Ether-Toolbars: Evaluating Off-Screen Toolbars for Mobile Interaction

Hanaë Rateau, Yosra Rekik, Edward Lank, Laurent Grisoni

Ether-Toolbars: Evaluating Off-Screen Toolbars for Mobile Interaction

Hanae Rateau
ESTIA & University of Lille Sci. and Tech, CNRS
hanae.rateau@inria.fr

Yosra Rekik
University of Lille Sci. and Tech, CNRS
yosra.rekik@univ-lille.fr

Edward Lank
University of Waterloo, INRIA
lank@uwaterloo.ca

Laurent Grisoni
University of Lille Sci. and Tech, CNRS
laurent.grisoni@univ-lille.fr

ABSTRACT

In mobile touchscreebn interaction, an important challenge is to find solutions to balance the size of individual widgets against the number of widgets needed during interaction. In this work, to address display space limitations, we explore the design of invisible off-screen toolbars (ether-toolbars) that leverage computer vision to expand application features by placing widgets adjacent to the display screen. We show how simple computer vision algorithms can be combined with a natural human ability to estimate physical placing to support highly accurate targeting. Our ether-toolbar design promises targeting accuracy approximating on-screen widget accuracy while significantly expanding the interaction space of mobile devices. Through two experiments, we examine off-screen content placement metaphors and off-screen precision of participants accessing these toolbars. From the data of the second experiment, we provide a basic model that reflects how users perceive mobile surroundings for ether-widgets and validate it. We also demonstrate a prototype system consisting of an inexpensive 3D printed mount for a mirror that supports ether-toolbar implementations. Finally, we discuss the implications of our work and potential design extensions that can increase the usability and the utility of ether-toolbars.

Author Keywords

Around-device interaction; ether-widgets; ether-toolbar; deformation estimation; model; user study; mobile interaction

INTRODUCTION

Modern mobile computing devices have ever-increasing processing power and memory, thus allowing increasingly feature-rich applications. Unfortunately, feature rich interfaces often require access to these features through on-screen widgets. These widgets, while necessary, occupy screen pixels on a device for which screen size is heavily constrained. To address this issue of limited on-screen space, some previous works have proposed interaction techniques that allow the user to take advantage of a virtual or invisible space around the device either by considering the device as a window into a larger display space [15], or by arranging additional display content around the device [14], *around-device interaction*.

This paper explores the design of single-screen application interfaces using an interaction paradigm in which one part of the application interface – typically widgets – exists outside of the display screen but is invisible, thus reserving the on-screen display solely for application content. In this way we leverage both human memory and mobile device surroundings. We call such invisible off-screen widgets, *ether-widgets*, inspired from past research on representations for virtual space between displays [2].

We focus here specifically on ether-toolbars; arguably the toolbar is the widget that best exposes single-click access in a high-density container. In order to explore the design of ether-toolbars, we first examine three competing display metaphors that put ether-widgets off-screen: a ghost display similar to a Manhattan-style focus+context interface [5]; a zoom-based metaphor, which zooms the display such that widgets are scaled and translated off-screen; and a translation-based metaphor, where widgets are simply directly translated off-screen. The results of this study argue for the advantages of a translation-based metaphor. We then design a simple implementation of an ether-toolbar, leveraging the translation metaphor, and measure performance on the ether-toolbar in contrast to performance with on-screen toolbars. From the collected data, we leverage a model that estimates the ether-toolbar deformation as perceived by users and we validate it. Finally, we present a prototype drawing application that demonstrates ether-widgets, including a simple computer vision system that supports self-calibration of around-display input. We conclude by discussing potential enhancements to ether-toolbars and implications for the design of additional

ether-widgets in applications.

RELATED WORK

Researchers have frequently explored how non-display physical space around screens can be used to boost the efficiency, accuracy, or expressivity of input during direct input. Most relevant to our research, this section explores work in off-screen or around-device interactions.

Different ways have been explored to make the device's surroundings interactive. For example, in SideSight [3], Butler *et al*.augment the mobile device with infra-red proximity sensors along the sides of the device. These sensors can detect and track finger position near the device enabling multi-touch interaction around the device. Hoverflow [16] describes an augmented Apple Iphone with proximity sensors that recognizes *gestures above* the device. Medusa [1] also uses proximity sensors all around a tabletop in order to sense users arms. This enables to map a touch to an arm and a user. The Abracadabra input technique [11] from Harrison *et al*.uses magnetometers added to a smartwatch and a small magnet attached to the user's finger to sense the finger position in the space around the watch. Toffee [19] enables around-device interaction using acoustic time-of-arrival. A device is augmented with four piezo discs. Thanks to this setup, touches can be located when performed. Thus, it allows to place virtual buttons around the device. These works propose to modify the device in order to make it aware of its near surroundings. In this work, we present a 3D printed mount that just clips onto the tablet enabling finger tracking and contour detection around the tablet.

There have been numerous investigations into understanding and evaluating the performance of off-screen interaction. Research has focused on understanding off-screen midair pointing in static visualization conditions. For instance, Hasan et al. [13] examine the advantage of off-screen pointing in terms of navigation time when comparing off-screen pointing with Peephole and standard Flick & Pinch for map navigation – with or without the help of an overview. Gustafson et al. [9] measured the performance of "coordinate-based imaginary pointing" when using the non-dominant hand as a reference. Their findings showed that the performance deteriorated significantly when increasing distance between the target and the reference hand. Finally, Ens et al. [6] investigated off-screen pointing performance for Fitt's task with dynamic and continuous feedback. Two novel pointing techniques were proposed namely,*launch-adjust* and *glide*. In both of these techniques, a continuous feedback on the target position and the finger position is provided by using either the *wedge* technique [8] or an overview of the scene. Their findings indicated that further targets should be larger to minimize the error. Another interesting result is that the direction has a significant effect on the movement time. In particular, diagonal direction increases the movement time and that users tend to undershoot the target.

