# Direct Manipulation and the Third Dimension: Co-Planar Dragging on 3D Displays

*Max Möllers, Patrick Zimmer, Jan Borchers*
*RWTH Aachen University, Germany*
{*moellers, patrick.zimmer, borchers*}*@rwth-aachen.de*

**ABSTRACT**

Recent advances in touch and display technologies are supporting a wide-spread use of touch-based direct manipulation techniques as well as 3D displays that give a perspectively correct view. Both techniques have consistency constraints including the following: With direct manipulation, a dragged object should stick to the finger tip. With viewer centered projection, head movement should update the scene's projection to preserve a sound 3D impression, e.g., leaning around a house should reveal its backyard. Unfortunately, these two contradict each other, making a combination, e.g., moving the head while touching or dragging an object, non-trivial. We introduce a design space of perspectively adjusted methods for direct manipulation to cope with this limitation, select nine different strategies from it, and evaluate six of them in depth. Participants dragged a box through a 3D maze with multiple, partially occluded levels. We identified one method to be among the fastest while yielding up to 32% less collisions than the other fast methods.

**ACM Classification:** H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

**Keywords:** motion parallax, direct manipulation, co-planar dragging, 3D interactions, 3D tabletops

**General terms:** Design, Experimentation, Human Factors

**INTRODUCTION**

More and more consumer displays are able to generate 3D output. One visual cue for this is stereoscopy: You render slightly different images for each eye and display them to each eye using shutter glasses or special lenses. Another important visual cue to perceive spatial relations is motion parallax: As soon as the user moves his head, a viewer centered projection is updated. This requires head-tracking but can be done using low-cost cameras found on devices such as smartphones, notebooks, or tablet PCs.

Orthogonal to this evolution, many current displays are touch enabled and employ the direct manipulation paradigm in which a dragged object always sticks to the finger. Our study looks into the combination of the two paradigms: Users touch and move the projections of 3D objects, thereby manipulating the objects themselves. To isolate our study from stereoscopic effects, such as the ones described in [18], we augmented our touch enabled tabletop exclusively with viewer centered projection.

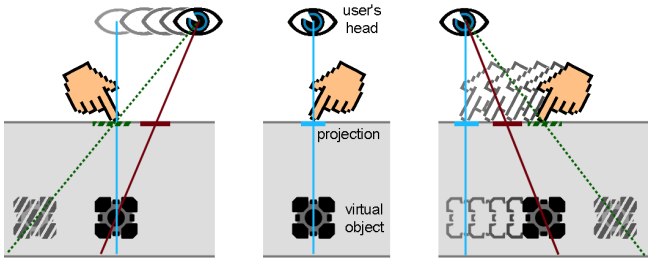


Figure 1: Direct manipulation in a naïve fish tank VR implementation. The colored lines show the gaze direction and the bars are the projections of the boxes. Bright blue is before and dark red after the movement; dotted green shows where the object would look to be "below" the finger from the user's point of view. Left) Moving the head does not change the 3D position of the object. Right) Moving the finger to the side moves the object equally. In both cases, the projections behave differently than the objects. Thus, the object is no longer below the finger, breaking the direct-manipulation metaphor.

One way to map the $(x, y)$-touch is to move the object in the same two dimensions and omit the third. We refer to this as *co-planar dragging*. Adding viewer centered projection causes head movement to change the projection of the 3D scene on the 2D display to provide a 3D impression. This means that the projections are moved on the display. In a naïve mapping (Fig. 1), touching one of these projections while moving your head will result in the projection of the touched object slipping off the finger—violating the sticks-to-finger property of direct manipulation. Another naïve option would be to move the touched object in 3D space based on the head movement—that way, the object would stick to the user's finger as expected, but now it would appear to move around in relation to its 3D environment as the user moves his head.

In this paper, we contribute new ways to perspectively adjust direct manipulation by presenting: (i) a design space for different input methods and (ii) a quantitative and qualitative


*ITS'12*, November 11–14, 2012, Cambridge, Massachusetts, USA.

analysis of six selected methods.

## RELATED WORK

Part of our research is related to the field of Virtual Reality (VR). While VR strives for perfect immersion covering multiple senses, we limit ourselves to a table size (150 cm x 80 cm) horizontal display, simulating 3D vision through motion parallax. These non-immersive systems are known as Fish Tank VR [19] and usually employ stereoscopy in addition to motion parallax. An early VR -motivated advance in the tabletop domain was the Responsive Workbench [1]. It featured a fully head-tracked stereo display for two users. However, its focus was on displaying content more than interacting with it as it only implemented indirect interaction through cyber gloves.

A comprehensive overview of other systems that use 3D technology on tabletops was done by Grossmann et al. in 2007 [6]. They defined a 3D tabletop taxonomy, differentiating systems by their physical properties as well as display and input properties. In this scheme, our setup joins the ImmersaDesk [5] in the group of table-sized systems with 2D input on a perceived 3D volume. While the ImmersaDesk was capaple of device-based input on its planar surface, its focus lay on interaction using six degrees of freedom flight sticks, also used in fully immersive VR systems like the CAVE [4]. Our setup differs as we strive for direct touch interaction [16], with no need for additional input devices.

Hachet et al. showed multitouch interaction with 3D objects hovering in the air [7]. They use an abstract 2D representation (based on handles) projected below the objects to allow for rotating, scaling, and translating them. We, however, focus on single touch direct manipulation and objects are shown beneath the physical interaction plane.

