EUROGRAPHICS 2012 / P. Cignoni and T. Ertl
(Guest Editors)

Volume 31 (2012), Number 2

Multitouch Gestures for Constrained Transformation of 3D Objects

Oscar Kin-Chung Au[1] Chiew-Lan Tai[2] Hongbo Fu[1]

[1]City University of Hong Kong [2]Hong Kong University of Science and Technology

Abstract

3D transformation widgets allow constrained manipulations of 3D objects and are commonly used in many 3D applications for fine-grained manipulations. Since traditional transformation widgets have been mainly designed for mouse-based systems, they are not user friendly for multitouch screens. There is little research on how to use the extra input bandwidth of multitouch screens to ease constrained transformation of 3D objects. This paper presents a small set of multitouch gestures which offers a seamless control of manipulation constraints (i.e., axis or plane) and modes (i.e., translation, rotation or scaling). Our technique does not require any complex manipulation widgets but candidate axes, which are for visualization rather than direct manipulation. Such design not only minimizes visual clutter but also tolerates imprecise touch-based inputs. To further expand our axis-based interaction vocabulary, we introduce intuitive touch gestures for relative manipulations, including snapping and borrowing axes of another object. A preliminary evaluation shows that our technique is more effective than a direct adaption of standard transformation widgets to the tactile paradigm.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Interaction Techniques— H.5.2 [Information Interfaces and Presentation]: User Interfaces—

1. Introduction

Most of existing 3D modeling packages are designed for single-point 2D input devices, and typically support two mechanisms of 3D manipulations in a single system: unconstrained manipulations and constrained manipulations of 3D objects. The former allows 3D transformations along arbitrary axes and is typically useful for coarse manipulations (e.g., various virtual trackballs [CMS88, Sho92, HSH04a] for 3D orientation manipulations). The latter supports fine-grained 3D manipulations and is often achieved with 3D transformation widgets [Bie87], which act as visual handles for 1D (axis) or 2D (planar) constraints (Figure 2(a)). As touch-screens have very different input properties from mouse- and pen-based input systems [Mos09, WBP*11], traditional 3D manipulation tools have to be re-designed to adapt to the tactile paradigm.

A few multi-touch user interfaces for unconstrained 3D manipulations [HCC07, HtCC09, RDH09, MCG10] have been proposed. They employ multi-touch input to enable simultaneous manipulation of several degrees of freedom (DOF). However, there is little research aiming at constrained 3D manipulations for multitouch interaction. Existing solutions [SSB08, CDH11] mainly attempt to tackle the problem of how to make the transformation widgets more easily selectable and manipulatable with a fingertip by redesigning the form and configuration of the widgets. However, like standard transformation widgets, their redesigned widgets directly associate different transformation tools with different widget components, thus requiring careful touch positioning to trigger appropriate tools.

Existing solutions tend to visually combine as many widgets as possible for simultaneous support of multiple transformation tools (e.g., Figure 2). Instead, motivated by the well-known Rotate-Scale-Translate multitouch gestures for 2D content interaction [KH11], we intend to *visually* disassociate transformation tools from widgets and transfer the manipulation power of those tools to multitouch gestures. Such indirect control of the widgets not only helps to minimize visual clutter but also enables an easy, seamless control of manipulation constraints (i.e., axis or plane) and modes (i.e., translation, rotation or scaling). This avoids tedious

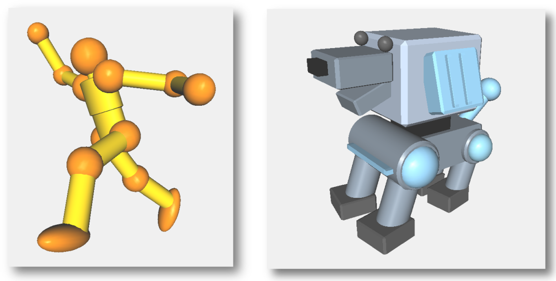
Figure 1: *Examples of models created using our multitouch manipulation techniques within a few minutes.*

editing steps such as mode/tool switching in traditional modeling scenarios.

Since constrained manipulations favor independent control of translations, rotations and scale, we show that two-finger gestures suffice for the determination of all the operations needed for axis-constrained manipulations: axis selection and transformation mode are determined by the orientation and movement of two touched points, respectively. Therefore, a single multitouch action (i.e., a single pair of finger-down and finger-up with touch movement) is sufficient to determine the desired transformations (i.e., translation, rotation or scaling) with respect to a desired axis. Such action does not require any complex transformation widget but candidate axes (Figure 2(b)) provided for visualization rather than direct manipulation, which is challenging with imprecise touch-based input. Similar 2-finger gestures are also designed for planar constraints to perform translation/scaling in plane. To further expand our axis-based interaction vocabulary, we introduce intuitive touch gestures for relative manipulations, including snapping and borrowing axis constraints from another object.

With only a small set of gestures and no widgets needed for direct manipulation, our button-free approach supports most manipulating capabilities found in commercial 3D modeling interfaces. A preliminary evaluation shows that users can efficiently use our interface to construct moderately complex 3D models or scenes given only a short training period (e.g., Figure 1). It is also demonstrated that our technique is more effective than a straightforward integration of standard transformation widgets to the tactile paradigm.

2. Related Work

Constrained transformations of 3D objects have been proved effective for fine-grained manipulations and are typically achieved through 3D transformation widgets [Bie87, CSH*92, SZH94], which are ubiquitous in 3D applications [GP95, Han97, CSB*05] and commercial modeling packages (e.g., Maya, Blender, 3DS Max). Although the visual design and implementation details of transformation widgets might vary, their functionality is largely the same: a widget typically contains a three-axis canonical frame as its basic component for independent control of axis-constrained manipulations; multiple DOF controllers for translation, rotation and scaling can be displayed at the same time [SSB08]. A successful manipulation relies on a precise selection of constraints and DOF controllers.