In the domain of pointing performance, Markussen et al. [18] investigated off-screen midair horizontal pointing performance on a distant display. They showed that the perceived space around the screen is modified depending on how far is the intended target position from the screen. Users tend to guess the intended position closer than it is from the screen. Hasan et al. [12] found that for midair target selection a *DownUp* gesture decreases the error rate but increases the trial time. In contrast, *LiftOff* gesture provides the best compromise between error rate and trial time. In their study of targeting around devices, they also indicated that the radial bins should be bigger than 4 cm wide and the input range should not extend beyond 40 cm around the device.

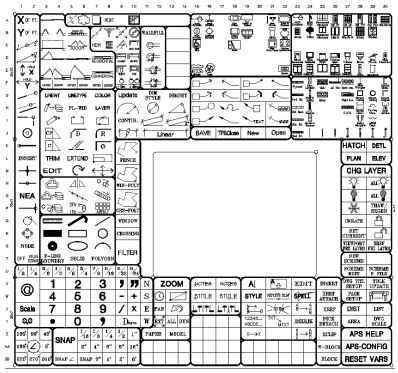


Figure 1: The AutoCAD tablet overlay.

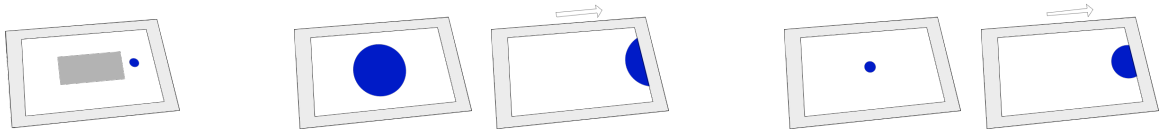
In this paper, we are particularly interested in the placement of widgets around the device. The inclusion of around device widgets to support interaction has been leveraged in commercial systems in the past to support feature-rich applications. As one simple example of this, versions of AutoCAD included tablet overlays (Figure 1) that positioned widgets on a data tablet outside the screen area. The advantage of this interaction was that a widget-space could be created for feature-rich applications significantly larger than the display. The disadvantage of using around-device space is the need to either memorize widget placement or the need to provide some targeting aid such as the AutoCAD overlay.

EXPLORING ETHER-WIDGET METAPHORS

Our goal in this work is to explore around-device space as a placement for widgets within an application. While others have explored the placement of content around the device, we are not aware of any work exploring, specifically, the placement of widgets.

In typical feature rich personal computer applications, extensive command sets are often contained in a toolbar-like interface such as a ribbon [10]. While every interface widget has trade-offs associated with it, one advantage of the ribbon is its ability to allow users to prioritize recognition over recall. This particularly benefits novice users: it exposes functionality so the functionality can be discovered visually during interaction. However, the problem with the ribbon is that it consumes significant screen real-estate. For example, Word's ribbon contains up to three rows of widgets. If one images porting this to a multi-touch computer, and one adopts Android design guidelines, then a ribbon like toolbar will consume at least 7mm minimum target size plus 1mm minimum target spacing time three rows of target = 2.5cm of on-screen space.

Our goal with off-screen target placement is to allow expert users to access features when they know the location of the



(a) Focus+Context (b) Translation: before translation (c) Translation: during translation (d) Zoomed: before zooming (e) Zoomed: during zooming

Figure 2: TThe three Ether-Widgets *metaphors* for target number 2. N.B: The arrow was not displayed to participants.

features. However, to support the novice experience, it would also be good if widgets could be displayed on-screen, could migrate off during interaction, and could be recalled when needed so that users can find their desired command.

Given a user interface with many widgets and a content area, if the desire is to move the widgets off-screen we propose three primary metaphors for off-screen placement. First, the screen could be considered a porthole into a larger space or map area, and widgets could be located at a fixed location somewhere else in the ether; a peephole or focus+context widget arrangement. Second, the display could zoom into the content area, allowing the widgets to drift naturally off the edge of the display. Finally, the widgets could be contained in a panel that simply moves off-screen. In this section, we evaluate the three metaphors for around-device widget placement, *i.e.*, the placement of ether-widgets. Figure 2 presents each of these options.

Focus+Context Ether-Widgets

The first metaphor considers the display to be a type of focus+context viewport with a high resolution focus and an alpha-blended context that shows a scaled view of around-device widget space. Many researchers have explored the idea of embedded high-resolution displays within a lower resolution or larger surrounding environment [17, 4]. Our implementation is most similar to a *Manhattan* embedding as described by Carpendale *et al*. [4] where a high-resolution focus area overlaps the context. We use alpha-blending to represent the surrounding context during interaction. This focus+context metaphor has analogs to, for example, overview maps embedded in command-and-control games.

