Master 2 in Computer Science - Interaction Specialty

Pressure based interaction for 3D manipulations on mobile devices

Author:
Xiyao WANG

Supervisors:
Lonni Besançon
Tobias Isenberg

Hosting lab/enterprise:
Inria Saclay - Aviz team

03/04/2017 – 08/09/2017

Secrétariat - tel: 01 69 15 66 36 Fax: 01 69 15 42 72
email: Murielle.Benard@u-psud.fr

Contents

Summary

Direct touch interaction has attracted much attention recently for interactive visualization and 3D manipulations. However, several major challenges still remain, such as the dismatch between 2D manipulation space and objects' 3D space, as well as the occlusion problems. In this report, we discuss the way to augment tactile interaction technique with pressure sensing on mobile devices and propose our augmented tactile interaction mapping. Our technique offers the possibility to manipulate in 3D with up to 6 degrees of freedom (DOFs) using only one or two fingers, and each DOF can be separated. Also, our proposed technique doesn't need to reserve screen area for manipulations, making it possible to reduce occlusion problems. We initially implement this technique for IOS devices with the help of its pressure sensors, and to generalize to those devices without pressure sensors, we then discuss some most used pseudo-pressure techniques and propose a suitable way for ours. At last, we do evaluation to compare it with traditional *RST* (Rotation, scaling and translation) technique which is most used for 3D manipulation on mobile devices.

Keywords

Human-centered computing, Interactive Visualization

CHAPTER 1

Introduction

The aim of my internship is to propose and evaluate the way of using pressure data to augment tactile interaction for 3D manipulations and eventually design a system that better supports 3D data exploration for scientific visualization.

In the domain of scientific visualization, interaction plays an important role of exploring three-dimensional datasets and there's a increasing number of researches where visualization and interaction are integrated together [19]. By interactively changing the view, scientists are able to immerse themselves in the data as they are exploring it, thus having a better vision on data's structure and it's evolution through time. Basic tasks includes navigation and manipulations (translation, rotation, scaling...) are required frequently [17]. Nowadays, major visualization environments include traditional workstation and complicated virtual reality environment such as the CAVE system. The former environment uses mouse as principal input device which reveals a low efficiency and less preference by users' study [5], that is no longer suitable for advanced tasks. The latter one provides better visual immersion and interaction tools can be added, however, the environment is complex and hard to calibrate and maintain.

To facilitate the 3D dataset's exploration, researchers have started to try using novel interaction input. Tactile input has yielded significant advantages compared with traditional ones and can be very useful for interactive exploration of three-dimensional data [5]. Yet, a few major challenges remain to be solved. The most common one is that tactile interaction is by nature two-dimensional while the manipulations need to be carried in three-dimension (see Fig 1.1).

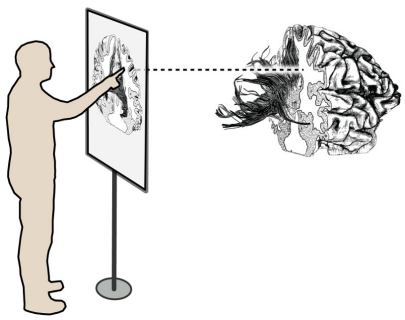
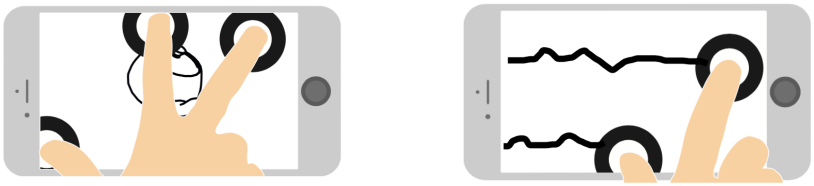


Figure 1.1: Dis-match between 2D tactile interaction and 3D data/manipulation, image from [17].

Also, each finger can only offer two degrees of freedom (DOFs), but generic tasks in 3D usually require manipulations up to at least 6 DOFs to specify the object's

position and orientation [17], such as translation along three axis, rotation around three axis and scaling. An increasing number of fingers could help but also increases occlusion problem. This problem seems not be a big one for some setups such as the large display, but it is an important issue for relatively small input surface like the mobile device: some content will be hidden beneath fingers, such as shown in Fig 1.2a. Apart from that, Besançon et al. [5] summed up by user studies that two-finger tactile manipulations are sometimes difficult to perform because two finger motions are all integrated, making it extremely hard to perform a manipulation changing only one DOF without affecting others.



(a) (b)

Figure 1.2: (a): Some part of content will be hidden with an increasing number of fingers on screen, image from [30]. (b): While doing the two-finger translation, users often perform pinching or other motions as well even if they're not intended to, image adjusted from [30].

There are already several ideas to facilitate or augment tactile interaction (see chapter 2 for details) in order to address these major challenges. The use of pressure is one possible way that has been partly discussed. In addition, recent release of mobile devices are equipped with pressure sensors built in the touch screen, such as 3D-touch[1] proposed by Apple or force-touch proposed by other companies, allowing us to get the pressure data directly from the screen, without the needs of assembling other electronic devices on the back or around the mobile as done by [14, 25], thus providing a much simpler way to combine tactile interaction with pressure. But research work for this combination is still very limited and none of them provided a complete design for manipulating objects in three dimensions.

This report is planned as follows: Chapter 2 begins with introducing previous work related to ours, including tactile interaction in 3D and the use of pressure. Next, Chapter 3 provides a detailed description for our design. And then, chapter 4 talks about the way we implement this design followed by chapter 5 who explains the experiments for evaluation. At last, chapter 6 gives a global conclusion of this work and discusses what possible future work could be done to improve.

[1] https://developer.apple.com/ios/3d-touch/

CHAPTER 2

Related Work

This project aims at augmenting tactile interaction by using pressure modality to fit the needs of 3D manipulations, related work mainly focuses on following fields: direct touch interaction with 3D environments and the use of pressure to augment tactile interaction including the use of pseudo-pressure.

2.1 Direct touch interaction with 3D environments

Direct touch interaction, where manipulations and visualization are located in the same space, has attracted much attention in human-computer interaction and information/scientific visualization, for example, Knoedel et al. [20]'s experiment compared it with indirect touch for 2D and 3D manipulations' task and showed that it has a faster completion time for both task.

Nowadays, techniques such as RNT (*Rotations* and *Translations*) [21] and RST (*Rotations*, *Scaling* and *Translations*) [22, 23], have been widely mentioned and explored for 2D manipulations (e.g., [13, 21]). But only a very limited number of studies have investigated 3D manipulations tasks.

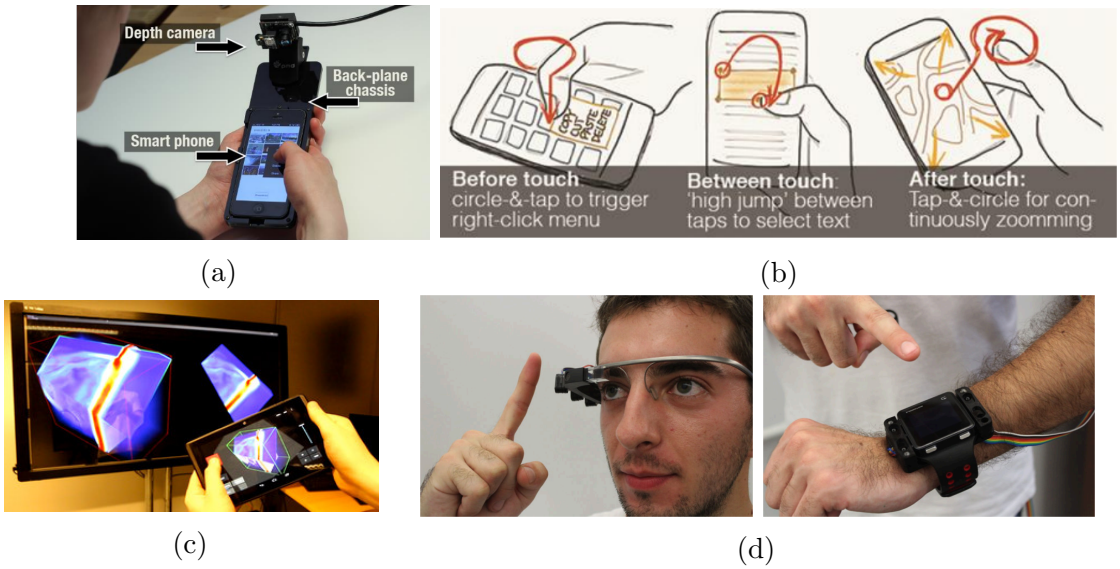
Hancock et al.[11] extended traditional RST to 3D and provided different mapping possibilities on direct-touch display using one-, two- and three-touch input to support 3D interaction: results showed that three-touch input has both a higher performance in terms of completion time and a higher user preference than others. They then presented another work offering a full 6DOF 3D interaction technique called *Sticky Tools*[12]: this technique mapped two fingers' motion for translations (along x, y and z) and rotation around z, and a third finger for rotation around x and y. Reisman et al.[27] also presented a method using three to four fingers to directly manipulate 3D objects, especially to address the challenge of rotation.