Steinicke et al. investigated the mix of stereoscopic and monoscopic content in a single GUI [17]. They also analyzed how people interact with stereoscopically rendered content [18], making it highly related to our work. However, we use view point correlation to monoscopically render our scene instead of showing fixed stereo images. While Steinicke et al. are trying to cope with the disparity of two different images displayed on the interaction surface, we are interested in the perceived alignment of touch point and virtual object.

Chan et al. [3] examined users' performance in 3D direct touch with an intangible reach-through display, realized with a fresnel lens. Similar to our motion parallax approach, the fresnel lens only displays a 2D projection, which is perceived as three dimensional. Chan et al. use two infrared cameras to calculate a 3D touch point above, on, or behind the image plane, which seems to be floating in mid air. Their results show that interaction in the z-axis (depth) is difficult, as there is no haptic feedback, neither active nor passive.

Interacting with 3D content does not necessarily require 3D displays; often 2D displays with a useful set of control options suffice. Recently, this field has branched towards tabletop displays with direct touch interaction. Interactions in the Air [12] controls content in three dimensions by additionally allowing contactless gestures above the table surface. Hancock et al. looked at direct touch gestures using one, two, or three contact points to manipulate objects with six degrees of freedom [9]. To keep interaction direct and feasible, they limited themselves to a shallow depth approach, where interactive content is reasonably close to the surface. Reisman et al. [15] present an in-depth analysis of the de-facto standard Rotate-Scale-Translate (RST) technique for manipulating 2D content and port it into the third dimension, using three or more contact points with novel interaction gestures.

Martinet et al. evaluated techniques for three dimensional object positioning on multitouch tabletops [13]. Their proposed Z-technique uses direct movement control on a plane with one finger, supported by a secondary indirect gesture to control depth. The use of single touch co-planar direct object movement gives a strong relation to our own work. Yet, we are facing the additional problem of a dynamic perspective and focus on techniques to cope with touch offset.

Hancock et al. showed how people (mis)interpret 3D orientation on tabletops when viewing off-axis [11] and offered insights in how to visualize content when perspectively accurate projection is not available or possible, e.g., when multiple users are involved [10].

An early version of the design space shown in this paper has been presented at a non-archival workshop at CHI 2012 [14].

## INTERACTIONS IN A FISH TANK

It is an often used paradigm that dragging an object makes it follow the user's finger or cursor. However, in a head tracked fish tank VR system, the two-dimensional dragging input can not be mapped to the 3D object easily.

The problem is the offset between the touch point on the surface and an object's projection on the same surface (Fig. 2). The offset depends on the current system state, which consists of the current head or view position $v$, an object handle $h$, and a touch position $t$. During a dragging operation, the following happens: The hand moves from $t_1$ to $t_2 = t_1 + \vec{m}$. The object naïvely mimics the movement of the user's finger, thus, being far below the interaction surface, appearing to be moving slower. Consequently, the object does not stick to the finger. To describe this effect, we use the offset $\vec{a}$. Moving the object to $h_i$ would eliminate the offset $\vec{a}$.

Furthermore, the illustration displays movement of the user's head from $v_1$ to $v_2$. The resulting additional offset is marked as $\vec{b}$. This offset is of special consideration when the user holds his hand still and tries to look around. By keeping track of head and touch movement, $\vec{a}$ and $\vec{b}$ can be easily distinguished. This gives us the option to deal with these effects either separately or in unison.

### Design Space

Offset occurs when a set of $v, t, h$ are not in a line. To compensate, we can either change the virtual object's position or we can change how the system takes the user's head into account by adjusting the virtual view point: Instead of moving the person's head, we move the digital representation of the head and ignore further updates (besides z movement, i.e., the person moving closer to the table). This means, we then

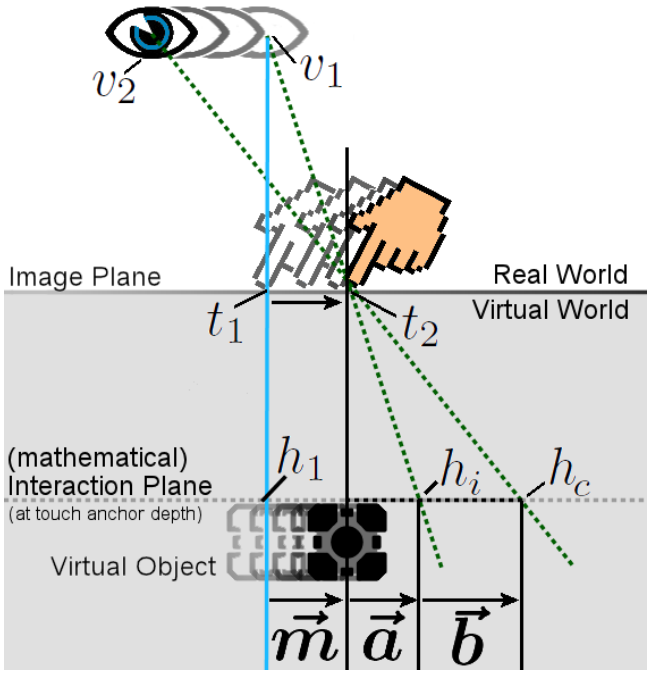


Figure 2: Offset created by dragging an object in a head tracked environment. Total offset can be split into a touch induced offset $\vec{a}$ and a head induced offset $\vec{b}$. From a graphics point of view, the near plane lies 60cm above the table, the projection plane matches the surface of the table, and the interaction plane is set to the top of the object.

have a new eye position and a changed viewport. This will shear the scene from the user's point of view. Table 3 shows a $2 \times 2$ matrix scheme where we oppose the offset triggers with the correction methods. For each type of offset we can choose one of two correction methods or none at all. This gives us three possible ways to deal with each offset: ignore the offset, move the object to compensate, or shear the scene to compensate. Linear combination of two corrections is an interesting field for future research.

| | **Trigger** | |
|---|---|---|
| **Correction** | Move Head | Drag Finger |
| Move Object | {0,1} | {0,1} |
| Shear Scene | {0,1} | {0,1} |

Figure 3: Design space for perspectively adjusted direct manipulation.