Traditional 3D transformation widgets are mainly designed for use with keyboard and single-point devices (e.g., mouse or pen) and rely on a large set of keyboard shortcuts or mode-switching buttons. Although they can be directly integrated into touch-based interfaces, their design is based on a tool-switching metaphor, which conflicts with the tool-free philosophy of the tactile paradigm. Further, small or crowded widgets in standard interfaces are difficult to operate due to the well-known fat finger problem (i.e., fingertip-blob input resolution of touch devices).

Few techniques [SSB08, CDH11] have been proposed to adapt constrained 3D manipulations to tactile interaction. To support the increasingly popular sketch-based modeling paradigm, Schmidt et al. [SSB08] present transient 3D widgets defined and controlled by stroke-based command gestures. Their interface is designed for single-point input only. Cohé et al. [CDH11] introduce *tBox*, a box-like widget for touch-screens whose faces are used for rotation and edges for translation and scaling. However, multi-touch input has not been fully exploited. The above two solutions mainly focus on the redesign of the form and configuration of the widgets such that the widgets can better tolerate imprecise touch-based input. The usage of the new widgets is largely the same as that of standard transformation widgets except that mouse-based input is replaced with pen- or tap-based input. Our work is inspired by those approaches, especially Schmidt et al.'s, but intends to exploit the rich information of multitouch input to minimize interface complexity and user interaction.

Although 3D manipulations with multiple inputs were studied a decade ago [ZFS97, BK99], such interface has not been popular until the rapid development of multitouch display devices. Recently, a few techniques [HCC07, HtCC09, RDH09, MCG10, MCG11] have been proposed to explore how to map multitouch inputs to 6 DOF of object control (translation and rotation). They focus on unconstrained 3D manipulations (e.g., free-form rotation), though sometimes certain constraints (e.g., screen-space constraint) may apply. Since constrained and unconstrained 3D manipulations have their own strength, we believe that our technique is complementary to and can co-exist with the unconstrained 3D multitouch techniques. For instance, we can determine the mode of interaction based on where the interaction happens, e.g., applying our axis-based interface for *off*-object interaction (given the location-independent nature of our interface) and unconstrained manipulations for *on*-object interaction.

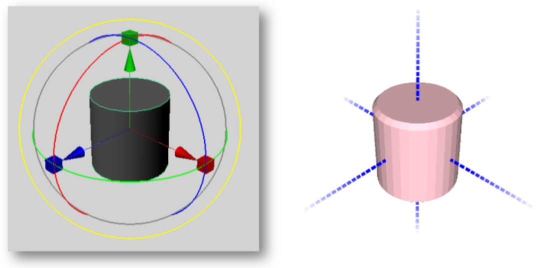

Figure 2: *(a) A standard 3D transformation widget in Maya, (b) our interface with canonical axes which are for visualization and not touchable handles.*

3. Design Rationale

Our design of multitouch interface for constrained 3D manipulations is motivated by the following observations on existing transformation widgets. The defining characteristic of a widget is providing a single interaction point for the *direct manipulation* of a given kind of data [Wik]. A transformation widget typically contains multiple visual handles (for different manipulation purposes e.g., Figure 2), requiring every single click to be exactly within the corresponding handles. A direct adaption of standard transformation widgets to the tactile paradigm is thus problematic due to fingertip-blob input resolution and occlusion by fingers [VCC*09]. Although the visual redesign of the widgets for touch screens may mitigate the problems [SSB08, CDH11], like standard widgets, they still require visual attention virtually at every touch, since every tap must be precisely and directly operated on proper elements of the widgets. Therefore, the visual redesign of transformation widgets faces a dilemma, especially for small-sized touch screens like those with smartphones: aiming to simultaneously display as many handles as possible while striving to make every handle easily manipulatable.

Our design goal for constrained 3D manipulations is largely orthogonal to the design of traditional transformation widgets: we intend to discard all visual widget handles in our UI design and move from the tedious widget manipulations to the more efficient gesturing mode. As multitouch inputs contain rich orientation (relative positions of multitouch points) and transformation (motion paths of touch points) information, they suit the need of 3D manipulations which require multi-dimension inputs. We thus introduce a small set of easy-to-use multitouch gestures to replace the traditional widget handles. The design of our gestures has been driven by the following set of guidelines:

Widgetless To avoid precise widget manipulation, no visual handles should be provided for direct manipulation.

Scale and location independence For gesturing to be effective, multitouch gestures must be recognized independent of scale and location [ZK03].

Seamless A constrained 3D manipulation involves three basic steps: constraint selection (axis or plane), transformation mode selection (translation/rotation/scaling), and transformation manipulation. These steps should be performed seamlessly with a single gesture.

Simplicity To make gestures easy to master, they should be simple and intuitive to use. The number of gestures should be small. Additionally, single-hand gestures should be designed for most operations, since in many scenarios the other hand is needed for other tasks (e.g., holding the touch device).

Context-awareness To make a small set of simple gestures sufficient to encompass all the available interactions provided by standard transformation widgets, reuse of gestures should be allowed and they can be interpreted as context-aware operations.

Here is a brief overview of our system. To enable scale and location independence, our gestures are recognized from the orientation and motion paths of the touching points rather than their contact locations. More specifically, a constraint is chosen from the candidate constraints of a selected object (e.g., Figure 3) based on the orientation of two fingers that touch the screen. The transformation mode and the magnitude of transformation are then determined based on the movement of the two fingers with respect to the chosen constraint. Such process of user interaction is seamless, involving only a single multitouch action (i.e., a single pair of finger-down and finger-up with touch movement).

4. Axis-Constrained Manipulations

In this section, we first introduce our multitouch gestures for manipulations constrained to a single axis, and then discuss manipulations constrained to a plane.