Despite the efficiency of increasing number of fingers used for interaction, occlusion problem appears too. Some researchers investigated to reduce the number of fingers while keeping a high DOF offered: Coh\'e et al.[10] designed *tBox* which could offer direct and independent control up to 9 DOF: 3 translations, 3 rotations, and 3 scalings by manipulating *tBox*'s border lines or other space of the screen.

Another way to reduce the number of fingers needed for carrying manipulations is investigated by Zeleznik et al.[33], Nijboer et al.[24] and Yu et al.[32], they proposed to reserve visualization space's boarders, map different screen areas as different manipulation functions to control interaction. For example, *FI3D* proposed by Yu et al.[32]: by starting and carrying the touch on different areas, this design allows users

to have full 7 DOF interaction control (3 translations, 3 rotations, and 1 uniform scaling) with just single- or dual-touch input.

Besides these techniques mentioned above, recently researchers do not only focus on the touch screen, but also try to combine with other interaction ways. For example, Besançon et al. [6] presented a hybrid tactile/tangible interaction technique for data exploration using a Google Tango tablet[1] (Fig 2.1c). Chen et al. [9] proposed *Air+Touch*. Their technique combined in-air gesture and touch events with the help of a depth camera (Fig 2.1a), thus allowing plenty of different interaction possibilities with only one finger (Fig 2.1b for some examples). Withana et al. [31] also used infrared sensors to recognize shallow depth gestures to address the challenges faced by spatially limited input device such as the touch screen (Fig 2.1d). And Hinckley et al. [16] designed system combining tactile modality with pre-touch sensing (fingers above the screen and grip around the mobile).



(a) (b)

(c) (d)

Figure 2.1: (a) Combining tactile interaction with air gestures using a depth camera, image from [9]. (b) Some examples of manipulations with *Air+Touch*, image from [9]. (c) Hybrid tactile/tangible interaction for 3D Data exploration, image from [6]. (d) *zSensing*: shallow depth gesture & tactile interaction, image from [31].

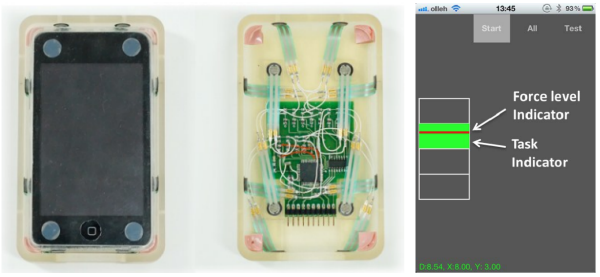
2.2 Pressure with touch interaction

Pressure can also be used as a primary input or an auxiliary factor in combination with other input. We are investigating the use of pressure to augment tactile

[1] https://get.google.com/tango/

interaction. Hancock et al.[12] mentioned that force-based interaction can be used to augment 3D manipulations, they proposed to firstly use a sensing technology to capture and translate physical information from real world, and then use a display technology to show the information in virtual world. But they didn't give any exact proposition or mapping examples of the way to use the pressure.

Limited to input device, most of the researchers tried to install pressure sensors by their own on input devices by now. For example, Heo et al.[14] captured pressure from both sides and back of an input mobile device (see Fig 2.2a) and proposed to use different pressure levels to distinguish manipulations such as drag and slide. According to their studies, a certain pressure level is not always easy to maintain during the manipulation, they later presented *ForceDrag* and *force lock*[15] to use pressure as an input modifier: selecting interaction mode with pressure before manipulations are proceeded. This work also suggested to add a force level indicator as virtual feedback (see Fig 2.2b).



(a) (b)

Figure 2.2: (a) Sensors installed on back and sides of a mobile, image from [14]. (b) Visual indicator, image from [15].

Besançon et al.[4] also integrated pressure sensors on back of the device to control gain factor. Apart from getting pressure data from back or sides of the mobile. Pelurson and Nigay[25] investigated to fix the sensor on the front bezel of the input device and use a non-dominant hand to augment navigation task for large 1D information.

2.3 Pseudo-pressure

What we would like is to record the pressure data while we're applying a touch on the screen without the needs of complex installation and calibration. Brewster et al.[8] designed a pressure-based text entry application with a resistive touchscreen where pressure data can be got easily without any modifications of the device's hardware. But nowadays, most mobile devices are equipped with capacitive touchscreen, the

pressure data can't be got naturally as the resistive screen they use, so before the wide spread of new pressure-sensitive screen (such as *3D Touch* by Apple and *Force Touch* by Huawei), the most common approach is to use pseudo-pressure. According to previous research work, we found three major ideas: contact area based approach and time based approach.

Contact area based method suppose that the contact area between our finger and screen augments with the pressure exerted. Benko et al.[3] initially investigated the contact area with different force to distinguish a cursor's tracking and dragging state. Boring et al.[7], as shown in the Fig 2.3, succeeded in combing multi-finger gestures into a single finger motion with the help of contact area based pseudo-pressure.

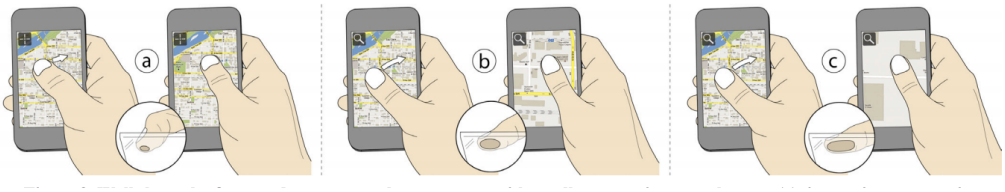


Figure 2.3: Using contact area based pseudo-pressure to perform manipulations initially required multi-touch gestures, image from [7].

Time based method suppose that a hard press usually requires a longer completion time than a tap, thus by recording the time of a touch, it is possible to simulate the pseudo-pressure. Arif et al.[2] talked about some example applications such as DooDle Buddy[2] but they yielded that this method has major challenges such a s a longer completion time. They also mentioned the contact area based approach's weak points: the contact area varies much from person to person and is dependent on type of touch. So they combined them together for a hybrid method calculating both average touch time and average touch area for a predictive text-entry application. They also used the pressure modality for an authentication system by recording both key sequences and pressure applied[1].

In addition, Apple company once revealed that their iPhone's accelerator can be an alternative to simulate the pressure: If users hold the mobile device while manipulating, a hard press usually gives a higher acceleration or displacement to this device. Even though differences are relatively small, their built-in sensors are sensitive enough to capture these differences and to distinguish different pressure levels. But we didn't find any relevant research papers about this topic.

[2] https://blog.pinger.com/tag/doodle-buddy/

CHAPTER 3

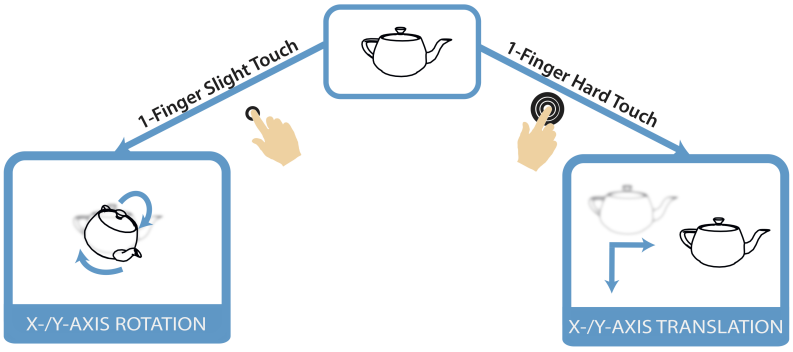
Technique

This chapter introduces the technique mapping of combining pressure data with tactile input and other elements of our system's design such as the calibration and feedback.

