Half-Window Gaussian filtered result for smoothing purposes, but this introduces 1–2 extra frames of latency. This extra latency can be found in Figure 9A, where the filtered result in red of Twice Half-Window Gaussian filter is pushed toward right for additional 2 time instances compared with the filtered result in blue of Half-Window filter (Figure 11A). However, unlike the Half-Window Gaussian latency which cannot be computed (since the 'true value' is unknown), the additional latency can be calculated as the subtraction between the Half-Window and Twice Half-Window Gaussian filter results. Furthermore, in observance that the additional latency can be is always smaller the original Half-Window latency. double the additional latency can thus be subtracted from the Twice Half-Window Gaussian filter result which makes the estimation closer to the 'true value'than the Half-Window Gaussian filter result. This approach works most of the time except at the transition state, and it leads to overshooting during change of direction, and during transitions from dynamic to static states as it is shown Figure 9B.

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 when the module is static, the observed data is very stable and close to the actual. Large collections of observed value show standard deviation is a good indicator of dynamic and static status of the sensor. Empirically, when the standard deviation is larger than 6, the module is in motion, otherwise it is in static or on the way to stopping (Figure 12). Therefore the adaptive algorithm computes the latency compensated value to the Twice Half-Window Gaussian filter in the following way: when the standard deviation is no smaller than 6, the compensated value is double of the subtraction of the Half-Window Gaussian filter by the Twice Half-Window Gaussian filter result; otherwise the compensated value is equal to the subtraction of the Twice Half-Window Gaussian Filter by the observed data (Figure 11C).if standard deviation >= 6

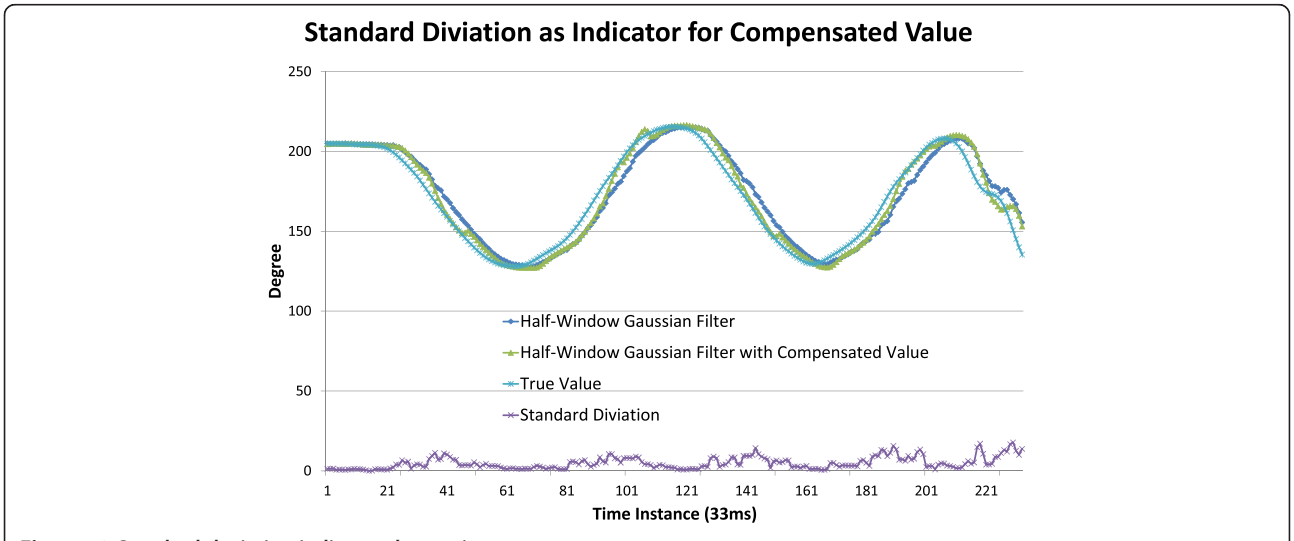


Figure 12 Standard deviation indicates the motion pattern.

