DOI: 10.1111/j.1467-8659.2012.03197.x
Pacific Graphics 2012
C. Bregler, P. Sander, and M. Wimmer
(Guest Editors)

Volume 31 (2012), Number 7

Two-Finger Gestures for 6DOF Manipulation of 3D Objects

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

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

Abstract

Multitouch input devices afford effective solutions for 6DOF (six Degrees of Freedom) manipulation of 3D objects. Mainly focusing on large-size multitouch screens, existing solutions typically require at least three fingers and bimanual interaction for full 6DOF manipulation. However, single-hand, two-finger operations are preferred especially for portable multitouch devices (e.g., popular smartphones) to cause less hand occlusion and relieve the other hand for necessary tasks like holding the devices. Our key idea for full 6DOF control using only two contact fingers is to introduce two manipulation modes and two corresponding gestures by examining the moving characteristics of the two fingers, instead of the number of fingers or the directness of individual fingers as done in previous works. We solve the resulting binary classification problem using a learning-based approach. Our pilot experiment shows that with only two contact fingers and typically unimanual interaction, our technique is comparable to or even better than the state-of-the-art techniques.

1. Introduction

Multitouch displays offer extra input bandwidth and thus afford more effective solutions for 3D manipulation of objects, which often involves the control of six Degrees of Freedom (6DOF): three for independent translation along x-, y-, and z-axes and three for independent rotation about x-, y-, and z-axes [RDH09]. However, since touch inputs are inherently two-dimensional and each contact finger provides only two DOF, there is no straightforward mapping between the combined DOF of multitouch inputs and the 6DOF required for 3D manipulation tasks.

Focusing on large-size touch tables or walls, all existing multitouch interaction techniques for full 6DOF 3D manipulation establish such mappings in terms of the number of fingers used for the interaction and/or the directness of each finger (i.e., whether or not the finger is in contact with the object) [MCG12] (Figure 1). Generally, at least three touch fingers (i.e., each for two DOF) are needed to support 6DOF control [HtCC09, RDH09]. However, three or more fingers appear too crowded to operate on touch screens of small size (e.g., common smartphones) and thus might seriously block the view. Moreover, many cost effective touch sensors (e.g., based on wave-guided infrared, surface acoustic wave, etc.) support only at most two touch points. The introduction of the directness of individual fingers makes it possible to achieve 6DOF control with one and two touch interaction [MCG12]. However, the performance of direct and indirect touch on an object is apparently sensitive to the size of the object on the screen and the rest of the screen, respectively.

This paper presents a new multitouch interaction technique for full 6DOF manipulation, which always uses only two-finger operations and is independent of the directness of fingers. Our technique thus can be used for either direct manipulation or indirect manipulation. The key idea is to carefully encode the movement of two fingers, which is interpreted as one of the following four two-finger gestures: panning, pinching, swiveling, and pin-panning (Figure 2). Full 6DOF manipulation can then be easily achieved by the following mapping: panning for xy-translation, pinching for z-translation, swiveling for z-rotation, and pin-panning for xy-rotation (see the accompanying video).

Two-finger panning, pinching, and swiveling gestures have been widely used for 2D manipulation and can be performed as a single *integral gesture*, often referred as the Rotate-Scale-Translation (RST) gesture in the context of 2D manipulation [HCV*06]. This integral gesture typically involves two *moving* fingers, which we characterize as Mode $2m$ (Figure 1). In contrast, the pin-panning gesture involves one moving finger and one fixed finger, corresponding to Mode $1m+1f$. Therefore, how to robustly separate Mode $2m$ from Mode $1m+1f$ is a key challenge to address in our

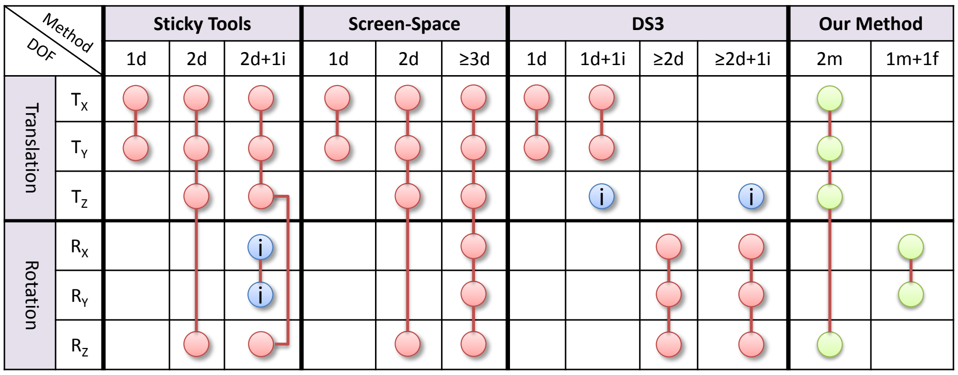


Figure 1: *The existing approaches (*Sticky Tools *[HtCC09],* Screen-Space *[RDH09], and* DS3 *[MCG12]) control the 6DOF based on the number of touch fingers and their directness (e.g., 2d+1i: two direct fingers with one indirect finger). In contrast, our technique always uses only two fingers and involves two modes determined by the movement of the two fingers (e.g., 1m+1f: one moving finger with one fixed finger). Circles connected with a single line represent the DOF that are integrated. Here, it is assumed that z-axis points toward the user and x-axis/y-axis lie on the screen plane.*

work. We solve this binary classification problem using a supervised learning approach. We show that our learning approach is able to achieve a high classification rate of around 95%.

