"Oh Snap" – Helping Users Align Digital Objects on Touch Interfaces

Jennifer Fernquist, Garth Shoemaker, Kellogg S. Booth

Department of Computer Science
201-2366 Main Mall, The University of British Columbia
Vancouver, BC Canada V6T 1Z4
{adara, garths, ksbooth}@cs.ubc.ca

Abstract. We introduce a new snapping technique, Oh Snap, designed specifically for users of direct touch interfaces. Oh Snap allows users to easily align digital objects with lines or other objects using 1-D or 2-D translation or rotation. Our technique addresses two major drawbacks of existing snapping techniques: they either cause objects to "jump" to snap locations, preventing placement very close to those locations, or they "expand" motor space so that on direct-touch interfaces objects lag behind the user's finger. Oh Snap addresses both of these problems using an asymmetric velocity profile similar to a technique for filtering degrees of freedom in multi-touch gestures that was introduced by Nacenta et al. (2009). Oh Snap applies the velocity profile to multiple "snapping" constraints. A user study revealed a 40% performance improvement over no snapping for 1-D translation, 2-D translation, and rotation tasks when snap lines or angles were targeted. We found that Oh Snap performs no worse than traditional snapping, while retaining its important functional benefits. The study also investigated optimal parameter settings and Oh Snap's accuracy in supporting the placement of objects near to, but not at, snap locations, which traditional snapping techniques do not support. Oh Snap was found to be competitive with non-snapping interfaces for these tasks.

1 Introduction

Touch interfaces, such as multi-touch tabletops, interactive wall displays and mobile devices, are growing in popularity. As a result, many researchers are investigating their usefulness for completing an increasingly diverse collection of tasks, including: the control of robots [19], the control of systems [8, 21], managing artifacts [2, 3, 13], and software engineering [9]. Most of these systems support the selection and manipulation of digital objects on the screen using direct touch, exploiting the naturalness of physical direct interaction. For example, users may touch and drag a digital robot to position its real-world counterpart, move and group digital documents, drag a 1-D slider, or rotate a dial.

Unfortunately, touch interfaces are sometimes not well suited to precise manipulation. The "fat finger problem" [27] makes selection of specific targets or placement of digital objects at precise locations or orientations difficult. Shortcomings

in current sensing technology and the difficulty inherent in resolving touch contacts also contribute to the problem.

There have been several techniques developed that attempt to facilitate precise touch interactions on both large and small touch interfaces [2, 3, 5, 22, 23, 29, 34]. Work has also been done to develop better methods for manipulating digital objects [15, 17, 25]. However, the fundamental issue surrounding the fat-finger problem remains. Very little has been done to improve the alignment and precise positioning of digital objects in a touch environment.

Alignment tasks in a computer interface are often assisted with "snapping" techniques [6, 7, 10, 24]. This dates back at least as far as Ivan Sutherland's groundbreaking Sketchpad system [28], which included snapping constraints. Snapping techniques are one of the most common object alignment methods, and are widely used in computer-aided design (CAD) and other drawing programs. Traditional snapping causes digital objects to instantaneously "jump" and then "stick" to a line or grid point that has snapping capabilities once the object is within some threshold distance from the snap location.

While this technique is sometimes sufficient for use with relative input devices, it is less suited to direct touch interfaces. Additionally, locations within the threshold area near snap locations may be inaccessible: if a user wishes to position a digital object within that area, the snapping functionality must be turned off. Toggling snapping functionality is onerous at best for single users, but it is even more of a problem on large collaborative displays, where the management of multiple widget sets and user states is a perennial problem [26].

More subtle snapping techniques have been developed, such as snap-and-go [4]. These permit digital object positioning near snap locations. Snap-and-go works by expanding motor space at a snap location, resulting in objects that stop, rather than jump. However, this method is not suitable for direct touch interfaces, because it can break the correspondence between finger and object.

In this paper we describe a new snapping technique, Oh Snap, designed specifically for direct touch interfaces. We employ a technique first introduced by Nacenta et al. [20]. Our technique is designed to support quick snapping to any one of a set of lines, angular orientations, or other constraints, while still allowing objects to be positioned in close proximity to one another and while maintaining a close correspondence of the user's finger with the dragged object throughout. This set of benefits is unique to our technique. Oh Snap provides a subtle snapping effect that needn't be explicitly enabled or disabled. It avoids limitations in existing alternatives and thus facilitates placements that other techniques do not. We compare our work to the earlier work by Nacenta et al. [20] in more detail after describing the new technique and a two-phase user study that was conducted to assess our technique.

2 Related Work

Many researchers have investigated approaches for supporting direct manipulation with objects on touch surfaces. Wobbrock et al. [33] investigated user-defined gestures for general interactions on multi-touch tabletops and found that touching-

and-dragging is the most natural method for translating digital objects, and that dragging by the corner is the most natural way to rotate objects. Similarly, Micire et al. [19] conducted an analysis of user-defined gestures for robot manipulation on a multi-touch tabletop. They too found that touch dragging was the most used gesture for positioning robots. Kruger et al. [15] and Liu et al. [17] developed additional methods for performing fluid rotation and translation of objects using direct touch.

Many tasks designed for a touch interface can benefit from precise positioning of digital objects. Nóbrega et al. [21] created LIFE-SAVER, an interactive visualization system for a touch interface to analyze emergency flood situations. Studies by Bjørneseth et al. [8] used a touch table for dynamic positioning of maritime equipment. This safety-critical task requires careful translation and rotation for specifying vessel positioning and heading, respectively.

2.1 The "Fat Finger" Problem

Many touch devices capture a touch contact 'point' that is actually a relatively large 2-D region [27]. This is often converted into a single (x,y) pixel coordinate for compatibility with traditional pointing models that assume a single point of interaction. However, there is no guarantee that this single point is the true contact point intended by the user. Despite many advances in technology, this problem persists in part due to a lack of sophisticated sensing techniques, but also because the intended point of interaction is inherently ambiguous. Techniques such as "focus+context" lenses have been designed to help mitigate this problem within information visualization applications [30], but no general solution exists for all types of applications.