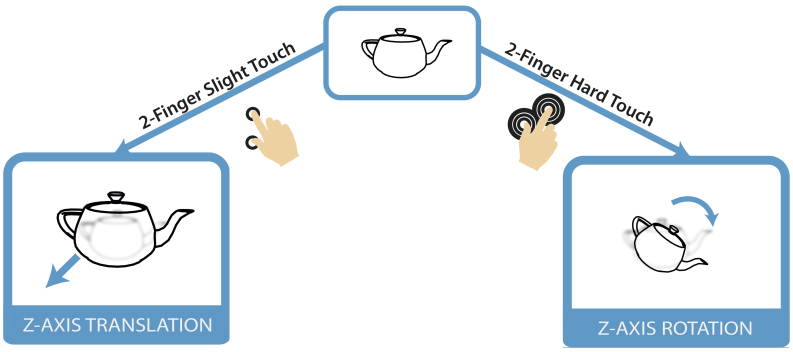
3.1 Interaction mapping

We propose to augment traditional RST tactile interaction technique (based on the most common mapping in mobile applications for 3D manipulations, see [5] for details) with pressure input. Initially, we use an iPhone 7 whose pressure sensitive screen allows us to get pressure data properly while we are touching the screen. So augmenting tactile interaction only needs to use this data directly while performing manipulations. However, it has been argued that keeping a stable level of force during whole manipulations is too difficult [14, 15], we choose to use quasi-postural mode [18]: *a posture whose initial configuration is augmented with a brief initial dynamic action but where this action's continuation is also used to parameterize the effect*. Specifically, instead of changing the data or the view, the beginning of a motion is used to select the interaction mode. And then all manipulations can be carried with only a slight force, that is to say that the pressure is only important at the start of a manipulation.

Even though it has been suggested that users can exert several different discrete force levels without too many difficulties [15], a binary mapping (distinguishing only light and hard touch) is enough to separate the 6 DOFs needed, and is easier and less frustrated to perform. Augmenting the number of force levels will only increase the difficulty of manipulations, so we chose to use the pressure only as a binary variable. Our proposed mapping is illustrated on Fig 3.1.



(a)



(b)

Figure 3.1: Manipulation mapping, images from [30].

To leave users a more flexible interaction way, we also allowed users to manipulate the 3D object using the traditional RST two-finger motions by selecting the right mode at the beginning. Specifically, we defined the following three modes:

- Light: This is the default mode, if one finger motion is triggered without a hard touch. This mode is used for rotation around x-/y-axes and translation along z/zooming.
- Hard: If one finger is put on the screen and this finger performed a hard touch, then it changes to the hard mode. This mode is used for translation along x-/y-axes and rotation around z-axes.
- Integrated: If two fingers are put down on the screen and one of them performed a hard touch. This mode is used for the RST two-finger motion: translation along the x-/y-axes(two-finger's translation), rotation around the z-axis(rotation gesture), and translation along the z-axis(pinching gesture).

The transition graph between different modes is represented on Fig 3.2. The transition is triggered by the movement of the touch, including both the change of position and the change of pressure. Transition conditions between different states are the number of fingers required and the relation between force exerted and force threshold, for example, the requirement $1/\geq$ of passing from *light* to *hard* means that it requires one finger (neither more nor less is possible) on screen and the force exerted should be greater than or equal to the force threshold. x means that this is not demanded specifically, any possibility is supported. The isometric force captured by IOS SDK is a continuous variable (see section 5.2 Fig 5.3 for experimental results), all touch began with only a very slight or near zero force, so the initial mode is always *light*, but a hard touch will attain the force threshold very fast while a light touch will never exceed this limit.

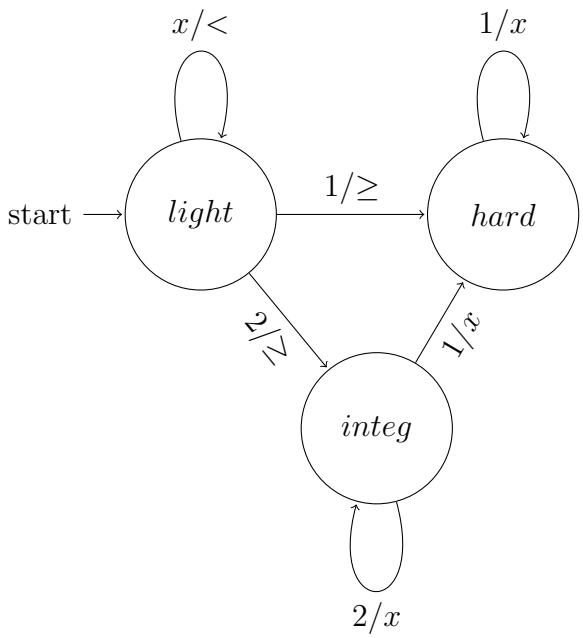


Figure 3.2: Transition graph of switching between different interaction mode.

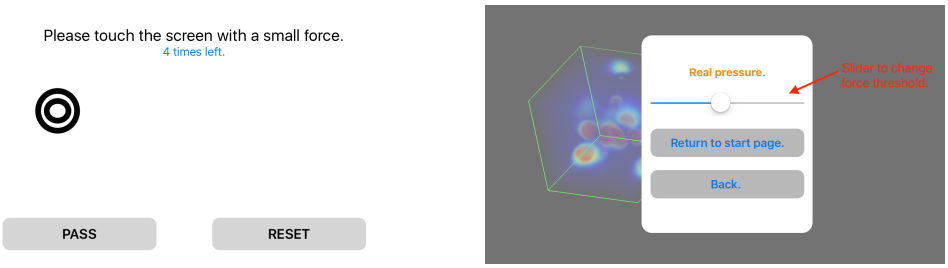
So the manipulation mapping for each of the DOF is defined as

- Translation along x/y: one-finger translation in hard mode or two-finger translation in integrated mode. The translation distance of the object on the display surface is equal to that of the finger.
- Rotation around x/y: one-finger translation in light mode. The mapping algorithm is constrained ArcBall [29].
- Translation along z/Zooming: pinching gesture in light or integrated mode. Zooming scale is equal to pinching scale.
- Rotation around z: two-finger rotation gesture in hard or integrated mode. Rotation angle of the object is equal to that of the gesture.

Specially for the rotation around z-axis/zooming, user should put first one finger on the screen, perform a hard touch, and then put the second finger on screen. To avoid unwanted translation (the first finger may move a bit) before the second finger is put down, we set a timer: if the second finger is put down in a very short time, we consider that the translation of the first finger is undesired due to mis-operating, then the data will be reset to the status before the first touch is effected. En contrary, if the second finger is put down after the time interval reserved, the data won't be reset.

3.2 Threshold calibration

The threshold of pressure can be firstly calibrated before manipulations to ensure that it is suitably defined for each user: users are asked to follow the indication on screen, tap or slide on some random positions on screens, with both small and heavy forces, and repeat several times (see Fig 3.3a). It's also possible to skip this phase, use default value, and then make changes manually later during the manipulations using a slider (see Fig 3.3b). This slider and the menu are displayed with a three-finger touch which is not used for 3D manipulations.