Zoomed Ether-Widgets

The second considers a zoomed version of the interface such that, as a result of zooming, the widgets move off-screen into the "ether" around the display. Zooming is a common technique used to allow detailed exploration of a region of interest while preserving the existence of context. In this implementation, during interaction the interface is enlarged such that content consumes the entire screen. This has the added benefit that ether-widgets are larger in size (and hence easier to target) than they would be if they existed on-screen.

Translation Ether-Widgets

In the third metaphor, we examine an approach where widgets are translated off the display. Translation, or linear movement, may be the easiest-to-track displacement of widgets because of the one-to-one mapping of perceived movement to actual movement (unlike, *e.g.*, zooming, where users must scale screen space to estimate distance). Translation is a common metaphor for toolbars and ribbons in standard graphical interfaces when these widgets are set to auto-hide. The animation depicts a translation of the widget into the windows titlebar. Given that the use translation is common for auto-hide in graphical interfaces, this metaphor may be the most familiar to users. As well, translation can be designed such that its perceived effect on content is minimized compared to zoom.

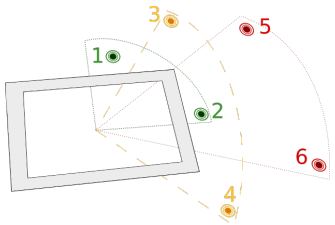


Figure 3: Targets positions where green targets are the near targets, the orange are the middle and the red ones are far ones.

EXPERIMENT 1: PILOT STUDY

We conducted a pilot study with eight participants to compare the impact of the three ether-widgets metaphors when selecting an off-screen target. The goal of this experiment was to contrast metaphors for off-screen target placement. The experiment ran on a 8.4-inch Samsung Galaxy Tab S is placed on a table. Participants sat at the table. To track the tablet, right hand and right index position, we used the OptiTrack system. One rigid body was attached to a glove we equipped the participants with. Another rigid body was attached to the tablet. For the index, we attached one marker to the fingertip. We implemented a simple tap recognizer that analyzed the Optitrack data. When a tap was performed, it was send to a NodeJS server with its coordinates in the reference frame of the tablet.

The task required participant to tap on an ether (off-screen) target as quickly and accurately as possible. Targets were displayed randomly one at a time within the off-screen space (see Figure 3). Each trial began after the previous target was selected and ended with the selection of the current target. In the *Focus+Context* condition, a gray rectangle representing the tablet surrounded by the off-screen space that contains the actual target are displayed on the screen (see Figure 2(a)). To start the trial, participant had to press the "start" button. In the *Translation* condition, first the target with its actual size (4 cm; 6 cm or 10 cm) and the background are displayed on the screen. Then participant had to press the "start" button. Both the target and background are then moved until the target reaches its

corresponding position in the ether-space (see Figure 2(b) and (c)). In the *Zoomed* condition, first the target with a diameter of 2 cm and the background are displayed on the screen. Then participant had to press the "start" button. The target is then simultaneously scaled and translated until it gets its corresponding size (4 cm; 6 cm or 10 cm) and position in the ether-space (see Figure 2(d) and (e)). We counterbalanced the ordering of ether-widget placement metaphors. We measured time, accuracy, and subjective responses from participants collected on a Likert scale.

RESULTS

We analyzed time, error, and subjective responses. All statistical analyses are multi-way ANOVA. Tukey tests are used post-hoc when significant effects are found. In the following, we report the results for each of the dependent variables.

We found mixed results for time and error dependent variables. For time, only *position* was significant ($F_{5,25} = 3.16$, $p = .023$), and the effect was not large. There were no other effects and no interactions.

Similarly, for accuracy, while significant effects were found for *metaphor* ($F_{2,10} = 10.27$, $p = .0038$), *size* ($F_{2,10} = 28.75$, $p < .0001$) and *position* ($F_{2,25} = 5.37$, $p = .0001$) with *metaphor* $\times$ *position* ($F_{10,59} = 2.35$, $p = .022$) and *size* $\times$ *position* ($F_{10,59} = 2.42$, $p = .019$) interactions on *accuracy*, post-hoc tests revealed that no one technique was consistently best. *Focus+context* is significantly more accurate than *zoomed* metaphor when selecting the targets 4 and 6 (far from the display), but *zoomed* has significant time $\times$ target effects. There are no significant advantages or disadvantages for the *translate* technique with report to other techniques.

In overall ranking, the *translation* metaphor was ranked 62.5% first, 25% second and 12.5% third. *Focus+context*, the next most preferred metaphor, was ranked 25% first, 37.5% second and 37.5% third. The *zoomed* technique was ranked 12.5% first, 37.5% second and 50% third.

ETHER-TOOLBAR DESIGN

In this section, we explore the design and efficacy of one potential ether-widget, an ether-toolbar. The overall goal of this follow-on study is to identify expected precision of ether-widgets for denser arrangements.

Results of the pilot study indicate that the translate metaphor is the most preferred and is statistically similar in speed and accuracy to other off-screen metaphors. As a result, we use the translate metaphor to place our ether-widget. One advantage to translate is that, conceptually, it is similar to widgets such as ribbons, where the animated auto-hide feature can be viewed as a translation of the widget to ether-space.

Ether-toolbars themselves may present both advantages and disadvantages as potential off-screen targets. First, their rectangular grid can be placed adjacent to the device, simplifying estimation of widget position; however, the density of their arrangement confounds accurate targeting, potentially nullifying any advantage of proximal placement. Second, toolbars have a natural migration pattern to drift off-screen via auto-hide options on standard toolbars, and this metaphor can be leveraged to support off-screen migration. Figure 4 shows a conceptual design for an Ether-Toolbar. During normal interaction, the toolbar is above the screen, and interacting in ether-space activates commands. Users can also choose to reveal the toolbar – for example, if they are novice users and do not know the location of controls. They do this using a bezel swipe, a standard mobile interaction technique that can be used to reveal above device controls.