This leaves us with nine control methods in total, all of which were examined in a pilot study. Six promising methods were selected for further analysis regarding speed and accuracy. The following sections summarize all of the nine method's properties and implementation details as well as the reasons for the removal of the three.

**Interaction Methods**

The overall algorithm works as follows (cf. Figure 2):

0. Given positions last frame; head $v_1$, finger $t_1$, object $h_1$.
1. Drag finger to $t_2$ and move head to $v_2$. Note: $t_2 - t_1 = \vec{m}$.
2. Cast ray from $v_1$ over $t_2$. Intersection with object plane yields $h_i$. Note: $h_i - h_1 = \vec{m} + \vec{a}$.
3. Cast ray from $v_2$ over $t_2$. Intersection with object plane yields $h_c$. Note: $h_c - h_1 = \vec{m} + \vec{a} + \vec{b}$.
4. Move object to $h_1 + \vec{m}$.
5. Select method. Update offsets.
6. Move object by offset assigned for positional correction.
7. Shear scene by offset assigned to be shifted.
8. Draw. Update $v_1$, $t_1$, $h_1$. Repeat.

All nine methods perform the same direct object movement $\vec{m}$. They then differ in the way they treat the offsets $\vec{a}$ and $\vec{b}$. Their naming is loosely based on the following scheme:

- *Correction* if they move the object to compensate for offset,
- *Shift* if they shear the scene to compensate for offset,
- *Team* if they do both,
- *Adaptive* if offset is mitigated over time,
- *Cross* if the compensations are switched, e.g., head induced offset is corrected by object movement.

Additional impressions can be gathered from our supplemental video figure.

$\left[\begin{smallmatrix}0&0\\0&0\end{smallmatrix}\right]$ UNCORRECTED

No correction is applied and the resulting offset is tolerated. The object is naïvely moved only by vector $\vec{m}$.

$\left[\begin{smallmatrix}1&1\\0&0\end{smallmatrix}\right]$ OBJECTCORRECTION

This method can be found in literature under the term Sticky Finger [2]. The object position is constrained to a plane and determined based on head and touch position. In our model this correlates to an applied movement of

$$\vec{m} + \vec{a} + \vec{b}$$

This way, the object always appears to be precisely under the touch. The consideration of $\vec{b}$ leads to head movement alone causing the object to move. Offset can only be forced when this movement can not be applied due to other constraints, e.g., boundaries in the virtual world.

$\left[\begin{smallmatrix}0&1\\0&0\end{smallmatrix}\right]$ ADAPTIVECORRECTION

When the finger is moved, the object follows precisely. Head movement, however, does not cause the object to move and lets the touch slip off. This method is designed to appear to a user as $\vec{m} + \vec{a}$ object movement, yielding the described effects. However, mathematically the total applied object movement is:

$$\vec{m} + \vec{a} + |\vec{a}| \cdot f \cdot \frac{\vec{b}}{|\vec{b}|}$$

That means $\vec{b}$ is only corrected alongside $\vec{a}$ if the latter is dominant. This also causes an offset touch to ease back towards alignment while it is moved. This has been done to avoid an offset $\vec{b}$ to be retained indefinitely which would make the method feel indirect despite the $\vec{a}$ correction applied.

The factor $f \in [0, \infty]$ determines the dependance of $\vec{b}$ on $\vec{a}$, i.e., how quickly alignment is restored from an offset state. During our study, it was set to $0.8$ as a trade-off between offset reduction and predictability.

$\left[\begin{smallmatrix}0&0\\1&1\end{smallmatrix}\right]$ SCENESHIFT

In this mode, we avoid the perceived offset between touch point and object without repositioning the object itself. While an object is grabbed, we take explicit control over the virtual view point to keep it in alignment with object and touch. This can make the scene look sheared from the user's point of view. However, from the manipulated view point the scene geometry is still soundly determined by the fish tank VR projection. This is the same effect a bystander would experience when another user's head position is tracked, not his own.

We achieve this viewpoint transformation by multiplying the projection matrix with the following shearing matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ x & y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}; \; x = \frac{a_x + b_x}{z}; \; y = \frac{a_y + b_y}{z}$$

When only the head is moved, this perspective correction exactly counteracts the perspective change created through head movement. This makes the scene appear frozen to the user, leaving him unable to look around obstacles. If the touch is moved, the object is naïvely moved along by $\vec{m}$, and the correction of the remaining offset $\vec{a} + \vec{b}$ causes a view point translation, making the scene shear in the direction of movement. In contrast to OBJECTCORRECTION, SCENESHIFT is even applied when the object is dragged into a boundary, never permitting the touch to be offset.

Releasing an object restores the correct perspective defined by the user's head position. Instead of snapping back instantly, we decided to ease back the scene using sine curve smoothing over a second.