4.1. Axis Selection

Candidate axes. Like standard transformation widgets, our method maintains a predefined set of candidate axes for the object selected for editing. Such candidate axes can be obtained either by simple methods (e.g., object-oriented bounding boxes, face normal and principal curvature directions) or by more complex shape analysis (e.g., reflective symmetry axes [PSG*06], upright orientations [FCODS08]). How to define such a set of axes is not our focus. We simply use a pre-defined set of axes for each scene object. Besides the object axes, we also include the axes of the world and screen canonical frames (Figure 3(f)). Each set of axes is displayed in a different color (e.g., blue for object-space, red for world-space, green for screen-space) to make them easily distinguishable. They are all drawn at the centroid of the selected object so that their relative projected orientations (angular difference) are immediately visible.

By projecting each candidate axis to the view plane, we

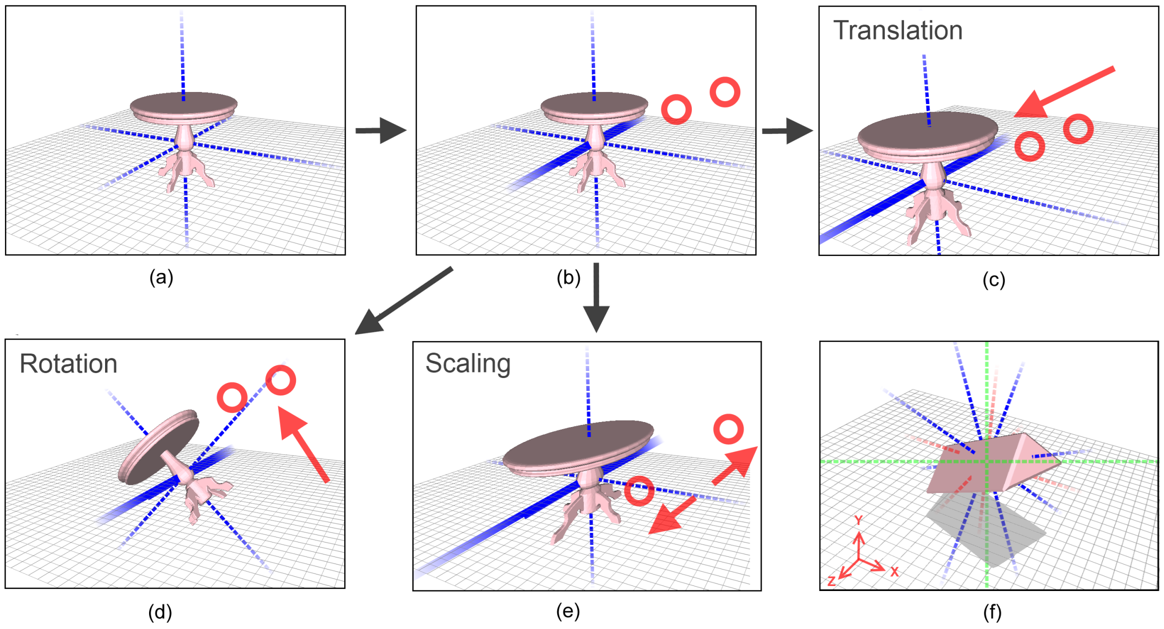

Figure 3: *Multitouch gestures for axis-based transformation manipulations. (a) Select an object for manipulation; its predefined axis set is then displayed. (b) Select an axis of interest (thick blue line) based on the orientation defined by two touch points (red circles). (c)-(e) Two-finger pan along the axis is recognized as axis-constrained translation, two-finger pan perpendicular to the axis is rotation about that axis, and two-finger pinch along the axis is axis-constrained scaling. The entire process is seamless, involving only a single multitouch action. (f) Manipulations constrained by world-space axes (red) or screen-space axes (green) are also supported.*

get a 2D line for each axis. Therefore it is sufficient to use a two-point touch to specify an axis of interest: the candidate axis with the smallest angular difference between its projected orientation and the *orientation* determined by the locations of the two touch points is used as the axis constraint for manipulation (e.g., the highlighted axis in Figure 3). We only require the two touch points roughly in the same direction of a desired axis. Such selection scheme is tolerant to imprecise touch-based input caused by the fat finger problem (unless the two fingers are placed too close to each other). Note that candidate axes are displayed for visualization only and are not touchable handles for direct manipulation. The two fingers can be placed anywhere on the screen, making the gesture independent of location and scale. From viewpoints where an axis is nearly parallel to the viewing direction, we disallow its selection since translation or scaling along such axis is generally difficult to predict.

A similar strategy for axis selection is used in [SSB08]. However, the usage of their stroke-based gestures is less seamless than ours, since their gestures are used to create translation/rotation widgets only, which then have to be manipulated using standard click-and-drag interaction. Our interface requires only a single multitouch action (two-finger touch and movement) to perform a constrained 3D manipulation.

The user can always re-touch the screen with two fingers to re-select an axis of interest. However, when two or more axes (e.g., from different sets of axes) are nearly coincident from the current viewpoint, picking a desired axis with a casual two-finger touch becomes more difficult. It is observed that object-space axes are more often the desired ones than other axes such as world-space axes and screen-space axes. Therefore, we give higher priority to the selection of object-based axes, by simply doubling the angular difference between the orientation of the two touch points and any candidate axis not in the set of object-based axes. Although the user can always change the camera view to one where the desired axis is more distinguishable, it is interesting to explore a mechanism for selection refinement (e.g., possibly based on the idea of timeout delimiter [HBRG05]).

4.2. Transformation Manipulations

Once an axis of interest is selected, the user keeps the two fingers touched and moves them for transformation manipulation. Our system automatically determines the transformation mode according to the movement of the two fingers relative to the chosen axis (along or perpendicular), and calculates the constrained transformation according to the displacement.