We conducted a pilot experiment to evaluate the performance of our technique compared to the state-of-the-art. We demonstrate that with only two contact fingers and no requirement on the directness of fingers, our technique outperforms the techniques of *Screen-Space* [RDH09] and *DS3* [MCG12] in terms of task completion time, and is comparable to the *Sticky Tools* technique [HtCC09], which requires at least three fingers for 6DOF control. We also show that bimanual interaction is highly demanded for all the existing techniques while a single hand is sufficient for the use of our technique, making our technique more applicable to different interaction scenarios.

2. Related Work

Traditional 3D manipulation techniques are mainly designed for single-point input devices and let users control one or two DOF in a single interaction and six DOF sequentially, which is particularly beneficial for 3D manipulation constrained to a certain axis or plane [SSB08]. However, such techniques typically require explicit changes of mode, which is not very convenient for complex transformations. Recently, some of these techniques have been adapted to the tactile paradigm [CDH11, ATF12] but remain focus on separate DOF control.

Extra interaction bandwidth afforded by multitouch devices allows multiple DOF to be controlled in parallel and thus improves the speed of complex manipulations. However, how to optimize the mapping between the 6DOF to be controlled for 3D manipulation and the DOF afforded by the multitouch inputs is a challenging problem, mainly due to the mismatch between the separable input structure of multitouch devices and the integral nature of 3D manipulation [MCG10]. Several mapping methods have been proposed to control the 6DOF in a partially or fully integrated manner, as summarized in Figure 1.

The *Screen-Space* technique proposed by Reisman et al. [RDH09] allows 6DOF control in a fully integral manner by simultaneously solving for the translation and rotation parameters via a least-squares optimization. The optimization is formulated to achieve direct manipulation such that the same points on an object being manipulated always stay underneath the same fingertips. This requires at least three fingers in contact with the object (Mode $\geq 3d$), though different subsets of 6DOF manipulations (e.g., xy-translation) can be achieved by one or two fingers only (e.g., Mode $1d$). While direct manipulation is indeed natural and intuitive, direct touch on the object might sometimes be tricky especially if the object is small on the screen and/or consists of irregular, thin parts.

Following the *Shallow-Depth* technique [HCC07], which supports only five DOF manipulation, Hancock et al. [HtCC09] introduce the *Sticky Tools* technique for 6DOF control, based on a partial DOF integration. The four degrees of freedom for rotation around z-axis and translation along x-, y-, z-axes (Mode $2d$) are integrated and controlled by a two-finger RST-like gesture. When the third finger is in contact with the touch screen (not necessarily the object), the control of the other two DOF in rotation can be performed in an indirect and integral way, but separately from the four DOF controlled by the first two fingers (Mode $2d + 1i$). We adopt this DOF separation strategy but control the separated two subsets of DOF based on the movement of two contact fingers instead of the number of fingers in contact with the screen and their contact order.

Martinet et al. [MCG12] present another 6DOF manipula-

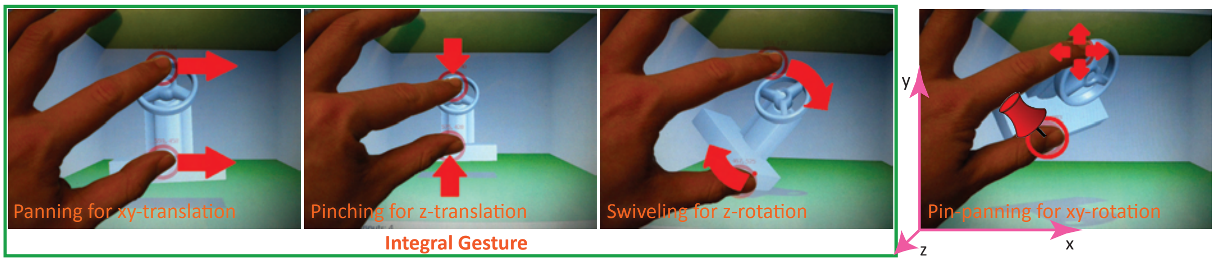

Panning for xy-translation
Pinching for z-translation
Swiveling for z-rotation
Pin-panning for xy-rotation
Integral Gesture
y
x
z

Figure 2: *The four degrees of freedom, i.e., for xy-translation, z-translation, and z-rotation, are controlled by an integral RST-style gesture (for Mode $2m$). The two degrees of freedom for xy-rotation are controlled by pin-panning gesture in an integral way (for Mode $1m+1f$), but separately from the other four DOF. Either one of the two fingers can be used to pin down at a fixed position and the other finger is then used for panning.*

tion technique called *DS3* (Depth-Separated Screen-Space), which separates the DOF for translation from that for rotation. The separated control is achieved by examining the directness of individual fingers. More specifically, two or more *direct* fingers indicate the mode for controlling the three DOF for rotation in an integral way (Mode $\geq 2d$), achieved based on a variant of the *Screen-Space* method. A single direct finger controls *xy*-translation and a second indirect finger controls *z*-translation (Mode $1d+1i$). Their pilot experiment shows that *DS3* is more efficient than *Screen-Space* and *Sticky Tools* on a specific 3D peg-in-hole task, where a hole is always fixed at the middle of a 3D rectangular parallelepiped of moderate size compared to the touch screen. However, no user study is carried out to evaluate the impact of the size of the object on the screen, which would apparently influence the performance of direct/indirect touch on an object (e.g., direct touch becomes difficult when an object appears too small).

