Using acceleration data from smartphones to interact with 3D medical data

Mateus de Souza*, Diego Dias Bispo Carvalho*, Peter Barth†, Jeferson Vieira Ramos*,
Eros Comunello*, Aldo von Wangenheim*
*LAPIX - Laboratory for Image Processing and Computer Graphics
Federal University of Santa Catarina
Florianópolis, Brazil.
e-mails:{mateussouza,diegodbc,jef,eros,awangenh}@inf.ufsc.br
†Hochschule RheinMain
Wiesbaden, Germany.
e-mail:peter.barth@hs-rm.de

Abstract*—Accelerometers integrated in modern smartphones pave the way to intuitively use gestures for collaboratively controlling interactive applications. Using and holding smartphones has become natural and ensures user acceptance as well as intuitive handling. We focus on using accelerators in several smartphones at the same time to interactively control a medical imaging solution. To this end, we introduce a framework to collect, modify, and distribute acceleration sensor data from multiple smartphones and integrate it with a medical imaging system which results in an environment suitable for e.g. doctors reviewing and explaining diagnostic findings. We performed some experiments to evaluate the usability of this approach and present an ongoing research in adapting the smartphone interface to physical simulation applications. ***Keywords*: mobile, acceleration sensor, smartphone, medical images**

I. Introduction

The adoption of smartphones and their ever expanding feature list [1] makes new forms of user interaction both possible and more readily accepted by users. In particular, the availability of acceleration sensors in recent smartphones – such as in Symbian-based phones (e.g. N95), Android phones, and iPhones – brings motion based user interaction to the masses. Increasingly, the user interaction goes beyond simply switching the screen orientation when a phone is turned to actually using three-dimensional acceleration data to control the position of objects. However, most scenarios are limited to controlling applications on the device itself. And even if acceleration data is used to control remote applications, it is typically limited to just one user and one device.

In this paper, we provide and evaluate a solution that allows to remotely control interactive applications with acceleration data potentially sent by several devices from several users simultaneously. To this end, we provide a framework to collect, manipulate, and distribute acceleration data from smartphones and integrate it with an interactive visualization application for three-dimensional medical image data. The solution provides accurate and intuitive browsing and positioning of medical image data for e.g. doctors that need to work with three dimensional data obtained from an MRI scan.

In section II we give an overview of related work using acceleration sensor data to remotely control interactive applications. In section III we introduce our solution to interact with medical images and detail the solution architecture. The main components, WS3D with a multimodal open input framework and PySensor to collect, manipulate, and distribute acceleration sensor data are detailed in section IV and section V respectively. We conduct some initial experiments and provide a first evaluation of our prototype in section VI and conclude in section VII.

II. Related Work

In previous decades, experiments to provide a new input device to three dimensional applications resulted in clumsy and sometimes unnatural interfaces [2]. First approaches with smartphones used the embedded cameras to track the device movement and transmitted those as commands to the target application [3]. With the recent advances in the accelerometer technology, it is possible to make this kind of input more user-friendly and less costly.

Sensors are becoming common devices in modern smartphones and used for controlling applications [1]. Most of the research seems to focus on bringing the applications to the device, e.g. [4] and not bringing the data to the application. But increasingly applications appear that remotely collect acceleration information and use them to better understand human behavior [5], [6]. Now it is possible to not use dedicated but common devices such as smartphones for controlling applications. Maybe the most prominent example of remote controlling with acceleration data from a smartphone is ShakerRacer [7].

The concept to remotely control applications with acceleration sensor data is well known, commercialized with the Wii-Controllers, and used in gesture recognition [8]. Luigi et al. [9] performed experiments in controlling the visualization of 3D medical data with the WII-Controllers.