Specifically, the axis-constrained translation is achieved by the gesture of two-finger pan *along* the selected axis (Figure 3(c)). The axis-constrained rotation is also achieved by the gesture of two-finger pan, but *perpendicular to* the selected axis (Figure 3(d)). Both the amount of translation and the amount of rotation are calculated according to the average distance between the current contact points and the initial contact locations. Our context-aware gesture is intuitive to use, since it is analogous to the movement patterns with standard transformation widgets (i.e., moving along the axis for translation and perpendicular to the axis for rotation, given the explicitly specified axis and manipulation mode). The axis-constrained scaling is designed as the gesture of two-finger pinch *along* the selected axis (Figure 3(e)), no matter if the two fingers are pinched together or apart. The amount of scaling is defined by the ratio between the current and initial distances of the two contact points. For all these operations, the transformation is applied to the object with respect to its centroid. Manipulations with respect to another origin (pivot) are discussed in Section 5.2. Note that two-finger pan and two-finger pinch are commonly used for 2D Rotate-Scale-Translate multitouch interaction [KH11], though their purpose is different in that context: two-finger pan for unconstrained translation and two-finger pinch for uniform scaling.

Gesture recognition. The gestures used in our system can be easily recognized according to the initial orientation of the two contact points and the subsequent movement relative to the selected axis. As common with existing gesture matching methods [SSB08], for robustness, the recognition is not immediately invoked until the contact points have displacements larger than a given threshold. Specifically, we adopt the *snap-with-buffer-zone* approach of Nacenta et al. [NBBW09], which supports separability of manipulations without obvious lagging or jumping. This approach also allows the user to cancel the current manipulation by moving the touching fingers back to the initial touch positions (i.e., no manipulation applied within the buffer zone). Once a gesture is recognized, the corresponding transformation mode is activated and remain unchanged until the fingers leave the touchscreen. We also note that, by adapting the idea of magnitude filtering proposed by Nacenta et al., it is possible to switch between different transformation modes in a single touch action. However, we did not implement this feature to avoid sudden mode change caused by unintended finger movement.

4.3. Plane-Constrained Manipulations

A plane constraint allows object manipulations within a given 3D plane which can be specified by selecting an axis orthogonal to the plane. To distinguish from the axis selection mode, we designate two-finger tap and touch (a tap followed by immediate touch) as the plane selection mode. A semi-transparent plane is displayed to visualize the selected plane constraint, as shown in Figure 4.

We implemented two common plane-constrained operations: translation and uniform scaling in plane. Similar to single-axis translation, translation in plane is performed by two-finger pan. The movement of the fingers is projected to the two defining axes on the plane to determine the amount of translation with respect to each axis. Uniform scaling in plane (no scaling along plane normal) is achieved by two-finger pinch, with the scale determined by the distance change between the two contact points (see Figure 4). Note that rotation in plane is redundant since it is equivalent to rotation with respect to the plane normal. To exit the plane-constrained mode the user simply lifts the fingers from the touchscreen.

5. Relative Manipulations

The operations discussed in the previous section are theoretically sufficient to construct any 3D transformation. However, all the transformations are always constrained to an axis or plane anchored at the object being manipulated. This can be quite cumbersome in practice, especially for the manipulations of multiple objects (e.g., object assembly). In this section we introduce interaction techniques which use constraints from other scene objects, thus supporting manipulations relative to any arbitrary object.

5.1. Active Snapping

Snapping provides an effective way to achieve precise relationships between objects. Snapping can be achieved either *passively* [BS86, OS05] or *actively* [PJBF03, SSB08]. Passive snapping relies on automatic detection of guiding lines or surfaces and thus often works only for nearby objects only. Active snapping is typically achieved with a user-specified path connecting two objects, which can be far away. The two objects are then automatically glued such that their position and/or orientation are aligned.

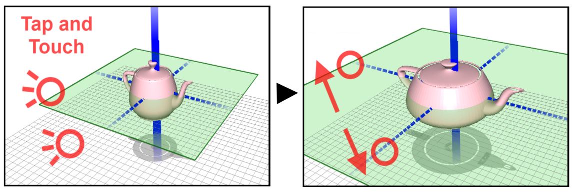

Figure 4: *Plane-constrained manipulation. A two-finger tap and touch with orientation along the normal direction of the plane (thicken line) selects the plane constraint. Subsequent two-finger pinch uniformly scales the object in the plane.*

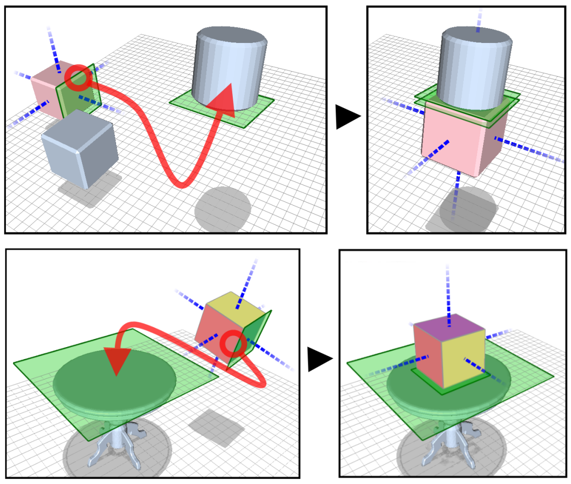

Figure 5: *Active snapping examples. The user draws a freeform curve from the source object to the target object. The tangent directions at the two ends of the curve determine the two axes to be aligned with while snapping the source object to the target object. Note that the snapping planes need not be visible (top example). The extra rotation DOF after axis alignment is determined by maximizing the original visible faces of the source object (bottom example).*