(a) (b)

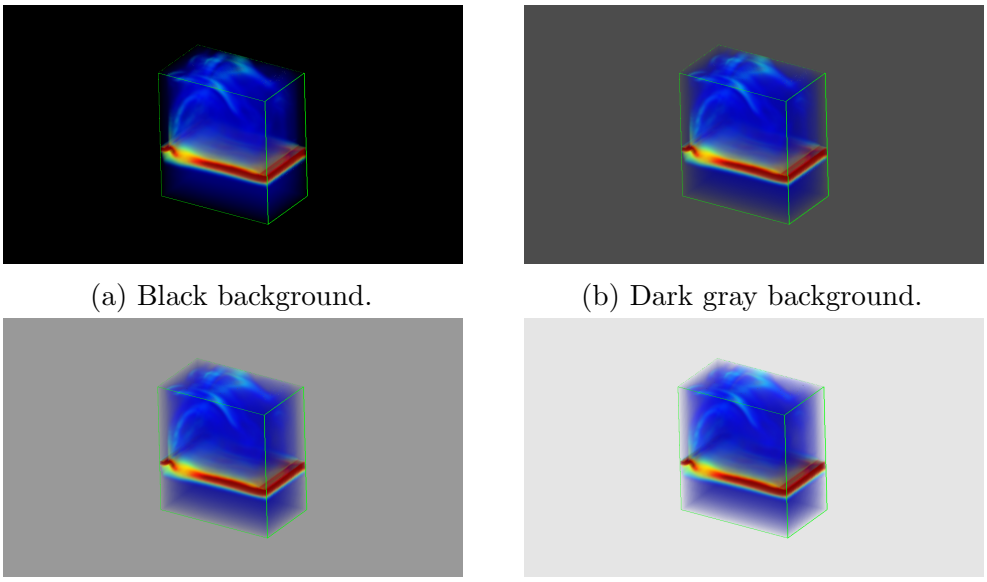
Figure 3.3: (a): Calibration phase before performing manipulations, the circle is the place asked to touch. (b): Possible to change force threshold during the exploration with the slider indicated.

3.3 Feedback

Moreover, the importance of feedback while controlling the pressure level has been pointed out [15, 26], we added both visual and haptic feedback. As the mobile screen is relative small, we prefer to use the maximal screen area to visualize 3D objects, so instead of adding a feedback bar (see Chap 2 Fig 2.2b), our visual feedback, as shown in the Fig 3.4, is to use different background color to indicate which interaction mode the user is currently in:

- Black background for no touch.
- Dark gray background for light mode.
- Light gray background for hard mode.
- Quasi-white background for integrated mode.

The choice of color is based on the visualization content, for now, we use only grayscale colors because they have no conflicts with the flow data that we are currently working with.



(a) Black background.

(b) Dark gray background.

(c) Light gray background.

(d) Quasi-white background.

Figure 3.4: Screen shots for visual feedback.

The haptic feedback means a short vibration of the device while passing from a slight touch to a hard one. One feedback might be enough for users to notice the change, but we still use two to get a better feedback and we think about evaluating the effect and preference of each feedback in future work.

3.4 The use of pseudo-pressure

The technique mentioned above requires a pressure sensitive screen which hasn't been widely equipped by most mobile devices. To address this challenge and to better generalize our method, we also propose a way to use this technique with pseudo-pressure instead of real pressure data.

We designed our pseudo-pressure technique based on results observed with simple data gathering experiments (see section 5.2). We use a combination of contact area based technique and time based technique to simulate pseudo-pressure. Based on experimental results, we assume that a hard touch has a large contact area and a hard mode selection takes a longer time for the finger to move.

CHAPTER 4

Implementation

This technique is initially implemented on an iPhone 7 (4.7 inches' screen diagonal, 750 × 1334 pixels, 326 ppi[1], iOS 10.0.0). We capture input events (both tactile and pressure inputs) using Swift 3. We also support scientific datasets ($*.vtk$ or $*.vti$ formats) that we first read using VTK 7.0 framework and then render with Open GL ES 3.0 using our own implementation to allow for more flexibility with the coding of the interaction technique. As the VTK library doesn't support Swift for now, we implement the 3D model and rendering parts in $C++$.

As our mobile application is an interactive system, we use the Model–view–controller (MVC) architecture which is popular for user interface design. As shown in Fig 4.1, an application is divided into 3 main parts: controller captures and processes users' behavior, convert to commands and send them to model; Model accept controller's commands to manage and change the data; View display the data and user interface.

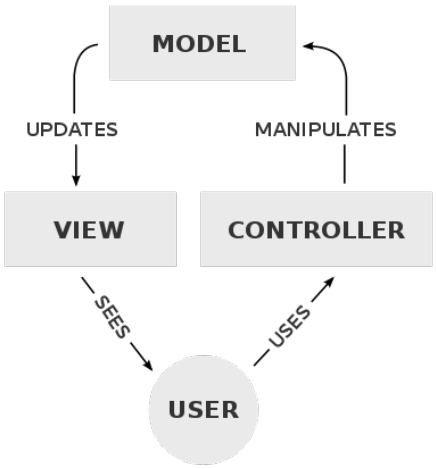


Figure 4.1: MVC diagram, image from https://en.wikipedia.org/wiki/Model-view-controller.

In our system, users touch the mobile's screen to manipulate the displayed 3D object. These touch motions are captured and processed by controller (Swift 3 and IOS SDK), and then controller interprets these behaviors and send commands to model (C++ and VTK 7.0). Model will manage the data according to the commands, specifically, the 3D object's position or orientation will change. At last the updated

[1]From https://www.apple.com/fr/iphone-7/specs/

data is displayed in real time by the view (C++ and OpenGL ES 3.0), so users can observe the results of their behaviors.

To get best use of the screen and to make the manipulable are as large as possible, we decided to use the full screen without reserving any concerns or margins and to force the display to be a landscape orientation. So the display/manipulation size is the same as the screen, its coordinate reference is shown as the Fig 4.2.

Figure 4.2: IPhone screen and its coordinate system. Image modified from the one of https://support.apple.com/iphone/repair.

4.1 3D visualization

Our implementation for 3D visualization is based on Open GL ES 3.0 which is a special version of OpenGL for embedded systems such as mobile phones. Its rendering pipeline is shown in Fig 4.3.

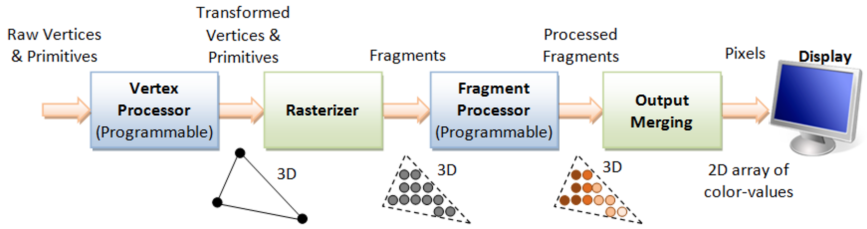


Figure 4.3: An simple illustration of OpenGL rendering pipeline, image from https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html.

Raw vertices & Primitives are data that we need to send to GL shaders to process, it contains vertex's spatial position and other useful information such as color, normal, texture and light sources. We need firstly to represent a 3D object with triangular segments, and for each vertex of these triangles, we map a color value

or a texture. In this pipeline, we need to concern two phases: vertex shading (*Vertex Processor* phase in Fig 4.3) and fragment shading (*Fragment Processor* phase in Fig 4.3)

Vertex shading phase will calculate object's position projected on screen. But we should first convert the 3D object's model coordinates into the real world space, and then to camera space which is a right-handed three-dimensional Cartesian coordinate system. At last the 3D space will be projected on 2D mobile screen display area as illustrated in Fig 4.4.

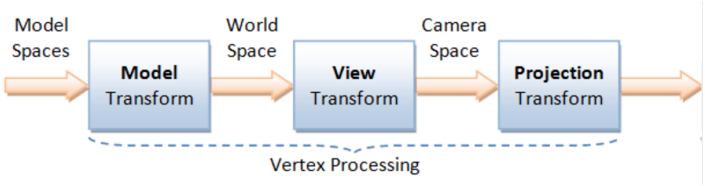


Figure 4.4: An illustration of vertex shading pipeline, image from https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html.

Model transform is used to convert object's model coordinates to the real world coordinate systems. And then, view transform makes real world coordinates be converted to the camera space. As it has been pointed out that *In Computer Graphics, moving the objects relative to a fixed camera (Model transform), and moving the camera relative to a fixed object (View transform) produce the same image, and therefore are equivalent.*[2] So we manage them together with a single model-view matrix. Interaction part communique with this matrix to change the 3D object's position and orientation (see section 4.2).