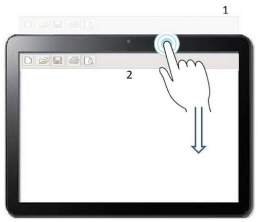


Figure 4: Conceptualization of Ether-Toolbar design. In typical interaction, the toolbar is positioned along the top of the tablet display, and, when a user interacts, they tap above the display on the toolbar in ether-space (1). By swiping down on the bezel, the toolbar can be revealed (2) and novice users can interact with it on the display. After interacting, it will be translated up and off the display again.

One advantage of this design is that it can co-exist naturally with standard mobile applications designed for tablets or smartphones (*e.g.*bezel swipe to reveal notifications). This technique, in novice mode, can be used identically. What ether-toolbar adds is an ability to directly access the above-device widgets without needing to reveal them, thus saving the bezel-swipe to reveal and the time-out to vanish, leaving the screen free for interaction.

EXPERIMENT 2 : ETHER-TOOLBAR ACCURACY

In this section, we evaluate the potential accuracy of ether-toolbars. We describe an experiment that varies the number of buttons and number of rows of buttons, we analyze precision, and we leverage this data to drive design recommendations for an overall implementation.

Participants

Ten new unpaid participants took part in our experiment. Participants' ages varied between 26 and 38 years. Nine participants were right handed.

Apparatus and Interface

The experiment ran on the same Samsung Galaxy Tab S. The tablet was placed on top of a 32" multi-touch screen which was inclined to horizontal. The 32" 3M multi-touch screen was used to detect tap locations around the tablet device. Participants sat in front of the tablet, which was placed in a fixed location at the bottom of the multi-touch screen as shown in Figure 5. An additional application was developed to log the touches on the multi-touch display – time and coordinates – in the reference frame of the tablet.

We implemented an Android applicationto simulate both an on-screen toolbar and an ether-toolbar in positions indicated in Figure 4. It included a control condition which pictured the toolbar on the screen and an experimental condition, with the

Figure 5: Screen-shot of the interface presented to participants. Here the grid is composed of two rows and ten buttons. The participant is asked to tap the 'B8' button.

ether-toolbar. The tablet displayed rectangles that represented the buttons at the top while a start button was 8cm under the first row of buttons. Each button had an identifier that gave the row (A-C, top to down - the letter "A" refers to the top row) and the column (1-number of buttons per row, right to left). For example, if we have a grid with two rows of eight buttons, B8 would identify the eighth button of the second row starting from the top-left corner of the grid (Figure 5). To perform a trial, a user would tap the start-button (pictured in Figure 5). The button label would change to reveal a target button to tap (row + column format), and the participant would then tap the button. The start button's position was changed for the control versus the experimental condition such that the distance made by the participant from start button to toolbar was fixed. This ensured that any timing differences were a result of the increase in task difficulty due to position estimation, and not due to increased distance to make.

Design

The experiment was a repeated measures design that included two conditions (on-screen vs ether-toolbar placement). We, also, examined 3 button densities within a row (5, 8, or 10 buttons per row) and we included 1, 2, or 3 rows of buttons. We limit our number of buttons per row to 10 because, in informal evaluation, it was deemed too difficult to accurately estimate the position of more than 10 buttons along the top of the display. Similarly, we use 3 rows of buttons because, in on-screen condition, users will interact with the widget on the display. Assuming minimum widget size of 1cm, 3 rows of buttons would cover 3cm on the display – approximately 25%.

Participants completed 15 trials within each configuration, where they clocked on five selected buttons three times each. We did not counterbalance condition (on-screen vs ether) between users. Instead, each user saw the on-screen condition first, to learn the toolbar placements and size, and then saw the ether condition second. This strategy was informed by our goal to evaluate the pointing precision on off-screen content that users habitually see on screen. For our other experimental factors, we began with one row of buttons, our simplest configuration, then allowed the complexity of the interface to gradually increase. We did this so we could easily determine at which point accuracy would begin to drop and to allow users to gradually learn button placement beginning with simple cases. Button density (5, 8, or 10) was counterbalanced.

Overall, our experiment was a $2 \times 3 \times 3$ repeated measures design clicking on five selected buttons three times each with 10 participants yield a total of 2700 individual button accesses.

Research Questions

We would expect ether-widget access to be both slower and more error prone than on-screen targeting. Given this assumption, the goal of this experiment is not to prove that fact – a failure for ether-widgets when compared to on-screen widgets – but instead to understand both the relative accuracy of users manipulating ether-widgets. With this in mind, we will examine overall speed and error rate, but then focus specifically on access patterns and how best to boost overall accuracy.

RESULTS

The dependent measures are *accuracy* (the proportion of correct button selection) and *trial time* (trial time is measured from the target apparition, to button successfully selected). All analyses are two-way ANOVA. Tukey tests are used post-hoc when significant effects are found.

Trial time and accuracy

Our primary interest is in how condition effects trial time and accuracy. RM-ANOVA did not reveal any condition effects for trial time, implying that there were no statistically significant differences between trial time when comparing ether or on-screen widget access. However, we did see significant main effects for accuracy.

