Hand-Posture-Augmented Multitouch Interactions for Exploratory Visualization

Peng Song[1]* Xiaoqi Yan[2]* Wooi Boon Goh[2] Alex Qiang Chen[3] Chi-Wing Fu[4]

[1]University of Science and Technology of China
[2]Nanyang Technological University, Singapore
[3]Carnegie Mellon University
[4]The Chinese University of Hong Kong

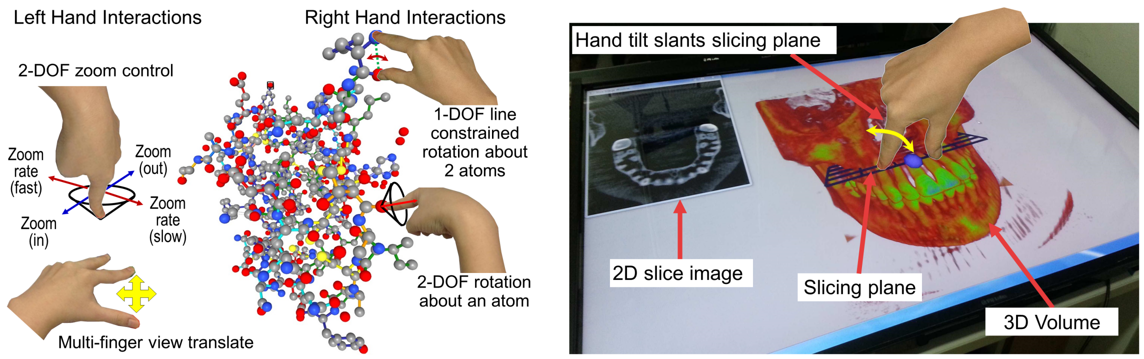


Figure 1: *Left: left- and right-hand-posture-augmented multitouch interaction designs for complex molecule visualization. Right: exploring the inside of a human skull by tilting and moving two fingers to simultaneously control both the orientation and location of a 3D slicing plane.*

Abstract

Conventional multitouch-based visualization systems mostly use just the touch information elicited from the touch surface to support the visualization applications, while rich contextual and dynamic information contained in user's hand postures above the surface remains untapped. This paper addresses this issue by proposing a family of finger-and-hand gestures that can be robustly recognized and employed over a multitouch tabletop surface. We describe how these gestures can be recognized in real-time with the use of depth sensing, and suggest several examples of how touch-based information augmented with information like hand differentiation, hand posture discrimination and finger tilt dynamics can improve conventional exploratory visualization. We present two case studies to show how the augmented gestures can be employed in different exploratory visualization scenarios more effectively.

Keywords: user interaction, visualization, hand gestures

Concepts: •**Human-centered computing** → *Interaction design;*

1 Introduction

Multitouch interaction is an effective means for data visualization as demonstrated in many existing multitouch-based visualization systems [Isenberg et al. 2013]. While these systems provide effective methods for intuitive data exploration, they employ mainly the finger contact information, i.e., the number of contact points made by a user and the contact-point movement over the surface. This contact-point-based interaction paradigm has inherent limitations:

- First, by restricting gestural designs only to contact information, contextual information provided by the user above the touch surface is ignored in the interaction design. Such information includes the user's hand posture during the contact and the nature of its above-surface dynamics.
- Second, multi-point contacts made by a single hand or performed bimanually cannot be distinguished. Neither can the left and right hands be differentiated. However, such hand differentiation and dynamic hand posture recognition allow the design of multitouch interactions that correlate more faithfully to the physical metaphor of particular operations, e.g., use a two-finger pinch to control a slicing plane, see Figure 1(right).

To resolve the above limitations, one promising approach is to employ depth sensing to acquire finger and hand postures on multitouch surfaces, so that we can incorporate the contextual knowledge of user's dynamic hand postures and bimanual status to enrich the multitouch interactions. Although some recent works [Jackson et al. 2012; Kratz et al. 2013] started to study hand gestures to enrich multitouch interactions, they only explored one or two specific gestures without examining how hand posture information can be used in different visualization scenarios. Other researchers [Wilson 2010; Dippon and Klinker 2011; Murugappan et al. 2012] proposed to detect touches and recognize hand postures on everyday surfaces by using a single depth camera. However, the touch fingers can be easily occluded by the hand or other fingers, and the touch detection accuracy is inferior as compared to off-the-shell multitouch devices.