The last step is to establish a projection matrix to map the position in 3D to 2Dy mobile screen displa, depending on types of projection. The default projection of our system is perspective projection because it simulates the best human's view: *objects in the distance appear smaller than objects close by*[3]. As shown in Fig 4.5, the view frustum is camera's field of view, all elements outside the frustum won't be seen, we map this space to our whole OpenGL space because we try to avoid missing any visualized elements; Fovy is the angle between the bottom and top of the projectors

[2] https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html
[3] https://en.wikipedia.org/wiki/3D_projection

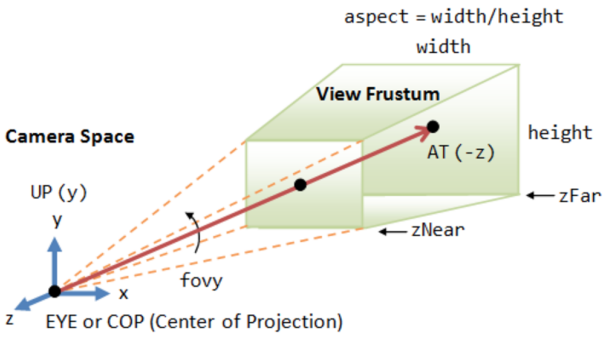


Figure 4.5: An illustration of perspective projection, image from https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html.

In addition, sometimes we also switch to the orthographic projection where the the view volume is a parallelepiped (Fig 4.6). In this case, the object's size doesn't change with its z distance, so they are mesearable or comparable regardless of their z-distance.

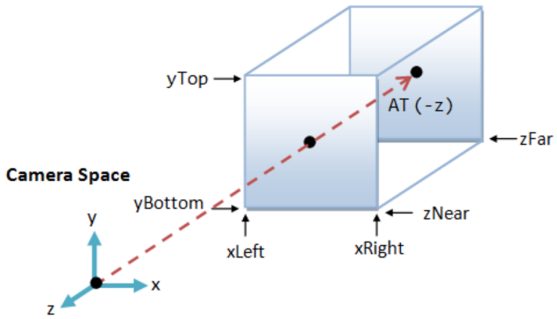


Figure 4.6: An illustration of orthographic projection, image from https://www.ntu.edu.sg/home/ehchua/programming/opengl/CG_BasicsTheory.html.

In conclusion, the position on screen of each vertex is computed with the equation 4.1.

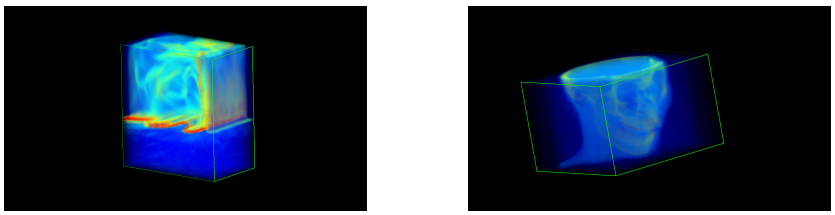
$$\text{postion} = \text{projection} * \text{model-view} * \text{vec4(3Dpositon,1)}. \tag{4.1}$$

Fragment shading phase computes the color or texture of a vertex, and then the color of other pixels inside a triangular fragment is interpolated with its three vertices. We don't need illuminations or shadows for now, so this phase is just to affect the vertex color with what we mapped.

Our main goal is to apply our technique for interactive exploration of scientific data, so we use VTK 7.0[4] to support some special kinds of scientific data, such as the

[4] http://www.vtk.org/

flow data (Fig 4.7a) and medical data (Fig 4.7b). With the help of VTK framework, our program reads the raw data from *.vtk* or *.vti* files and then process the data to form segments compatible to OpenGL shaders and then according to different visualization requirements, mapped each vertex with specific color.



(a) (b)

Figure 4.7: (a): Flow data. (b): A model of human head.

4.2 Tactile interaction

After the program captures all the touch events and their related information such as the position, force and contact radius, we compute a transformation matrix for each movement of the finger and then multiply it with the formal model-view matrix to update the object's position or orientation.

For our technique, the rotation center is the center of the data model, not the center of screen nor the center of camera space. So we also define a center point of data model $O_object = (O_x, O_y, O_z)$, each time a translation is effected, the center should be translated with the same distance too. For rotation motion, we can compute the translation axis each time and then compute the rotation matrix around this axis, but this method is complex, so instead we chose to use another way: the rotation matrix is computed according to the normal axis of the world, but instead of updating the model view directly, we translate the model to the center of the world space, rotate, and put the data back to the position where it belongs to. As the position is represented as a column in homogeneous coordinate (compatible to OpenGL shaders), the update of view for rotation is shown in the equation 4.2.

$$\text{model-view} = \begin{bmatrix} 1 & 0 & 0 & O_x \\ 0 & 1 & 0 & O_y \\ 0 & 0 & 1 & 0_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \text{rotation matrix} \times \begin{bmatrix} 1 & 0 & 0 & -O_x \\ 0 & 1 & 0 & -O_y \\ 0 & 0 & 1 & -0_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \text{model-view} \tag{4.2}$$

And for other transformations, the view matrix is updated by equation 4.3.

$$\text{model-view} = \text{transformation matrix} \times \text{model-view} \tag{4.3}$$

Rotation around x and y - constrained Arcball

Rotating around x/y is mapped as sliding one finger with a small force on screen. We use the Arcball technique to realize this transformation. This technique is proposed by Shoemake [29] aims initially at adjusting the spatial orientation of an object, and then being investigated for tactile interaction. It offers a consistent way to do rotation freely at any axis.

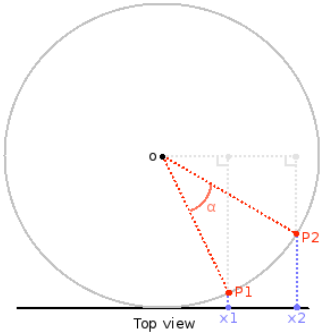


Figure 4.8: Arcball rotation representation, image from https://en.wikibooks.org/wiki/OpenGL_Programming/Modern_OpenGL_Tutorial_Arcball.

As shown in the Fig 4.8, it is supposed that there's a invisible ball behind the screen, rotating the view on screen is seen as rotating the ball. Supposing P_1 is the start point and P_2 is the end point of a motion, the rotation matrix is computed by:

1. Convert the pixel position of to space, which means convert x from $[0 width]$ to $[-1+1]$ and y x from $[0 height]$ to $[-1+1]$. r is the radius of the ball, we usually fix it to 1.

$$x = -(x\_pixel\text{–}width/2)/(r * width/2)$$
$$y = (y\_pixel\text{–}height/2)/(r * height/2);$$

2. Compute vectors $\overrightarrow{OP_1}$ and $\overrightarrow{OP_2}$.
if $x^2 + y^2 > 1$, then $\overrightarrow{OP} = (x/sqrt(x^2+y^2), y/sqrt(x^2+y^2), 0)$,
else $\overrightarrow{OP} = (x, y, sqrt(1-x^2-y^2))$

3. Compute rotation angle and rotation axis.

$$\text{rotation angle: } w = P_0 \cdot P_1$$
$$\text{rotation axis: } (x, y, z) = P_0 \times P_1$$

4. Compute rotation matrix.

$$\text{rotation matrix} = \begin{bmatrix} 1-2y^2-2z^2 & 2xy-2zw & 2xz+2yw & 0 \\ 2xy+2zw & 1-2x^2-2z^2 & 2yz-2xw & 0 \\ 2xz-2yw & 2yz+2xw & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

But Shoemake also mentioned at the last of his article [29] that certain manipulations could become easier if we add some constraints by fixing the axis of rotation. For our implementation, we constrained the rotation axis separately for x and y, thus we only need to know the rotation angle θ, and the rotation matrix can be computed easily by

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) & 0 \\ 0 & sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

$$R_y = \begin{bmatrix} cos(\theta) & 0 & sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

Translation along x and y

Translation along x and y is mapped to a one-finger slide on screen preceded by a hard touch. We mapped the distance of translation for the object is same as the distance translated by the finger. When the required hard touch is performed, the program begins to record the touch's position on the screen and then compute its movement along x and y axis from the previous position, noted dx and dy. And then translation matrix is simply computed by

$$\text{translation matrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

Translation along z/Zooming

According to our prospective projection, translation along z and zooming have similar effect. It is mapped to a pinching gesture that can be recognized by IOS SDK. We choose to implement the scaling operation instead of translating along z for mainly two reasons. The first is that zooming scale and pinching scale can naturally match

without the need of additional calculations. The second reason is that zooming still work even if we change the view to the orthographic projection.