III. Overview

Our main objective is to couple the complexity of the visualization of a medical dataset in 3D with a multimodal input

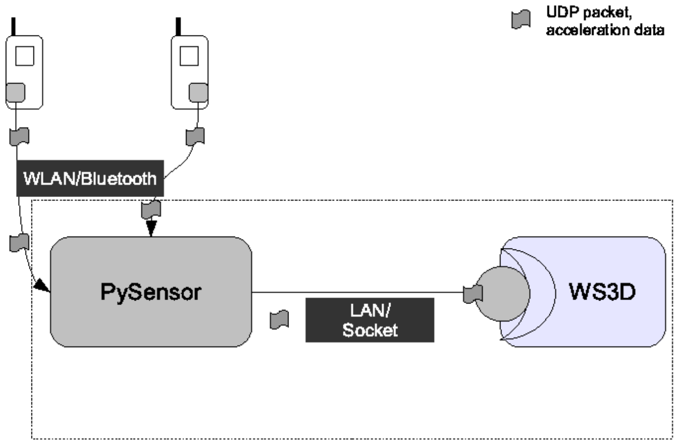


Figure 1: Architecture of the framework

environment. Despite the recent advances in input devices for three-dimensional scenarios, most users still rely on mouse and keyboards to interact with computers. Many of the new devices target audiences like videogame consumers. Using game controlers may demotivate potential users in a platform designed to serve health professionals. According to Bowman et al. [10] "the problem currently facing 3D UI researchers is that despite this broad coverage and extensive knowledge of 3D interaction techniques, the usability of 3D UIs in real-world applications is still surprisingly low in many cases". We believe that some part of the rejection by users is that they are just hesitant to touch and learn when dealing with a different device. Therefore, we focused on a device that physicians are more familiarized with: smartphones.

As users are already acquainted on how a smartphone works, it is easier for them to use it as an input device. Moreover, smartphones have better capabilities than other input devices, such as mice and other accelerometer-based hardware. They are capable of giving the user feedback (vibrating, playing sound / videos and so on), which mice cannot. In addition, users probably already use their phones in their everyday work, so they're always present and ready for use. Finally, smartphones can give multiple ways of input, that could be explored in the future, such as image tracking from the camera or even voice commands.

Nowadays smartphones have different functionalities and can track different input signals: from the simple press on a click button to the change of the device rotation angle. Since this kind of input can be tracked and tailored to specific application needs we chose to use then to control the visualization of data in a 3D environment. The smartphone connectivity possibilities allow to build a scenario where many users can interact, without worrying about physically connecting a cable to the PC running the application. The connection is performed through the smartphones wireless capabilities, there is no problem related to the distance between the user and the computer that runs the application, something that can happen with device that communicates by a Bluetooth connection, such as the WII-Controller.

We focus two different scenarios:

- A physician is analysing a patient's data with more doctors in the same room. He points to the patient data's characteristics and asks for the others physicians' opinions. Provided the possession of a smartphone, any other physician in the room can join the interaction.
- All the physicians are in different places. They comunicate with each other through a web conference. They are sharing the view of a remote application. At any time, each one of the physicians could manipulate the model that was supposed to be displayed to everyone.

The first scenario is easy to figure out. In fact, a different input may not seem a real improvement in the users interaction. But if we realize that probably a single mouse would be available at the moment, instead of a smartphone for every user, we can see that a lot of progress could be achieved specially where two or more users are accessing the application at the same time.

There are many tools that allow interactions through web conferences. However our advantage is that only the machine that runs the application requires distinguished hardware configurations. Since our signals are sent directly to the internet, the smartphone can be anywhere a wireless connection is available. Users send signals via the internet and can visualize the result on screens. In this scenario users should be visualizing their results on a third party software (that could be running on a PC or the smartphone), but they could input data freely.

With these two scenarios in mind we performed a set of tests. The first goal was to find a more natural way to use the cellphone capabilities in dealing with 3D data. So we used the accelerometer's commands to perform geometrical transformations in 3D environment's objects.

IV. Visualizing and Interacting with Medical Images

The Workstation3D (WS3D) [11] is an application to visualize and manipulate medical images in a 3D environment. The application builds volumes from images gathered from CT or MRI examinations. To this end, the WS3D extracts 3D polygonal meshes from the volumes by radiological density or curve selection. It also allows the application to apply VOIs (Volumes of Interest) on discarding polygons from the meshes and to measure tubular meshes cross sections and length. Typical user interaction with the WS3D is via a desktop user interface with typical widgets. The actual interaction to position the 3D object is primarily done with mouse which can be operated in different modes to allow to control several aspects of the visualization. As we do not aim to use the smartphone as a full replacement for keyboard and mouse, we focus on replacing the 3D interaction and do not cover replacing the existing typical desktop widgets nor using these with the smartphone. Therefore, we take advantage of the acelerometer to control geometric transformations on the camera. The visualization tool relies on trackball and free-look visualization allowing camera repositioning and zoom operations. To allow continuously working with the smartphone without being interrupted by necessary small mode settings with the keyboard, we also use the keyboard on the smartphones to do most basic commands with single key strokes.

