A Motion Sensor-Based User Interface for Construction Drawings Navigation

Mehdi NOURBAKHSH\textsuperscript{1}, Robert STEVENSON \textsuperscript{2}, and Javier IRIZARRY \textsuperscript{2}

\textsuperscript{1} PhD Student, School of Building Construction, College of Architecture, Master’s Student in Computer Science, Computer Graphics, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332; email: mehdi.nourbakhsh@gatech.edu

\textsuperscript{2} Research Scientist, Georgia Tech Research Institute, 250 14th ST, Atlanta, GA., email: robert.stevenson@gtri.gatech.edu

\textsuperscript{3} Assistant Professor, School of Building Construction, College of Architecture, Georgia Institute of Technology, Atlanta, GA 30332-0680, USA; PH (404) 385-7609; FAX (404) 894-1641; email: javier.irizarry@coa.gatech.edu

ABSTRACT

Nowadays, most viewing of electronic drawings on construction sites takes place on mobile devices with limited screen size. Most of the current software interfaces still use a static display method, scaling drawings to fit the screen. Even though there are limitations due to small screen size on mobile devices, they have the benefit of being equipped with a variety of sensors such as magnetometers, accelerometers and gyroscopes. Some studies have looked at the use of these sensors to further enhance user interaction in areas such as gaming and texting. However, no study has considered the design of a user interface for construction drawing navigation using these sensors. To this end, the objective of this study is to develop a touch-free interface to navigate construction drawings using motion sensing. The interface was developed on Apple Inc.’s iOS platform and tested on an iPad tablet device. Motion sensor data, from the accelerometer and gyroscope, were used to pan and zoom the document with spatial movements. This study provides an innovative means to navigate large, layered drawings on small tablet devices. The algorithm used to develop the interface can be embedded in different applications for better navigation of layered construction drawings.

INTRODUCTION

Most viewing of construction drawing now takes place on an electronic display rather than in print form. With current advances in computer graphics, the resolution of such drawings has increased over the years. However, the typical viewing size of drawings has decreased as significant image viewing takes place on mobile devices with limited screen size (Dachselt and Buchholz 2009). Unfortunately, most of the software interfaces available on the market (e.g. GoBIM, FingerCAD, Architect’s Formulator, etc.) still use a static display method, scaling drawings to fit the screen. This ignores the flexibility of displays especially when drawing resolutions, as presented on an A0 sheet, may be about 10 times greater than the display resolution (comparing with a small device such as an iPhone).

This difference between the size of drawings and displays becomes even more apparent while considering different layers of drawings in one project (e.g. architectural, structural, mechanical, electrical, etc.). Imagine that a construction manager wants to see a detail of a particular spot on the architectural layer and
switches to other layers to find the perspective spot. The manager needs to zoom in, 
pan to find the spot, check the details, and then zoom out, load another layer and 
repeat the process again in the other layer. This is very difficult to deal with in small 
screen devices.

Even though there are limitations due to small screen size on mobile devices, 
they have the benefit of being equipped with a variety of sensors such as 
magnetometer, accelerometer, and gyroscope. A number of studies and systems have 
been developed that use these sensors to further enhance the interface in different 
areas, such as gaming (Chehimi and Coulton 2008; Gilbertson et al. 2008; Wei et al. 
2008), texting (Goel et al. 2012), and navigation (Sánchez et al. 2012). The extent of 
the system that is presented in this paper builds on many of these techniques and it is 
expected that users will find it to be intuitive and natural to use.

**Background and problem**

Viewing large documents, such as a drawing, on a limited viewing area like 
on that of a tablet computer is largely a solved problem. The user is given a full view 
of the document and utilizes pinch-to-zoom and panning gestures giving the user the 
necessary techniques for viewing overviews and details within a large document 
(Hornb et al. 2002). These interaction techniques provide a good interface. However, 
this interaction model is being approached from a more realistic point of view with 
the inclusion of augmented reality. It is envisioned that the user would be provided 
the perception that the large document that they are viewing is actually sitting on the 
table in front of them. This sort of interaction model best suits a real world analogy 
where the user is accustomed to dealing with large documents on a table. When a user 
is dealing with documents such as a drawing, it’s common that they are viewing 
multiple layers of the same structure. When the user is viewing these real paper 
documents and wants to compare and contrast the different layers, the action to do so 
can be unwieldy and only gets more complex when more than two layers are being 
evaluated.