Given the imprecise nature of touch interactions, we design a new gesture for active snapping, which is primarily based on the shape of motion path rather than precise location of touch points. Similar to [SSB08], we use the predefined set of candidate axes to define snapping proxies, making the interface applicable smooth surfaces as well. For each axis, we pre-define a parallel pair of bounding planes perpendicular to the axis as candidate snapping planes (e.g., those in green in Figure 5). To activate snapping, the user draws a freeform touch path connecting the selected object to the object to which it will be snapped (Figure 5). Our system then uses the tangent directions at the starting point and ending point of the path to automatically determine the snapping planes at the selected object and target object, respectively. Specifically we compare the projected axes of the object with the starting tangent and ending tangent of the 2D path. The snapping plane whose axis direction is most similar to the tangent direction at the starting (ending) point is chosen as the source (target) object. Unlike active snapping methods in [PJBF03, SSB08], which relies on accurate position specification of the path, our strategy allows more flexibility since it relies only on the tangent directions at the two ends of the drawn curve. This allows snapping of objects even when the snapping faces are invisible at the current camera view, greatly reducing the camera control effort.

The selected object is transformed such that the center and the normal of the snapping planes are aligned. There remains one DOF for the source object, i.e., rotation about the aligned normal of the snapping planes. We use a simple strategy to determine the best rotation based on keeping as much of the originally visible faces visible. This is done by computing the dot product of each axis of the source object and the viewing direction, and selecting the alignment that minimizes the sum of the differences between the dot products before and after the snapping (see Figure 5, bottom). Note that users always have the freedom to adjust this rotation angle after snapping. Overall, our active snapping operation avoids the otherwise tedious manipulations of the objects, giving a convenient interface for multiple object manipulations.

5.2. Axis and Center Borrowing

Setting up arbitrary axes or pivots is typically tedious and time-consuming [SSB08]. However, it is often the case that the desired manipulation constraint of an object exists in the candidate constraints of another scene object. Our system

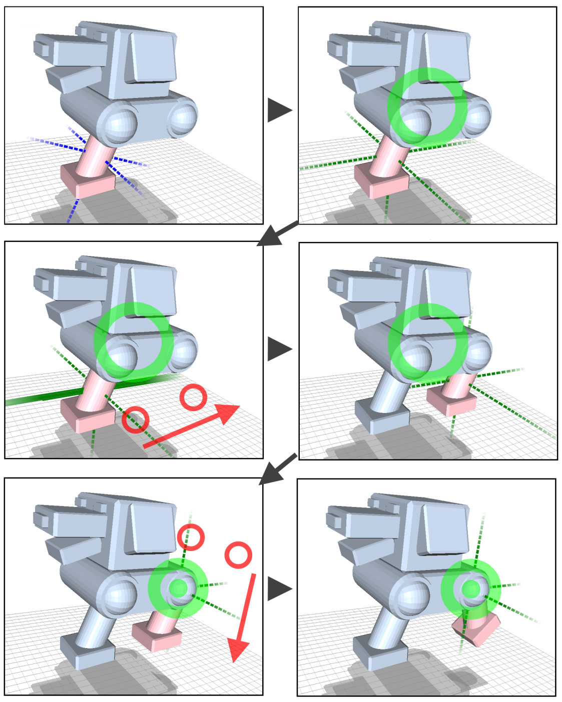

Figure 6: *Axes and center borrowing example. The leg is selected for manipulation. A one-finger long press on the body borrows the axis set of the body for manipulating the leg. A two-finger touch selects an axis constraint and the front leg is duplicated and translated along the chosen axis. (Object duplication is activated with a three-finger gesture not shown here, see Section 6.) Another one-finger long press toggles to the mode of borrowing the center of the body as well for rotating the hind leg.*

supports simple interaction to borrow axes and center of another object for manipulations, allowing relative manipulations of an object with respect to *any* object.

To use the axes of another object for manipulating a selected object, the user simply marks the desired axis provider using *single-finger long press*. The new set of axes are then displayed centered at the selected object, which can then be manipulated with respect to the new axis set (see Figure 6). Another single-finger long press on the axis provider toggles to the mode of borrowing not only the axis set but also the center of the provider for manipulation. Since the center of the provider is being used, the borrowed axes remain drawn at the provider. In Figure 6, the hind leg is rotated with respect to the center and an axis of the body. Note that this is similar to pivot objects in other interactive manipulation systems (e.g., [SSB08]). However, our system allows any object to serve as an axis-center provider and does not explicitly distinguish between editable objects and pivot objects.

6. Supporting Operations

In this section, we briefly describe the supporting operations, which together with the manipulation operations introduced earlier form a complete multitouch interface for constrained 3D manipulations. Our gestures below are mainly for a proof of concept and subject to change when adapting our gestures for constrained manipulations to existing multitouch systems. Some of the operations below rely on a realtime finger and palm registration technique [AT10], which automatically determines which hand of the user is touching and which contact points belong to which fingers whenever three or more contact points are detected.

Camera control. A perspective camera is adopted. For camera orbit, we use one finger movement (starting from empty space) to trigger a virtual trackball interface [HSH04b]. Panning and zooming are implemented using five-finger slide and pinch gestures, respectively.

Object selection. For selecting or deselecting an object, we use a one-finger tap on the object. A one-finger double-tap centers the tapped object in the screen space. To select multiple objects, two-finger tap on another object to add it to the current selection. A tap on the empty space deselects all the objects.

Uniform scaling. Given an object of interest (i.e., being selected), five-finger pinch with the left hand is recognized as uniform scaling.

Screen-space rotation. When any of the screen-space axes is selected, a two-finger *rotation* is interpreted as screen-space rotation, rotating the object about the viewing direction.

Object duplication. Duplication operation is activated with three-finger pan. Depending on whether one or two objects are being selected, the duplicated object either undergoes axis-constrained translation (translation axis determined by the three-finger movement) or is directly transformed by the relative transformation between the two selected objects, respectively.