Using acceleration data as input data to control the 3D position of a medical model requires to bring the acceleration data to the application. Thus, the receiving side – the medical image application the WS3D – has to be open to accept acceleration data or in general different input devices. BILL [12] is part of the WS3D and serves as a multimodal input framework to allow dynamic binding between different input devices and software events.

For every input device BILL needs a respective representation, which we call Input Device, that translates regular events to BILL's events. Once the event is inside the framework, already translated, it is distributed to every listener, here called Action, that was previously bound to that specific event. Then, the Action performs whatever the application is supposed to when recieving that event.

In this smartphone example, we defined a Cellphone Device, which translates the events from the smartphone (buttons pressed, accelerometer data) into BILL's events. Then, we constructed a Cellphone Action, crafted to apply to the applications the commands from the smartphone. This structure, shown in Figure 2, has a very modular nature, as every application that uses the cellphone as input only needs to design its own Cellphone Action. In fact, it only needs to do that if the commands applied to that application differs from the previously defined Cellphone Actions. Also, the Cellphone Device and all the events in BILL, if well constructed, can be reutilized as a basic cellphone input structure.

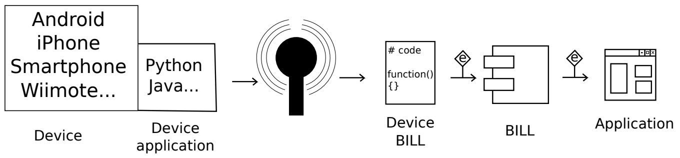


Figure 2: Representation of the communication between devices, BILL and the application.

BILL's purpose is to improve code reusability and development time speed while decoupling the event management from a prototype application. Every action in the code repository could be employed in any application without expressing hard coded connection between any input device; e.g., a joystick, camera, or in this case cellphone. The application prototype used in the experiments in this paper were conceived to be manipulated with mouse and keyboard passed to BILL through the wxWidgets library. Therefore, we linked different events with some already implemented actions. As it would be non intuitive to manipulate an application with many toolbars and buttons entirely with a smartphone – and since the input layer allows multimodal inputs – it is still possible to use the prototype with mouse and keyboard alongside the cellphones. It was easily possible to accomodate the client side of the PySensor comunication layer as single logical input device on BILL. In the PySensor environment acceleration data for different devices is collected and prepared to serve as input data for the application. Thus, while the WS3D and BILL are capable of dealing with data from different smartphones, they are not aware of that multiplicity and will treat all the events as they were from one source.

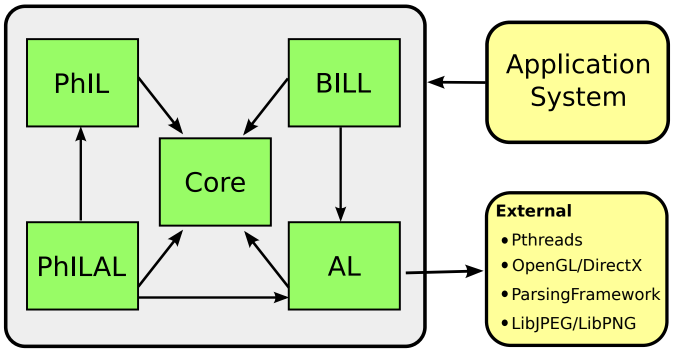


Figure 3: C3DE libraries internal in UML component model[11].

V. Collect, manipulate, and distribute acceleration sensor data

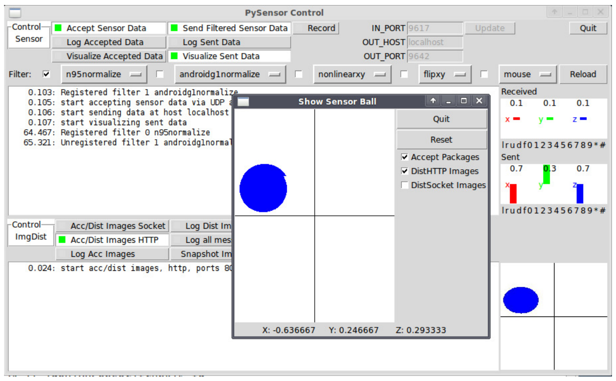
PySensor [13] is an open environment to collect acceleration sensor data from various mobile devices, manipulate the data to include for example gesture recognition, and distribute the resulting data to client applications. We first give an overview of PySensor and then detail the specific data preparation for browsing medical images.