Various technologies have been introduced for detecting hand and tracking gesture input, allowing the manipulation of 3D objects in virtual reality applications with non-tactile hand gestures. However, these systems usually require extra hardware support such as multiple-camera configuration [SK10], color glove [WP09] or depth sensor [CCL*11]. Such specific hardware requirements limit the usability and practicability of these systems. Our work aims at designing an effective interface for 3D object manipulation using commonly available multitouch input devices, from handheld smartphones to desktop computers equipped with touch screens or pads.

3. Design Rationale

We design our multitouch technique for 6DOF manipulation, driven by the following set of guidelines:

Supporting different sized multitouch screens An ideal technique should be consistently applicable to different sizes of common touch screens, ranging from a few inches (e.g., 3.5-inch screen of iPhone 4S) to dozens of inches (e.g., 82-inch multitouch collaboration wall by Perceptive Pixel). Existing techniques are mainly designed for touch screens of large sizes.

Two-finger operations only Two-finger operations are preferred for the following reasons. First, many multitouch devices still can support at most two touch points due to their cost-effective choice of touch hardware. Second, the more fingers touching the screen, the more hand occlusion is introduced [VCC*09], posing a serious problem especially to small touch screens (e.g., common smartphones). Thirdly, two-finger operations are simple and easy to operate by a single hand. Lastly, single-touch interaction is often already reserved for common operations like object selection and camera control in a complete multitouch system for 3D manipulation.

Supporting unimanual interaction Multitouch manipulation with a single hand is preferred in many scenarios [NBBW09], e.g., when the touch screen is too small for two hands, or when the other hand has to hold the touch device or another input device. Although the previous techniques for 6DOF manipulation do not necessarily require the use of multiple hands, bimanual interaction is highly demanded by those techniques, as we will explain later.

Independent of fingers' directness A technique should ideally be adaptive to both direct manipulation and indirect manipulation. This can make the performance of such a technique largely insensitive to the size of touch screen and/or the size of the object being manipulated on the screen.

Seamless operations Explicit mode switching, for example by clicking mode buttons, should be avoided to preserve the freehand nature of the interactions. Ideally, operations for selecting or switching between different DOF subsets should perform in a seamless manner.

4. Two-Finger Gestures and Operations

In this section we introduce our proposed two-finger gestures and their corresponding operations for 6DOF manipulation. The algorithm for gesture recognition will be presented in Section 5.

To make our technique applicable to most multitouch devices, we only assume that each finger in contact with the touch screen gives us only the 2D position information of the contact point. With such information, it is well known that 4DOF control can be easily achieved by a two-finger RST-style gesture (e.g., see Mode $2d$ for *Sticky Tools* in Figure 1). Our technique also adopts this RST-style gesture to control 4DOF. The remaining challenge here is in how to carefully control the other two DOF with the same two fingers, without depending on the directness of individual fingers, or the name and associated hand of each finger in contact.

Our key idea for full 6DOF control with only two contact fingers is to classify the gestures into two manipulation modes by examining the moving characteristics of the contact fingers. Note that the RST-style gesture constitutes three primitive gestures: panning, pinching and swiveling, all of which typically involve the *simultaneous* movement of both fingers. Motivated by this observation, we define two modes based on whether both fingers are moving (Figure 1):

Mode $2m$: This mode involves two moving fingers, which control four DOF by an RST-style gesture.

Mode $1m+1f$: This mode involves one fixed finger and one moving finger, which together control the remaining two DOF by a *pin-panning* gesture.

Below we briefly describe the mapping between the 3D operations and our gestures, as illustrated in Figure 2. The detailed implementation will be discussed in Section 6. Without loss of generality, we assume that the touch screen is aligned with the xy-plane of the virtual 3D space, and z-axis is parallel to the viewing direction and points toward the user. We also assume that we are working with a perspective camera. Like the previous works, our technique focuses on 6DOF control for 3D rigid transformation of an object. Let $(T_x, T_y, T_z, R_x, R_y, R_z)$ denote the involved transformation parameters, where T_x, T_y, and T_z are for translation along x-, y-, and z-axes respectively, and R_x, R_y, and R_z are for rotation about x-, y-, and z-axes respectively.

In Mode $2m$, the four DOF corresponding to xy-translation, z-translation, and z-rotation are controlled by the RST-style gesture, i.e., the two-finger gestures of panning, pinching, and swivelling. In Mode $2m$ both fingers must move, which is not required but is typical with the traditional RST-style gesture. With this requirement, for pinching/swivelling, the two fingers typically move/rotate with respect to the midpoint between the two initial contact points. In Mode $1m+1f$, one finger is fixed and the other finger moves. This pin-panning gesture operates the xy-rotation. Since there are only two DOF involved, they are essentially controlled by the moving finger using a trackball-like interface.