Sensing limitations may give rise to a variety of issues when manipulating objects, such as unintentional movement of the object. For example, when a finger lifts up from the screen its contact area changes shape, which may result in a change to the calculated pixel coordinates. If a user were attempting to precisely place a digital object, this might cause the object to shift from the desired target.

There has been work to resolve the fat finger problem by obtaining a more accurate touch contact point [12, 31], providing feedback to the user about the success/failure of the touches [32], and incorporating selective zooming [29]. Although these techniques can provide a more accurate touch contact location, they do not assist substantially in object alignment tasks.

2.2 Existing Snapping Techniques

Traditional snapping techniques, such as snap-dragging [7], cause objects to automatically jump to snap locations once they are within a predefined distance from the snap location. Basic snapping is highly effective in assisting alignment tasks; however, while it is sometimes sufficient for use with relative input devices, such as computer mice, it is less well suited to direct touch interfaces. It is highly unintuitive if an object a user is touching suddenly jumps around underneath the user's finger.

Worse yet, traditional snapping does not support the placement of objects near to, but not exactly at, snap locations. Disabling snapping can address this problem, but explicitly toggling snapping is highly undesirable for a touch interface, especially in a collaborative environment. The snap toggling function would either have to be a global function, affecting all users, or a local function, which would require additional information to determine the identity of the activator. It might also have to be placed in one or more menus, accessible to all users, or activated with a possibly complex gesture, requiring at least some additional training of users [11].

Snap-and-go, introduced by Baudisch et al. [4], is a snapping technique that does not require toggling on or off. It functions by expanding motor space at snap lines, resulting in objects stopping at the desired location as opposed to automatically jumping there once they are within some threshold distance. This allows objects to be placed near snap lines as well as directly on them, unlike traditional snapping.

Snap-and-go works well for relative input devices, such as mice. With relative devices, snap-and-go stops a dragged object at the snap line as the user keeps moving the mouse beyond the snap line. After a short distance, the object begins moving with the mouse again. Unfortunately, snap-and-go is not suitable for direct touch interfaces where the correspondence between a user's finger and an object under manipulation should ideally be maintained at all times. If a user were to drag an object across a snap-and-go line, the finger would permanently move out ahead of the object. Indeed, the more snap lines an object crosses, the farther the object would lag behind the finger that was dragging it, in effect "losing" any direct object-finger correspondence.

There have been other attempts to solve these problems. Pseudo-haptic feedback has been used to improve interactions with graphical user interfaces, causing screen widgets to "feel" sticky, magnetic, or repulsive [16]. Researchers have developed sticky widgets [18] and "force fields" [1] to help with window alignment in mouse-based environments. These ideas could be adapted to the translation and alignment of objects on touch tables; however, they would suffer from the same drawbacks as snap-and-go, because the finger could "lose" the object. The problem remains of maintaining a close correspondence between a user's finger and the object that is being manipulated.

3 The "Oh Snap" Technique

Oh Snap is a snapping technique designed specifically for touch interfaces. It possesses several benefits, including: it eases alignment of objects with snap lines, it doesn't require toggling modes, it maintains the correspondence of finger to object, and objects can be placed close to snap lines or other objects without snapping interfering.

The basic idea behind the Oh Snap technique is shown in Fig. 1. To begin, a user drags an object as she normally would until the object first touches a snap line, at which point the object stops moving even if the user does continue to drag her finger. The object remains stationary unless the user's finger travels a small distance (the *snap-width*) beyond where snapping has occurred. Once the finger travels beyond the *snap-width*, the object starts moving at a rate faster than the finger is moving. Once

the finger travels further, beyond the *catch-up width*, the object will have caught up to the finger, and dragging continues as usual. Of course, if the user lifts her finger while the object is snapped to the line, the object remains in its aligned position.

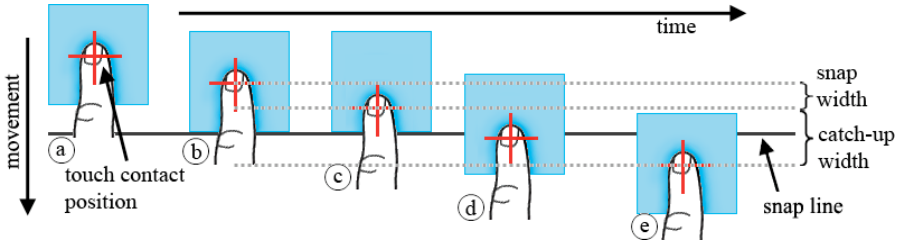


Fig. 1. As a finger moves an object downward to a snap line (a) the object "snaps" when its leading edge touches the snap line (b). As the finger continues downward, the object remains snapped to the line (c). When the finger is beyond the snap width, the object un-snaps and starts catching up to the finger (d). When the finger reaches *snap width + catch-up width* the object has returned to its original position relative to the finger (e).

During the catch-up phase an object travels at a rate faster than the finger. The motion of an object in the catch-up phase is defined by a linear interpolation that calculates the object's position proportional to the finger's position within the catch-up region. Pseudo code for the algorithm is shown in Fig. 2. The number of pixels the object travels for each pixel the finger travels is determined by the ratio calculated in Eqn. 1, which is shown on the left in Fig. 3. This is the rate at which an "un-snapped" object catches up to the finger. The ratio can also be considered to be the size of a *super pixel* relative to a real pixel, the distance moved by the object each time the user's finger moves one real pixel.

```
linearInterpolation(fingerX, fingerOriginX, snapWidth,
    catchUpWidth)
{
    return fingerOriginX + (fingerX-fingerOriginX-snapWidth) *
        (snapWidth + catchupWidth) / catchUpWidth;
}
```

Fig. 2. Code fragment for the linear interpolation function that returns the position of the object moving in the x direction if the object is snapped and the finger position is in the 'catch-up' area. In this code `fingerX` is the current finger position, and `fingerOriginX` is the position the finger was in when snapping occured.