In this work, we employ a low-cost depth camera to acquire dynamic finger-and-hand gestures and bimanual status over a multitouch tabletop surface in a controller-free manner. By this, we can *augment the multitouch interaction* and enhance the interaction design for a visualization system in terms of richness, efficiency, and extended capabilities. This paper makes the following contributions:

- First, we explore and present a family of dynamic finger-and-hand gestures that can be recognized over a multitouch tabletop surface with a low-cost depth camera. By carefully combining the 3D depth and 2D touch information, our hand-posture recognition method can improve the recognition robustness against frequent hand/finger occlusion.
- Second, we discuss and demonstrate how the dynamic finger-and-hand gestures can be applied to improve conventional multitouch-based visualization. In particular, we present two case studies on exploratory visualization: interactive visualization of molecular structures (Figure 1(left)) and interactive volume data exploration (Figure 1(right)).

*joint first authors

SA '16 Technical Briefs, December 05-08, 2016, Macao
ISBN: 978-1-4503-4541-5/16/12
DOI: http://dx.doi.org/10.1145/3005358.3005363

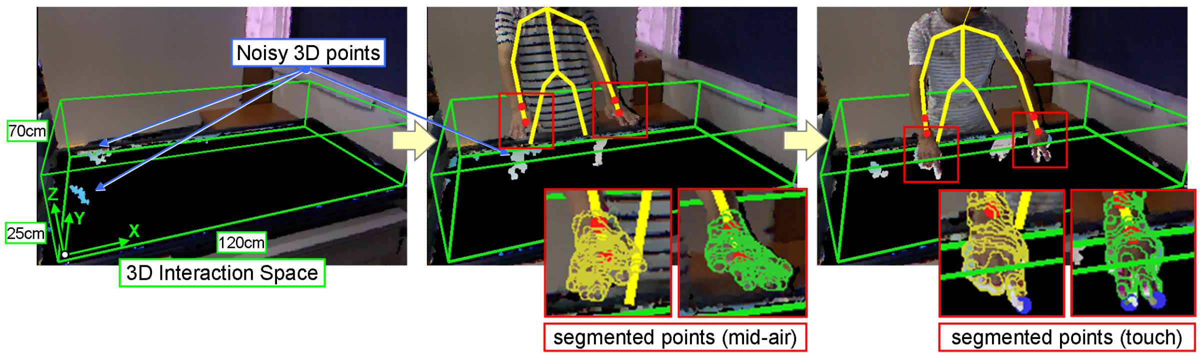


Figure 2: *Left: 3D interaction space (120x70x25 cm^3) above the multitouch surface as seen from the Kinect view. Middle: hands just enter this 3D interaction space; and we can locate the hand centers from the skeleton. Right: hands just contact the multitouch surface (blue dots).*

2 Hand-Posture Knowledge Extraction

This section presents the system setup, and describes how we segment 3D points extracted from the Kinect and recognize hand postures over a multitouch tabletop surface.

2.1 System Setup and Calibration

Hardware. Figure 3 illustrates the setup. The multitouch-enabled LCD display is a normal 55-inches TV set equipped with the PQ-Labs G4 overlay and placed horizontally at ∼0.9m above ground, with a Kinect sensor mounted next to it (∼0.7m above it) for acquiring hand actions. Both the multitouch hardware and Kinect sensor are connected to the same server PC.

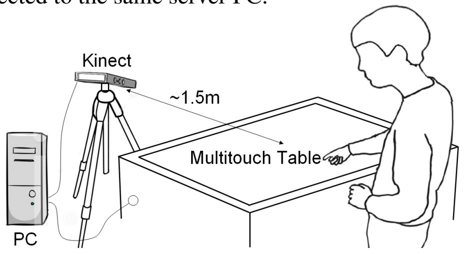


Figure 3: *Our prototype system setup.*

Software. We used the PQ-Labs multitouch API to retrieve the touch information from the multitouch hardware, and the Kinect for Windows SDK to obtain the color, depth, and skeleton information from the Kinect sensor.