7. Preliminary Evaluation and Discussion

We conducted pilot experiments to investigate the effectiveness of our interface design.

Apparatus. The experiments were conducted on an Acer Iconia Tab notebook that is equipped with a 10.1-inch touch screen, supports multitouch input up to 4 points (here all supporting operations involving 5-finger gestures were invoked without tracking the little finger contact point) and tracks finger contacts at approximately 100 Hz. The notebook was placed on a table (with height of 74 cm), with the screen facing upward, and participants were free to adjust its location and orientation.

Participants. Six university students from computer science department participated (5 males and 1 female with a mean age of 22). Most of them regularly use multitouch applications such as image and map browsing tools on mobile devices. None of them has extensive experience with graphics modeling software.

Task I. We compared our widgetless interface with its widget-based version (see Figure 7) in order to measure the performance difference between widget and widgetless paradigms. To provide similar manipulations for both interfaces, we put the widgets on the candidates axes. Dragging the ball widget along the axis performs axis-constrained translation. Dragging the bar widget orthogonal to the axis rotates the object about the axis. For axis-constrained scaling, the user first touches both ball widgets on the axis and pinch them along the axis. To evaluate the touch precision factor for the widget-based UI, we define the acceptance radius as the maximum distance of the detected touch point to the nearest widget center and tested two different acceptant distances (30 and 45 pixels).

For performance comparison, we chose the 3D docking task introduced by Zhai and Milgram [Zha98] since 3D manipulation always involves the specification of positions, orientations and scale of 3D objects, which can be considered as a sequence of docking tasks with respect to target locations in the user's mind. Participants were required to scale, translate, or rotate a source object to a target location and orientation in the virtual 3D space as quickly and accurately as possible. During the test, participants were allowed to perform camera control to change the view point. The source object is a man-made object consisting of shape primitives (we use the elk model in Figure 7) with clearly distinguishable orientation. For each interface, each participant was asked to perform 15 trials, docking the same source object, initially centered on screen, to 15 different targets. A trial is considered to be completed if the object is transformed

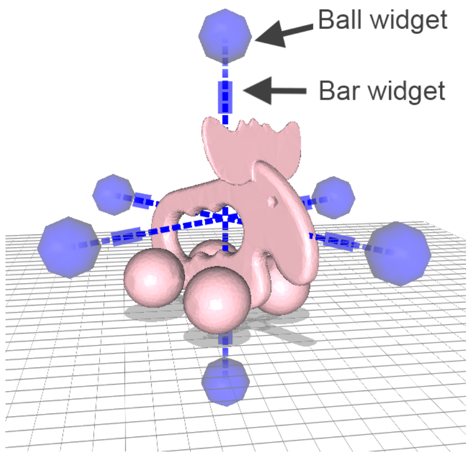

Figure 7: *Widget-based UI used in preliminary evaluation.*

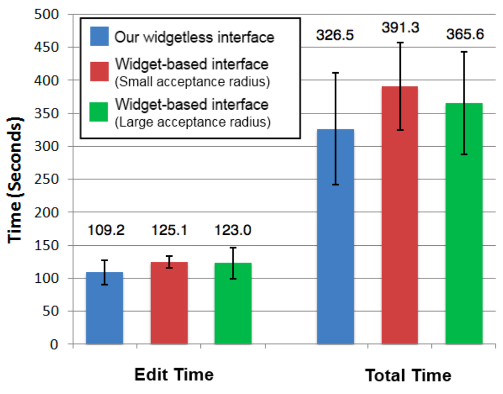

Figure 8: *Trial performance mean and standard deviation for different user interfaces.*

close enough to the target location, orientation and scale, determined by comparing the difference of the transformation matrices of the source and target objects against a given threshold. Individual participants randomly used any of the interfaces first. Before the test, the subjects were first briefed on both interfaces and practiced until they felt comfortable (this lasted from a few minutes to a dozen of minutes).

We recorded the overall task completion time and the editing time (time spent on translating, rotating and scaling the object, excluding camera manipulations) of all trials as well as the number of each type of manipulation operations used. An informal questionnaire was designed to collect the participants' feedback about the controllability and acceptability of both interfaces.

Results. We performed an analysis of variance (ANOVA) on the performance data for the last 10 trials (i.e., the first 5 trials were discarded to reduce learning effects). The overall completion time and the editing time are shown in Figure 8. Overall, our interface was not slower than the widget-based version. For the editing tasks, the performance improvement of our widgetless interface over the widget-based interface was statistically significant ($F(1,16) = 4.07$, $p = 0.0056$). A main possible reason was that widget-based interface requires precise control of the widgets for manipulation operations, but the editing object was often occluded by the controlling hand and fingers, thus causing slower manipulation. These data support our belief that multitouch input with orientation information is an efficient alternative to traditional widget-based interfaces for 3D object manipulations on tactile input devices. In terms of the total completion time there was no significant difference between our interface and the widget-based interface ($F(1,16) = 2.68$, $p = 0.116$), possibly because participants spent much more time on camera control and browsing operations, which are needed with both interfaces. It is interesting to note that there was no significant difference between widget-based interface with different acceptance radii in terms of both the editing time ($F(1,10) = 0.050$, $p = 0.826$) and total completion time ($F(1,10) = 0.449$, $p = 0.515$). We speculate that the small radius (diameter 1cm) we used was already sufficiently large for accurate touching of the widgets.