Table 1 Comparison between the UM-AR-GPS-ROVER and ARMOR configuration

Device	UM-AR-GPS-ROVER	ARMOR	Comparison
Location Tracking	Trimble AgGPS 332 using OmniStar XP correction for Differential GPS method	Trimble AgGPS 332 using CMR correction broadcast by a Trimble AgGPS RTK Base 450/900	OmniStar XP provides 10~20 cm accuracy. RTK provides 2.5 cm horizontal accuracy, and 3.7 cm vertical accuracy
Orientation Tracking	PNI TCM 5	PNI TCM XB	Same accuracy, but ARMOR places TCM XB rigidly close to the camera
Video Camera	Fire-I Digital Firewire Camera	Microsoft LifeCam VX-5000	LifeCam VX-5000 is lightweight, small and uses less wire
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 and smaller
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 outputs charging both GPS receiver and HMD
Backpack Apparatus	Kensington Contour Laptop Backpack	Load Bearing Vest	Extensible and easy to access equipment

Compensation Value = 2 * (Half-Window Gaussian Filter Result – Twice Half-Window Gaussian Filter Result)

else

Compensation Value = Twice Half-Window Gaussian Filter Result – Observed Data

ARMOR hardware architecture

To test SMART framework's correctness and robustness, ARMOR, an evolved product from UM-AR-GPS-ROVER (Behzadan and Kamat 2009), is designed as a hardware platform implementation of SMART. As a prototype design, the UM-AR-GPS-ROVER meets the need of a proof-of-concept platform for visual simulation. However, there are two primary design defects that are inadequately addressed: accuracy and ergonomics. First, the insecure placement of tracking devices disqualifies the UM-AR-GPS-ROVER from the centimeter-accuracy-level goal. Secondly, co-locating all devices, power panels, and wires into an ordinary backpack makes it impossible to accommodate more equipment like the Real Time Kinematic (RTK) rover radio. The weight of the backpack is also high for even distribution around the body.

Table 2 Power voltage demands of different devices

Component	External power supply	Voltage	Power
Head-mounted Display	MP3750	5 V	<1.25 W
Electronic Compass	AAA batteries	3.6 V~5 V	0.1 W
Web Camera	Through USB	Unknown	Unknown
RTK Rover Receiver	MP3750	10 V~32 V	4.2 W
RTK Rover Radio	Through RTK Rover Receiver	10.5 V~20 V	3 W
RTK Base Receiver	Integrated Internal battery	7.4 V	<8 W
User Command Input	AA batteries	3 V	Unknown
Laptop	Integrated Internal battery	11.1 V	17.76 W

ARMOR represents a significant upgrade from the UM-AR-GPS-ROVER. The improvements can be divided into three categories: 1) highly accurate tracking devices with rigid placement and full calibration; 2) lightweight selection of input/output devices and external power source, and 3) load-bearing vest to accommodate devices and distribute weight evenly around the user's body. An overview comparison between UM-AR-GPS-ROVER and ARMOR is listed in Table 1.

Tracking devices

Position tracking device — RTK-GPS

The UM-AR-GPS-ROVER uses the AgGPS 332 Receiver with OmniStar XP mode to provide a user's position (i.e., the position of camera center) with 10~20 cm accuracy. This level of accuracy is sufficient for creating animations of a construction process simulation, where slight positional displacement may not necessarily compromise the validation purpose. However, for precision critical applications, 10~20 cm accuracy is not sufficient for visual accuracy.

The AgGPS 332 Receiver used in UM-AR-GPS-ROVER is thus upgraded and three objectives are pursued: 1) The upgraded GPS must be able to support centimeter-level accuracy; 2) The hardware upgrade should have minimum impact on the software communication module; and 3) The existing device should be fully utilized given the cost of high-accuracy GPS equipment. Ultimately, the AgGPS RTK Base 450/900 GPS Receiver is chosen for implementing the upgrade for three reasons. First, it leverages RTK technology to provide 2.5 cm horizontal accuracy and 3.7 cm vertical

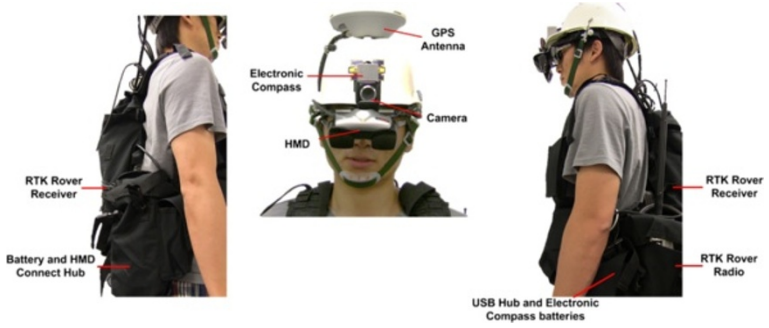


Figure 13 The profile of ARMOR from different perspectives.