PySensor includes for each mobile platform a device-specific software that sends raw UDP packets containing acceleration and keypress data. The software on the device itself is dumb and does nothing but collect and distribute data. Optionally, images can be sent to the device and visualized providing for a full user interaction loop. The most common smartphone platforms such as Symbian and Android, as seen in Figure 4, are supported and porting to new platforms is straightforward. A server (PC) part allows further processing, logging, recording, and visualization of this data, which is finally distributed to a client application.

Figure 4: Android OS capable smartphone executing PySensor

As most smartphones nowadays include wireless LAN this is the preferred option to send packets. Alternatively, Bluetooth can be used to send the data packets. The server runs the main

PySensor framework which is written completely in Python. Data processing is realized as a filter chain and allows to manipulate the data stream (with access to past packets) with user provided python functions. Data packets from several sources can be mixed, altered and sent as combined data stream to a receiving application. To integrate receiving client applications, libraries in C, Java, and Python are provided that can be used to generate asynchronous user interaction events. In addition, a simple user interface as shown in Figure 5 allows visualization and interactive development of user-provided filters such as ones used for gesture recognition. A recording and playback facility of acceleration data streams allows even unit testing of gesture recognition algorithms.



IN_PORT 9617
OUT_HOST localhost
OUT_PORT 9642

Filter: ✓ n95normalize, androidg1normalize, nonlinearxy, flipxy, mouse

```
 0.103: Registered filter 1 androidg1normalize
 0.105: start accepting sensor data via UDP a
 0.106: start sending data at host localhost
 0.107: start visualizing sent data
64.467: Registered filter 0 n95normalize
65.321: Unregistered filter 1 androidg1normal
```

```
 0.024: start acc/dist images, http, ports 80
```

X: -0.636667 Y: 0.246667 Z: 0.293333

Figure 5: PySensor configuration and signals values

For the described scenario the following filters have been realized. First, all acceleration sensor input data of each device is normalized to a -1 to 1 cube to ensure device independence and make sure that the neutral position on a table with earth gravity is really (0, 0, -1). In addition, earth gravity is removed from the data and neutral position then is (0, 0, 0). Users feel uncomfortable if even slight deviation from a neutral position starts changing the model. Note that movement might occur if the device lies on a table that is not a 100 percent level. Thus, we set the cube from -0.1 to 0.1 to not have any effect. Lastly, as exact positioning is needed as well as fast movement, we apply a nonlinear function – after some experiments we settled for quadratic – to the acceleration data to allow for a faster movement if the acceleration is large and a more fine grained control if the acceleration is small. This also accomodates the inherent unprecision in the delivered acceleration data. Precise control over the actual position can be exercised near the neutral position. In addition, we felt that holding a key for a specific selection feels unnatural to the users. Thus, we make the keys that select a specific part under control of the acceleration data behave as On/Off keys. To this end, we make sure that within a specific time limit a stream of "key pressed" events generates just a single "key press" event. Finally, note that two devices that send roughly 30 packets per second result in 60 packets (or three devices in 90 packets), which was too much for the receiving application to handle, as each time the complete model has to be shifted and redisplayed. Thus, we limit the data rate of the combined generated packets to a level comfortable to the receiving application – 15 packets per second – which still maintains the illusion of direct manipulation.

VI. Experiments and Evaluation

We designed three applications to test the framework. In the first one, we bound some of the commands used in the Workstation3D to the cellphone, so it's focused on the medical imaging part of the framework. The second is a more generic application, so we can run the tests over a more diverse group and focus on usability. With the third prototype we tested the framework reusability and adaptability.