A temporal diagram of the Oh Snap technique is shown in Fig. 3. The graph shows how an object first travels normally, how it then snaps to a line while the user's finger is in the snap region, and how it eventually catches back up to the finger because it travels faster than the finger in the catch-up region.

3.1 Snap width and catch-up width

The ratio in Eqn. 1, and thus the size of the super pixels, must be carefully chosen. When the *catch-up width* is very large, the ratio approaches one so objects effectively never catch up. Conversely, when the *catch-up width* is close to zero, the ratio approaches infinity and objects jump to their original position underneath the user's finger as soon as they un-snap. This can make it difficult to position an object at a location closer than the *snap width* to a snap line.

$$\text{ratio} = ((\text{snap width}) + (\text{catch-up width})) / (\text{catch-up width}) \quad (1)$$

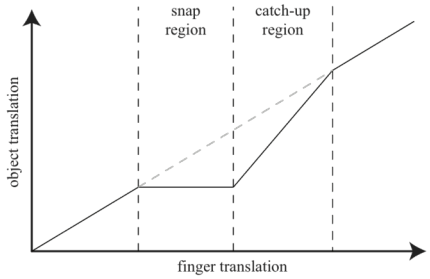


Fig. 3. The relationship of object translation relative to finger translation as an object moves, first normally, then snapped (stationary) in the snap region, and eventually catching up to the finger again as it leaves the catch-up region.

Ideally, the magnitude of *catch-up width* + *snap width* should be less than the width of an average touch contact. Wang and Ren [31] found this to be 36 pixels (0.4mm/pixel) for an oblique touch from an index finger. The *snap width* should be large enough to accommodate users overshooting a target, otherwise objects un-snap before the user's finger has a chance to stop. A balance must be struck so that catching-up is imperceptible but still occurs quickly enough to be useful. If the ratio is too close to one, some positions near a snap line can be unreachable, due to quantization effects, unless the touch sensors provide sub-pixel resolution. For example, if the ratio (and super pixel size) is 2, a position 3 pixels away from a snap line is unreachable, because the object will be moving in steps of 2 pixels as the finger moves in steps of 1 pixel. This can be mitigated either by lifting the finger right after crossing the snap line, which flags the object as un-snapped, and then putting it down again and resuming with normal dragging, or dragging the object far enough away from, and then back towards, the snap line. Neither seems like a very good solution, which emphasizes the need for proper selection of *catch-up* and *snap widths*. We discuss the selection of these parameters in section 5.2 and evaluate three different parameter sets in the second phase of our user study.

3.2 Benefits of the Oh Snap technique

The benefits of Oh Snap are summarized in Table 1. First, Oh Snap preserves the position of the user's touch point on a digital object relative to that object. This feature is especially useful if users drag objects across snap lines when they have no

intention of aligning the object to those lines. Although the object will temporarily snap to those lines as it crosses them, the object will eventually catch up with the user's finger and return to its original relative position underneath it. This is crucial for touch tables or other direct touch interfaces. Second, Oh Snap allows users to place digital objects near snap lines as well as align them with snap lines without having to toggle the snap capabilities on and off. This is important in collaborative environments where toolbars that may hold the snap toggle might not be accessible (reachable) by some users. Lastly, because Oh Snap supports all object positioning tasks, there is no need to incorporate mode switching functionality into the interface.

Table 1. Comparison of snapping techniques.

Technique	Fast snapping	Mapping maintained	Close placement
Oh Snap	Yes	Yes	Yes
snap-and-go	Yes	No	Yes
traditional	Yes	Yes	No
no snapping	N/A	Yes	Yes

4 Implementation

```
OhSnap(objectBorder, snapLineX, snapWidth, catchUp){
   if (objectBorder.rightX == snapLineX && !isSnapped){
      isSnapped = true;
      fingerPositionSnapped.x = currentFingerPosition.x;
   }
   if (isSnapped){
      fingerDiff = currentFingerPosition.x -
                   fingerPositionSnapped.x;
      if (fingerDiff <= snapWidth)
         return snapLineX;
      else if (fingerDiff > snapWidth &&
               fingerDiff <= snapWidth + catchUp)
         return linearInterpolation(
            fingerPositionSnapped.x, currentFingerPosition.x,
            snapWidth, catchUp);
      else
         isSnapped = false;
   }
   return objectBorder.rightX;
}
```

Fig. 4. Code fragment for the 1-D Oh Snap function for the case when the object is moved from left to right (lower to higher X coordinate). In this code `objectBorder` is a variable that references the boundary of the object and `rightX` is the x position of its rightmost edge. Additional cases are implemented similarly.

We implemented Oh Snap in C#, running on a SMART Table. We implemented three object movement types that are frequently performed on touch tables: 1-D translation, 2-D translation, and rotation. Rotation has been shown to be very useful on touch

tables, especially in collaborative situations [14]. For translation, the position of each edge of an object is checked against the position of all environment lines it is parallel to. Orthogonal edges in the 2-D environment can be snapped independently. For rotation, the snap width is measured as an angle rather than as a pixel count.

If a user snaps an object to a line and then lifts her finger from the screen, the object is no longer flagged as being snapped, whether it is aligned with the line or is within the (*snap width*) + (*catch-up width*) region. This is primarily useful if a user wishes to place an object near a snap line but has snapped the object and is having difficulty reaching the destination due to the object moving along super pixels. A code snippet for the 1-D Oh Snap function that handles an object moving from left to right is presented in Fig. 4. This uses the linear interpolation function in Fig. 2.

5. User Study

In order to objectively evaluate the performance of the Oh Snap technique, we performed a user study. The first phase focused on comparing the performance of Oh Snap to traditional snapping and to no-snapping for alignment tasks. The second phase investigated participants' ability to drag an object close to but not onto a line (i.e. to not snap the object to the line) when the Oh Snap feature is turned on.