Table 1 presents a snapshot of post-hoc accuracy analysis over our data set. These accuracies assume that button width and height were identical to on-screen button width and height, specifically button row height was 1cm and button width was 35mm per button for the 5 buttons configuration, 22mm per button for the 8 buttons configuration, and 17mm per button for the 10 buttons configurations, a result of the display width measuring 17.5cm.

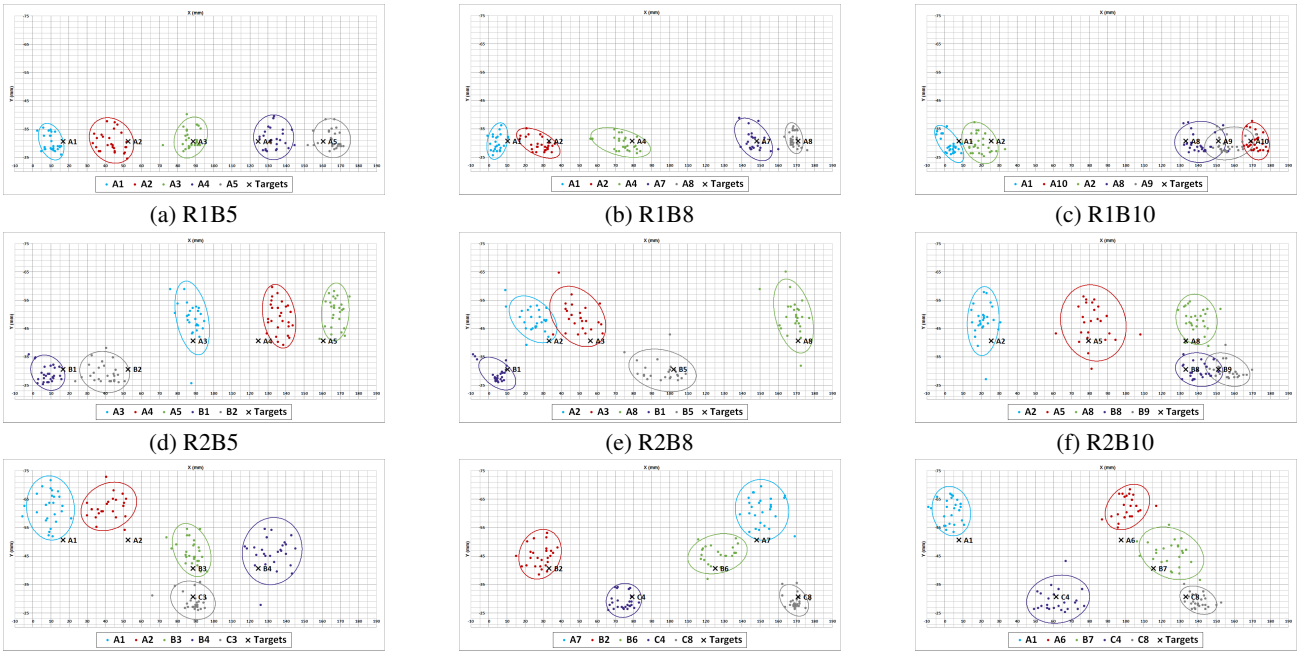
Overall, we see that for 1-row, 5- and 8- button configurations, accuracy of off-screen targeting is quite high (85% for one row of five buttons and 87% for one row of eight buttons, respectively (not significant)). One interesting factor we observed was that, in multi-row configurations, buttons in the row farthest from the tablet had lower accuracy than buttons closer to the tablet, again unsurprising do the imprecision in distance estimation when moving away from the edge of the tablet. Interestingly, when examining Figure 6, we confirm that the farther the row is the larger it is perceived by participants. Additionally, Figure 6 shows that for the farthest row, participants consistently over-estimate its height.

Participants comments

All our participants felt that the task became more difficult and that they needed to concentrate to accomplish it from either R2B8 or R2B10 configurations. All our participants also preferred not to use 10 buttons per row for the ether-widgets condition. Some quotes are: "*10 buttons is really too much... I think I am unable to make the difference between having 10 or 11 or 12 buttons (or even more)...*", "*if I have the choice, I will stop at 8 buttons per row*" or "*I don't like*