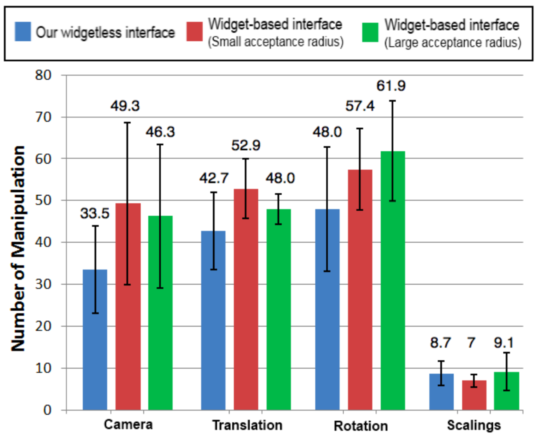

Figure 9: *Average number of manipulations and standard deviation for different user interfaces.*

Figure 9 shows that our interface can help to significantly reduce the overall numbers of 3D manipulations on tactile input devices (camera: $F(1,16) = 5.20$, $p = 0.033$; translation: $F(1,16) = 6.24$, $p = 0.020$; rotation: $F(1,16) = 5.00$, $p = 0.036$); there was no significant reduction for scaling ($F(1,16) = 0.223, p = 0.641$) possibly due to the small sampling size, since in our preliminary evaluation, only 3 out of 10 recorded trials required scaling manipulations). Feedback from the participants indicated that the widgets and the manipulated object sometimes obstruct each other, requiring

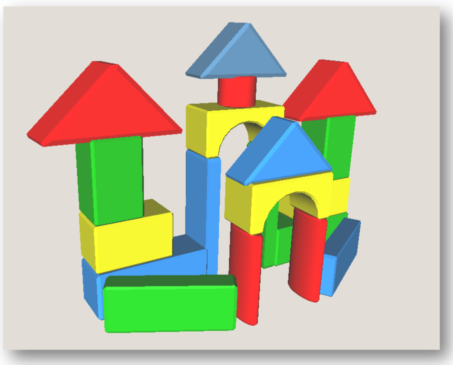

Figure 10: *Participants were asked to construct this castle from given building blocks using our interface.*

more camera manipulations to find better views to manipulate the object and thus also leading more transformation operations. Finally we noticed that there was no significant difference between the numbers of different types of manipulation when different acceptance radii were used for the widgets-based interface.

Task II. Participants were asked to use our interface to perform an assembly task of constructing a castle from a given set of building blocks. They were shown the image in Figure 10 and asked to construct one as similar to that as possible. This was an informal study, without any accuracy requirements. Each participant's feedback was collected after completion.

Results. The results of this informal experiment indicated that our interface was effective and has good controllability for basic 3D manipulations. All participants could successfully construct the castle within 5 to 10 minutes. Participants with richer experience of using multitouch devices took relatively less time to complete the assigned task. All the participants reported that after the learning session, they could easily place the objects at desired locations. The operation most preferred by participants was active snapping, since it can be achieved without considering the actual transformation for snapping, allowing automatic alignment of objects in a single view without changing viewing direction.

Participants also reported that they sometime forgot to select the source object before using active snapping, ending up camera orbit instead of snapping. This is a common limitation of context-aware interfaces since users need to remember which operations are supported under the current context. In contrast, we noticed that users did not have this problem with axis-based transformations, apparently because the candidate axes of an object are displayed only after the object is selected. This extra visualization information gives an implicit hint to the users about the operations order. It is possible to assign another gesture to the active snapping operation (e.g., tap and drag with one-finger), which would then eliminate the requirement of first selecting the source object before snapping. However, to keep the user interface simple and consistent as well as avoiding accidental object manipulations, we retain the object selection requirement before snapping.

8. Conclusions

We presented a small but effective set of multitouch gestures for constrained manipulations of 3D objects. Our interface makes use of the extra input bandwidth of multitouch screens and delegates the manipulation power of standard transformation widgets to the multitouch gestures. Our interface has no widgets as touchable handles, and thus successfully tolerates imprecise touch-based input. It supports a seamless control of constraint and transformation manipulation using a single multitouch action. The introduction of active snapping and axes/center borrowing further enhances the controllability of our system. The effectiveness of our user interface is supported by the pilot experiments.

There are a few possible directions for future study. One is to enrich the visualization and improve interface design in order to provide more hints for context-aware operations, thus reducing the occasions of incorrect operations. It is also interesting to integrate our gestures to existing multitouch systems supporting unconstrained manipulations of 3D objects. Note that as our current interface design mainly focuses on single-hand operations, there is room for designing more complex operations using two-hand gestures. Finally, as a pilot study, our results are positive and encouraging. However, a more comprehensive and formal evaluation is still needed for future development.

Acknowledgement

The authors would like to thank Lu (Arthur) Chen for his help in implementation and video editing, Michael Brown for video narration. We appreciate the helpful comments from the anonymous reviewers. This work was supported by a grant from the Innovation & Technology Fund of Hong Kong (project ITS/117/09). Oscar Au was supported in part by a grant from CityU (No. StUp7200266). Hongbo Fu was supported in part by grants from CityU (No. SRG7002533), and the HKSAR Research Grants Council (No. 9041562).

References

[AT10] Au O. K.-C., Tai C.-L.: Multitouch finger registration and its applications. In *OZCHI '10* (2010). 7

[Bie87] Bier E. A.: Skitters and jacks: interactive 3D positioning tools. In *I3D '86* (1987), pp. 183–196. 1, 2

[BK99] Balakrishnan R., Kurtenbach G.: Exploring bimanual camera control and object manipulation in 3d graphics interfaces. In *CHI '99* (1999), pp. 56–62. 2

[BS86] BIER E., STONE M.: Snap-dragging. *ACM SIGGRAPH Computer Graphics 20*, 4 (1986), 233–240. 5

[CDH11] COHÉ A., DECLE F., HACHET M.: tbox: A 3D transformation widget designed for touch-screens. In *Proceedings of the 2011 annual conference on Human factors in computing systems* (2011), ACM, pp. 3005–3008. 1, 2, 3

[CMS88] CHEN M., MOUNTFORD S. J., SELLEN A.: A study in interactive 3-d rotation using 2-d control devices. In *SIGGRAPH '88* (1988), pp. 121–129. 1