The gestures of panning, pinching and swivelling are inherently integrated. Therefore, it is not necessary to apply xy-translation, z-translation, and z-rotation operations sequentially to achieve complex transformations. In fact, the four DOF involved can be controlled simultaneously with a single compound gesture, as shown in the accompanying video. The two DOF for xy-rotation are controlled separately from the above-mentioned four DOF. Fortunately, as shown in the video, the transition between the two modes is seamless and it is even possible to control full 6DOF using a single multitouch action (i.e., a single pair of finger-down and finger-up with touch movement).

Our technique naturally supports unimanual interaction. Due to the opposable nature of the thumb, it is highly recommended to use the thumb and one of the other four fingers from the same hand to perform two-finger operations (e.g., the thumb and index finger as shown in Figure 2). Whenever a pin-panning gesture is intended, *either* the thumb *or* the other finger can be used to pin down at a fixed position. Clearly, bimanual interaction can also be easily supported by using one finger from each hand.

5. Gesture Recognition

In this section we present a robust algorithm for gesture recognition. Since panning, pinching and swiveling are inherently integrated, it is unnecessary to recognize these gestures individually. Therefore, our key problem is to distinguish the RST-style gesture (for Mode $2m$) from the pin-panning gesture (for Mode $1m+1f$), which is essentially a *binary classification* problem.

Classification simply by checking finger immobility does not work for two reasons. First, in practice it is difficult to keep one finger completely fixed when the other finger is moving, especially when both of the fingers are from the same hand. Second, imprecise multitouch input might cause slightly different positions even for a stationary contact finger.

When the user interacts with a multitouch screen, the device continuously generates a stream of low-level touch events corresponding to *touch-down*, *touch-move* and *touch-up*. Our gesture recognition begins with exactly two fingers in contact with the screen (touch-down) and ends when either of the two fingers is lifted (touch-up). Our discussion below hence focuses on touch-move events from the two touch fingers. The order of the touch points is irrelevant; without loss of generality, let p_i^t denote the current position of a touch finger with touch id i, $i = 0, 1$ at time t.

5.1. Linear Classification

A straightforward solution to the classification problem is by simple thresholding, i.e., to explicitly check whether or not a finger is moving at a *speed* above a certain threshold.

Since touch-move events are generated at the rate of hundreds per second, using only the current and previous positions of a touch point at time t and t' is unstable for measuring its moving speed. Therefore, we use a simple Exponentially Weighted Moving Average (EWMA) approach, which exponentially decreases the influence of past speed so that the approximated value is close to the real speed while keeping the approximated value stable. Specifically, the moving speed of a touch point with touch id i at time t is estimated as:

$$S_i^t = (1-\alpha)S_i^{t'} + \alpha\|p_i^t - p_i^{t'}\|/\|t-t'\|, \quad (1)$$

where $\alpha \in (0,1)$ is the degree of weighting decrease (we use $\alpha = 0.1$).

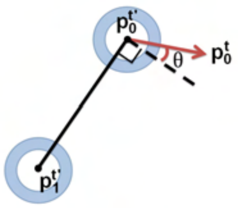
Let $S_{min}^t = \min(S_0^t, S_1^t)$ and $S_{max}^t = \max(S_0^t, S_1^t)$. Figure 3 (Left) plots pairs of min and max speed values in the S_{max}-S_{min} coordinate system, when several users were asked to perform specific gestures. As expected, the points for the RST-style gesture (especially for panning) are close to the line $S_{max} = S_{min}$, since the two fingers typically move at a similar speed.

A simple solution for the separation of the points for pin-panning from those for the RST-style gesture is to set a single threshold $S_{threshold}$ such that the RST-style gesture can be captured by $S_{min}^t > S_{threshold}$ (i.e., two moving fingers) and the pin-panning gesture is intended when $S_{min}^t < S_{threshold}$ & $S_{max}^t > S_{threshold}$ (i.e., one fixed and one moving). When $S_{max}^t < S_{threshold}$, neither of the gestures is recognized. The value of $S_{threshold}$ can be set by trial and error or learnt from the training data. However, as shown in Figure 3 (left), the points for the pan-pinning and RST-style gestures may highly overlap each other, causing no clear separation boundary in the S_{min}-S_{max} space. Frequently misclassified gestures would easily lead to unpleasing 3D manipulation experience.

5.2. Nonlinear Classification

Our preliminary experimentation on the linear separation model, as discussed in the previous subsection, motivated us to seek a better separation model as well as a better feature descriptor.

Feature descriptor. Although the speed S_i^t provides a good distinguishing characteristic for the movement of individual fingers, it fails to capture the movement correlation between the two fingers. Such correlation characteristic is useful for gesture recognition, especially when the two fingers are from the same hand, where the muscle bulks that move each finger are partly blended, preventing completely independent movement of individual fingers. Since pin-panning often involves rotational movement of one finger with respect to the other (fixed) finger, by experimenting a few alternatives we found that *the magnitude of centripetal acceleration* is another discriminative feature especially for distinguishing pin-panning from swiveling that rotates typically around a central point. Specifically, our current implementation approximates the magnitude of centripetal acceleration



as follows:

$$a^t = \|(\omega_1^t)^2 - (\omega_2^t)^2\| * d^t, \quad (2)$$

where ω_1^t and ω_2^t are the approximated relative angular velocities at time t for the two points, and d^t is the distance between the two contact points. Similar to the computation of the moving speed in Equation 1, the relative angular velocities are estimated by an EWMA approach:

$$\omega_i^t = (1-\alpha)\omega_i^{t'} + \alpha S_i^t \cos\theta/\|p_0^{t'} - p_1^{t'}\|, \quad (3)$$

where $S_i^t \cos\theta$ is the movement orthogonal to the orientation determined by the two contact points (see the inset figure above) and recall that α is the degree of weighting decrease.