Each time the gesture is recognized, we compute the zooming matrix and then adjust the view. But the gesture recognizer is only capable of registering the scale from the initial status, not the scale from the previous movement. So each time this gesture is recognized, we initialize a variable $s = 1.0$, when the fingers move, we compute uniform zooming matrix and then we update $s = scale$.

$$\text{zooming matrix} = \begin{bmatrix} s/scale & 0 & 0 & 0 \\ 0 & s/scale & 0 & 0 \\ 0 & 0 & s/scale & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.8}$$

Rotation along z

The rotation along z is mapped to the two-finger rotation gesture that can be recognized by IOS SDK. The rotation angle for the object is mapped naturally to the fingers' rotation angle. We record the rotation angle for each movement, compute the difference angle θ between the current the previous movement of fingers and thus we compute the rotation matrix around z axis:

$$R_z = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.9}$$

CHAPTER 5

Experiments

This chapter describes the experiment procedure that we would like to do. But we we are still waiting for the agreement of COERLE(Le comité opérationnel d'évaluation des risques légaux et éthiques)[1] Inria, and we can not begin our evaluations without it.

5.1 Evaluation of the interaction technique

The principal experiment aims at evaluating the performance of our pressure augmented manipulation technique. We chose to compare it with the traditional RST technique because for now RST is the most investigated for manipulating 3D objects on mobile devices, other tactile based interaction techniques, such as *tBox* and *FI3D* are not yet commonly used and they all require to add elements on screen which will make the visualization space narrow.

We will ask participants to do 3D docking tasks with traditional RST technique and our augmented one. The docking task is similar to that done by Besançon et al. [5] (Fig 5.1).

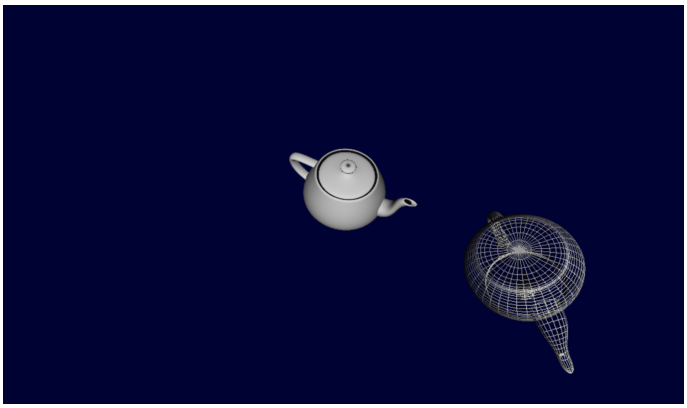


Figure 5.1: An example of docking task, image from [5].

The phone will be put on the table in front of the participant. Participants are not allowed to move the phone and they will be asked to manipulate this object to try to best fit the target's position and orientation (the right one on Fig 5.1). Each

[1] https://www.inria.fr/institut/organisation/instances/coerle

user will perform 15 tasks with each technique. The position and orientation of both the object to manipulate and the target object are randomly generalized and will be different from other tasks.

We will investigate to evaluate task completion time, precision (euclidean distance and rotational angle), workload, fatigue, and preference. The last three parameters will be evaluated with several questionnaires: before the experiment, a first questionnaire for their demographic information including the familiarity with such manipulations; after each technique, we will ask users to fill in a questionnaire reporting their workload (based on NASA's Task Load Index[2]) and fatigue level (based on Shaw's approach [28]); at the end, we will ask users to fill in a final questionnaire assigning their strategy, preference, and other remarks.

5.2 Identification and evaluation of pseudo-pressure

We also generalize our technique to those mobile phones without pressure sensors with the help of pseudo-pressure, but even though some research work has been done for pseudo-pressure such as what we introduced in section 2.3, none of them is perfect and we don't know which is better for 3D manipulations either.

The global flow chat for establishing the pseudo-pressure model is shown in Fig 5.2. We first need to gather touch related information to identify a initial model to simulate the pseudo-pressure. And then we will do experiments to verify if this model works well. If it gives satisfying results, this model is validated and can be used to replace pressure sensing on old mobile devices. If not, we will correct and update the model based on feedback and then re-do the evaluations until a suitable one is found.

[2] http://humansystems.arc.nasa.gov/groups/tlx/downloads/TLXScale.pdf.

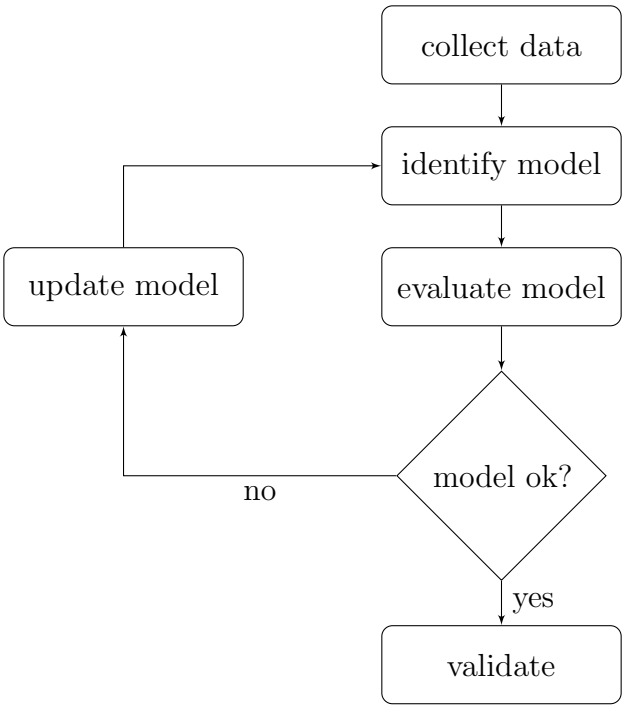


Figure 5.2: Flow chat of evaluation process.

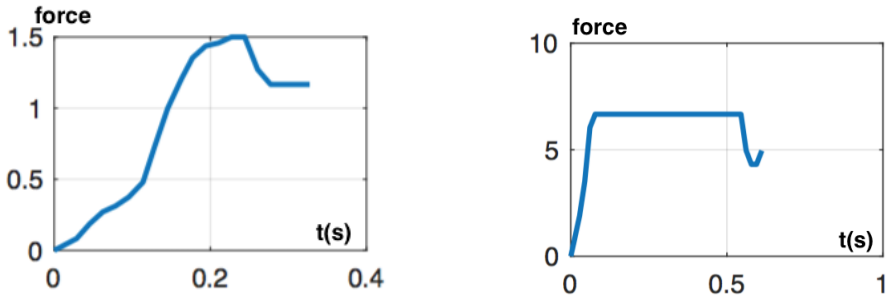
Gathering data

The first step is to gather related information for slight and hard touch and thus use the most distinguishable attributes to establish the model.

The phone will be put on the table in front of the participant. Participants are not allowed to move the phone and they will be asked to perform mode selection tasks(the same way as our pressure based interaction technique required) and then slide on screen. Participants will use successively his/her right hand thumb, right hand index finger, left hand thumb, and left hand index finger to perform the tasks. With each finger, participants need firstly perform a light mode slide on screen three times, which means participants should touch the screen with a slight force and then slide en screen. And then with the same finger, participants are asked to press first hardly on screen, and without releasing their finger, slide on screen with normal force for three times. The touch position and slide trajectory will be changed each time.