The initial experiments on limited user groups show promising results. For the first application, we asked two doctors and four non-specialist users to interact with it, while trying to achieve some simple goals. Initially, we gave each user a manual with the commands that the application can perform and how to apply each of them properly. In addition, the manual included an explanation of every goal to be performed. When the users understood the instructions, they had a period of 5 minutes to play around with the application and get used to the commands and look and feel of the interaction. Then, we asked them to perform the tasks. The focus of the first experiments was not speed, but assesing how comfortable and natural it is to use the smartphone to interact with the imaging solution.

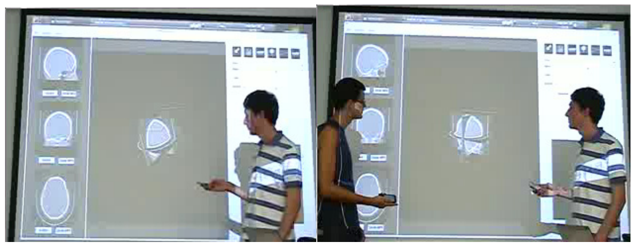

(a) (b)

Figure 6: (a) A user can manipulate the software alone in front of an audience, distant from the computer. (b) Two users interacting with the application through smartphones.

In the experiments the users were asked to visualize a polygonal mesh reconstruction of a skull gathered from a CT examination as shown in Figure 7. Users were asked to perform the following steps:

- Look at a reconstruction from the left and right side
- Zoom on the nose
- Find two implant teeth

The doctors immediately felt more acquainted in controlling the application in comparison to the non-specialized users and completed the tasks much faster in a natural way. However, the non-specialized users did not feel at ease navigating through the skull of a human being. After completing the tasks the users were interviewed with open questions about their experience and prompted to provide suggestions and advices

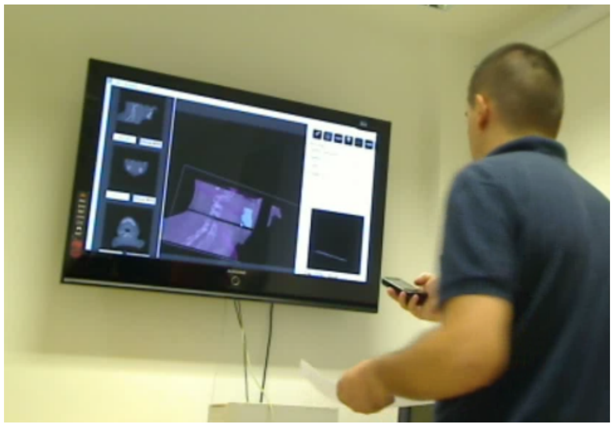


Figure 7: PySensor - WS3D interaction

in order to improve the application. In the next paragraph, we concentrate on the feedback of the two medical doctors.

One doctor is a dentist and works as a professor at a university. It took her about five minutes to get used to using the smartphone as an interaction device. She controlled the application very well, but tended to move her shoulder alongside the hand toward the direction the smartphone was turned. She suggested to switch the directions in zoom control, such that pulling towards the body means zoom in and not move back[1] which basically means zoom out. All casual users made this suggestion after trying to move the smartphone in the "wrong" zooming direction first. The other doctor is a radiologist. He performed all the tasks after the initial phase focused, with ease and faster than all other users. He stated that the application could work as useful tool in discussing examinations within a group of specialists. Both doctors made suggestions and critics of features of the workstation concerning software user interface which is independent of the interaction with the smartphone. A common complaint was the lack of a indication of the three axis origin in the volume and mesh visualization.

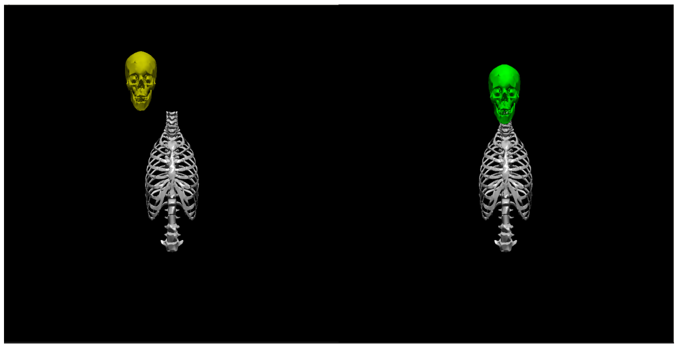


Figure 8: The second test application, before and after the user achieved the goal

For the second application, we gathered a group of 15 testers. The tests were performed individually and no time was given for the users to get used to how the application or the controls worked.