We then define the feature vector for classification as $(S_{max}^t, S_{min}^t, a^t)$. Since our feature descriptor is encoded only based on the speed and distance information, it is inherently orientation independent, making our classification independent of gesture orientations, which is important to deal with a wide variety of touch devices (e.g., tabletop surfaces, smartphones etc.).

Classification. By examining the 2D projection of our 3D feature vector (Figure 3), we found that although there seems a feasible binary classification boundary, it is not easy to manually specify such boundary. We address this problem by taking a learning-based approach based on Support Vector Machines (SVM), though other classifiers might work equally well [Bis06]. SVM has been widely used for classification due to its excellent empirical performance. By using a nonlinear kernel function, SVM is able to model nonlinear dependencies among features. This is desirable since our three features are nonlinearly dependent.

To train an SVM classifier, we asked 12 users to repeatedly perform specific gestures (i.e., panning, pinching, swiveling, pin-panning), resulting in around 5000 labeled training samples. As the number of features is small (only three) compared to the number of instances in the training data, a common approach is to map the data to higher dimensional spaces [CL11]. We adopt the commonly used RBF kernel with $\gamma = 0.5$, and $\rho = -0.369$, which nonlinearly maps the samples into a high dimensional space. We

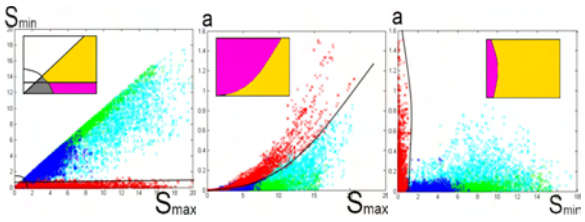


Figure 3: *2D projections of our 3D feature vectors to 2D coordinate systems $S_{max} - S_{min}$ (left), $S_{max} - a$ (middle), and $S_{min} - a$ (right): green for panning, blue for pinching, cyan for swiveling, and red for pin-panning. The inset shows the result by nonlinear classification: yellow for Mode 2m, pink for Mode $1m+1f$, and gray for unclassified.*

achieve a high cross validation accuracy 96.03% with this general model. Figure 3 plots the 2D projections of the found separation boundary, which is essentially a nonlinear hyperplane in 3D space. Similar to the linear classification model, the points for $S^t_{max} < S_{threshold}$ are marked as unclassified.

Validation. We use an independent test dataset to validate our method. The test dataset consists of thousands of labeled feature vectors from the same group of users for training data collection. The classification accuracy of our trained SVM model varies with the users, from 79% to 94% with average 91.4% (sd = 4%). We have also tried a user-tailored SVM model which is learnt using the training data from individual users. Such personalized SVM classifiers generally lead to high classification rate, with an average of 94.8% (sd = 2.9%).

The same test dataset is also used to validate the linear separation model (Section 5.1), achieving a classification rate of 89.7% on average (sd = 4.1%). The classification rate difference between the linear model and general SVM model is not huge, but both are much smaller than that of the individual trained SVM models. This is possibly owing to individual users having their own manipulation habits and different degrees of coordination/independence of finger movements, leading to different movement patterns for the same gestures.

Note that since touch-move events come at the rate of hundreds per second, individual touch-move events typically lead to small transformations. Therefore, occasionally misclassified gestures cause subtle artifacts. As shown in the supplemental video, the individual trained SVM models produce smoother manipulation than the linear classification model.

6. Direct/Indirect Manipulation

Although recent works on multitouch interaction have focused on direct manipulation of 3D objects, it is well known that direct control and indirect control have their own strengths and weaknesses [MH08, SBG09]. A recent study by Knoedel and Hachet [KH11] shows that indirect interaction favors efficiency and precision for the RST docking task. Unlike *DS3*, our technique does not rely on the directness of fingers to trigger different manipulation modes. Therefore our technique is applicable to both direct manipulation and indirection manipulation.

Below we briefly describe how we implement the 6DOF control in our current system. As we expected that direct finger contact would get difficult when the object appears small, which has been confirmed by our user study (Section 7), our implementation does not require direct finger contact with the object for manipulation. To achieve indirect manipulation, for each touch point not on the object, we find the point on the object whose projected screen-space point is closest to the screen-space touch point. During interaction, we enforce this closest point to undergo the same displacement as its corresponding contact point (see the accompanying video for demonstration). We use an optimization similar to that used in the *Screen-Space* method [RDH09], but with unknowns t_x, t_y, t_z, r_z in Mode $2m$ and with unknowns r_x, r_y in Mode $1m+1f$. Since in Mode $2m$, we have two fingers and four DOF to solve, it is possible to use a more direct mapping without any optimization, similar to the *Sticky Tools* technique [HtCC09]. Similarly, *xy*-rotation can be controlled using an Arcball-like rotational controller [Sho92], which supports both indirect and direct interaction.