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 1st order accuracy), and tracks the same satellites as an RTK rover. The carrier phase measurement is used to calculate the real-time differential correction that is sent as a Compact Measurement Record (CMR) through a radio link to the RTK rover within 100 km (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. The second reason for choosing this receiver is that, despite the upgrade, the RTK rover outputs the position data in NMEA format (acronym for National Marine Engineers Association) (NEMA 2010), which is also used in OmniStar XP. No change therefore applies to the communication module. The third and final reason is that the original AgGPS 332 Receiver is retained as an RTK rover with its differential GPS mode being changed from OmniStar XP to RTK. A SiteNet 900 radio works with the AgGPS 332 Receiver to receive the CMR from the base station.

Improvement has also been made to the receiver antenna placement. The UM-AR-GPS-ROVER mounted to the receiver antenna on a segment of pipe that was tied to the interior of the backpack. This method proved to be inefficient in preventing lateral movement. Therefore ARMOR anchors the GPS receiver with a bolt on the summit of the helmet, so that the phase center of the receiver will never shift relative to the camera center. The relative distance between the receiver phase center and the camera center is calibrated beforehand and added to the RTK rover measurement.

ARMOR can work in either indoor or outdoor mode. Indoor mode does not necessarily imply that the GPS signal is totally lost, rather the qualified GPS signal is absent. The GPS signal quality can be extracted from the $GGA section of the NMEA string. The fix quality ranges from 0–8, for example 2 means DGPS fix, 4 means Real Time Kinematic, and 5 means float RTK. The user can define the standard (i.e., which fix quality is deemed as qualified) in the hardware configuration file. When a qualified GPS signal is available, the geographical location is extracted from the $GPGGA

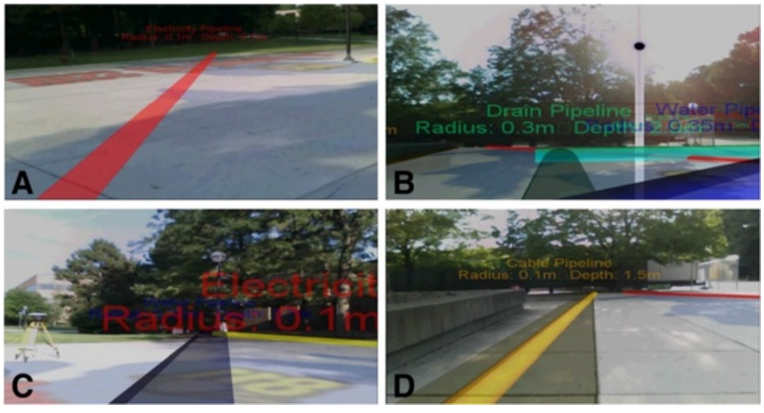


Figure 14 Labeling attribute information and color coding on the underground utilities. (**A**) Shows the proposed excavation area (the semi-transparent layer); (**B, C**) show the geometric and attribute information of the pipelines; (**D**) shows the buffering zone of the detected pipelines.

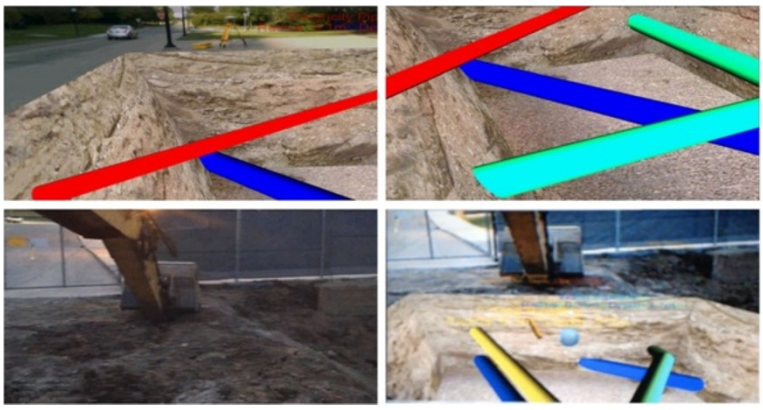


Figure 15 An x-ray view of the underground utilities.

section. Otherwise, a preset pseudo-location is used, and this pseudo-location can be controlled by user input.

Orientation tracking device—electronic compass

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

Theoretically the electronic compass is applied to measure the orientation of the user's head. However, since a user's eyes are obstructed by a Video See-Through Head Mounted Display (HMD) and that the eyes' function is replaced by a video camera, the electronic compass is applied to measure the attitude of the camera instead of the eyes.