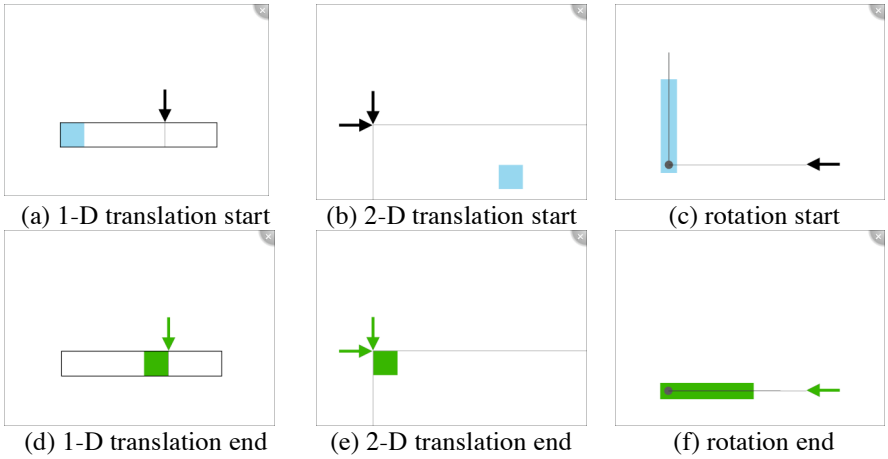
5.1 Comparison of Snapping Techniques – Phase 1 of the User Study

The purpose of this phase of the user study was to evaluate the Oh Snap technique and compare its performance to traditional snapping as well as to no snapping. We investigated performance with three object movement types: 1-D translation, 2-D translation, and rotation.

Task. A participant's task was to use a single finger on the right hand to move a digital blue square, as quickly as possible, so that its edge(s) were aligned with the desired target line(s). The target lines were indicated with a black arrow, which turned green when the object was aligned with the indicated line. When all edges were aligned correctly (1 edge for the 1-D and rotation tasks, 2 edges for the 2-D task), the square also turned green, indicating success. Screenshots of the start and end of successfully completed 1-D, 2-D, and rotation tasks are shown in Fig. 5.

At the start of a trial, a button at the bottom of the screen labeled 'GO' became active after one second. When the active button was touched, the button disappeared, the square became active (indicated by its changing color from grey to blue), and the participant could then begin the trial. The timer began when the participant first touched the square and it stopped when the participant lifted the finger from the screen with the appropriate edge(s) of the square aligned to the target line(s). Each trial required successful alignment. If a participant lifted the finger when the square was not fully aligned, the participant would have to touch the square and move it again to complete the trial.

The layouts of the three tasks were designed so that their components occupied the lower portion of the screen. This allowed participants to sit at the table and comfortably reach the digital objects. For the 1-D translation task, participants moved the square so that its right edge was aligned with a single target line. The right edge of the square began at a distance 305 pixels (17cm) to the left of the target line. In the 2-D translation task, the square was to be aligned so that its top edge and left edge were aligned to horizontal and vertical target lines, respectively. At the start of a trial, the top left corner of the square was 490 pixels (27cm) from the intersection corner of the target lines. In the rotation task, participants rotated a rectangle, anchored at one end, so that the central line protruding from it was parallel to the target line. The target line was horizontal and the anchored rectangle began vertical at the start of a trial, rotated 90 degrees from the target line. A pilot study showed that crossing multiple distracter snap lines did not affect performance time. To simplify instructions to participants we did not use distracter lines in the final version of the study.



(a) 1-D translation start (b) 2-D translation start (c) rotation start

(d) 1-D translation end (e) 2-D translation end (f) rotation end

Fig. 5. Screenshots of the three Phase 1 tasks at the start and end of a trial.

Interface techniques. There were three snapping interface conditions: *no snapping*, *Oh Snap*, and *traditional snapping*. Trial tasks were identical across conditions, but the snapping behaviour of the target lines differed. In the no snapping condition, the target line did not cause objects to snap, and the edge of the square had to be placed within $\pm$2 pixels of the appropriate target line(s). Pilot testing revealed that the no snapping tasks were nearly impossible to complete without a small tolerance due to the touch sensing limitations of the SMART Table and the fat finger problem.

In the traditional snapping condition, the target lines had the traditional snapping behaviour with a snapping threshold of 10 pixels so that if the appropriate square edge was within that threshold of the target line, it would automatically be translated to the target line position. In the Oh Snap condition, the target lines had the Oh Snap behaviour with a snap width set to 10 pixels and the catch-up width set to 10 pixels.

To conduct a fair comparison, we gave Oh Snap and traditional snapping the same threshold to make them as equivalent as possible.

Experimental Design. The study was a fully counter-balanced, within subjects 3×3 (Snapping Technique × Movement Task) design with 20 trials for each treatment. For every trial, we recorded the task completion time and the number of times participants lifted a finger from the table (referred to as a *touch up*).

Participants were given training and the opportunity to practice each movement task (with no snapping) at the start of a session. For data analysis, we discarded the first 5 trials of each treatment of 20 trials to reduce learning effects.

Participants. Eighteen right-handed participants (2 female) between the ages of 21 and 40 (μ = 26.7) with normal color vision were recruited from our institution. Six participants had previously used a tabletop display, but only briefly during demos or to play games. Each participant received $10 for participating. Prior ethical approval was obtained from our university's behavioral research ethics board.

Apparatus. The user study was conducted on a SMART Table from SMART Technologies. The table had a 57.2cm × 42.9cm screen, with a resolution of 1024 × 768 pixels, and a 70Hz refresh rate. The application used for this user study was written in C# using the SMART Table SDK.

Hypotheses. We had two hypotheses: (1) Participants would perform faster with Oh Snap than with no snapping, and (2) participants would not perform slower with Oh Snap compared to traditional snapping. We were also interested in seeing how snapping technique would impact *touch ups*, the number of times users lift a finger from the table during a task.