The application itself was fairly simple. We reconstructed a skull and a thorax, moving the skull away from the thorax. Then, we asked the user to put the skull back into its place, right above the thorax. They had to use the phone to translate the skull and, when it reached the right position, it would turn green, indicating that the user succeeded. After a few seconds, another skull would appear on the screen, but now rotated from its original orientation instead of translated. The users had to rotate the skull, making it face forward. Again, when the skull reached the target orientation, it would turn green. Finally, a third skull is shown, now translated and rotated at the same time. On completion of the final task, the program exits and the test is over.

We timed every test and, as shown in Figure 9, the users fall roughly in three groups. The first group (users A through E), are experienced computer gamers and very habituated to mouse control rotations and translations. They also share knowledge of how the smartphone accelerometers and the application itself work. Group two (users F through K) share the same experience in computers with the first group but were unfamiliar with the smartphone. The last group (formed by the rest of the users) did not use the computer much and were unfamiliar even with rotation and translation using the mouse.

Some of the users that were more used to the interface provided by the smartphone to perform rotations and translations could perform the tasks easily, while the ones that had been just introduced to this new input method took a while to apply any conscient transformation to the skull. In fact, eleven users took less time to perform the last task (which requires translation and rotation combined) than to set the second skull right (which only requires rotation) as it can be seen in Figure 11. Probably because they became more skilled in performing the rotation action over time.

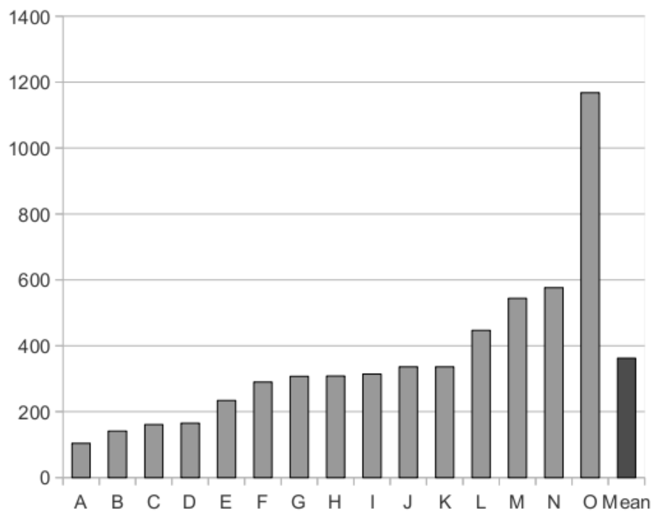


Figure 9: User x Time spent in the test in seconds

[1]The opposite of an average FPS game

Observation revealed that most participants initially expected the smartphone position to reflect the position of the 3D object. They needed some time – much more time for the non-specialized users – to get acquainted to the idea of using the acceleration to indicate position "change". Nonlinear controls and an area without effect is instantly accepted by the user. The eyes are focussed on the screen and the smartphone is not checked at all. However, the natural "zero" position for a smartphone when held is not flat (like on a table) but slightly raised, as seen in Figure 10, which should be accomodated in the future.

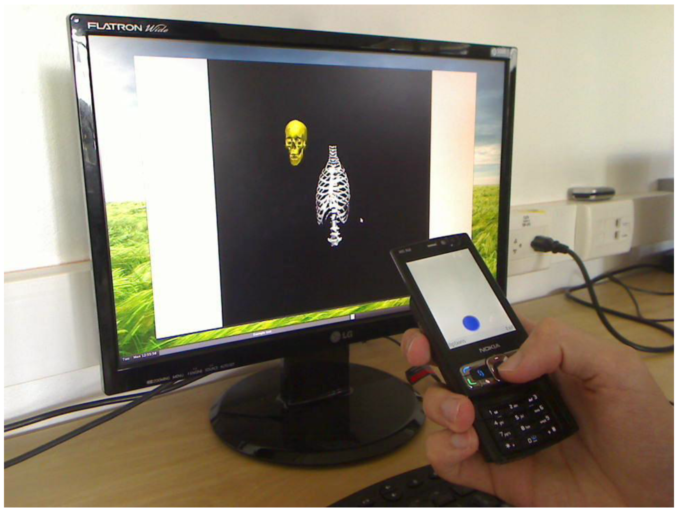


Figure 10: Cellphone natural holding position

A usual complaint was about the lack of a third axis for rotation. Accelerometers cannot provide data for rotations in the Z-axis, but the framework is easily adaptable to make use of that axis whenever it's acessible in hardware. But, after explanations that it is only possible to apply rotations in the X and Y-axis, users quickly adapted and performed the tasks with relative ease.