[CSB*05] COLEMAN P., SINGH K., BARRETT L., SUDARSANAM N., GRIMM C.: 3D screen-space widgets for non-linear projection. In *GRAPHITE '05* (2005), pp. 221–228. 2

[CSH*92] CONNER B. D., SNIBBE S. S., HERNDON K. P., ROBBINS D. C., ZELEZNIK R. C., VAN DAM A.: Three-dimensional widgets. In *I3D '92* (1992), pp. 183–188. 2

[FCODS08] FU H., COHEN-OR D., DROR G., SHEFFER A.: Upright orientation of man-made objects. *ACM Trans. Graph. 27*, 3 (2008), 42. 3

[GP95] GRIMM C., PUGMIRE D.: Visual interfaces for solids modeling. In *UIST '95* (1995), pp. 51–60. 2

[Han97] HAND C.: A survey of 3D interaction techniques. In *Computer Graphics Forum* (1997), vol. 16, pp. 269–281. 2

[HBRG05] HINCKLEY K., BAUDISCH P., RAMOS G., GUIMBRETIERE F.: Design and analysis of delimiters for selection-action pen gesture phrases in scriboli. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), ACM, pp. 451–460. 4

[HCC07] HANCOCK M., CARPENDALE S., COCKBURN A.: Shallow-depth 3d interaction: design and evaluation of one-, two- and three-touch techniques. In *CHI '07* (2007), pp. 1147–1156. 1, 2

[HSH04a] HENRIKSEN K., SPORRING J., HORNBÆK K.: Virtual trackballs revisited. *Visualization and Computer Graphics, IEEE Transactions on 10*, 2 (2004), 206–216. 1

[HSH04b] HENRIKSEN K., SPORRING J., HORNBÆK K.: Virtual trackballs revisited. *IEEE Trans. Vis. & Comput. Graph. 10*, 2 (2004), 206–216. 7

[HtCC09] HANCOCK M., TEN CATE T., CARPENDALE S.: Sticky tools: Full 6DOF force-based interaction for multi-touch tables. In *ITS '09* (2009), pp. 145–152. 1, 2

[KH11] KNOEDEL S., HACHET M.: Multi-touch rst in 2D and 3D spaces: Studying the impact of directness on user performance. In *3DUI* (2011). 1, 5

[MCG10] MARTINET A., CASIEZ G., GRISONI L.: The effect of dof separation in 3d manipulation tasks with multi-touch displays. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology* (2010), ACM, pp. 111–118. 1, 2

[MCG11] MARTINET A., CASIEZ G., GRISONI L.: Integrality and separability of multi-touch interaction techniques in 3d manipulation tasks. *IEEE Trans Vis Comput Graph* (2011), Accepted for publication. 2

[Mos09] MOSCOVICH T.: Contact area interaction with sliding widgets. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology* (2009), ACM, pp. 13–22. 1

[NBBW09] NACENTA M. A., BAUDISCH P., BENKO H., WILSON A.: Separability of spatial manipulations in multi-touch interfaces. In *GI '09* (2009), pp. 175–182. 5

[OS05] OH J., STUERZLINGER W.: Moving objects with 2D input devices in cad systems and desktop virtual environments. In *Proceedings of Graphics Interface 2005* (2005), Canadian Human-Computer Communications Society, pp. 195–202. 5

[PJBF03] PEREIRA J., JORGE J., BRANCO V., FERREIRA F.: Calligraphic interfaces: Mixed metaphors for design. *Interactive Systems. Design, Specification, and Verification* (2003), 154–170. 5, 6

[PSG*06] PODOLAK J., SHILANE P., GOLOVINSKIY A., RUSINKIEWICZ S., FUNKHOUSER T.: A planar-reflective symmetry transform for 3D shapes. *ACM Trans. Graph. 25*, 3 (2006), 549–559. 3

[RDH09] REISMAN J. L., DAVIDSON P. L., HAN J. Y.: A screen-space formulation for 2D and 3D direct manipulation. In *UIST '09* (2009), pp. 69–78. 1, 2

[Sho92] SHOEMAKE K.: Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface* (1992), vol. 92, pp. 151–156. 1

[SSB08] SCHMIDT R., SINGH K., BALAKRISHNAN R.: Sketching and composing widgets for 3D manipulation. *Computer Graphics Forum 27*, 2 (2008), 301–310. 1, 2, 3, 4, 5, 6, 7

[SZH94] STEVENS M. P., ZELEZNIK R. C., HUGHES J. F.: An architecture for an extensible 3D interface toolkit. In *UIST '94* (1994), pp. 59–67. 2

[VCC*09] VOGEL D., CUDMORE M., CASIEZ G., BALAKRISHNAN R., KELIHER L.: Hand occlusion with tablet-sized direct pen input. In *Proceedings of the 27th international conference on Human factors in computing systems* (2009), ACM, pp. 557–566. 3

[WBP*11] WIGDOR D., BENKO H., PELLA J., LOMBARDO J., WILLIAMS S.: Rock & rails: extending multi-touch interactions with shape gestures to enable precise spatial manipulations. In *CHI '11* (2011), pp. 1581–1590. 1

[Wik] WIKIPEDIA: GUI widget. `http://en.wikipedia.org/wiki/GUI_widget`. 3

[ZFS97] ZELEZNIK R. C., FORSBERG A. S., STRAUSS P. S.: Two pointer input for 3D interaction. In *I3D '97* (1997), pp. 115–120. 2

[Zha98] ZHAI S.: Quantifying coordination in multiple DOFs movement and its application to evaluating 6 DOF input devices. In *SIGCHI* (1998), pp. 320–327. 7

[ZK03] ZHAI S., KRISTENSSON P.: Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2003), ACM, pp. 97–104. 3