Results for Phase 1. We performed a repeated measures ANOVA on the dependent variable trial time. A Bonferroni adjustment was applied to all pair-wise comparisons. Mean trial times for all snapping techniques across all movement types are shown in Table 2.

Table 2. Mean trial time (in seconds) for each of the snapping techniques across all movement types.

Snapping Technique	Movement Type			
	All	**1-D**	**2-D**	**Rotation**
No snapping	2.507	2.301	3.517	1.703
Oh Snap	1.513	1.582	1.667	1.287
Traditional snapping	1.387	1.362	1.651	1.147

Performance. As shown in Table 2, the average trial time in seconds was 2.507 for no snapping, 1.513 for Oh Snap, and 1.387 for traditional snapping. There was a significant main effect of snapping technique ($F(2,34)=30.062$, $p<.0005$, $\eta^2=.639$) and movement type ($F(2,24)=27.529$, $p<.0005$, $\eta^2=.618$). There was a significant interaction effect of snapping technique × movement type ($F(4,68)=9.493$, $p<.007$).

We ran pair-wise comparisons between techniques and found that both Oh Snap ($p<.0005$) and traditional snapping ($p<.0005$) were faster than no snapping, but traditional snapping was not significantly faster than Oh Snap. We also ran pair-wise comparisons on movement types and found significant differences between all pairs: 1-D and 2-D ($p<.005$), 1-D and rotation ($p<.0005$), and 2-D and rotation ($p<.0005$).

To understand how snapping technique impacted performance for each movement type, we ran a repeated measures ANOVA for each of the three movement types. There was a significant effect of snapping technique for rotation ($F(2,16)=17.634$, $p<.0005$, $\eta^2=.688$), 1-D translation ($F(2,16)=14.411$, $p<.0005$, $\eta^2=.643$), and 2-D translation ($F(2,16)=10.128$, $p<.001$, $\eta^2=.559$). Both Oh Snap and traditional snapping were significantly faster than no snapping for all movement types ($p<.0005$). Traditional snapping was significantly faster than Oh Snap only in the 1-D task ($p<.01$); however, it was only 14% faster whereas Oh Snap was 31% faster than no snapping in the 1-D task. A chart showing the mean trial times for each movement type grouped by snapping technique is in Fig. 6.

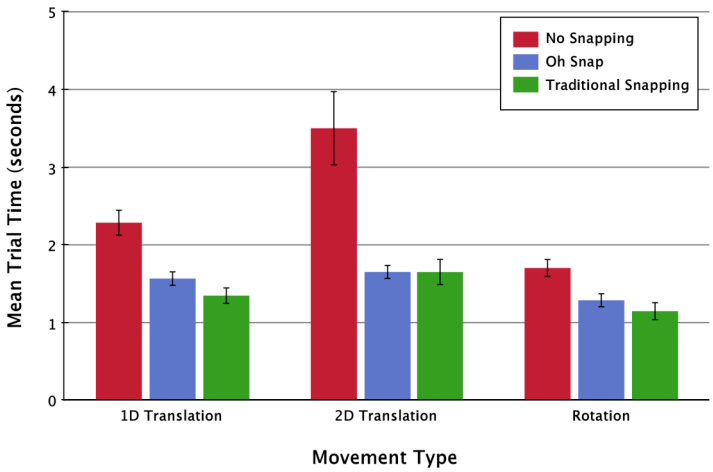


Fig. 6. Mean trial time by movement type grouped by snapping technique. Error bars represent standard error.

Number of touch-ups. We also conducted a 3×3 (Snapping Technique × Movement Task) repeated measures ANOVA for the average number of touch ups in a set of trials. There was a significant main effect of both snapping technique ($F(2,34)=21.427$, $p<.0005$, $\eta^2=.558$) and movement task ($F(2,34)=9.45$, $p<.001$, $\eta^2=.357$). Pair-wise comparisons showed that participants lifted their finger up

significantly more in the no snapping condition (μ=1.438) compared to either Oh Snap (μ=1.19) or traditional snapping (μ=1.185) with $p<.0005$ for both. There was no significant difference in the number of touch ups between Oh Snap and traditional snapping. There was a significant interaction effect for snapping technique × movement type ($F(4,68)=6.274$, $p<.023$).

Questionnaire data. Questionnaires were administered after each snapping technique to collect subjective ratings and receive comments using TLX-based Likert-style questions. Pair-wise comparisons using Wilcoxon Signed Ranks Tests for each of the questionnaire questions are presented in Table 3.

Participants ranked the three snapping techniques from best to worse in terms of preference. A Friedman test showed that technique significantly impacted rankings $\chi^2_{(2,N=18)}=10.371$, $p<.006$). Both Oh Snap and traditional snapping had a mean rank of 1.67 while no snapping had a mean rank of 2.56 (lower is better).

Discussion of Phase 1. Both the Oh Snap and the traditional snapping techniques were significantly faster than no snapping, supporting hypothesis (1). On average, Oh Snap was 40% faster than no snapping. Because traditional snapping was not significantly faster than Oh Snap on average, hypothesis (2) is also supported. As reported in the results, although traditional snapping was significantly faster than Oh Snap in the 1-D tasks, it was only 14% faster, but Oh Snap was 31% faster than no snapping for such tasks. This small degradation suggests that Oh Snap is a reasonable alternative to traditional snapping for alignment tasks on touch interfaces, and it has the very important added benefit that users can place objects in close proximity to snap locations.

Table 3. Summary of significance differences between snapping techniques for each post-condition questionnaire question. **A**=Oh Snap, **B**=No snapping, and **C**=Traditional snapping. Non-significant comparisons are labeled *n.s.*

Significant differences between snapping techniques in subjective answers			
Question	**Snapping Technique**		
	A better than B	**C better than A**	**C better than B**
Mental demand	.013	*n.s.*	.003
Physical demand	.033	*n.s.*	*n.s.*
Task difficulty	.005	.021	.001
Success	.03	*n.s.*	.007
Hard work	.003	*n.s.*	.003
Fulfilled	.001	*n.s.*	.0005

Although this phase of the user study showed that Oh Snap performs well for alignment tasks, it did not investigate how Oh Snap might benefit proximity placement. To properly assess this feature, we conducted a second phase of the user study to compare several variations of the snap width and catch-up width parameters to find the best values for each.