$\left[\begin{smallmatrix}0&0\\1&0\end{smallmatrix}\right]$ ADAPTIVESHIFT

This method is derived from SCENESHIFT but differentiates between offsets. Without touch movement, the view point stays locked, maintaining alignment. Touch movement may cause offset, which is slowly mitigated by additional head movement. The same factor $f = 0.8$ applies, analog to ADAPTIVECORRECTION. The resulting values $x$ and $y$ fed into the shearing matrix are as follows:

$$x = \vec{b} + \frac{|\vec{b}| \cdot f \cdot a_x}{|\vec{a}| \cdot z}; \; y = \vec{b} + \frac{|\vec{b}| \cdot f \cdot a_y}{|\vec{a}| \cdot z}$$

$\left[\begin{smallmatrix}0&1\\1&0\end{smallmatrix}\right]$ TAGTEAM

This method not only differentiates between offsets, it corrects them differently: additional object movement compensates touch induced offset; perspective shearing compensates head induced offset. That means, moving the touch causes the object to stick to the finger. Without touch movement the view point gets locked, as in SCENESHIFT. The total applied object movement, therefore, is:

$$\vec{m} + \vec{a}$$

The factors to be fed into the shearing matrix are

$$x = \frac{b_x}{z}; \; y = \frac{b_y}{z}$$

During object movement, view point changes are limited, as the head-coupled projection only reacts to the user as long as it counteracts the touch induced offset. In this method, touch offset is only possible when objects get dragged against a boundary, although contrary to the OBJECTCORRECTION method, head movement can mitigate this offset to a degree.

$\left[\begin{smallmatrix}1&0\\0&1\end{smallmatrix}\right]$ CROSSTEAM

This method is an adaptation of TAGTEAM with switched roles. Here, touch induced offset is compensated by perspective adjustment and view offset by correcting object position. Note that the following formulas for total object movement and shearing show switched roles for $\vec{a}$ and $\vec{b}$, but otherwise are the same as for the TAGTEAM method. Total movement:

$$\vec{m} + \vec{b}$$

Shearing factors:

$$x = \frac{a_x}{z}; \; y = \frac{a_y}{z}$$

Dragging the object in this environment automatically shears the scene, while moving the head has the potential of actively moving the object, depending on the direction of offset and existing projection adjustment.

$\left[\begin{smallmatrix}1&0\\0&0\end{smallmatrix}\right]$ CROSSCORRECTION

This methods applies adaptive positional correction, but unlike the ADAPTIVECORRECTION method, the correction here is linked to the other trigger, so here it is head movement that causes positional correction. Touch movement, on the other hand, causes no correction and can lead to offset. The offset

is steadily corrected when the head is moved similar to other adaptive methods. The total applied object movement is:

$$\vec{m} + \vec{b} + |\vec{b}| \cdot f \cdot \frac{\vec{a}}{|\vec{a}|}$$

The known factor of $f = 0.8$ applies.

$\left[\begin{smallmatrix} 0 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ CROSSSHIFT

This methods applies adaptive projectional correction, but unlike the ADAPTIVESHIFT method, the correction here is linked to the other trigger. Touch movement drags the box along with $\vec{m}$ and shears the scene to preserve alignment. The factors used in the shearing matrix are the following:

$$x = \vec{a} + \frac{|\vec{a}| \cdot f \cdot b_x}{|\vec{b}| \cdot z};\ y = \vec{a} + \frac{|\vec{a}| \cdot f \cdot b_y}{|\vec{b}| \cdot z}$$

Here, head movement causes no correction and can lead to offset, enabling the user to look around freely without loosing control of the box. When the touch is moved, existing offset is steadily corrected by shearing the scene, similar to adaptive methods.

## STUDY

The aim of this study is to evaluate the aforementioned different control methods for object dragging. To analyze this (in)direct manipulation, we designed a task in which the user needs to move a box through a maze with several levels.

During initial testing that used a within-subject design, the three CROSS methods consistently received negative comments as touch induced offset could lead to viewport changes and head movement could move the object. All in all, users were very unhappy with (a) their head movement causing collisions, (b) the need to release the object to safely look around, and (c) the unexpected jumping of the box shortly after releasing it. Many users stated confusion and frustration. Some interpreted this kind of control as "moving the space around the object". CROSSSHIFT, e.g., was described as "weird", "unsettling", and "confusing".

Consequently, we focused on the remaining six methods: UNCORRECTED, OBJECTCORRECTION, ADAPTIVECORRECTION, SCENESHIFT, ADAPTIVESHIFT, and TAGTEAM.

### System Setup

A sketch of the system used is displayed in Figure 4. The base of our system is a multi-touch table with outer measures of 155x136x92 cm. The visible size of the horizontal display is 140x80 cm. It is backlit by three Full-HD projectors, giving us a total resolution of 3240x1920 px on the tabletop. The table uses FTIR [8] for touch detection. Two 120 fps cameras with 640x480 px resolution scan the surface for reflected infrared light from the user's fingertips. This lead to an input resolution of 20dpi. The system is powered by a Mac Pro with a single ATI Radeon HD 5870 graphics card.

To augment the tabletop with head tracking, we used an industry grade tracking solution. Five cameras were mounted on a frame above and to the right of the table yielding sub millimeter accuracy around and over the table. To be seen by the cameras, users had to wear a hair circlet on which four reflective markers were mounted.