The UM-AR-GPS-ROVER places the electronic compass on the zenith of the helmet, which makes it very hard to align the camera and the electronic compass. ARMOR chooses to anchor the electronic compass rigidly close to the camera anchored on the visor of the helmet, and parallel to the line of sight, making the physical discrepancy calibration much easier. The calibration approach is described in Section 4.2.1. For safety reasons, the electronic

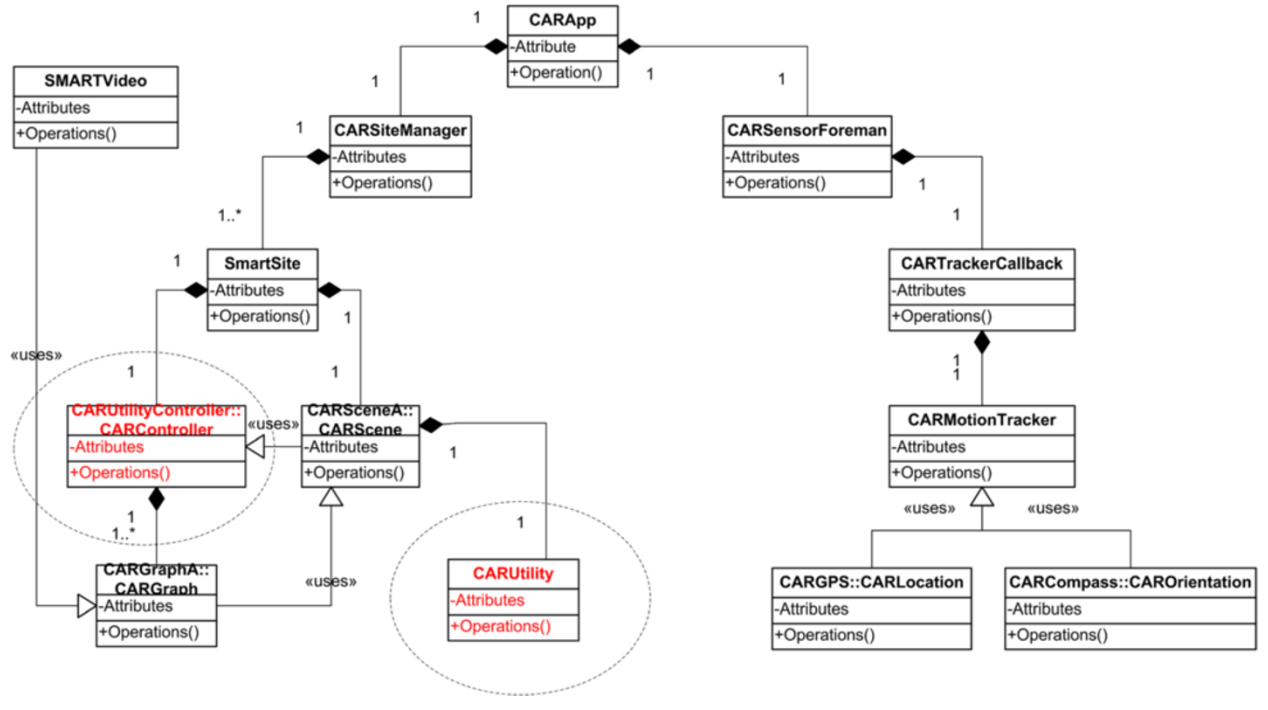


Figure 16 SMART adaption for visualization of underground utilities.

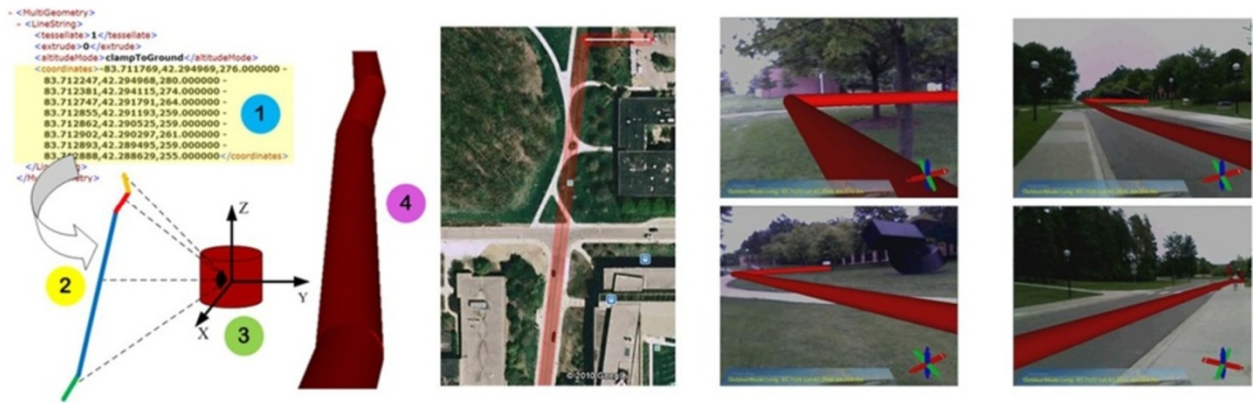


Figure 17 Conduit loading procedure, conduits overlaid on Google Earth and field experiment results.