5.2 Parameters for Snap Line Proximity – Phase 2 of the User Study

This phase investigated the variation of the Oh Snap parameters *snap width* and *catch-up width*, and measured their effect on user performance for positioning digital objects in close proximity with (but not aligned to) a snap line. We compared Oh Snap variants to the no snapping technique. We hoped to find a balance between a small ratio (Eq. 1) to reduce super pixel size, and minimizing the sum *catch-up width* + *snap width* so that it was not much larger than the average touch contact width.

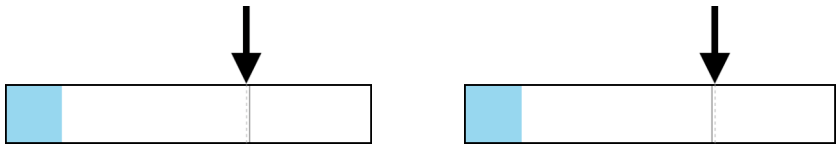
Apparatus and Participants. The apparatus and the participants were the same as in Phase 1. Participants started Phase 2 shortly after they completed Phase 1.

Task. The task was very similar to the 1-D alignment task from Phase 1. Participants were asked to translate a square so that it was close to, but not quite aligned with, a snap line. A dotted target line, with an arrow pointing to it, was placed 5 pixels before or 5 pixels after the Oh Snap line. Participants were asked to drag the square so that its right edge was aligned with the dotted target line. As in Phase 1, once the square was aligned with the target line, both the arrow and the square turned green to indicate successful alignment.

A screenshot of the start of each of the tasks (after the 'GO' button had been pressed) is shown in Fig. 7.

Interfaces. There were four sets of snapping parameters.

- no snapping
- a 4/3 ratio with 15px catch-up width and 5px snap width
- a 3/2 ratio with 20px catch-up width and 10px snap width
- a 4/3 ratio with 30px catch-up width and 10px snap width



(a) Target line is *before* snapping line. (b) Target line is *after* snapping line.

Fig. 7. Screenshots of each of the two tasks at the start of a trial in Phase 2.

Traditional snapping was not used because the target lines were within the traditional snapping threshold area and thus impossible to reach with that technique.

We chose these parameter sets based primarily on our findings during pilot testing, when we found that ratios of less than 3/2 made it difficult to accomplish proximity tasks. We chose 40 pixels, or 2.24cm on the SMART Table, as the largest *catch-up width* + *snap width* to investigate if a width larger than the average touch contact width of 1.44cm [31] would yield positive results. A large value for *catch-up width* + *snap width* allows for tenacious snapping of an object's position while also maintaining imperceptibility, both of which are desirable. If that size gave good results, future Oh Snap implementations would have to take great care not to overlap the snap and catch-up regions of different snappable lines.

Experimental Design. The study was a partially counter-balanced, within subjects 4×2 (Snapping Parameter Set × Proximity Task) design with 20 trials for each treatment. For each trial, we recorded the task completion time. As in Phase 1, participants were given training and the opportunity to practice each movement task (with no snapping) at the start of a session. For the purposes of data analysis, we discarded the first 5 trials of each treatment of 20 trials to reduce learning effects.

Hypotheses. We had two hypotheses: (3) The Oh Snap conditions would be no slower than the no snapping condition, and (4) participants would perform fastest with the largest catch-up-size-to-snap-width-ratio and the largest snap-width. When the ratio is large (i.e. the catch-up size is 3 times larger than the snap width) the size of super pixels is decreased so it should be easier to perform fine-grained movement of a digital object that has been snapped. We expected that participants would perform better placing objects just before the snap line than just after it. Because objects could avoid being snapped when placing them before the snap line, this proximity task would be less difficult.

Results for Phase 2. We ran a repeated measures ANOVA on the dependent variable trial time. A Bonferroni adjustment was applied to all pair-wise comparisons. Mean trial times for all Oh Snap parameter sets across both proximity tasks are presented in Table 4.

Performance. There was a significant main effect of Oh Snap parameter set ($F(3,51)=12.702$, $p<.0005$, $\eta^2=.428$) and proximity task ($F(1,17)=6.626$, $p<.02$, $\eta^2=.28$). Mean trial times for all parameter sets are presented in Fig. 8.

Table 4. Mean trial time (seconds) for each of the Oh Snap parameter sets across both proximity types (just before and just after a snap line).

snap width	catch-up width	Ratio	Proximity Type		
			Both	Before	After
No snap	No snap	N/A	2.309	2.236	2.382
5px	15px	4/3	3.350	3.689	3.012
10px	20px	3/2	3.559	3.919	3.198
10px	30px	4/3	3.662	4.128	3.196

Pair-wise comparisons revealed that the no snapping condition was significantly faster than all parameter sets ($p<.0005$ for the first, $p<.002$ for the second, $p<.0005$ for the third). Participants did not complete trials significantly faster with any Oh Snap parameter set compared to any other. There was a significant interaction effect of parameter set × proximity task ($F(3,15)=3.884$, $p<.031$). Mean trial times for each proximity task type grouped by Oh Snap parameter set are shown in Fig. 8.

Surprisingly, the proximity task that required participants to position the square just before the snap line took significantly longer than positioning the square just after the line ($p<.02$). We anticipated that because the objects could avoid being snapped when placing them before the snap line, this proximity task would be less difficult and

faster as a result. In sharp contrast, squares positioned after the snap line must be snapped and therefore we thought they would take longer.