**Related works**

There are two navigation methods commonly used together, panning and 
zooming. The pan allows users to move horizontally and vertically on the screen 
while the zoom changes the scale at which the dataset is viewed, allowing users to 
view regions of interests at a greater resolution (Nekrasovski et al. 2006). These 
navigation techniques have been studied over the years and only current work within 
this domain is presented in this section.

The interactive navigation of interfaces using gesture has been investigated in 
different studies. Harrison and Dey (2008) used a lean to zoom technique in laptops. 
In their system, the zoom was directly proportional to the extent of the lean. When 
users leaned towards the laptop, the interface zoomed in. They used a simple vision 
tracking system with web cameras on laptops to calculate the distance between users 
and their laptops. Boulos et al. (2011) developed a natural user interface using 
Microsoft Kinect to navigate Google Earth. Zooming required a two-hand gesture, 
enabling users to move their hands either closer or farther. The distance between two 
hands was computed to detect whether they are moving together or apart.
Mistry et al. (2009) designed a wearable gestural interface to virtually augment surfaces or physical objects the user is interacting with into the tangible world. They used a camera and a projector mounted on a hat to project information onto surfaces, walls, and physical objects. Users can interact with the system using a natural hand and arm gesture to zoom in or out.

There are several approaches to control the zoom on interfaces in mobile devices. One of the approaches Apple Inc. has been using in Mobile Safari is to render web pages in a much bigger resolution of the device, then use a physical window of the device as a sliding window that can be moved across the high resolution screen. Thus, users are able to zoom into spots on a web page as well as zoom out.

Dachselt and Buchholz (2009) conducted a study to address the seamless combination of large displays with sensor-enabled phones. An accelerometer-enabled mobile phone was used to detect intuitive gesture interactions of users. The continuous and stepwise tilt gestures were used to map a variety of interactions typical for media-centered applications. The study also proposed a throw gesture to transfer data between the mobile device and a distant display.

Tsang et al. (2003) demonstrated an approach to virtual reality, designing an interactive 3D kiosk to view 3D designs of objects. The display acted as a window to the real world using a motion capture sensor mounted on a table, and also a 6-axis arm to measure the position and orientation.

Joshi et al (2012) illustrated a sensor fusion methodology to combine face tracking with sensor data. A front-facing camera to detect 3D position of viewer and a low-latency gyroscope to sense changes in direction of the device were used. The prototype showed natural pan-tilt-zoom interfaces for many forms of imagery such as multi-viewpoint image sets depicting parallax in a scene, multiple images stitched to create a single viewpoint 360-degree panorama, and street side interfaces integrating both multi-perspective panoramas and single viewpoint 360° panoramas. A touch-free panning and zooming for maps were achieved by tilting, moving the device away or close to the viewer.

The objective of this study is to design an interface to enable users to: 1) Pan on drawings while they are moving the device towards north, south, west, and east, and 2) Switch between the layers of drawings while they are moving the device up or down.

It is also the objective of the study to identify the benefits and challenges of such an interaction method and discuss its implications for information access in the Architecture, Engineering, and Construction (AEC) industry.

IMPLEMENTATION

Development environment

The Large Document Browser is implemented in the iOS environment in Objective C and built using XCode version 4.5 on Mac OS X using the iOS 6 API. Since the application depends on the sensor readings from the device, the development and debugging was done using an iPad as opposed to the built in
simulator in XCode. The iPad with retina display (3rd generation) was the target device configuration.

**Application overview**

When the application loads the user is presented with a large document viewing area, a simple bottom menu bar, and a lock icon in the upper left (Figure 1). The menu bar contains a simple toggle switch to select the particular mode the user wishes to use the application when face up. Table mode is for sliding it around a table, and hand mode is to hold the device. The lock icon is to give the user a visual indicator when the document is locked and ignoring all sensor movements in the Z direction.

The application is meant to first be opened while holding it vertically with two hands. Holding the device in this manner, the user is given a scaled view of the document with familiar pinch to zoom and drag to pan to explore the document.

![Surface panning is active](image)

**Figure 1. Application interface**

The application transforms when the user rotates the device so it is face up either on a table, or still held in hands. The use of the gyroscope in this scenario gives us the ability to detect the orientation. When switching orientations a smooth animation is employed to zoom the interface so the user doesn’t lose context. In addition, when the user is panning the document while holding it vertically, a “bounce” animation is implemented to smoothly pan the document back within the bounds of the view so the document doesn’t get lost outside of the view from too much panning.