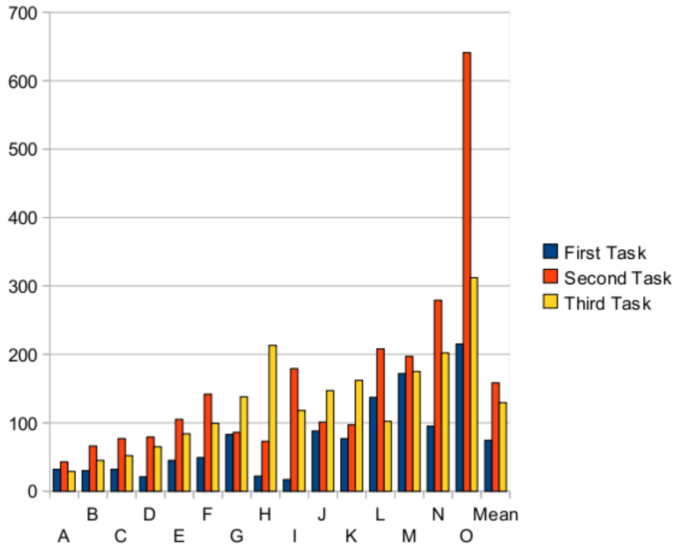


Figure 11: User x Time spent in each task for the second application

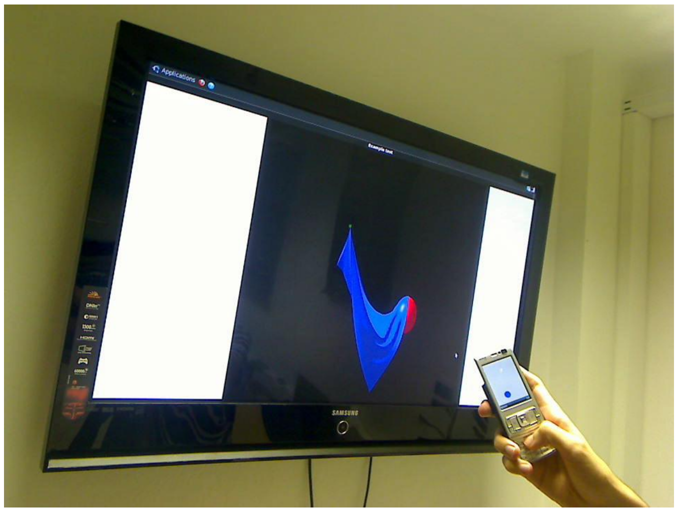


Figure 12: Smartphone interaction with the Spring Mass System

For the last application, we added smartphone controls to a physical simulation prototype. Instead of controlling a skull, the user could translate a ball through the 3D space, where it would interact with a mesh. Users could also pause the simulation and change some of the mesh properties. Little changes were made to the Cellphone Action, but the rest of the framework remained the same. As some modifications are inevitable, we believe the framework achieved a very good level of reusability. Moreover, smartphone events were easily mapped to match the application needs, so the framework could be used in a wide range of applications with different ends.

VII. Conclusion

We have provided a prototypical solution to interactively visualize medical data with commonly available smartphones using acceleration sensor data. The setup does not need any specialized hardware and provides a good user experience due to its intuitive controls and good user acceptance due to its use of smartphones. Both doctors used the application intuitively after a few minutes and did no longer concentrate on the interaction/navigation but on the generated images as anticipated. Collaborative scenarios can be easily supported with current hardware. A simple environment to remotely control applications with acceleration sensor data has been provided and helps to easily develop application specific interpretation of the acceleration data and is very useful to iteratively tune the user interaction.

In the second experiment we realized that users with experience in manipulating 3D data can work with the smartphone more effectively. Despite the initial difficulties in understanding how the input commands work, the results showed a promissing learning curve across the experiment's three stages.