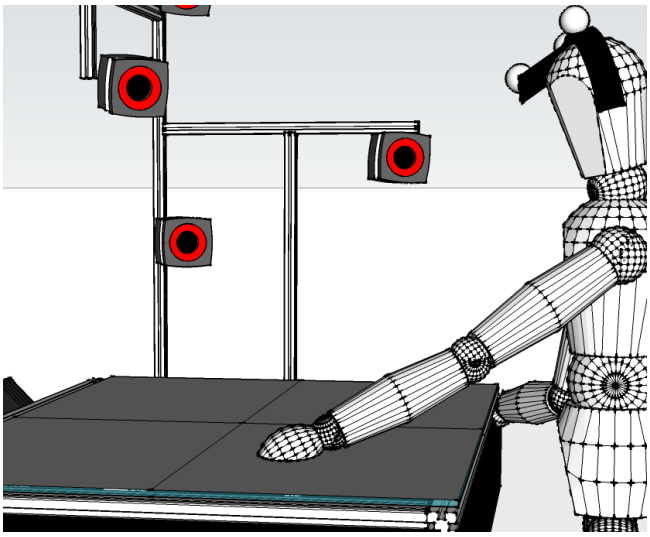

Figure 4: User standing in front of the tabletop. Infrared cameras in the background track the markers on the user's head. Head position is translated into a perspectively correct view of the virtual scene.

### Task

To accustom themselves with the hardware prior to the test, participants had two minutes to play around in a 2D dragging setting. Afterwards the previously assigned interaction method was activated and the participants could play around with it in a single level 3D playground for as long as they liked before they entered the maze itself. Participants had to drag a cuboid through this maze, avoiding collisions with its walls (Fig. 5). The maze was divided into three stages. Starting on the top level, participants progressed downwards by moving the box into a marked hatch on the end of each level. Levels did not disappear when finished, so participants had to deal with the added challenge of occlusion.

The box can be touched at one of two visible handles on its top (Fig. 5). Other objects would be controlled via similarly placed touch anchors or dynamically via ray casting to determine a touch anchor. We restricted grabbing the box to single touch interaction. The box follows the user's touch, according to the rules defined by the underlying design paradigm. While grabbed, the box is subject to a simple physics engine that allows it to pivot around its touch point depending on speed, direction, and friction. The physics layer is strictly two dimensional and works with the projections of the box and maze to calculate movement and collision behavior. When the box is released it immediately stops, so no flinging operations can be performed. However, to counteract touches getting lost while dragging, due to, e.g., bad tracking or blind spots, a small but noticeable delay is coded into the action of releasing an object. Yet, this allows for head induced box movement a few milliseconds after release.

The shape and physical movement model of the box gave the

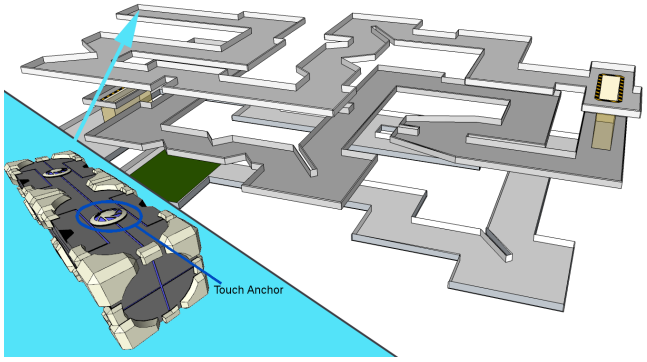


Figure 5: Task design: Participants had to drag the box through the maze, starting from the far left corner on top level to the green field on the bottom level. The boxes touch anchors are highlighted in blue.

users a task they found intriguing, challenging, and enjoyable. Many users described dragging the box as "driving a truck", referring to the tendency of the box's rear to cut corners when pulled behind. In case of collisions, a sound is played to encourage more careful behavior.

### Procedure

We measured completion time in seconds, number of occlusions, and number of re-grabs. We used a between-subject design for the test with three repetitions. A within-subject design for six techniques was not feasible with the long task time and three repetitions. Before the test we collected demographic data: gender, age, height, prior experience with touch devices. During the playground settings, we encouraged participants to share their impressions. However, during the actual test we asked them to focus on moving the box without too many collisions or slowdown. Participants were offered breaks between each level of the maze and each of the three trials. Relax times were part of the schedule, as each level began at the first touch of the box.

Because users tended to cause multiple contacts or follow-up collisions due to the physical properties of the box, we decided to measure a maximum of one collision every two seconds. Besides time and collisions, we also recorded the number of re-grabs, counting one for every time the box is released and grabbed again, either to reposition it for corners, set back the perspective, unlock the camera to look around, reduce the touch offset, or personal preference. Participants were not told that re-grabs were tracked or in any way important for their performance.

The study host took notes based on the feedback during the playground and while observing. After the test, users filled out a questionnaire consisting of nine statements rated on a five point Likert scale: The input method (1) allows for precise control, (2) lets me control the box directly, (3) forces me to keep my head still, (4) allows for good orientation while dragging, (5) is taking away my control over the system, (6) is comprehensible, (7) causes unpredictable jumping of the box, (8) tempts me to re-grab often, (9) negatively influences my 3D impression.

While users were filling out the form, an open dialogue with the study host was offered to convey additional comments or answer open questions.

### Participants

A total of 91 participants aged 18-30 ($M = 24.59$, $SD = 2.97$), 78 male, took part in our study. 91% were students, almost all of them computer science or related. 88% had previously experienced 3D vision, 87% had experienced mobile multi-touch, and 30% had already interacted with a touch-interactive tabletop. Half of the participants had some kind of vision impairment, but all of them had them corrected or only had minor impairments.

### Results

Means and confidence intervals of time, collisions, and re-grabs are reported in Figures 6, 7, and 8. We opted to stack the bars for each level, but will point out significant effects between levels when they exist. To increase readability, we focus on significant and impactful results of our analysis, e.g., we do not elaborate on the fact that trial almost always had a (strongly) significant effect on every independent variable as this can be interpreted as a learning effect. The full factorial analyses as well as all pairwise comparisons for each method and independent variable can be found in the supplemental data.