Offline Calibration. To support real-time recognition and association of hand postures with multitouch, we need to deduce the transformations among the following coordinate systems: (i) the 3D physical space in Kinect view, say $\mathbb{S}_K$; (ii) the 3D physical space above the multitouch surface, say $\mathbb{S}_M$; and (iii) the 2D multitouch screen-space coordinates, say $\mathbb{S}_m$.

- *Between $\mathbb{S}_K$ and $\mathbb{S}_M$.* Similar to [Araùjo et al. 2012], we put four small red discs at the corners of the multitouch display. Then, we examine the Kinect color image to determine a group of pixels for each red disc in the Kinect view, and look at each pixel group within the Kinect depth image to compute the 3D coordinate of each red disc in $\mathbb{S}_K$. By these, we can determine the transformation between $\mathbb{S}_M$ and $\mathbb{S}_K$.
- *Between $\mathbb{S}_M$ and $\mathbb{S}_m$.* Here we tap 20 points randomly and evenly on the multitouch surface, and record their 2D contact positions, and their associated fingertip coordinates obtained from the Kinect. By using the transformation obtained from the previous calibration step, we can map the fingertip coordinates from $\mathbb{S}_K$ to $\mathbb{S}_M$. Then, by assuming that $\mathbb{S}_m$ and the horizontal XY-subspace of $\mathbb{S}_M$ are linear, we apply a least-squares fit on the points association, and determine the transformation between $\mathbb{S}_M$ and $\mathbb{S}_m$.

By the deduced transformations, we can define a *3D interaction space* above the multitouch surface, see the green rectangular box in Figure 2(left) with its corresponding XYZ axes, in the space over the multitouch surface, i.e., $\mathbb{S}_M$.

2.2 Hand-Points Segmentation

Our real-time method to recognize 3D hand postures has two key steps: i) *hand-points segmentation* and ii) *hand-posture recognition*. The first step aims to segment out the Kinect 3D points of a hand when it moves in the 3D interaction space:

First, we collect all Kinect 3D points inside the 3D interaction space. Since the glass on the multitouch surface reflects IR light, it is mostly not detected by the Kinect, but random noisy points may appear near the surface, see Figure 2. To avoid them, we ignore points that are too close (within 1cm proximity) to the multitouch surface. Though this strategy also removes the valid points related to the fingertip when it contacts the multitouch surface, we can use the multitouch data to recover the fingertip positions.

There are two cases in segmenting out the Kinect 3D points of a hand. In the first case, our hand just enters the 3D interaction space, so we extract the hand joint center, say c_0, in the body skeleton from the Kinect SDK, see Figure 2(middle) for the two red dots, which represent hand wrist and hand center. To recover a stable hand center, we further perform the followings: i) collect Kinect 3D points within a virtual sphere (radius: 15cm) centered at c_0; ii) determine the point (among the point set) that is the farthest away (-Y) from the user (it is usually the fingertip), say p_{tip}; iii) find a subset of points within a radius of 20cm from p_{tip}, and take their centroid as the hand center, say c_1. In the second case, where our hand moves in the 3D interaction space, we use the previous hand center, i.e., c_k ($k{>}0$), to construct the virtual sphere to collect Kinect 3D points in substep (i) above, and then follow the same procedure to compute the next hand center.

With the above procedure, we can obtain a stable hand center subject to changes in hand postures, e.g., one-finger to multi-finger pointing. It is also more stable compared to that from the body skeleton (from Kinect API) since the skeleton hand center can be easily deviated by noise, and may shift backwards when our hand contacts the touch surface, see Figure 2 (right).