compass is encapsulated in a custom-sized aluminum enclosure that is free of magnetic forces.

Magnetometer calibration also needs to be carried out for the purpose of compensating for distortions to the magnetic field caused by the host system and the local environment (PNI 2009). Given that ARMOR's entire periphery could have magnetic impact on the sensor—for example GPS receivers, HMDs, and web cameras—the TCM XB needs to be mounted within the host system that is moved as a single unit during the calibration.

Input/output devices and external power supply

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 a small volume and lightweight. Microsoft LifeCam VX5000 is chosen for the following reasons—the size is only 45 mm × 45.6 mm and it does not compromise on resolution (640 × 480) or connection speed (480 Mbps). More importantly, it takes samples at 30 Hz, which is the same speed as the electronic compass.

Augmented view output: head-mounted display (HMD)

The augmented view generated by the video compositor is ultimately presented by the Video See-Through HMD. The eMagin Z800 3DVisor is chosen as the HMD component of ARMOR because it has remarkable performance in primary factors, including wide view angle, large number of colors, lightweight frame, and comfort. Furthermore, the stereovision capability is another important rendering effect that helps the user to better appreciate the 3D augmented space.

External power supply

External power supplies with variant voltage output are indispensible for powering devices without integrated internal batteries. ARMOR upgrades 'POWERBASE' of UM-AR-GPS-ROVER to 'Tekkeon myPower ALL MP3750' which shows improvements over 'POWERBASE' in four ways: 1) both the volume (17 cm x 8 cm x 2 cm) and weight (0.44 kg) of MP3750 are only 1/5 of POWERBASE's volume and weight; 2) the main output voltage varies from 10 V to 19 V for powering the AgGPS 332 Receiver (12 V), and an extra USB output port can charge the HMD (5 V) simultaneously (Table 2); 3) it features automatic voltage

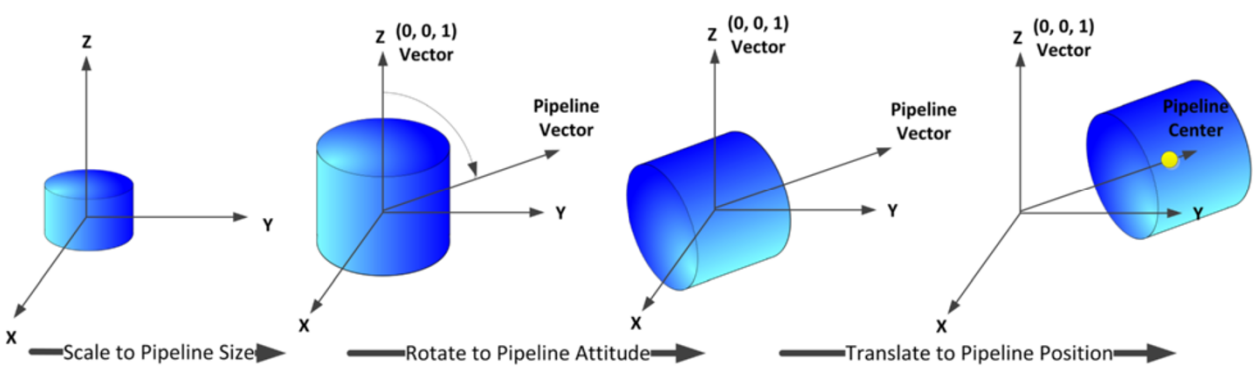


Figure 18 Process of creating pipeline segment from unit cylinder.

detection with an option for manual voltage selection; and 4) an extended battery pack can be added to double the battery capacity (Tekkeon 2009).

Load-bearing vest

The optimization of all devices in aspects of volume, weight, and rigidity allows the author to compact all components into one load-bearing vest. Figure 13 shows the configuration of the backpack and the allocation of hardware. There are three primary compartments: the back pouch accommodates the AgGPS 332 Receiver; the SiteNet 900 is stored in the right side pouch, and the left-side pouch holds the HMD connect interface box to a PC and the MP3750 battery; and the ASUS N10J netbook is securely tied to the inner part of the back. All of the other miscellaneous accessories—like USB to Serial Port hubs, or AAA batteries—are distributed in the auxiliary pouches.