	R1B5					R1B8					R1B10			
	On-screen		Ether-toolbar			On-screen		Ether-toolbar			On-screen		Ether-toolbar	
	Mean	s.d	Mean	s.d		Mean	s.d	Mean	s.d		Mean	s.d	Mean	s.d
A1	100	0	100	0	A1	100	0	90	9.97	**A1⋆**	100	0	70	29.93
A2	100	0	73.33	21.33	A2	100	0	76.66	23.95	**A2⋆**	100	0	60	23.45
A3	100	0	80	19.95	A4	100	0	86.66	17.42	**A8⋆**	96.66	6.53	50	22.31
A4	100	0	76.66	17	A7	100	0	83.33	14.60	**A9⋆**	96.66	6.53	66.66	21.77
A5	100	0	93.33	8.71	A8	100	0	96.66	6.53	**A10⋆**	100	0	86.66	17.42
	R2B5					R2B8					R2B10			
	On-screen		Ether-toolbar			On-screen		Ether-toolbar			On-screen		Ether-toolbar	
	Mean	s.d	Mean	s.d		Mean	s.d	Mean	s.d		Mean	s.d	Mean	s.d
A3⋆	93.33	8.71	18.33	20.9	**A2⋆**	100	0	16.66	22.3	**A2⋆**	96.66	6.53	10	9.97
A4⋆	96.66	6.53	23.33	19.59	**A3⋆**	100	0	23.33	23.95	**A5⋆**	90	13.94	18.33	15.74
A5⋆	100	0	20	17.42	**A8⋆**	96.66	6.53	13.33	14.44	**A8⋆**	93.33	8.71	10	13.94
B1	96.66	6.53	93.33	8.71	B1	96.66	6.53	90	19.59	B8	96.66	6.53	53.33	27.88
B2	86.66	17.42	73.33	25.39	B5	90	13.94	66.66	27.54	B9	100	0	68.33	26.87
	R3B5					R3B8					R3B10			
	On-screen		Ether-toolbar			On-screen		Ether-toolbar			On-screen		Ether-toolbar	
	Mean	s.d	Mean	s.d		Mean	s.d	Mean	s.d		Mean	s.d	Mean	s.d
A1⋆	100	0	13.33	19.95	**A7⋆**	96.66	6.53	16.66	10.88	**A1⋆**	100	0	6.66	8.71
A2⋆	100	0	3.33	6.53	**B2⋆**	86.66	10.66	41.66	25.41	**A6⋆**	96.66	6.53	3.33	6.53
B3⋆	86.66	10.66	46.66	22.20	**B6⋆**	93.33	13.06	43.33	30.87	**B7⋆**	96.66	6.53	15	12.36
B4⋆	96.66	6.53	23.33	17	C4	96.66	6.53	90	9.97	C4	96.66	6.53	56.66	21.88
C3	96.66	6.53	86.66	13	C8	90	9.97	100	0	C8	100	0	63.33	22.73

Table 1: Mean and s.d of accuracy.The significant difference between *on-screen* and *ether-toolbar* are **highlighted⋆**.



(a) R1B5 (b) R1B8 (c) R1B10

(d) R2B5 (e) R2B8 (f) R2B10

(g) R3B5 (h) R3B8 (i) R3B10

Figure 6: The taps of every participants for each configuration. The blue dots represent the taps in *expert mode*.

selecting the off-screen buttons when there are 10, it is really very difficult".

To better understand how participants performed, we report the different strategies described by participants in order to localize their ether-button target. This data was extracted from

	Lower (mm)	Upper (mm)	Size (mm)
Closest row	14.67	28.57	13.89
Middle row	28.57	43.91	15.35
Farthest row	43.91	62.91	19.00

Table 2: Corrected boundaries of the ether-toolbar rows in mm. The Y axis origin is the tablet top side.

post-experiment interviews with participants.

- **Considering the two extremities of the tablet and its middle as references.** All participants used this strategy in order to estimate the target button's location. For instance, participants knew that the third button was exactly at the center of the screen in the 5 buttons configurations, whereas in the 8 buttons configurations the middle line of the screen was the line separating the fourth and the fifth button. Similarly with the extremities of the screen, participants knew the first and last buttons were at the extremities and then could estimate these buttons close neighbors.

- **Counting the number of buttons.** This strategy was mostly used at the very beginning of the experiment for five and eight buttons and was the primary strategy in the 10 button configurations: participants physically counted with their fingers the buttons while estimating the size of the buttons.

ETHER-TOOLBAR DEFORMATION MODEL

Results from Experiment 2 showed that the accuracy drops drastically for buttons in rows above the closest one from the tablet. Interestingly, when looking at the graphs in Figure 6 we can identify consistency in errors. Participants seem to estimate the buttons further and larger than they actually are but always in the same proportions for buttons of the same row. This observation means that a model that reflects how users perceive mobile surroundings for an ether-toolbar can be leveraged from the collected data with the ultimate goal of improving the accuracy. In addition, due to the deterioration of the performance for the configurations where there were 10 buttons and as also our participants felt that these configurations are difficult, we decide to limit the number of buttons per row to a maximum eight.

Deformation Model

Our goal here is to leverage a model that estimates the ether-toolbar deformation as perceived by users. Towards this ultimate goal, we, first, assume that we have 95% accuracy for the collected data for the different ether-configurations. Given this assumption, we then consider only the following 95% of the collected data: for each configuration, for each target button, we calculated the two barycenters x_{mean} and y_{mean} of respectively the x coordinate (*i.e.*, the position of the tap along the top side of the tablet) and y coordinate (*i.e.*, the position of the tap above the tablet) of the all collected taps for this button. For each barycenter (x_{mean} and y_{mean}), we then extract 95% of the data (X_{95} and Y_{95}) that are the nearest to it. In the following, we describe the deformation of the ether-button through the y axis and then through the x axis.

In order to determine the deformation of the ether-buttons through the y axis, we consider the data extracted in Y_{95} and ran a partitioning algorithm based on the Fisher-Jenks grouping algorithm [7]. The Fisher-Jenks algorithm classifies unidimensional values in a predefined number of groups (three in our case) by minimizing the intra-group variance. The output of the partitioning gives us the barycenters of the three groups (corresponding to the barycenters of the three rows) and the intervals representing the boundaries of the groups on the y axis (corresponding to the height of the each row). Table 2 summarized the results. We then consider participants comments and our findings for the farthest row to refine our model as follow: we multiply the height of the farthest row by two to make them as if they have an "unlimited height".