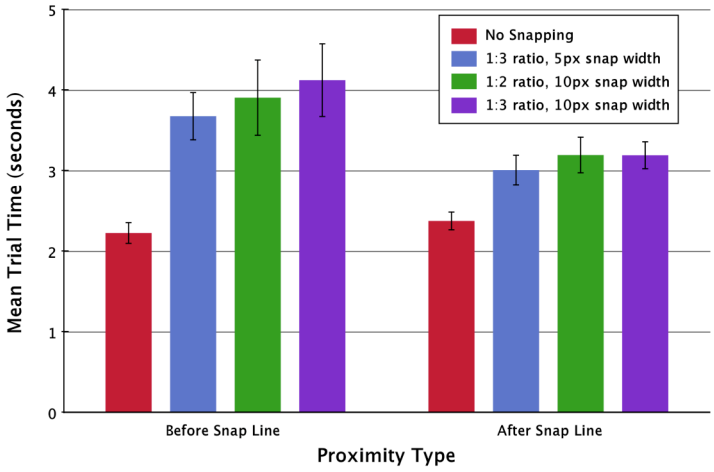


Fig. 9. Mean trial time by proximity type grouped by Oh Snap parameter set. Error bars represent standard error.

Perhaps participants knew that snapping would occur in the 'after' task and learned how to adjust their movements to accommodate it. Conversely, in the 'before' task, participants may have worked slower so that they did not snap to the line. Participants may have occasionally overshot the target position, though not always, resulting in inconsistent behaviour preventing them from mastering the task.

Number of touch ups. We conducted a 4 × 2 (Parameter Set × Proximity Task) repeated measures ANOVA for the number of touch ups in each treatment. There was a significant main effect of both parameter set ($F(3,51) = 6.79$, $p<.001$, $\eta^2=.285$) and proximity task ($F(1,17)=5.236$, $p<.035$, $\eta^2=.235$). Pair-wise comparisons of Oh Snap parameters revealed the no-snapping condition had significantly fewer touch ups than the second ($p<.002$) and fourth ($p<.008$) parameter sets, but not the third.

Discussion of Phase 2. Hypotheses (3) and (4) were not supported: the no snapping condition was faster than the three Oh Snap conditions and there were no significant differences in mean trial time between the parameter sets. While this means that there is a price, in terms of raw speed, for using Oh Snap, if snapping is desired the no-snapping condition is not a viable option. Oh Snap's real competitors are other snapping techniques. The advantage it has is that it permits placement of objects at any location, including right near a snap line. There is very little penalty for doing this compared to no snapping, and it cannot be done at all with other techniques.

There was no difference between the number of touch ups in the no snapping condition and in the condition using the third Oh Snap parameter set (3/2 ratio, 10px snap width). We recommend those parameters for future Oh Snap implementations.

6 Conclusions & Future Work

We introduced and evaluated a novel snapping technique supporting the alignment of digital objects on a touch interface. Oh Snap maintains the position of objects underneath the user's finger. It allows placement of objects close to snap lines, without having to toggle the snap capabilities off and on. This feature is especially important for collaborative environments. Oh Snap offers a mean 40% performance improvement over no snapping for 1-D, 2-D and rotation tasks, and it does not perform significantly worse than traditional snapping for those tasks on a touch table.

We investigated three parameter sets for Oh Snap in two proximity placement tasks. The no snapping condition performed better than the parameter sets, but the differences were not very large. Placing objects near a snap line is an infrequent task compared to snapping objects or placing objects in open space, both of which are tasks at which Oh Snap excels. We argue that, for direct touch surfaces, Oh Snap is superior to no snapping or traditional techniques because of its unique combination of attributes (Table 1). Furthermore, the lack of a significant difference between parameter sets for Oh Snap indicates that there is probably a wide range of suitable values, meaning that developers can choose values that best suit their users' needs.

6.1 Comparison to Nacenta et al.'s prior work

The Oh Snap velocity profile is similar to the "magnitude filtering" used by Nacenta et al. [20] to block selected degrees of freedom. In their approach velocity profiles are used to determine whether motion is the beginning of a constrained gesture or if it is "noise" in the current gesture. In contrast to this, Oh Snap employs the profile not just when movement is *initiated*, but *whenever* a snap line is encountered. Oh Snap uses the velocity profile *within* a chosen degree of freedom (1-D or 2-D translation, or rotation) to decide when to *stop*, whereas Nacenta et al. use it to *change* or *start* a chosen degree of freedom.

Nacenta et al. use symmetric constraint regions. Our regions are asymmetric; no snapping occurs approaching a snap line, only after it is passed. This makes it less likely a target just ahead of a snap line will be "missed." If a target just after a snap line is missed, "backing up" will not be handicapped because the asymmetry means that no snapping is applied in the reverse direction once the object becomes un-snapped.

6.2 Extensions to Oh Snap

Oh Snap works equally well on a traditional single-touch table or on a multi-touch table. If multi-touch is available, the number of touches could be used as a mode

switch, which would allow seamless transitions between snapping and no-snapping. Future work for these and other snapping techniques will involve a new bimanual Oh Snap interaction that does not require explicit implementation of system-defined snap lines. An Oh Snap user will touch the interface with two fingers of the same hand and which define a temporary snap line to which they can align an object.

Acknowledgements

This research was funded in part by the Networks of Centres of Excellence of Canada through GRAND, the Graphics, Animation and New Media NCE, by the Natural Sciences and Engineering Council of Canada under the postgraduate scholarship program and the discovery grants program, by the University of British Columbia through a University Graduate Fellowship, and by an equipment donation from SMART Technologies. Defense Research and Development Canada supplemented the NSERC postgraduate scholarship. Additional equipment and other infrastructure necessary for the research were provided by funding from the Canadian Foundation for Innovation and the British Columbia Knowledge Development Fund.