We observed similar phenomenon based on this data gathering:

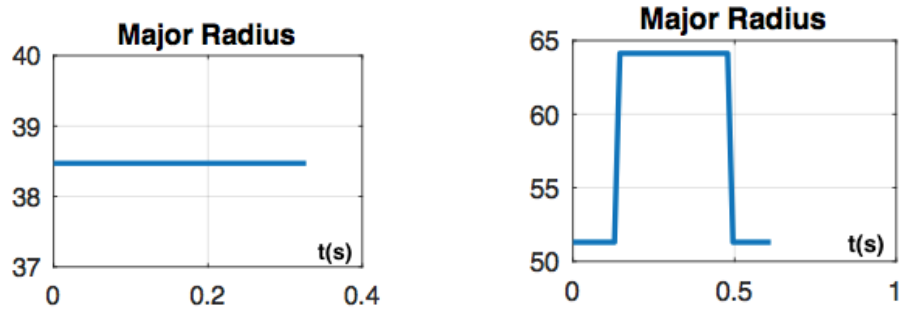
- Pressure is captured as a continuous variable; all types of touches begin with a quasi-zero pressure.



(a) (b)

Figure 5.3: (a): Pressure-time plot for a user performing a light touch with right hand index finger. (b): Pressure-time plot for the same user performing a hard touch with the same finger.

- For the same person using the same finger, a hard touch usually has a larger contact radius (leading to contact area) compared with a light touch.

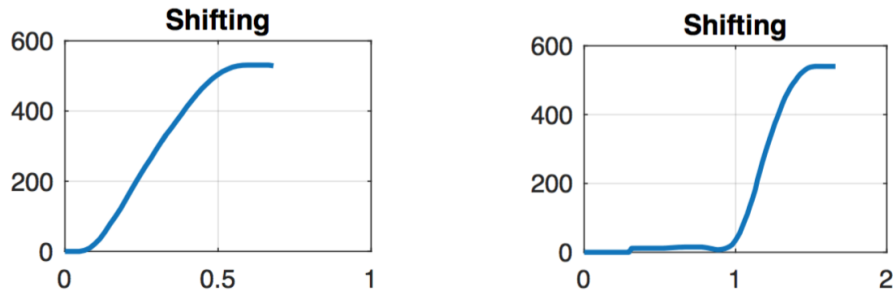


(a) (b)

Figure 5.4: (a): Major contact radius-time plot for a user performing a light touch with right hand index finger. (b): Major contact radius-time plot for the same user performing a hard touch with the same finger.

However, the API offered by IOS SDK for detecting major contact radius is inaccurate, the radius doesn't change continuous but with a certain range, which makes it incapable of detecting relatively small variations. So we don't use only the contact surface for pseudo-pressure.

- The change from light mode to hard mode requires some time to complete. For motions in light mode, touches move almost directly after the finger is put down while for motions in hard mode, touch usually firstly stays at the same position, after a shot time when the selection is finished, it then begins to move (see Fig 5.5).



(a) (b)

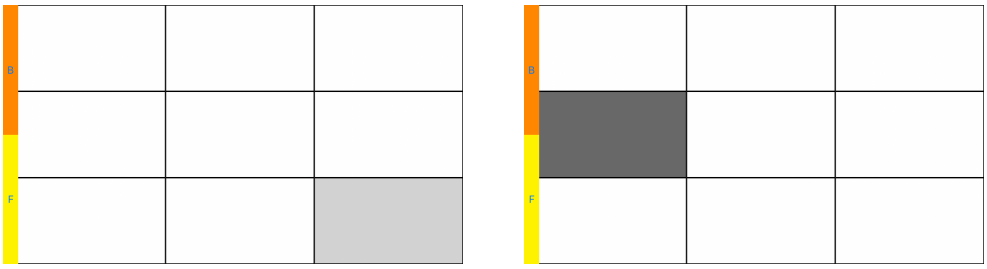
Figure 5.5: (a): Touch shifting compared with initial touch position-time plot for a user performing a light touch with right hand index finger. (b): Touch shifting compared with initial touch position-time plot for the same user performing a hard touch with the same finger.

- The phenomenon above remain the same for all fingers, only values change.

Improvement

We observed that if we put the phone on a table, people's habit of touching the screen with their index finger varies: for some people, they move the whole arm and twist to make their finger reach the target position while others prefer to keep their arm unmoved, and clicking on different positions by changing their finger's joints. In the second situation, even if the user is performing the same motion (for example, rotating the object along x/y) with the same finger, touches' force and contact surface could be different depending on the touch area, thus leading to an influence of defining the force threshold.

So we conduct this experiment to verify if the touching area has a non-neglectable influence on the choice of threshold. We created a simple application as shown in the Fig 5.6. The screen is mainly divided into $3 \times 3 = 9$ parts except the two reserved area on the left boarder. The upper part is used to clear the previous motion and re-do this motion to avoid the influence of mis-operation. The lower left border is used to finish the task because after performing required motions, the program won't end directly after performing all required tasks because we always give users some freedom to think if he/she needs to go back and redo the last motions.



(a) (b)

Figure 5.6: (a): A light gray indicates the participant needs to touch this area with a light force. (b): A dark gray indicates the participant needs to touch this area with a hard force.

The phone will be put on the table in front of the participant. Participants are not allowed to move the phone and they will be asked to touch the 9 spaces both slightly and hardly in a random order (the space to touch and the force level required are both random) with his/her index finger, first right hand and then left hand. We use the different color to indicate the task required: a light gray as Fig 5.6a means a light touch and a dark gray as Fig 5.6b means a hard touch. Users could only touch the required space, touching other space will not have any influences on the process and will not be recorded. When users have finished all the tasks required, there would be no colored areas.

Validation

After we have established the contact area based and time based model of pseudo-pressure, we conduct an experiment to see if this model can distinguish most slight and hard touches successfully.

Participants will be asked first to follow the guide to calibrate the contact radius and time thresholds for their right hand index finger. In this phrase, preceded by sliding, users are asked to press on specific area of the screen 5 times with light force and 5 times with a hard force. The threshold will be set with the middle value. And then they are asked to use this finger to perform 20 times a light mode slide and 20 times a hard mode slide. After finishing, they will be asked to change to the left hand index finger and repeat.

We will compare the prediction results with the real situation to verify if this model is good enough to replace the pressure sensing and make our interaction technique useful for old devices.

CHAPTER 6

Conclusion and perspectives

We present a pressure augmented tactile interaction way to support 3D data exploration on mobile devices. Compared with others tactile techniques, ours can separate each of the DOF, thus offering a more flexible way to interactively explore scientific data. Apart from that, our technique allows users to manipulate in 3D with a minimal number of fingers and don't need to reserve some specific areas or draw specific control tool bars to perform the manipulations, thus saving valuable screen space and reducing the occlusion problem. With our work, we expect to offer researchers a more flexible and more efficient way to explore 3D data.

Future work could focuses on the following points: looking for additional solutions to improve the use of pseudo-pressure; testing if the screen size will have an influence on people's manipulation performances because a larger screen size may change the way to interact with the touch screen; extending to a more general usage scenario because now we ask users to put the mobile device on table to interact while sometimes users would prefer to hold the device. In this case, several things will change, for example, users might prefer to use thumb instead of index finger to perform one-finger motion, and besides, users might be willing to use two hands to accomplish two-finger motions; and combining our interaction design into more complex data exploration tasks such as selection, seeding, cutting plane and more.

Publication related to this thesis

We have already submitted a poster to IEEE VIS 2017:
Xiyao Wang, Lonni Besançon, Mehdi Ammi, and Tobias Isenberg. Augmenting Tactile 3D Data Exploration With Pressure Sensing. IEEE VIS 2017, October 2017.
Extended abstract: https://hal.inria.fr/hal-01570442
Video demonstration : https://www.youtube.com/watch?v=nPW-cnMtnaM

And we're planning to write a paper to ACM CHI 2018.

Acknowledgments

I would like to thank in the first place the Aviz team members and other inria personnel for they have given me this internship's opportunity. Particularly, I want to thank my supervisors Lonni Besançon and Tobias Isenberg who guided me throughout this internship and helped me learn a lot for both this scientific area and the research process.

I would like to thank Anastasia Bezerianos of Université Paris-Sud and Yolaine Bourda of CentraleSupélec. I could not acquire related knowledge nor be capable of accomplishing this internship without their help and encouragement.

I would also like to acknowledge all those teachers, friends, and colleagues who have helped me during this internship and who have given me valuable comments on this thesis.

Finally, I must express my thanks to all participants of the experiments related to my internship. Only with their help could I evaluate our technique and finish this project.