7. Pilot Study

We conducted a pilot experiment to evaluate the effectiveness of our two-finger 6DOF manipulation technique, with our trained nonlinear classifier.

Goals. We have the following goals in mind when designing our experiment:

- ◇ To compare our technique with the state-of-the-art techniques, namely, *Sticky Tools (ST)* [HtCC09], *Screen-Space (SS)* [RDH09] and *DS3* [MCG12], especially in terms of completion time.
- ◇ To evaluate the performance of different techniques against different sizes of screen and object on the screen.
- ◇ To check the preference of unimanual interaction or bimanual interaction.

Participants. Ten university students with a mean age of 23.6 (SD 1.7) participated. Among them, eight were male and two were female. Participants have variable experience with multitouch interaction, though most of them were familiar with multitouch applications such as image and map browsing tools on mobile devices.

Apparatus. Although we were interested in various screen sizes, for simplicity, we used a single multitouch device only, i.e., an HP TouchSmart tx2 laptop with 12.1-inch screen, and simulated different screen sizes by fixing different application window sizes. The laptop was placed on a table (with height of around 70cm), with the screen facing upward. Participants sat in front of the table such that both their hands could easily reach the whole screen. Participants were free to adjust the location and orientation of the laptop.

Docking task. For performance comparison, we chose the 3D docking task introduced by Zhai and Milgram [ZM98]. Participants were required to translate and rotate a source object to a target location and orientation in the virtual 3D world as quickly and accurately as possible (Figure 4). As the number of participants was not big, a repeated measures design was adopted and each participant was asked to complete the same set of controlled trials. A trial was automatically determined as completed if the source and target objects were close enough. The source orientation and location for next trial were set the same as the target orientation and location in its previous trial. If a trial could not be completed within two minutes, the participants had an option to skip such trial.

As shown in Figure 4, the source object we chose was a CAD-style object which had holes and was easier to directly touch from its front view than its side view. To test the influence of finger directness, we manually specified the orientation of the target object for different trials. We set two rotation levels: *simple* rotation around only one axis (x, y, or z) with small rotation angles < 30 (e.g., Figure 4 (Left)) and *complex* rotation around a random mix of x-, y-, and z-axes with large rotation angles > 60 (e.g., Figure 4 (Right)). The size of the object on the screen varied by specifying different depth values, roughly categorized into three groups: *small* size (Figure 4 (Left)), *normal* size (Figure 4 (Right)), and *large* size.

Each participant had to successfully complete the same set of trials, 10 trials in our case, using all the four techniques one by one. The order of the techniques was counterbalanced across participants, to reduce effects of training on our analysis. Given the popularity of smartphones and tablets, we were interested in two window/screen sizes, 5-inch and 11-inch. Each trial was repeated for these two sizes. In summary, our experiment involved $10 \times 4 \times 10 \times 2 = 800$ trials.

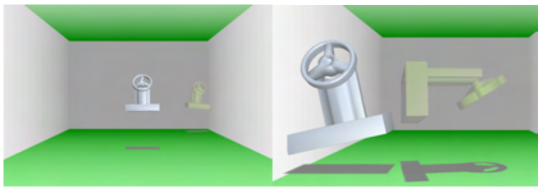


Figure 4: *Left: Docking task with simple rotation and small object size. Right: complex rotation and normal size. Target position and orientation are illustrated in pale green. The (perspective) camera was predefined and always fixed during object interaction. Shadow effect was introduced to help determine the depth.*

Before starting the experiment with a new technique, the participants were briefed on the technique and practised until they felt comfortable (this lasted from a few minutes to a dozen of minutes). The participants were suggested to use a single hand for interaction unless they felt that bimanual manipulation is much more flexible or efficient. The experiment ended with a qualitative feedback from the participants about the controllability and acceptability of individual techniques. During the experiment, our system recorded the following information for quantitative analysis: the completion time of individual trials, number of operations used, contact time, number of touch-down events, and finger travelling distance.

7.1. Quantitative Analysis

Skip rate. The skip rate reflects the degree of ease of use. We found that for simple trials (i.e., involving simple rotation), there was no trial skipped for our technique, *ST* or *DS3*, but the skip rate of *SS* was 2%. For complex tasks, the skip rates of our technique, *ST*, *SS* and *DS3* were 2%, 0%, 13%, and 7.5% respectively. The higher skip rates of *SS* and *DS3* imply that these two techniques are relatively difficult to use for complex transformations. As reported by the participants, the main reason is that since both *SS* and *DS3* require direct manipulation of the object, precise control becomes difficult when working on small-size screens or objects (imagine how difficult to precisely control the degree of rotation when two contact fingers have to stay very close to each other).

As the participants might quickly skip a trial according to their previous experience, the spent time for the skipped trials were not reliable to quantitatively evaluate the performance in terms of completion time. Therefore, skipped trials are removed from the subsequent analysis. Below we report the individual impact on average completion time per trial by different techniques, complexity levels, screen sizes, and object sizes. We found that similar conclusions are applicable to the traveling distance, contact time, and number of touch-down events, which are not shown here due to space limitation.