The configuration of the vest has several overall advantages: 1) the design of the pouches allows for an even distribution of weight around the body; 2) the separation of devices allows the user to conveniently access and check the condition of certain hardware; and 3) different parts of the loading vest are loosely joined so that the vest can fit any body type, and be worn rapidly even when fully loaded. ARMOR has been tested by several users for outdoor operation that lasts continuously for over 30 minutes, without any interruption or significant discomfort.

Results and case study

The robustness of ARMOR and the SMART framework have been evaluated in an Augmented Reality application designed to visualize subsurface utilities during ongoing excavation operations for improved context awareness and accident avoidance.

Current practice of excavation damage prevention and its limitation

Every U.S. state's "One-Call Center" is a message handling system for underground utility owners. It collects excavation requests from contractors and distributes them to all of its members (utility owners). The standard procedure can be summarized as the following steps: 1) the contractor issues a text-based request to the "One-Call Center" describing planned excavation locations; 2) "One-Call Center" distributes the request to its members who may own buried assets in the proposed excavation area; 3) Each facility owner dispatches field locators to mark the approximate location of the underground facilities; 4) the excavator operators carry out the excavation activities based on the visual guidance marked on the ground.

Despite its effectiveness of reducing utility lines hit to a great extent, there are several weaknesses in the current practice of damage prevention to buried utilities that can be improved using Augmented Reality. First, the description about the proposed excavation is a text-based request ticket that can be vague and ambiguous, and it is up to the field locators to interpret the digging zone from the written request. In case of misinterpretation, the field locators either fail to cover the entire proposed excavation area, or take significant extra effort to mark the ground that is unnecessary. Second, even though the surface markings (paint, stakes, flags, etc.) serve as visual guidance for the excavation operator, they are vulnerable to heavy traffic, severe weather, and excavation activity which scrapes the surface and then removes the top soil, thus destroying the surface markings. This makes it challenging for an excavator operator to maintain spatial orientation, and must rely on memory and judgment to recollect marked utility locations as excavation proceeds (Talmaki and Kamat 2012).

Proposed practice for improvement of the "One-call" damage prevention practice with augmented reality

Augmented Reality can help to accurately visualize the proposed excavation area and digitize the located underground utilities, which bridges the communication among contractors, "One-Call Center", facility owners, field locators and excavation operators. First, the contractor can issue a "visual" ticket to the "One-Call Center" and facility owners by superimposing a semi-transparent layer above the precise proposed excavation range; second, dispatched locators come to the field, "see" the proposed digging area (Figure 14A) and can survey and mark precisely within that area. The markings are stored digitally and contain information about utility type and geometry attribute (Figure 14B, C) and buffer zone indicating the uncertainty of the survey result (Figure 14D); third, upon the completion of the digital marking, excavation operators can thus persistently "see" what utilities lie buried in a digging machine's vicinity, and consciously avoid accidental utility strikes (Figure 15).

Visualization of buried utilities based on SMART framework

The ultimate goal of SMART framework is to speed up the process of inventing engineering applications based on AR, so that domain experts can focus on building application logic and leave the AR logic to SMART framework. Figure 16 highlights the modifications compared with Figure 1 to adapt SMART for visualizing buried utilities. Essentially, two parts need to be overwritten: 1) The animation module (CARStatementProcessor and CARAnimation) is replaced by the utilities generation module (CARUtility), and the detail implementation of CARUtiliy is explained in the next section; 2) CARUtilityController replaces CARControllerA to respond to customized user inputs for creating, manipulating, visualizing, and archiving utilities.

Visualization logic of buried utilities

The workflow of CARUtility is briefly summarized here (Dong and Kamat 2012a). The digitized survey results are exported as KML (Keyhole Modeling Language) files. The following procedure interprets KML files and builds conduit (e.g. pipe) models in the augmented space (Figure 17):

(1) Extract the spatial and attribute information of pipelines from the KML file using libkml, a library for parsing, generating, and operating in KML (Google 2008). For example, the geographical location of pipelines is recorded under the Geometry element as "LineString" (Google 2012). A cursor is thus designed to iterate through the KML file, locate "LineString" elements, and extract the geographical locations.