The framework is open to support several different client applications and scenarios. In particular, more elaborate gesture recognition algorithms can be performed within the framework and tailored to specific use cases. Furthermore, the collaboration possibilities of using accelerometer based input devices need to be further investigated and evaluated.

VIII. Future Works

At the present moment we worked with the smartphones as mere input to control and to visualize objects in a three-dimensional environment, but they can also receive signals. In the near future we intend to increase the user interaction, allowing the cellphones in displaying images and/or videos generated by the application executing on a PC. In a more elaborated scenario, users may participate in a web conference using just the smartphones, without concerns about downloading medical images and heavy 3D models. In the PySensor interface users interact with the smartphone's touchpad just as it was with the keypad. Since touchable devices are widely spread nowadays, we could take more advantage of that; allowing the user to control an interface with graphical buttons and touch gestures.

It is hard to find a good device that simulate displacement in surgical simulator with an acceptable cost. The devices are specially developed for that goal and are focused in capturing movement precision [14]. On the other side, the interaction with three dimensional objects through devices such as mice constraints to users, because the movements are limited by a degree of freedom. We also intend to perform some experiments in simulating physical behaviours with the cellphone, probably using the vibrate sensors present in many cellphones to act as a response tool.

References

[1] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz, "Sensing techniques for mobile interaction," in *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, (New York, NY, USA), pp. 91–100, ACM, 2000.

[2] B. Fröhlich and J. Plate, "The cubic mouse: a new device for three-dimensional input," in *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, (New York, NY, USA), pp. 526–531, ACM, 2000.

[3] S. Jeon, J. Hwang, G. J. Kim, and M. Billinghurst, "Interaction techniques in large display environments using hand-held devices," in *VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology*, (New York, NY, USA), pp. 100–103, ACM, 2006.

[4] K. Kodama, N. Fujita, Y. Yanagisawa, M. Tsukamoto, and T. Yoshihisa, "A rule engine to process acceleration data on small sensor nodes," in *ICPS '08: Proceedings of the 5th international conference on Pervasive services*, (New York, NY, USA), pp. 189–190, ACM, 2008.

[5] J. Yang, "Toward physical activity diary: motion recognition using simple acceleration features with mobile phones," in *IMCE '09: Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics*, (New York, NY, USA), pp. 1–10, ACM, 2009.

[6] G. Bieber and C. Peter, "Using physical activity for user behavior analysis," in *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, (New York, NY, USA), pp. 1–6, ACM, 2008.

[7] A. Jakl, "Shakerracer: Real rc car controlled with the n95 acceleration sensor," 2007.

[8] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a wii controller," in *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, (New York, NY, USA), pp. 11–14, ACM, 2008.

[9] L. Gallo, G. De Pietro, and I. Marra, "3d interaction with volumetric medical data: experiencing the wiimote," in *Ambi-Sys '08: Proceedings of the 1st international conference on Ambient media and systems*, (ICST, Brussels, Belgium, Belgium), pp. 1–6, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[10] D. A. Bowman, J. Chen, C. A. Wingrave, J. Lucas, A. Ray, N. F. Polys, Q. Li, Y. Haciahmetoglu, J.-s. Kim, S. Kim, R. Boehringer, and T. Ni, "The international journal of virtual reality, 2006, 5(2):3-14 new directions in 3d user interfaces."

[11] A. Silva, D. Carvalho, T. Nobrega, R. Inacio, and A. von Wangenheim, "A multi-layered development framework for medical imaging applications," in *Computer-Based Medical Systems, 2009. CBMS 2009. 22nd IEEE International Symposium on*, pp. 1 –4, 2-5 2009.

[12] A. F. B. Silva, D. D. B. Carvalho, T. H. C. Nobrega, R. T. Inácio, and A. von Wangenheim, "Framework for interactive medical imaging applications," in *COLIBRI 2009 - Colloquium of Computation: Brazil / INRIA* (Sbc, ed.), (Porto Alegre), pp. 126–129, 2009.

[13] pysensor, "Pysensor." http://code.google.com/p/pysensor, 2009.

[14] C. Bruyns and K. Montgomery, "Generalized interactions using virtual tools within the spring framework: Probing, piercing, cauterizing and ablating," 2002.