2.3 Hand-Posture Knowledge

From the multitouch hardware, we can obtain and then map each finger touch point from $\mathbb{S}_m$ to $\mathbb{S}_M$. From the hand segmentation, we can know the center of each hand in $\mathbb{S}_M$. Hence, we can associate each finger touch point with the user's left/right hand by checking the proximity. Therefore, even though the contacting finger is occluded by the hand or other fingers in the Kinect view, we can still recon-

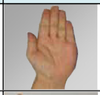
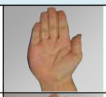
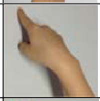
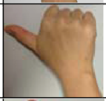
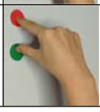
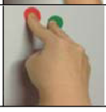
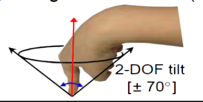
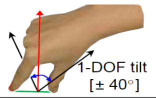
Category	Discriminated Hand Posture Type			
Hands		Left Hand		Right Hand
1-finger		Index finger 2-DOF tilt		Thumb No tilt
2-finger		Pinch/Grip 1-DOF tilt		Point/Cut No tilt
Tilt angle range about normal (red) to contact point or line (green)  2-DOF tilt [± 70°]  1-DOF tilt [± 40°]				

Table 1: *Summary of finger-and-hand gestures that can be robustly extracted, see associated tilt degree-of-freedom (DOF) and range.*

struct a hand skeleton from the hand center to each finger contact. With this, we are capable of recognizing the following categories of hand gestures (see alongside with Table 1):

- *Left and right hand discrimination.* First, we can easily discriminate *left and right hands* in the 3D interaction space by checking the Kinect skeleton data.
- *One-finger touch group.* Second, we can differentiate hand gestures with one-, two-, or multi-finger contacts on the multitouch surface. In the one-finger touch group, we consider only the *thumb* and *forefinger* (see Table 1(2nd row)) because performing one-finger touches with other fingers is ergonomically uncomfortable. To differentiate between the thumb and forefinger, we obtain the forearm vector from the Kinect skeleton and calculate the vector from the hand center to the touch point in $\mathbb{S}_M$. Then, we project these two vectors onto the multitouch surface, and compute the angle in-between for differentiating between a thumb touch and a forefinger touch.
- *Two-finger touch group.* For two finger touches from the same hand, we consider only the *pinch* and *cut* gestures (see Table 1(3rd row)) since performing others are ergonomically uncomfortable. To differentiate between pinch and cut, we construct a vector from the hand center to the middle of the two touch points, project such vector and the forearm vector onto the multitouch surface, and then compute the angle in-between to differentiate between a cut and a pinch.
- *Hand dynamic: finger joystick.* Besides static hand gestures, we can also recognize hand-gesture dynamics. Our *finger joystick* design allows a contacting forefinger to dynamically tilt about its fixed touch point on the multitouch surface to realize a joystick controller. To achieve this, we compute the vector from the touch point to the hand center in $\mathbb{S}_M$ per frame, and apply the vector's rotation angles for 2-DOF control.
- *Hand dynamic: two-finger pinch tilt.* We also explored the tilt dynamics of the two-finger pinch gesture. In detail, when the two fingers contact the multitouch surface, we determine the line joining the two touch points. Then, when we perform the pinch tilt with the touch points being fixed, we can measure the rotation of the hand center about the line, and use the rotation angle for 1-DOF control.

3 Multitouch Interaction Design with Depth Sensing

This section presents several interesting multitouch interactions designs made possible by incorporating some of the dynamic finger-and-hand gestures listed in Table 1.

Left and right hand discrimination. Many geographical visualization systems employ standard view exploration operations like pan, zoom, and rotate. When an appropriate view of the working canvas has been reached, editing operations such as annotate, draw, erase, etc., can be applied to desired elements on the canvas. Current multitouch interaction design typically approaches this problem by using a mode-switch button to inform the application if the subsequent touch gestures are for view exploration or for editing. Since our system supports left and right hand discrimination, this feature can be employed to increase the flexibility and efficiency of multitouch interaction design by associating groups of operations with different hands. By merely remembering the assigned functional roles of each hand (similar to table manners that dictate our fork and knife hands), users can seamlessly transit between exploration and editing operations without explicit mode switching, see Figure 4.