(2) Convert consecutive vertices within one "LineString" from the geographical coordinate to the local coordinate in order to raise computational efficiency during the registration routine. The first vertex on the line string is chosen as the origin of the local coordinate system, and the local coordinates of the remaining vertices are determined by calculating the relative 3D vector between the rest of the vertices and the first one, using the Vincenty algorithm (Behzadan and Kamat 2007).

(3) In order to save storage memory, a unit cylinder is shared by all pipeline segments as primitive geometry upon which the transformation matrix is built.

(4) Scale, rotate, and translate the primitive cylinder to the correct size, attitude, and position (Figure 18). For simplicity, the normalized vector between two successive vertices is named as the pipeline vector. First, the primitive cylinder is scaled along the X- and Y-axis by the radius of the true pipeline, and then scaled along the Z-axis by the distance between two successive vertices. Secondly, the scaled cylinder is rotated along the axis—formed by the cross product between vector <0, 0, 1> and the pipeline vector—by the angle of the dot product between vector <0, 0, 1> and the pipeline vector. Finally, the center of the rotated pipeline is translated to the midpoint between two successive vertices. This step is applied to each pair of two successive vertices.

(5) Translucent rectangles (Figure 12D) are created and laid underneath the pipelines to represent the buffer zone for excavation.

(6) An inverted pyramid without bottom is created to represent the X-ray vision of the underground pipelines. Each cylinder segment and its associated rectangle (buffer zone) are tested against the surface of pyramid, and the parts falling outside are truncated (Figure 15).

Discussion and conclusion

This paper has presented the design and implementation of a robust mobile computing platform composed of the rigid hardware platform ARMOR and the application framework SMART with open access to the community. Researchers who are interested in improving the AR graphical algorithms or developing new AR engineering applications can take advantage of the existing AR framework implemented in this research, and can prototype their ideas in a much shorter lifecycle.

Targeting outdoor AR applications at centimeter-level accuracy, algorithms for both static and dynamic registration have been introduced. Several dynamic misregistration correction approaches are also described and evaluated. It is found that dynamic misregistration continues to be an open research problem and continues to be under investigation by the authors. Several efforts are being made in ongoing work, which include: 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 provide additional heuristics about the angular speed).

A video demo about SMART and ARMOR, and its application in excavation damage prevention can be found at <http://pathfinder.engin.umich.edu/videos.htm>. A video demo about its occlusion capability can also be found at the same site. SMART is an open source project that can be downloaded at <http://pathfinder.engin.umich.edu/software.htm>.

Competing interests
Both authors declare that they no competing interests.

Authors' contributions
SD coded the software package SMART, designed the hardware implementation ARMOR, and carried out the field experiment. VK co-authored the software package SMART, and co-designed the field experiment. All authors read and approved the final manuscript.

Author details
[1]Department of Civil and Environmental Engineering, University of Michigan, 2350 Hayward Street, Suite 1380, Ann Arbor, MI 48105, USA. [2]Department of Civil and Environmental Engineering, University of Michigan, 2350 Hayward Street, Suite 2340, Ann Arbor, MI 48105, USA.

Received: 11 December 2012 Accepted: 29 April 2013
Published: 12 June 2013

References

Azuma, R. (1997). A survey of augmented reality. *Teleoperators and Virtual Environments, 6*(4), 355–385.

Azuma, R., Hoff, B., Neely, H., & Sarfaty, R. (1999). *A motion-stabilized outdoor augmented reality*. Houston, TX: Proceedings of the Virtual Reality. IEEE.

Behzadan, A. H., & Kamat, V. R. (2007). Georeferenced registration of construction graphics in mobile outdoor augmented reality. *Journal of Computing in Civil Engineering, 21*(4), 247–258.

Behzadan, A. H., & Kamat, V. R. (2009). Automated generation of operations level construction animations in outdoor augmented reality. *Journal of Computing in Civil Engineering, 23*(6), 405–417.

Behzadan, A. H., Timm, B. W., & Kamat, V. R. (2008). General-purpose modular hardware and software framework for mobile outdoor augmented reality applications in engineering. *Advances Engineering Informatics, 22*, 90–105.

Dong, S., & Kamat, R. V. (2012a). *Scalable and Extensible Augmented Reality with Applications in Civil Infrastructure Systems*. Ann Arbor: University of Michigan.

Dong, S., & Kamat, R. V. (2012b). Real-Time Occlusion Handling for Dynamic Augmented Reality Using Geometric Sensing and Graphical Shading. *Journal of Computing in Civil Engineering*. in press.