When the application is in table mode, and held vertically in the hands the user can quickly switch between the document layers by using a two fingers swipe gesture. This allows a user to view a particular area of the drawing and quickly compare and contrast the various details in the other layers.
**Table mode**

In this mode, the device is meant to be resting on a table (Figure 2). When the user puts the device down, the orientation of the device is remembered and any rotation of the device will cause the document to remain at a stationary view to the user. The user pans the document in this mode by moving the device along either the X- or Y-axis. When doing so the document will then pan in the opposite direction by using sensor readings from the accelerometer only.

![Figure 2. Perspective view of the Table mode. (The drawing outside of the device is semi-transparent to show how it will look to the user as they pan)](image)

**Hand mode**

In this mode, users hold the iPad in their hand to interact with the interface. To pan the user first tilts the device in the direction they intend to pan, and then move the iPad in X and Y direction, the document pans towards these directions respectively.

A “clutch” mechanism is also implemented in this mode, where users pick up the iPad and move it to a new position, while keeping the display fixed. This mechanism is implemented using gyroscope data. This mode also supports the switching between document layers by reading sensor data (Figure 3). Users can switch between three different document layers by moving the device about 5 inches in either up or down direction. Since the sensor data is noisy, the surface (X, Y) panning motion is stopped while moving towards Z and the Z movement is stopped while panning on the surface. This means that if the user is panning on the surface, the Z movement does not work. Similarly, if the user is moving towards the up or down direction, the panning does not work. A ‘Lock’ mechanism is defined in which users are required to tap the iPad to lock the panning on the surface. Then, the movement towards Z will be activated to switch between layers. When a new layer is displayed, the Z movement will be locked automatically and the surface panning can be resumed as before.
Architecture overview

The application was architected as a single view iOS application. Upon launching, the first class loaded is the AppDelegate. The AppDelegate handles creating a window object and passes along the main UIViewController class, which is named ViewController. The ViewController is where much of the behind the scenes code is handled.

View controller

The ViewController class is the main gatekeeper for the application. The viewDidLoad function initializes a number of classes that handle the routing of the movement sensor data readings to delegate functions, in addition a special GLKViewController object is created to handle the redrawing of the DocumentView at a reasonable frame rate. Also, this function will load the document image files that are read from a Property List file. These image files are used to then to do a one-time splitting up of the large image files into smaller “tiles” that can then be used as texture images for the DocumentView. The images are split up only the first time the application loads using a utility class that is created called ImageTiler.

Sensor notifications

One of the biggest challenges in this part was how to handle noise from sensors and have a smooth pan effect on the screen. Different sets of filters are used to overcome this challenge. Low pass filter was used to filter out the portion of the accelerometer data caused by gravity from the portion of the data caused by motion. High pass filter was applied to isolate sudden changes in movement from the constant effect of gravity.

Kalman filter was implemented, using equation 1, to estimate the future states of the movement. Indeed, the Kalman filter is useful when there is noise and other inaccuracies because it estimates the unknown variables in the future state based on the current status of the data (Welch and Bishop 1995).

\[ X_k = X_{k-1} + K_k (Z_k - X_{k-1}) \]  
(1) (adopted from Welch and Bishop 1995)
$X_k$: Current Estimation; $K_k$: Kalman Gain; $Z_k$: Actual Measurement $X_{k-1}$: Previous Estimation

One of the other important challenges in this study was determining the direction in which the iPad is moving. Since the sensor data are noisy, there are always positive and negative acceleration data even in one second. For the hand mode, gyroscope data are used to determine the direction. However, only the acceleration data was used to determine the movement direction in the Table mode. Figure 4 shows the simplified algorithm for movement towards X in the Table mode.

![Figure 4. Simplified algorithm for movement towards X on the table mode](image)

Figure 5 shows the accelerometer data when the iPad is moving towards the X direction on the Table mode.