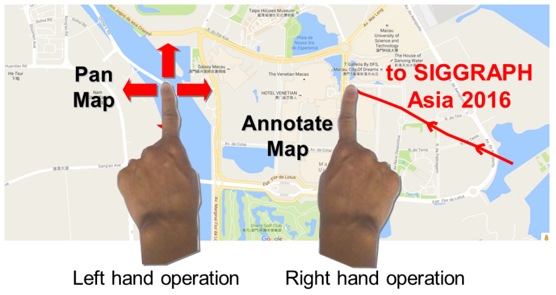


Figure 4: *The ability to distinguish between hands allows one to seamlessly pan a map with the left finger and annotate with the right finger without any mode switching operation.*

Forefinger/thumb gesture discrimination. To remove the need for explicit mode switching when moving from one operation to another, different hand postures of the same hand can also be mapped to different multitouch operations. By drawing analogue to common physical actions, certain hand-posture gestures can be naturally mapped to certain application operations. For example, one would write on the sand with the forefinger, but use the side of the thumb to erase a crease. Based on such physical analogue, one can assign the draw (annotation) operation to a single forefinger touch and an erase operation to a thumb touch, see Figure 5.

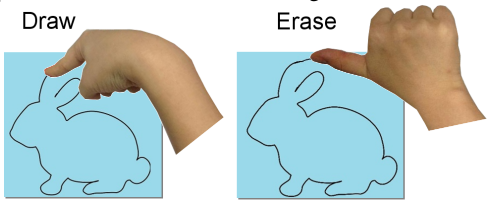


Figure 5: *Examples of augmented multitouch interactions for 1-point touch operations. The draw (left) and erase (right) operations have an intuitive mapping to a forefinger and a thumb-based gesture.*

One-finger joystick. Interactive adjustment of parameters with a continuous value range is a common task in many visualization applications, e.g., fine-grain color selection, tuning thresholds, etc. Here, the slider bar widget is often employed as a means for users to adjust parameters. However, using such widget requires finger

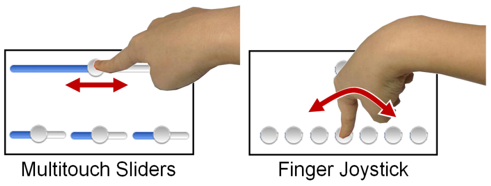


Figure 6: *Dynamic finger tilt about a contact point (right) is used to adjust continuous parameters. Unlike the traditional multitouch slider bar (left), the finger joystick design is more space friendly.*

movement across the screen, and this is often constrained by the lack of screen space in the user interface layout. The ability to vary dynamics of hand posture about a stationary contact point like that provided by the one-finger joystick gesture can be used to overcome the limitations of the slider bar widget, see Figure 6(right). Since it requires no movement on the touch surface, a GUI with dense and space-friendly arrangement of variable inputs can be realized.

Two-finger pinch tilt. We extend the idea of dynamic hand tilt to a two-finger touch implementation, where control of a 1-DOF tilt operation constrained about two arbitrary points can be easily implemented and differentiated from a 2-DOF tilt provided by a one-finger joystick gesture. Figure 7 illustrates an application of the two-finger pinch tilt dynamic gesture.

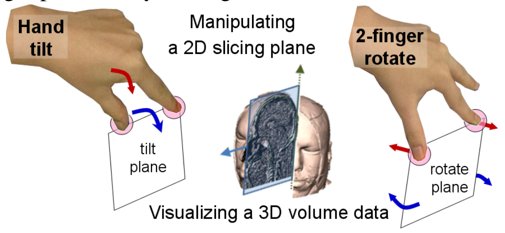


Figure 7: *Example of how two-finger hand tilt and rotate gestures can be used to interactively control a 2D slicing plane for exploratory visualization of a 3D CT volume data.*

The dynamics of the hand tilt and the movement of the finger contact points can be combined to produce interesting multitouch interaction designs, allowing one to instantaneous vary multiple continuously changing parameters as shown in Figure 8. By setting appropriate angular thresholds, we can also map a high or low hand tilt to different multitouch operations without explicit mode switching.

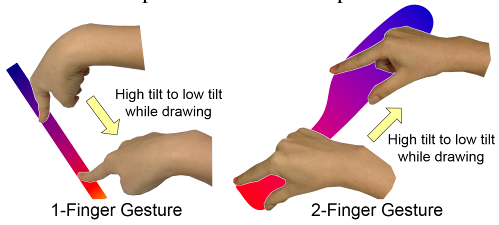

High tilt to low tilt while drawing
High tilt to low tilt while drawing
1-Finger Gesture
2-Finger Gesture

Figure 8: *Examples of how instantaneous hand tilt of a user can control the changing color of a line (left) and the changing tonal gradation between two points with varying width (right).*

4 Case Studies: Visualization Applications

This section presents two case studies, showing how our proposed finger-and-hand gestures can be employed in visualization applications (see the supplementary video for the demonstration).