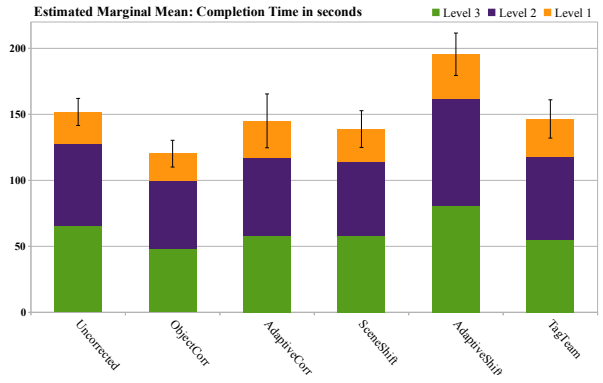


Figure 6: Mean completion time in seconds. Mean bars are sub-divided into single levels. 95% Confindence intervals are shown for the sum over all levels.

Time measurements were found not to be normally distributed (Shapiro-Wilk $W = .68, p < .0001$) but were sufficiently normal after a $log(y + 1)$ transform ($W = .99, p = .54$). Mixed effect analysis of variance (treating the user as a random effect) revealed significant main effects of method ($F_{(5,208.7)} = 3.49, p = .0048$) and level ($F_{(2,680)} = 471.56, p < .001$) on time. No interaction between method and level was found (Fig 6).

Collision was analyzed as Poisson distributed count data using Generalized Estimating Equations (reported as Wald's Chi-Square). We found a significant main effect of method ($\chi^2_{(5,N=91)} = 14.11, p = .015$) and level ($\chi^2_{(2,N=91)} = 390.32, p < .0001$) on the number of collisions. Interactions were found between method and level ($\chi^2_{(10,N=91)} =$

$57.75, p < .0001$). In level 1/2/3 the method had a significant main effect on the number of collisions ($\chi^2_{(5,N=91)} = 24.98/12.46/15.39, p < .0001/ = .029/ = .009$).

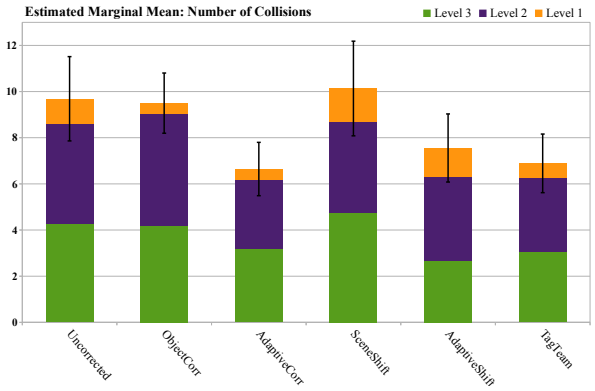


Figure 7: Number of collisions. Mean bars are subdivided into single levels. 95% Confindence intervals are shown for the sum over all levels.

Re-grabs were also analyzed as Poisson distributed count data using Generalized Estimating Equations (reported as Wald's Chi-Square). We found a significant main effect of method ($\chi^2_{(5,N=91)} = 114.26, p < .0001$) and level ($\chi^2_{(2,N=91)} = 557.56, p < .0001$) on the number of re-grabs. Interactions were found between method and level ($\chi^2_{(10,N=91)} = 45.99, p < .0001$). In level 1/2/3 the method had a significant main effect on the number of collisions ($\chi^2_{(5,N=91)} = 81.38/76.48/111.24, p < .0001$).

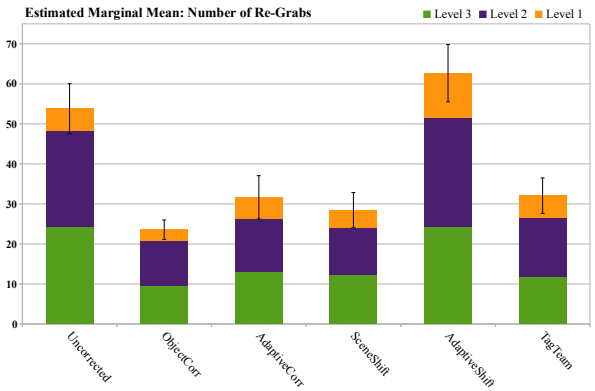


Figure 8: Number of re-grabs. Mean bars are subdivided into single levels. 95% Confindence intervals are shown for the sum over all levels.

The significant results of the pairwise comparisons for time, collisions, and re-grabs can be found in Fig. 9. All three independent variables are significantly correlated ($p < .0001$): collisions and time, $r(89) = .35$, re-grabs and collisions, $r(89) = .29$, and re-grabs and time, $r(89) = .79$.

In our questionnaire, users rated statements on a five point Likert Scale from **1** (*strongly disagree*) to **5** (*strongly agree*) (see Table 1). Kruskal-Wallis revealed a significant effect

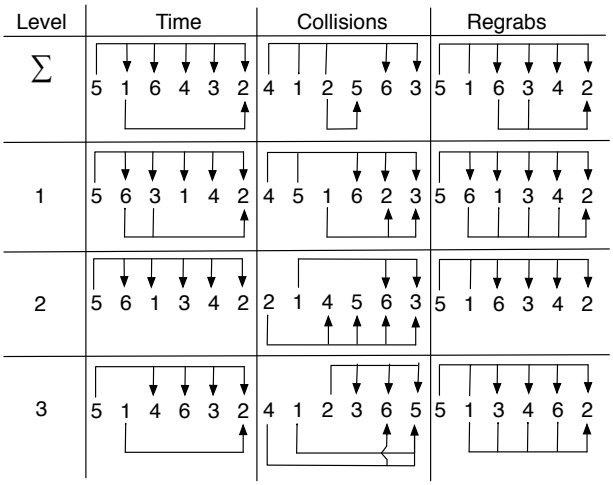


Figure 9: Results summed and by each level. Tukey's HSD for time and Wilcoxon signed rank sum test for collisions and re-grabs. Arrows denote a significant difference. Methods are sorted descending by their mean and are labeled as follows: (1) Uncorrected, (2) ObjectCorrection, (3) AdaptiveCorrection, (4) SceneShift, (5) AdaptiveShift, (6) TagTeam.