We then made the same analysis for the x axis though the accuracy rate is 89.73%. We found that the barycenters shift by 2.36mm on average. The major deformations (5mm) happen for the left-most and right-most buttons and their closest neighbors. This can be explain by the fact that participants took the screen extremities as reference points for the left-most and right-most target buttons. Using only the Fisher-Jenks classification algorithm, we found the groups sizes were always smaller than the actual buttons size. Due to these minor deformations along the x axis, we decided to only modify the ether-toolbar along the y axis.

Experiment 3: Validation

A third experiment, similar to the second, was used to validate our ether-widgets deformation estimation. The same apparatus, task, and design were used, but with only 2 buttons densities conditions (5 and 8) and 3 new participants: 3 participants x 2 toolbar placements x 3 row conditions x 2 buttons densities x 5 buttons x 3 repetitions = 540 individual button accesses.

RM-ANOVA did not reveal any condition effects for the accuracy, with an overall of accuracy of 94.23% (sd=2.83) for the ether-toolbar condition and 99.62% (sd=.73) for the on-screen toolbar condition. This findings indicate that our estimation of the deformation of ether-toolbar performed well with our validation data.

PROTOTYPE: DRAWING WITH ETHER-WIDGETS

While much of our design work explored ether-toolbars, we designed a prototype application that explores the ether-widgets space more fully. We implemented "Ether-Draw", a prototype application that supports ether-widgets and includes a low-cost mirror mount that supports self-calibration of around-display input.

When the user starts a new project with the Ether-Draw application, the ether-toolbar is not integrated into the application GUI. However, Ether-Draw provides the users the ability to create new personalized ether-widgets in real time and depending on their need. When creating a new ether-button, users have to memorize the position of the button and the associated command. To help them remember, and to make the presence of the button more explicit for demonstration purposes, users can draw a marker at the position of the ether-button. This physical representation of the button is depicted in our Figures for clarity, but is strictly optional.

Two main ether-button types can be created: (1) *ether-button export* and (2) *ether-button import*. In the *ether-button export* case, users can configure a particular command through the GUI (*e.g.*, red circle) and save it outside the screen. To save the command, users have to perform a long tap on the current tool widget to activate the export mode. When a tap is detected, an ether-button is created outside the screen where the tap was made. In the *ether-button import* case, users can draw a shape on the paper-sheet next to the tablet and import it in the application. Like the *ether-button export*, a button is created at the drawing location so that the shape can also be reused later in the application.

Hardware and implementation details

The prototype was implemented on a Samsung Galaxy Tab S of 8.4". We designed a 3D printed mount where a mirror was fixed above the front camera of the tablet. The mount allowed users to adjust the mirror so it shows the space at the right side of the tablet (See Figure 7).

The drawing application supports different shapes (rectangles, ellipses and lines) and different stroke styles, thicknesses and colors, as well as filling colors for closed shapes. A calibration button allows users to see the camera image so they can adjust the mirror position and orientation (Figure 8). The application runs on Android. The OpenCV library is used to process the camera image in order to track the finger and detect the drawing contours for *tool import*. Since we only need to map 2D coordinates of the virtual buttons in the camera image and the finger position, we do not need the finger position in space. We only use the image coordinate. The finger is tracked using an OpenCV background removal algorithm.

A tap sound recognizer that runs in the background has also been implemented to allow users to export and select tools. When a tap is detected, a virtual button that saves the current command is created where the finger was at the moment of the tap. If not, we search the nearest virtual button from the finger position that is not further than 100 pixels.

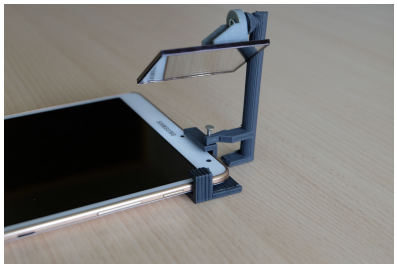


Figure 7: 3D printed mount that supports the mirror.

DISCUSSION AND IMPLICATIONS

Ether-toolbars leverage simple computer vision and spatial memory to support interaction. Thanks to the simple deformation model provided, ether-toolbars obtain an accuracy that is statistically equivalent to on-screen targets. We observe from the model that the further the row from the tablet, the larger it has to be. This confirms the results of the works of Gustafson *et al*.[8] and of Ens *et al*.in [6] discussed in the related work concerning target pointing with report to a reference point (hand or device). Whereas the study presented by Markussen *et al*.[18] found that for *midair* gestures around a distant TV screen, users tend to guess the target position closer than it actually is.

The added advantage is that relatively low-cost hardware additions to current commodity smartphones can support their implementation.

In the domain of intelligent user interfaces, one important area of inquiry is an understanding of how much intelligence is needed to support interaction. Whether complex algorithms must be implemented or significant external sensing must be used to support novel user interactions is often an open question. Some past research has explored this, including the use of time-of-arrival signals [19] to support around-device interaction. Our research falls within this domain, exploring specifically the question of whether simple computer vision algorithms can support around-device interaction.

Alongside the question of simplicity, another area of inquiry in around-device interaction involves precision. Specifically, our work provides evidence that high-density interaction surfaces can be supported in the space around portable devices, provided one understands how distance affects targeting precision. We do this via the high degree of consistency between our two user studies of toolbar targeting, where we show that targeting precision is statistically equivalent – even for up to 24 tiled buttons, 3 rows of 8 – located immediately adjacent to the display. This represents a significant enhancement over other recent work in around-device interaction, where fewer, larger, more separated spaces were used in around-device interaction [12].