Technique. Repeated measures analysis of variance (ANOVA) found a significant difference between the four techniques on completion time ($F_{3,36} = 5.62$, $p < 0.003$). Pairwise comparison showed that our technique (average 18.7s) was significantly faster than *SS* (average 38.0s; $F_{1,18} = 9.21$, $p <= 0.01$) and *DS*3 (average 36.7s; $F_{1,18} = 10.8$, $p <= 0.005$). However, there was no significant difference between our technique and *ST* (average 21.2s; $F_{1,18} = 0.22$, $p = 0.65$). That is, our technique is comparable to *ST*, which requires at least three contact fingers for interaction. The conclusion here is consistent with that for the skip rate. Our finding is inconsistent with that by Martinet et al. [MCG12], which concluded that *DS3* was faster than *ST*. This is possibly because the sensitivity of *DS3* to screen and object sizes could not be exposed given their setup, i.e., fixed target of moderate size on a large touch screen.

Complexity. The ANOVA found a significant main effect of complexity on task completion time ($F_{1,78} = 28.5$, $p < 9E-07$). More effort was needed to complete complex tasks for all the techniques, as shown in Figure 5 (a) and (b). It is the case particularly for *SS*, possibly due to the mismatch between the perceptual structure of the task and the control structure of the multitouch device.

Screen size. As expected, the ANOVA revealed a significant main effect of screen size on completion time ($F_{1,78} = 12.98$, $p < 0.001$). Working on 5-inch screen, it generally took more time for all the methods to complete the trials (Figure 5), however our method gives the best performance. Among other methods, *DS3* was the most sensitive to screen size, possibly because direct touch on the object was difficult on small screen, especially when the object was distant from the viewer. Note that although the reported completion time of *SS* for 5-inch screen was less than that of *DS3*, it was not

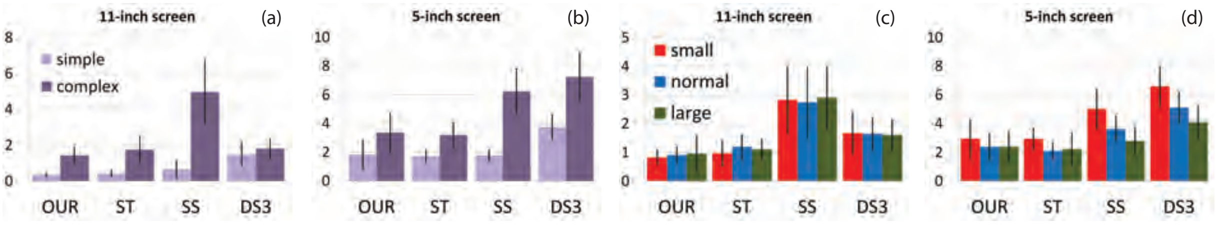


Figure 5: *The effect of task complexity (a and b) and object size (c and d) on average task completion time (in 10 seconds). The error bars represent the corresponding standard deviation.*

sufficient to conclude that *SS* was faster than *DS3* given the much higher skip rate of *SS*.

Object size. For 11-inch screen, we found no significant effect in object size (Figure 5 (c)). This is possibly because the smallest object still appeared large enough for manipulation on the 11-inch screen. However, for 5-inch screen, a significant difference was found for *SS* and *DS3* (Figure 5 (d)) (*SS*: $F_{2,27} = 2.82$, $p = 0.07$; *DS3*: $F_{2,27} = 2.85$, $p = 0.08$). This was expected since direct contact with the object was more difficult for smaller object sizes on the 5-inch screen.

7.2. Qualitative Analysis

Number of hands. We manually recorded the number of hands used by each participant for different methods. Eight of the ten participants completed all the trials using only a single hand with our technique. The other two occasionally used two hands. In contrast, for the other three techniques, all the participants almost always used two hands for interaction. Bimanual interaction is deliberately considered for the design of *SS*. Since for z-translation *DS3* strictly demands one direct finger and one indirect finger, it is more natural to perform such operation using two hands. Among the three existing techniques, *ST* is probably the most promising technique supporting unimanual interaction. Like ours, their Mode $2d$ naturally supports unimanual control and the index and thumb fingers are typically used. Their Mode $2d+1i$ can also be possibly achieved with a single hand, e.g., by using the index and middle fingers for pinning and the thumb from the same hand for panning. However, it is difficult to achieve seamless transition between the two modes due to different numbers of fingers and different moving characteristics of individual fingers. In addition, due to the physical constraints by the fixed index and middle fingers, the free movement of the thumb is less comfortable.

The feedback from the participants confirmed our conclusions above. When they were asked to vote the best technique, six of them went for our technique, three for *ST* and one for *SS*. Nobody was in favor of *DS3* mainly because of its poor performance on the small-size screen. Most of the participants had difficulty using *SS* for both 5-inch and 11-inch screens, largely due to its unpredicted results. Nevertheless, two participants could use *SS* very fluently after 10-minute practice, and they had similar good performance for all the other three techniques. Our technique was preferred to *ST* especially when working with 5-inch screen, since interacting with three fingers on a small screen caused too serious hand occlusion.