of the method on criterion (5) loss of control ($\chi^2_{(5,N=91)} = 18.28, p = .0026$). For this criterion, a pairwise comparison using Wilcoxon rank sum test revealed that UNCORRECTED is lower than OBJECTCORRECTION, ADAPTIVECORRECTION, and ADAPTIVESHIFT. Also, OBJECTCORRECTION is higher than ADAPTIVECORRECTION.

| | UNCORRECTED | OBJECTCORR | ADAPTIVECORR | SCENESHIFT | ADAPTIVESHIFT | TAGTEAM |
|---|---|---|---|---|---|---|
| precise control | 3.0 | 3.7 | **3.9** | 3.5 | 3.3 | 3.5 |
| direct control | 3.3 | 4.0 | 4.0 | **4.1** | 3.7 | 4.1 |
| keep head still | 1.5 | **2.6** | 1.6 | 1.5 | 1.7 | 1.8 |
| good orientation | 3.9 | 3.3 | **4.1** | 3.1 | 3.4 | 3.4 |
| loses control | 1.3 | **2.6** | 1.9 | 2.4 | 2.4 | 1.7 |
| comprehensible | 4.3 | 4.5 | 4.2 | 3.9 | 3.6 | **4.7** |
| unexpected jumping | 2.0 | **2.7** | 1.8 | 2.1 | 1.9 | 2.6 |
| requires re-grabs | **3.7** | 3.4 | 2.5 | 2.6 | 3.3 | 2.9 |
| lessens 3D impression | 1.7 | 1.7 | 1.4 | 2.1 | **2.3** | 1.6 |

Table 1: Average ratings of the questionnaire statements. A five point Likert scale was used.

## DISCUSSION AND DESIGN SUGGESTIONS

Before taking a closer look at each method, we can see that re-grabs and time have a tight connection. Obviously, performing a re-grab takes time. However, after a re-grab the offset is nullified, yielding more precise control, which in turn might speed up task time. The positive correlation (**.79**) indicates that the former is the stronger effect, yet we recom-


Figure 10: Collisions for each level across all methods. They primarily occurred at bottlenecks and in corners.

mend to allow methods to use re-grabs to let the user correct offset while minimizing the amount of re-grabs the user has to perform.

The number of collisions increased with the depth and primarily occurred at bottlenecks and in corners. This was similar for all methods. We observed that the box was mostly controlled using the front handle. However, to navigate around corners, some people used the back handle to rearrange the box.

OBJECTCORRECTION was in the group of fastest methods and its (insignificant) leader. They same is true for the number of re-grabs. However, many users stated confusion and frustration with OBJECTCORRECTION due to (a) their head movement causing collisions, (b) the need to release the object to safely look around, and (c) the unexpected jumping of the box shortly after releasing it. ADAPTIVESHIFT should be avoided regardless of the expected occlusion or interaction depth as it was slow and required a lot of re-grabs (Fig. 6,8,9). UNCORRECTED required many re-grabs but users did complete with average times. This could hint that re-grabs, though necessary, were easier and faster to execute for this method.

The number of collisions can vary greatly with the level due to the amount of occlusion and interaction-depth. However, SCENESHIFT and UNCORRECTED performed badly overall (Fig. 7,9). OBJECTCORRECTION worked well at least for the first level, but did have problems with lower levels due to the aforementioned implicit box movement. The ADAPTIVESHIFT method was one of the worst on the first level and the best on the last level regarding collisions. Participants also criticized the view snapping back, classifying the method as precise but weird. ADAPTIVECORRECTION performed great among all levels (32% fewer collisions), closely followed by TAGTEAM. Both methods offer direct control. While TAGTEAM is restrictive about object-finger alignment, ADAPTIVECORRECTION allows to break it in case the user just wants to look around a bit.

With ADAPTIVECORRECTION being received so well, we propose it to be the first choice. This method seems to be a confusion free, surprise free, accurate, low entry barrier approach to 2D direct manipulation of 3D media. Although OBJECTCORRECTION is (insignificantly) faster, the method received bad feedback on unexpected behavior. Also, ADAPTIVECORRECTION has a comparably low number of re-grabs, which might be important for some groups of users such as physically handicapped people.

All in all, we have identified head movement interfering with box control as one of the main problems. We also found that offset is a problem users struggle with and that performance seems to decline whenever offset is involved. However, not all methods that allow for offset perform poorly. Dealing with offset smartly seems to be a viable option, other than avoiding it completely.

Several extensions to our study setup may lead to further interesting results: using multiple fingers for single object control, controlling multiple objects at once, or adding stereoscopy to enhance the 3D impression. Further extensions to our methods are also possible by changing the adaptive correction factor $\boldsymbol{f}$, or by trying to linearly combine correction methods for given offset triggers.

Another option would be to "zoom in" on the object to interact with. That means, we would set the projection plane to the object's top, thereby lifting it up and removing the problems introduced due to object depth. However, this would mean we would break the fish-tank metaphor thereby loosing context (e.g., the upper floors) and realism. Still, in other domains, this might be a feasible approach that would warrant further studies.