Feiner, S., Macintyre, B., & Hollerer, TA. (1997). A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment. In *Proceedings of 1997 International Symposium on Wearable Computers* (pp. 74–81). Piscataway, NJ: IEEE.

Golparvar-Fard, M., Pena-Mora, F., Arboleda, C. A., & Lee, S. (2009). Visualization of construction progress monitoring with 4D simulation model overlaid on time-lapsed photographs. *Journal of Computing in Civil Engineering, 23*(6), 391–404.

Google. (2008). *Introducing libkml: a library for reading, writing, and manipulating KML*. http://google-opensource.blogspot.com/2008/03/introducing-libkml-library-for-reading.html.

Google. (2012). *KML Documentation Introduction*. https://developers.google.com/kml/documentation/.

Jacobs, M. C., Livingston, M. A., & State, A. (1997). Managing Latency in Complex Augmented Reality Systems. In *Proceedings of the 1997 Symposium on Interactives 3D Graphics* (pp. 49–54). New York, NY: ACM.

Kamat, V. R., & El-Tawil, S. (2007). Evaluation of augmented reality for rapid assessment of earthquake-induced building damage. *Journal of Computing in Civil Engineering, 21*(5), 303–310.

Liang, O., Shaw, C., & Green, M. (1991). On temporal-spatial realism in the virtual reality environment. In *Proceeding of 1991 Symposium on User Interface Software and Technology*. New York, NY: ACM.

Martz, P. (2007). *OpenSceneGraph Quick Start Guide*.

NEMA. (2010). *NEMA data*. http://www.gpsinformation.org/dale/nmea.htm.

Piekarski, W., Smith, R., & Thomas, B. H. (2004). Designing Backpacks for High Fidelity Mobile Outdoor Augmented Reality. In *Proceedings of the 2004 IEEE and ACM International Symposium on Mixed and Augmented Reality* (pp. pp. 280–281). Piscataway, NJ: IEEE.

PNI. (2009). *User Manual Field Force TCM XB*.

Roberts, G., Evans, A., Dodson, A. H., Denby, B., Cooper, S., & Hollands, R. (2002). The Use of Augmented Reality, GPS and INS for Subsurface Data Visualization. In *Proceedings of the 2002 FIG XIII International Congress* (pp. 1–12). Copenhagon, Denmark: International Federation of Surveyors, FIG.

Shreiner, D., Woo, M., Neider, J., & Davis, T. (2006). *OpenGL Programming Guide* (6th ed.). Boston, MA: Addison-Wesley.

Stafford, A., Piekarski, W., & Thomas, H. B. (2006). Implementation of God-like Interaction Techniques for Supporting Collaboration Between Outdoor AR and Indoor Tabletop Users. In *Proceedings of the 2006 IEEE and ACM International Symposium on Mixed and Augmented Reality* (pp. 165–172). Piscataway, NJ: IEEE.

Talmaki, S., & Kamat, V. R. (2012). Real-Time Hybrid Virtuality for Prevention of Excavation Related Utility Strikes. *Journal of Computing in Civil Engineering*.

Tekkeon. (2009). *MP3450i/MP3450/MP3750 datasheets*.

Thomas, B., Piekarski, W., & Gunther, B. (1999). Using Augmented Reality to Visualise Architecture Designs in an Outdoor Environment. In *Proceedings of the Design Computing on the Net*.

Trimble. (2007). *AgGPS RTK Base 900 and 450 receivers.: Trimble*.

Vincenty, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application o fnested equations. *Survey Reviews, XXIII*, 88–93.

Wang, X., & Dunston, P. (2007). Design, strategies, and issues towards an augmented reality-based construction training platform. *Journal of Information Technology in Construction (ITcon), 12*, 16.

Wang, X., Kim, M. J., Love, P. E. D., & Kang, S.-C. (2013). Augmented Reality in built environment: Classification and implications for future research. *Automation in Construction, 32*, 13.

Webster, A., Feiner, S., Macintyre, B., & Massie, W. (1996). Augmented Reality in Architectural Construction, Inspection, and Renovation. In *Proceedings of 1996 ASCE Congress on Computing in Civil Engineering*. New York, NY: ASCE.

Yeh, K.-C., Tsai, M.-H., & Kang, S.-C. (2012). On-site building information retrieval by using projection-based augmented reality. *Journal of Computing in Civil Engineering, 26*(3), 14.

doi:10.1186/2213-7459-1-1
Cite this article as: Dong and Kamat: SMART: scalable and modular augmented reality template for rapid development of engineering visualization applications. *Visualization in Engineering* 2013 **1**:1.