8. Conclusion and Future Works

This work for the first time introduced a single-hand, two-finger multitouch technique for 6DOF manipulation of 3D objects. Our key contribution is the introduction of the immobility of contact fingers for different manipulation modes, instead of using the number of fingers and/or the directness of individual fingers, as done in previous works. With fewer fingers and fewer hands, our technique outperforms the *Screen-Space* and *DS3* methods and is comparable to the *Sticky Tools* method. Among all the techniques, ours is the least sensitive to the sizes of screen and object.

As in a small time window it is rather difficult to determine whether a finger is still in motion or being moved extremely slowly, our technique might not be very suitable for fine-tuning control of model transformations. This problem might be alleviated by using the snapping approaches (see [ATF12] and references therein) together with our technique. Our current technique has been evaluated on a single touch device only. We are interested in examining the performance of our technique on screens of a wider range in size. We are also interested to explore how to robustly evaluate the immobility for touch sensors with different response rates.

Acknowledgements

The authors would like to thank Lu (Arthur) Chen for his help in video editing, Yutong Zhao for video narration. We appreciate the helpful comments from the anonymous reviewers. This work was partially supported by grants from the Research Grants Council of HKSAR (No. 113610, GRF619611), and the City University of Hong Kong (No. StUp7200266 and No. SRG7002776).

References

[ATF12] AU O. K.-C., TAI C.-L., FU H.: Multitouch gestures for constrained transformation of 3D objects. *Computer Graphics Forum 21*, 2 (2012). 2, 8

[Bis06] BISHOP C.: *Pattern recognition and machine learning*, vol. 4. Springer New York, 2006. 5

[CCL*11] CHEN C.-P., CHEN Y.-T., LEE P.-H., TSAI Y.-P., LEI S.: Real-time hand tracking on depth images. In *Visual Communications and Image Processing (VCIP), 2011 IEEE Conference* (Nov. 2011), IEEE, pp. 1–4. 3

[CDH11] COHÉ A., DÈCLE F., HACHET M.: tbox: a 3D transformation widget designed for touch-screens. In *CHI* (2011), pp. 3005–3008. 2

[CL11] CHANG C.-C., LIN C.-J.: LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology 2* (2011), 27:1–27:27. 5

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

[HCV*06] HANCOCK M. S., CARPENDALE S., VERNIER F. D., WIGDOR D., SHEN C.: Rotation and translation mechanisms for tabletop interaction. In *IEEE International Workshop on Horizontal Interactive Human-Computer Systems* (2006), pp. 79–88. 1

[HtCC09] HANCOCK M., TEN CATE T., CARPENDALE S.: Sticky tools: full 6DOF force-based interaction for multi-touch tables. In *ACM International Conference on Interactive Tabletops and Surfaces* (2009), pp. 133–140. 1, 2, 6

[KH11] KNOEDEL S., HACHET M.: Multi-touch RST in 2D and 3D spaces: Studying the impact of directness on user performance. In *3D User Interfaces* (2011), IEEE, pp. 75–78. 6

[MCG10] MARTINET A., CASIEZ G., GRISONI L.: The effect of dof separation in 3D manipulation tasks with multi-touch displays. In *VRST* (2010), pp. 111–118. 2

[MCG12] MARTINET A., CASIEZ G., GRISONI L.: Integrality and separability of multitouch interaction techniques in 3d manipulation tasks. *IEEE Transactions on Visualization and Computer Graphics 18*, 3 (2012), 369–380. 1, 2, 6, 7

[MH08] MOSCOVICH T., HUGHES J.: Indirect mappings of multi-touch input using one and two hands. In *CHI* (2008), pp. 1275–1284. 6

[NBBW09] NACENTA M., BAUDISCH P., BENKO H., WILSON A.: Separability of spatial manipulations in multi-touch interfaces. In *Graphics Interface* (2009), pp. 175–182. 3

[RDH09] REISMAN J. L., DAVIDSON P. L., HAN J. Y.: A screen-space formulation for 2D and 3D direct manipulation. In *UIST* (2009), pp. 69–78. 1, 2, 6

[SBG09] SCHMIDT D., BLOCK F., GELLERSEN H.: A comparison of direct and indirect multi-touch input for large surfaces. In *Human-Computer Interaction–INTERACT* (2009), pp. 582–594. 6

[Sho92] SHOEMAKE K.: Arcball: a user interface for specifying three-dimensional orientation using a mouse. In *Graphics Interface* (1992), vol. 92, pp. 151–156. 6

[SK10] SCHLATTMANN M., KLEIN R.: Efficient bimanual symmetric 3d manipulation for bare-handed interaction. *Journal of Virtual Reality and Broadcasting 7*, 8 (Oct. 2010). 3

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

[VCC*09] VOGEL D., CUDMORE M., CASIEZ G., BALAKRISHNAN R., KELIHER L.: Hand occlusion with tablet-sized direct pen input. In *CHI* (2009), pp. 557–566. 3

[WP09] WANG R. Y., POPOVIĆ J.: Real-time hand-tracking with a color glove. In *ACM SIGGRAPH 2009* (New York, NY, USA, 2009), SIGGRAPH '09, ACM, pp. 63:1–63:8. 3

[ZM98] ZHAI S., MILGRAM P.: Quantifying coordination in multiple dof movement and its application to evaluating 6 dof input devices. In *CHI* (1998), pp. 320–327. 6