![Figure 5. Acceleration data (iPad is moving on the X direction)](image)

The panning movement will not be smooth if only the received data from the accelerometer is considered. For instance, the document should go forward, backward, forward, and backward if our algorithm only used the data presented in Figure 5. An algorithm was designed and implemented to predict the movement direction based on the real time sensor data. The algorithm listens to the first 30 millisecond of the movement and predicts the future state of the iPad. All of the displacements will be transferred towards the positive or negative direction based on the results of this function.
Document view

The DocumentView is where all of the displaying of the document is done. Overall this view is very similar to the built in iOS UIScrollView. Some interactions behind this view around the UIScrollView are modeled. The UIScrollView is a class that is very useful in dealing with displaying a large document; however it was not suited for the performance of the application. In order to give the user a more responsive and smooth scrolling application driven by the accelerometer and gyroscope, the DocumentView has to be fast. The processing of the sensor data is bound very much on time, and having the most readings from these sensors is imperative to the performance of the application.

OpenGL

To give the DocumentView the best performance, OpenGL for the drawing was chosen. This meant that all of the drawing, scaling, rotating, and transposing of the document has to be done at a low level. In order to handle the large image size the drawing was cut up into smaller tiles. The maximum image texture size on the iPad is 2048 x 2048; however using the cut technique, the application was able to support images much larger than that.

When the class loads, the ViewController will set up the document by passing the image size and the number of X and Y tiles. Next it will call a couple of functions to create the tiled geometry and the loading of the texture tiles into OpenGL. When a frame is drawn from the GLKViewController an orthographic projection was used for the view that had a size of the bounds of the frame. The view projection is then multiplied by values that make up the scaling, panning, and rotation for each frame. These values are manipulated either directly from the ViewController via the sensor data, or changed by processing the zoom and pan gesture data from the ViewController. In addition, this class also handles animating the movement of the document when switching between held and face up. The scale, pan, and rotation attributes are linearly animated for half a second.

DISCUSSION

One of the challenges in this study was defining various thresholds to control the interface actions. For instance, if the iPad is stationary on a table, the accelerometer data will fluctuate between ±0.02*Gravity (G). If there is no threshold for the stationary mode of the device, the document pans in different directions. Similarly, if users hold the iPad in their hand, there would be an acceleration associated with the user’s hand motion. Even if the user tries to hold it in a fixed position, the orientation of the device and instance shaking of the hand will create a new threshold for this mode. A threshold for moving towards X, Y, and Z in the Table mode as well as the Hand mode was considered. The program will face a problem if a higher value is being used for a threshold than what it is supposed to be. For instance, in the Table mode, if the stationary threshold is increased to ±0.1G, the algorithm will inform the device that does not move if the acceleration is in this range. Since the stationary mode’s threshold is high, the device may not pan or it may pan slowly on the surface because the acceleration of the device while moving with
acceleration less than ±0.1G. To tackle this problem, a user study is required to evaluate these thresholds.

The unit of displacement resulted from our algorithm is mm while the unit of displacement is pixel in iPad. If the displacement unit to pixel was solely converted, the document would pan very slowly. A speed parameter is required to avoid this slow motion. In this study, different speed parameters were tested and tried to find the best speed. However, a user experiment study is required to validate these parameters.

One way to explore a better user feel for the speed of the panning is to implement a calibration setting where the user would follow a set of directions to move the device in certain scenarios. The application would then analyze the sensor data to come to a custom setting that would adjust the speed and movement thresholds accordingly.

FUTURE ENHANCEMENTS

The movement algorithms that were developed could be optimized since they involved a significant number of trial and error operations. These algorithms would certainly benefit from further research that could further improve their performance. These ongoing adjustments would benefit from user input that would result from user studies. These user studies would adhere to Human Computer Interaction theories and principles to develop objective data to drive attributes in the algorithms.

In addition, the application as it stands now loads a static set of documents that are read from a configuration file. It would be useful if the user could load any large document from a variety of sources and formats. For example, the application currently only supports loading of image files. Portable Document Format (PDF) is another format that could be loaded into the application. Another useful data format that could be loaded by the application would be mapping tiles from services such as Apple Maps or Google Maps.

CONCLUSION

This study developed an interface to navigate large documents on the iPad device with spatial movements. Using two different modes, Table mode and Hand mode, users are able to interact with the interface. Some features developed to accommodate the table mode are panning towards X and Y while moving the device on the table, switching between layers with swipe, fixing the document rotation relative to the initial orientation of the device on the table, and zooming in/out with holding the device down/up. Likewise, the features of the Hand mode are panning towards X and Y while moving the device with hand, fixing the movement with clutch function, and switching between layers with Z movement. All of these features work together to make up a successful representation of our original vision of developing a document browser that utilizes an intuitive interaction model.

REFERENCES