Non-flat objects could benefit from more sophisticated touch processing as people would expect or prefer objects to mimic physical behavior: start rotating (along the axis orthogonal to the plane defined by the drag begin point, drag end point and the object's center of gravity) until the object "tails" behind the finger. Separated manipulation modes could be another option, we would then try to adapt or compare some of the up to three finger combinations [9]. Another question is how to deal with varying heights of the scene, e.g., moving an object along a slope.

### RepliCHI

As proposed in [20], we encourage replication of this study. To this end, all our datasets can be found at `http://tinyurl.com/buygrmq` for replication and further analysis.

## CONCLUSION

In this paper, we presented a design space of interaction methods for touch-enhanced displays that use motion parallax for 3D. These methods address problems that arise with perspectively adjusted dragging in which users interact with an object's projection rather than the object itself. The two main problems identified are the perceived offset between touch and object projection, and head movement inferring object translation. We presented nine different control methods and evaluated six of them in depth. We identified the ADAPTIVECORRECTION method to be among the fastest, while yielding up to 32% less collisions than other fast methods.


## ACKNOWLEDGMENTS

We thank our numerous study participants and Chatchavan Wacharamanotham for supporting us in analyzing the data. This work was funded in part by the German B-IT Foundation.


## REFERENCES


1. Maneesh Agrawala, Andrew C. Beers, Ian McDowall, Bernd Fröhlich, Mark Bolas, and Pat Hanrahan. The Two-User Responsive Workbench: Support for Collaboration Through Individual Views of a Shared Space. *Proc. SIGGRAPH '97*, pages 332–388.

2. Doug A Bowman, Ernst Kruijff, and Joseph J. Laviola. *3D User Interfaces: Theory and Practice*, page 157. Addison-Wesley Longman, Amsterdam, 2005.

3. Li-Wei Chan, Hui-Shan Kao, Mike Y Chen, Ming-Sui Lee, Jane Hsu, and Yi-Ping Hung. Touching the Void: Direct-Touch Interaction for Intangible Displays. *Proc. CHI '10*, pages 2625–2634.

4. Carolina Cruz-Neira, Daniel J Sandin, Thomas A DeFanti, Robert V Kenyon, and John C Hart. The CAVE: audio visual experience automatic virtual environment. *Communications of the ACM*, 35(6):64–72.

5. Marek Czernuszenko, Dave Pape, Daniel Sandin, Tom DeFanti, Gregory L. Dawe, and Maxine D. Brown. The ImmersaDesk and Infinity Wall projection-based virtual reality displays. *Proc. SIGGRAPH '97*, pages 46–49.

6. Tovi Grossman and Daniel Wigdor. Going Deeper: a Taxonomy of 3D on the Tabletop. *Proc. TABLETOP '07*, pages 137–144.

7. Martin Hachet, Benoit Bossavit, Aurélie Cohé, and Jean-Baptiste de la Riviére. Toucheo: Multitouch and Stereo Combined in a Seamless Workspace. *Proc. UIST '11*, pages 587–592.

8. Jefferson Y Han. Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. *Proc. UIST '05*, pages 115–118.

9. Mark Hancock and Sheelagh Carpendale. Shallow-depth 3D Interaction: Design and Evaluation of one-, two- and three-touch Techniques. *Proc. CHI '07*, pages 1147–1156.

10. Mark Hancock and Sheelagh Carpendale. Supporting Multiple Off-Axis Viewpoints at a Tabletop Display. *Proc. TABLETOP '07*, pages 171–178.

11. Mark Hancock, Miguel Nacenta, Carl Gutwin, and Sheelagh Carpendale. The effects of changing projection geometry on the interpretation of 3D orientation on tabletops. *Proc. ITS '09*, pages 157–164.

12. Ottmar Hilliges, Sharam Izadi, Andrew Wilson, and Steve Hodges. Interactions in the air: adding further depth to interactive tabletops. *Proc. UIST '09*, pages 139–148.

13. Anthony Martinet, Gery Casiez, and Laurent Grisoni. The Design and Evaluation of 3D Positioning Techniques for Multi-touch Displays. *Proc. 3DUI '10*, pages 115–118.

14. Max Möllers, Patrick Zimmer, and Jan Borchers. Direct Manipulation and the Third Dimension: A Design Space for Co-Planar Dragging on 3D Displays . *CHI EA '12: Workshop on the 3rd dimension of CHI*.

15. Jason L Reisman, Philip L Davidson, and Jefferson Y Han. A Screen-Space Formulation for 2D and 3D Direct Manipulation. *Proc. UIST '09*, pages 69–78.

16. Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *Computer*, 16(8):57–69, 1983.

17. Frank Steinicke, Timo Ropinski, Gerd Bruder, and Klaus Hinrichs. Interscopic User Interface Concepts for Fish Tank Virtual Reality Systems. *Proc. VR '07*, pages 27–34.

18. Dimitar Valkov, Frank Steinicke, and Gerd Bruder. 2D Touching of 3D Stereoscopic Objects. *Proc. CHI '11*, pages 1353–1362.

19. Colin Ware, Kevin Arthur, and Kellogg S Booth. Fish tank virtual reality. *Proc. INTERCHI '93*, pages 37–42.

20. Max L Wilson, Wendy Mackay, Ed Chi, Michael Bernstein, Dan Russell, and Harold Thimbleby. RepliCHI - CHI should be replicating and validating results more: discuss. *CHI EA '11*, pages 463–466.