Bibliography

[1] Ahmed Sabbir Arif, Ali Mazalek, and Wolfgang Stuerzlinger. The use of pseudo pressure in authenticating smartphone users. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS '14, pages 151–160, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[2] Ahmed Sabbir Arif and Wolfgang Stuerzlinger. Pseudo-pressure detection and its use in predictive text entry on touchscreens. In *Proceedings of the 25th Australian Computer-Human Interaction Conference: Augmentation, Application, Innovation, Collaboration*, OzCHI '13, pages 383–392, New York, NY, USA, 2013. ACM.

[3] Hrvoje Benko, Andrew D. Wilson, and Patrick Baudisch. Precise selection techniques for multi-touch screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 1263–1272, New York, NY, USA, 2006. ACM.

[4] Lonni Besançon, Mehdi Ammi, and Tobias Isenberg. Pressure-based gain factor control for mobile 3D interaction using locally-coupled devices. In Juan Pablo Hourcade, Daniel Wigdor, and Caroline Appert, editors, *Proceedings of the Annual Conference on Human Factors in Computing Systems (CHI, May 6–11, Denver, CO, USA)*, pages 1831–1842, New York, 2017. ACM.

[5] Lonni Besançon, Paul Issartel, Mehdi Ammi, and Tobias Isenberg. Usability comparison of mouse, touch and tangible inputs for 3D data manipulation. Technical Report 1603.08735, arXiv.org, March 2016.

[6] Lonni Besançon, Paul Issartel, Mehdi Ammi, and Tobias Isenberg. Hybrid tactile/tangible interaction for 3D data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):881–890, January 2017.

[7] Sebastian Boring, David Ledo, Xiang 'Anthony' Chen, Nicolai Marquardt, Anthony Tang, and Saul Greenberg. The fat thumb: Using the thumb's contact size for single-handed mobile interaction. In *Proceedings of the 14th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '12, pages 39–48, New York, NY, USA, 2012. ACM.

[8] Stephen A. Brewster and Michael Hughes. Pressure-based text entry for mobile devices. In *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '09, pages 9:1–9:4, New York, NY, USA, 2009. ACM.

[9] Xiang 'Anthony' Chen, Julia Schwarz, Chris Harrison, Jennifer Mankoff, and Scott E. Hudson. Air+touch: Interweaving touch & in-air gestures. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 519–525, New York, NY, USA, 2014. ACM.

[10] Aurélie Cohé, Fabrice Dècle, and Martin Hachet. tbox: A 3d transformation widget designed for touch-screens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 3005–3008, New York, NY, USA, 2011. ACM.

[11] Mark Hancock, Sheelagh Carpendale, and Andy Cockburn. Shallow-depth 3d interaction: Design and evaluation of one-, two- and three-touch techniques. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 1147–1156, New York, NY, USA, 2007. ACM.

[12] Mark Hancock, Thomas ten Cate, and Sheelagh Carpendale. Sticky tools: Full 6dof force-based interaction for multi-touch tables. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '09, pages 133–140, New York, NY, USA, 2009. ACM.

[13] Mark S. Hancock, Sheelagh Carpendale, Frederic D. Vernier, Daniel Wigdor, and Chia Shen. Rotation and translation mechanisms for tabletop interaction. In *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, TABLETOP '06, pages 79–88, Washington, DC, USA, 2006. IEEE Computer Society.

[14] Seongkook Heo and Geehyuk Lee. Force gestures: Augmented touch screen gestures using normal and tangential force. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 1909–1914, New York, NY, USA, 2011. ACM.

[15] Seongkook Heo and Geehyuk Lee. Forcedrag: Using pressure as a touch input modifier. In *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, pages 204–207, New York, NY, USA, 2012. ACM.

[16] Ken Hinckley, Seongkook Heo, Michel Pahud, Christian Holz, Hrvoje Benko, Abigail Sellen, Richard Banks, Kenton O'Hara, Gavin Smyth, and William Buxton. Pre-touch sensing for mobile interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2869–2881, New York, NY, USA, 2016. ACM.

[17] Tobias Isenberg. Interactive exploration of three-dimensional scientific visualizations on large display surfaces. In Craig Anslow, Pedro Campos, and Joaquim

Jorge, editors, *Collaboration Meets Interactive Spaces*, chapter 6, pages 97–123. Springer, Berlin/Heidelberg, 2016.

[18] Tobias Isenberg and Mark Hancock. *Gestures* vs. postures: 'Gestural' touch interaction in 3D environments. In Ken Anderson, Leena Arhippainen, Hrvoje Benko, Jean-Baptiste de la Rivière, Jonna Häkkilä, Antonio Krüger, Daniel Keefe, Minna Pakanen, and Frank Steinicke, editors, *Proceedings of the CHI Workshop on "The 3rd Dimension of CHI: Touching and Designing 3D User Interfaces" (3DCHI, May 5, Austin, TX, USA)*, pages 53–61, 2012.

[19] Daniel F Keefe. Integrating visualization and interaction research to improve scientific workflows. *IEEE Computer Graphics and Applications*, 30(2), 2010.

[20] Sebastian Knoedel and Martin Hachet. Multi-touch rst in 2d and 3d spaces: Studying the impact of directness on user performance. In *3D User Interfaces (3DUI), 2011 IEEE Symposium on*, pages 75–78. IEEE, 2011.

[21] Russell Kruger, Sheelagh Carpendale, Stacey D. Scott, and Anthony Tang. Fluid integration of rotation and translation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 601–610, New York, NY, USA, 2005. ACM.

[22] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The design of a gui paradigm based on tablets, two-hands, and transparency. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, CHI '97, pages 35–42, New York, NY, USA, 1997. ACM.

[23] Tomer Moscovich. Multi-touch interaction. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems*, pages 1775–1778. ACM, 2006.

[24] Menno Nijboer, Moritz Gerl, and Tobias Isenberg. Exploring Frame Gestures for Fluid Freehand Sketching. In Marc Alexa and Ellen Yi-Luen Do, editors, *Eurographics Workshop on Sketch-Based Interfaces and Modeling*. The Eurographics Association, 2010.

[25] Sebastien Pelurson and Laurence Nigay. Bimanual input for multiscale navigation with pressure and touch gestures. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, ICMI 2016, pages 145–152, New York, NY, USA, 2016. ACM.

[26] Gonzalo Ramos, Matthew Boulos, and Ravin Balakrishnan. Pressure widgets. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 487–494, New York, NY, USA, 2004. ACM.

[27] Jason L. Reisman, Philip L. Davidson, and Jefferson Y. Han. A screen-space formulation for 2d and 3d direct manipulation. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 69–78, New York, NY, USA, 2009. ACM.

[28] Christopher D Shaw. Pain and fatigue in desktop vr: Initial results. In *Proceedings of the Graphics Interface 1998 Conference, June 18-20, 1998, Vancouver, BC, Canada*, pages 185–192, June 1998.

[29] Ken Shoemake. Arcball: A user interface for specifying three-dimensional orientation using a mouse. *Proceedings of the Conference on Graphics Interface '92*, 1992.

[30] Xiyao Wang, Lonni Besançon, Mehdi Ammi, and Tobias Isenberg. Augmenting Tactile 3D Data Exploration With Pressure Sensing. IEEE VIS 2017, October 2017. Poster.

[31] Anusha Withana, Roshan Peiris, Nipuna Samarasekara, and Suranga Nanayakkara. zsense: Enabling shallow depth gesture recognition for greater input expressivity on smart wearables. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3661–3670, New York, NY, USA, 2015. ACM.

[32] Lingyun Yu, Pjotr Svetachov, Petra Isenberg, Maarten H. Everts, and Tobias Isenberg. Fi3d: Direct-touch interaction for the exploration of 3d scientific visualization spaces. tvcg, 16(6):1613–1622, 2010. doi¿ 10.1109/ tvcg.2010.157 f. keefe and tobias isenberg: Re-imagining the scientific visualization interaction paradigm 7 daniel f. kee. *IEEE, the ACM, ACM SIGGRAPH, and*, 2013.

[33] Robert C Zeleznik, Andrew S Forsberg, and Paul S Strauss. Two pointer input for 3d interaction. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 115–120. ACM, 1997.