References

1. Ahlström, D., Hitz, M., Leitner, G.: An evaluation of sticky and force enhanced targets in multi target situations. In Proc. NordiCHI '06, pp. 58--67, (2006)
2. Albinsson, P.-A., Zhai, S.: High precision touch screen interaction. In Proc. CHI '03, pp. 105--112, (2003)
3. Aliakseyeu, D., Subramanian, S., Lucero, A., Gutwin, C.: Interacting with piles of artifacts on digital tables. In Proc. AVI '06, pp. 159--162, (2006)
4. Baudisch, P., Cutrell, E., Hinckley, K., Eversole, A.: Snap-and-go: helping users align objects without the modality of traditional snapping. In Proc. CHI '05, pp. 301--310, (2005)
5. Benko, H., Wilson, A.D., Baudisch, P.: Precise selection techniques for multi-touch screens. In Proc. CHI '06, pp. 1263--1272, (2006)
6. Bier, E.A.: Snap-dragging in three dimensions. In Proc. I3D '90, pp. 193--204, (1990)
7. Bier, E.A., Stone, M.C.: Snap-dragging. In Proc. SIGGRAPH '86, pp. 233--240, (1986)
8. Bjørneseth, F.B., Dunlop, M.D., Strand, J.P.: Dynamic positioning systems: usability and interaction styles. In Proc. NordiCHI '08, pp. 43--52, (2008)
9. Frisch, M. Dachselt, R.: Benefits of interactive display environments in the software development process. In Proc. CHASE '08, pp. 53--56, (2008)
10. Gleicher, M.: Image snapping. In Proc. SIGGRAPH '95, pp. 183--190, (1995)
11. Gutwin, C. Greenberg, S.: Design for individuals, design for groups: tradeoffs between power and workspace awareness. In Proc. CSCW '98, pp. 207--216, (1998)
12. Holz, C., Baudisch, P.: The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In Proc. CHI '10, pp. 581--590, (2010)
13. Khalilbeigi, M., Steimle, J., Muhlhauser, M.: Interaction techniques for hybrid piles of documents on interactive tabletops. In Proc CHI EA '10, pp. 3943--3948, (2010)

14. Kruger, R., Carpendale, S., Scott, S.D., Greenberg, S.: How people use orientation on tables: comprehension, coordination and communication. In Proc. GROUP '03, pp. 369--378, (2003)
15. Kruger, R., Carpendale, S., Scott, S.D., Tang, A.: Fluid integration of rotation and translation. In Proc. CHI '05, pp. 601--610, (2005)
16. Lecuyer, A.: Simulating haptic feedback using vision: A survey of research and applications of pseudo-haptic feedback. Presence: Teleoper. Virtual Environ., 18(1):39--53, (2009)
17. Liu, J., Pinelle, D., Sallam, S., Subramanian, S., Gutwin, C.: TNT: improved rotation and translation on digital tables. In Proc. GI '06, pp. 25--32, (2006)
18. Mandryk, R.L., Rodgers, M.E., Inkpen, K.M.: Sticky widgets: pseudo-haptic widget enhancements for multi-monitor displays. In Proc. CHI '05, pp. 1621--1624, (2005)
19. Micire, M., Desai, M., Courtemanche, A., Tsui, K.M., Yanco, H.A.: Analysis of natural gestures for controlling robot teams on multi-touch tabletop surfaces. In Proc. ITS '09, pp. 41--48, (2009)
20. Nacenta, M.A., Baudisch, P., Benko, H., Wilson, A.: Separability of spatial manipulations in multi-touch interfaces. In Proc. GI '09, pp. 175--182, (2009)
21. Nóbrega, R., Sabino, A., Rodrigues, A., Correia, N.: Flood emergency interaction and visualization system. In Proc. VISUAL '08, pp. 68--79, (2008)
22. Olwal, A., Feiner, S.: Spatially aware handhelds for high-precision tangible interaction with large displays. In Proc. TEI '09, pp. 181--188, (2009)
23. Olwal, A., Feiner, S., Heyman, S.: Rubbing and tapping for precise and rapid selection on touch-screen displays. In Proc. CHI '08, pp. 295--304, (2008)
24. Raisamo, R., Raiha, K.-J.: A new direct manipulation technique for aligning objects in drawing programs. In Proc. UIST '96, pp. 157--164, (1996)
25. Reetz, A., Gutwin, C., Stach, T., Nacenta, M., Subramanian, S.: Superflick: a natural and efficient technique for long-distance object placement on digital tables. In Proc. GI '06, pp. 163--170, (2006)
26. Ringel Morris, M., Paepcke, A., Winograd, T., Stamberger, J.A.: TeamTag: exploring centralized versus replicated controls for co-located tabletop groupware. In Proc. CHI '06 pp. 1273--1282, (2006)
27. Siek, K.A., Rogers, Y., Connelly, K.H.: Fat finger worries: How older and younger users physically interact with PDAs. In Proc. INTERACT '05, pp. 267--280, (2005)
28. Sutherland, I.E.: Sketchpad: a man-machine graphical communication system. In Proc. AFIPS '63, pp. 329--346, (1963)
29. Vogel D., Baudisch, P.: Shift: A technique for operating pen-based interfaces using touch. In Proc. CHI '07, pp. 1063--1072, (2007)
30. Voida, S., Tobiasz, M., Stromer, J., Isenberg, P., Carpendale, S.: Getting practical with interactive tabletop displays: designing for dense data, "fat fingers," diverse interactions, and face-to-face collaboration. In Proc. ITS '09, pp. 109--116, (2009)
31. Wang, F., Ren, X.: Empirical evaluation for finger input properties in multi-touch interaction. In Proc. CHI '09, pp. 1063--1072, (2009)
32. Wigdor, D., Williams, S., Cronin, M., Levy, R., White, K., Mazeev, M., Benko, H.: Ripples: utilizing per-contact visualizations to improve user interaction with touch displays. In Proc. UIST '09, pp. 3--12, (2009)
33. Wobbrock, J.O., Morris, M.R., Wilson, A.D.: User-defined gestures for surface computing. In Proc. CHI '09, pp. 1083--1092, (2009)
34. Yang, L., Wang, F., Deng, H.: A survey on target selection technique for touch sensing devices. In Proc. MVHI '10, pp. 534--537, (2010)