CONCLUSIONS AND NEXT STEPS

We presented ether-widget design for mobile interaction. Ether-widgets leverages both users short term spatial memory and sensing technologies of mobile devices to expand the interaction space around the device. We first explored three ether-widget metaphors. Through a pilot study, we showed that the translation metaphor is the most preferred while providing the same accuracy and speed than Focus+context and zoomed metaphors. We then explored the design and efficiency of one particular type of ether-widgets, the ether-toolbar while using the translation metaphor. We designed an experiment to understand the relative accuracy of users when interacting with an ether-toolbar. Our findings indicate that 10 buttons per row is not adapted for ether-toolbar as it is both less preferred by participants and deteriorated significantly the performance. Importantly, our findings indicate that the farther the row is from the tablet the larger it is perceived by users. Through Fisher partitioning algorithm, we then proposed a model that estimate the ether-toolbar deformation as perceived by users. We validate our model with three new participants and we reach 94% of accuracy with no statistical difference with on-screen toolbars. This result supports the idea that off-screen interaction around the device is suitable for feature rich applications without deteriorating the performance compared to on-screen interaction. We finally presented "ether-draw", a prototype application that supports personalized ether-toolbar. We hope that our findings will contribute towards the adoption

(a) Calibration (b) Exportation (c) Importation (d) Use of imported drawing

Figure 8: Application to support drawing using ether-widgets: (a) calibration step; (b) a user exporting a yellow circle command outside the screen; (c) a user creating/importing a new sheep command; (d) a user using the sheep command.

of ether-widget design and will enable the design of additional ether-widgets in applications.

Among possible future work, it would be interesting to study the effect of the different sides of the tablet on the performance of ether-toolbar as well as the effect of the device (*e.g.*, using a smartphone). Further work would, also, investigate new prototype applications that take advantage of the combination of the mobile device and its surrounding space. In the long term, it would also be interesting to evaluate user expertise development on non-visual targets, and how expertise influences accuracy and ether-widgets space perception.

REFERENCES

1. Michelle Annett, Tovi Grossman, Daniel Wigdor, and George Fitzmaurice. 2011. Medusa: A Proximity-aware Multi-touch Tabletop *(UIST '11)*.
2. Patrick Baudisch, Edward Cutrell, Ken Hinckley, and Robert Gruen. 2004. Mouse Ether: Accelerating the Acquisition of Targets Across Multi-monitor Displays *(CHI EA '04)*.
3. Alex Butler, Shahram Izadi, and Steve Hodges. 2008. SideSight: multi-"touch" interaction around small devices. In *(UIST '08)*.
4. Sheelagh Carpendale, John Ligh, and Eric Pattison. 2004. Achieving Higher Magnification in Context *(UIST '04)*.
5. Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. 2009. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Comput. Surv.* (2009).
6. Barrett Ens, David Ahlström, Andy Cockburn, and Pourang Irani. 2011. Characterizing User Performance with Assisted Direct Off-screen Pointing *(MobileHCI '11)*.
7. Walter D. Fisher. 1958. On Grouping for Maximum Homogeneity. *J. Amer. Statist. Assoc.* 53, 284 (1958), 789–798.
8. Sean Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang Irani. 2008. Wedge: Clutter-free Visualization of Off-screen Locations *(CHI '08)*.
9. Sean Gustafson, Daniel Bierwirth, and Patrick Baudisch. 2010. Imaginary Interfaces: Spatial Interaction with Empty Hands and Without Visual Feedback *(UIST '10)*.
10. Jensen Harris. 2008. The Story of the Ribbon. *Presentation at MIX08* (2008).
11. Chris Harrison and Scott E. Hudson. 2009. Abracadabra: Wireless, High-precision, and Unpowered Finger Input for Very Small Mobile Devices *(UIST '09)*.
12. Khalad Hasan, David Ahlström, and Pourang Irani. 2013. Ad-binning: Leveraging Around Device Space for Storing, Browsing and Retrieving Mobile Device Content *(CHI '13)*.
13. Khalad Hasan, David Ahlström, and Pourang P. Irani. 2015. Comparing Direct Off-Screen Pointing, Peephole, and Flick & Pinch Interaction for Map Navigation *(SUI '15)*.
14. Khalad Hasan, David Ahlström, Junhyeok Kim, and Pourang Irani. 2017. AirPanes: Two-Handed Around-Device Interaction for Pane Switching on Smartphones *(CHI '17)*.
15. Bonifaz Kaufmann and Martin Hitz. 2012. X-large Virtual Workspaces for Projector Phones Through Peephole Interaction *(MM '12)*.
16. Sven Kratz and Michael Rohs. 2009. HoverFlow: Expanding the Design Space of Around-device Interaction *(MobileHCI '09)*.
17. Edward Lank and Son Phan. 2004. Focus+Context Sketching on a Pocket PC. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems (CHI EA '04)*.
18. Anders Markussen, Sebastian Boring, Mikkel R. Jakobsen, and Kasper Hornbæk. 2016. Off-Limits: Interacting Beyond the Boundaries of Large Displays *(CHI '16)*.
19. Robert Xiao, Greg Lew, James Marsanico, Divya Hariharan, Scott Hudson, and Chris Harrison. 2014. Toffee: Enabling Ad Hoc, Around-device Interaction with Acoustic Time-of-arrival Correlation *(MobileHCI '14)*.