4.1 Interactive Visualization of Molecular Structures

When exploring the visualization of complex molecular structures, two basic types of exploratory operations are often required. The first relates to the virtual camera view, including view translation, orientation, and zoom control. We assign these multitouch operations to the user's left hand, see Figure 1(left). The 2-DOF finger joystick gesture allows the user to simultaneously control the zoom direction and rate to explore the visualization quickly and precisely.

The second type relates to the manipulation of the virtual 3D model. For our purpose, the ability to rotate the model about a specific atom and a constrained rotation about a line joining selected atoms are desired. The finger joystick gesture is employed in the former case to give a 2-DOF rotation about an atom selected (touched) by the user's forefinger. For the latter case, a two-finger pinch gesture allows the user to select two atoms that make up a line, about which the 1-DOF hand tilt offers a constrained rotation, see Figure 1(left).

4.2 Interactive Volume Data Exploration

The second case study is the visualization of 3D volume data obtained from CT and MRI scans. Figure 1(right) shows the internal structure of a human skull being explored with a 2D sliced image according to a 3D virtual slicing plane. By using our proposed one-handed two-finger pinch gesture, we can simultaneously control the orientation and location of the slicing plane: position from the contact points made by the two fingers, and orientation by the tilt angle of the gesture. As a result, we can interactively explore tiltable 2D sliced images over the 3D volume by simply using two fingers.

5 Comparative User Studies

Two comparative user studies were performed to evaluate the quantitative improvement when conventional multitouch interaction designs are augmented with knowledge of the proposed hand gestures. In user study 1, the finger joystick interaction design is compared to a conventional slider bar widget to evaluate the efficiency and ease of use when adjusting a variable parameter to a random target value. Results indicate that performing the task with the finger joystick provides significant improvement in task completion time. In user study 2, the user's task is to modify marked squares by applying a color or erase operation using either conventional multitouch interaction designs or multitouch designs augmented with the proposed gestures. Results show that the augmented designs are more efficient and less frustrating. The supplementary material provides more details.

6 Conclusion

This paper presents a family of dynamic finger-and-hand gestures that can be used to enhance and enrich multitouch interaction designs on tabletop surfaces for exploratory visualization. We present two case studies to show how these gestures can be applied for exploratory visualization and two user studies to show their efficiency.

Limitation. Current setup requires extended floor-space behind the multitouch tabletop, which can be problematic in confined situations.

Acknowledgments

This work is supported by the Singapore MOE Tier-1 Grant (RG 29/11), and the Chinese University of Hong Kong strategic recruitment fund and direct grant (4055061).

References

Araùjo, B. R. D., Casiez, G., and Jorge, J. A. 2012. Mockup Builder: direct 3D modeling on and above the surface in a continuous interaction space. In *Graphics Interface*, 173–180.

Dippon, A., and Klinker, G. 2011. KinectTouch: Accuracy test for a very low-cost 2.5D multitouch tracking system. In *ITS*, 49–52.

Isenberg, P., Isenberg, T., Hesselmann, T., Lee, B., von Zadow, U., and Tang, A. 2013. Data visualization on interactive surfaces: A research agenda. *IEEE CGA 33*, 2, 16–24.

Jackson, B., Schroeder, D., and Keefe, D. F. 2012. Nailing Down Multi-Touch: anchored above the surface interaction for 3D modeling and navigation. In *Graphics Interface*, 181–184.

Kratz, S., Chiu, P., and Back, M. 2013. PointPose: Finger pose estimation for touch input on mobile devices using a depth sensor. In *ITS*, 223–230.

Murugappan, S., Vinayak, Elmqvist, N., and Ramani, K. 2012. Extended Multitouch: Recovering touch posture and differentiating users using a depth camera. In *UIST*, 487–496.

Wilson, A. D. 2010. Using a depth camera as a touch sensor. In *ITS*, 69–72.