Spreadsheets on Interactive Surfaces: Breaking through the Grid with the Pen

VINCENT CAVEZ, CAROLINE APPERT, and EMMANUEL PIETRIGA, Université Paris-Saclay, CNRS, Inria, France

Spreadsheet programs for interactive surfaces have limited manipulations capabilities and are often frustrating to use. One key reason is that the spreadsheet grid creates a layer that intercepts most user input events, making it difficult to reach the cell values that lie underneath. We conduct an analysis of commercial spreadsheet programs and an elicitation study to understand what users can do and what they would like to do with spreadsheets on interactive surfaces. Informed by these, we design interaction techniques that leverage the precision of the pen to mitigate friction between the different layers. These enable more operations by direct manipulation on and through the grid, targeting not only cells and groups of cells, but values and substrings within and across cells as well. We prototype these interaction techniques and conduct a qualitative study with information workers who perform a variety of spreadsheet operations on their own data.

CCS Concepts: • **Human-centered computing** → **Interaction techniques**; **Graphical user interfaces**; **Touch screens;**

Additional Key Words and Phrases: Digital pen, multi-touch interaction, ink, spreadsheets

ACM Reference format:
Vincent Cavez, Caroline Appert, and Emmanuel Pietriga. 2024. Spreadsheets on Interactive Surfaces: Breaking through the Grid with the Pen. *ACM Trans. Comput.-Hum. Interact.* 31, 2, Article 16 (January 2024), 33 pages.
https://doi.org/10.1145/3630097

1 INTRODUCTION

In his reflections about the use of spreadsheets in organizational life, Paul Dourish observes that *"a spreadsheet starts not blank but empty"* [12]. This intriguing statement captures a key characteristic of spreadsheet programs. The *grid* effectively creates a layer above the values, which plays a central role in enabling many of the direct manipulations that users perform on the spreadsheet's columns, rows and cells. But because it covers the entire workspace, this grid layer captures most input events, often complicating straightforward tasks like selecting a substring within a cell. Additionally, spreadsheet programs designed for interactive surfaces introduce yet another layer that captures basic touch actions for the purpose of navigation.

These three layers create tension between the different types of interactions: *grid-level* interactions (cell, row and column selection & manipulation); *value-level* interactions (text, number and

Authors' address: V. Cavez, C. Appert, and E. Pietriga, Université Paris-Saclay, CNRS, Inria, Bat 660, rue René Thom 91190 Gif-sur-Yvette, Orsay, France; e-mails: vincent.cavez@inria.fr, caroline.appert@universite-paris-saclay.fr, emmanuel.pietriga@inria.fr.

1073-0516/2024/01-ART16 $15.00
https://doi.org/10.1145/3630097

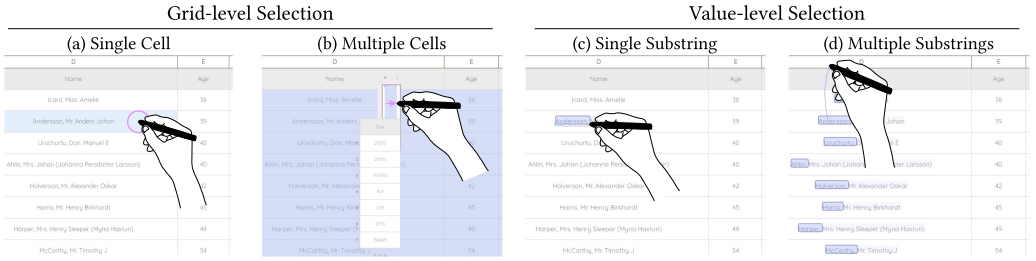

Fig. 1. Our techniques enable seamless direct manipulations of diverse elements in a spreadsheet primarily by enabling a broader variety of selection actions than existing spreadsheet programs. With only one or two pen marks, users can select different types of scope. Representative selections at the grid and value levels: (a) a single cell; (b) large sets of cells without navigation – here columns B–J extending far beyond the current viewport; (c) a substring within a cell; (d) multiple substrings matching a pattern across cells in a column.

formula editing); and *navigation* interactions (panning and zooming the worksheet). This tension between layers often breaks the direct manipulation paradigm, where users expect to effortlessly select an element and manipulate it. This tension exists even when performing the most elementary action: *selection*. While it is easy to select a single cell, selecting only part of a text or number inside a cell is tedious, requiring multiple actions to traverse the grid and adjust the selection. Additional problems occur when selecting sets of elements. Selecting elements such as, *e.g.*, rows or columns is possible but interferes with navigation actions when all elements are not visible simultaneously. And while spreadsheets are designed to manipulate sets of cells, it is never possible to select only a part of the value across multiple cells at once. Such manipulations are useful though, for instance to select a specific suffix such as the country code in a list of cities to remove them all, or to separate surname from firstname in a list of people with a single manipulation.

This article introduces a set of novel direct manipulation techniques that enable seamless interaction at both grid and value levels, within and across cells, as illustrated in Figure 1. With these techniques, users can achieve diverse operations with a few pen marks and touch actions. Building upon previous work on pen + touch [24, 44], we implement a clear division of labor between input modalities, dedicating the pen to selection, touch to manipulation, and multi-touch to navigation. A key element of our approach is to use the pen not only as a tool to select grid-level elements, but as a tool to *break through the grid* as well, enabling precise, seamless value-level selections without mode switching. We further leverage the pen's unique precision to design small pen-operated widgets that users can invoke to jump to distant locations in the spreadsheet or to perform advanced, non-contiguous selections of tabular data rows and columns.

After a review of the literature, we examine a representative sample of commercial spreadsheet programs over a range of operating systems and pen + touch hardware: handheld tablets, slate PCs, large digital drawing boards. We highlight inconsistencies among them, as well as the main sources of friction between layers. We then conduct an elicitation study to gather empirical data about users' expectations when using pen + touch to interact with spreadsheets. Informed by both our analysis of commercial spreadsheet programs and our elicitation study, we design a set of interaction techniques that enable seamless access to all layers, leveraging the expressive power of pen + touch input to enable the *effortless selection of a variety of elements in spreadsheets*. These elements include individual cells, groups of cells, as well as values and substrings within cells or spanning multiple cells. We implement this set of interaction techniques in a prototype that we use to conduct a semi-structured qualitative study with six information workers performing a variety of spreadsheet operations on their own data. We discuss how participants used the techniques,

and how their feedback helped us improve on them. We conclude with opportunities to explore as future work.

2 RELATED WORK

Spreadsheets have received significant attention from the HCI research community. Research works include studies about how information workers use regular spreadsheet programs (see, *e.g.*, [2, 8]); solutions to effectively deal with errors and hidden dependencies (see, *e.g.*, [26, 55]); algorithms that help automatically transform data [19, 22, 28]; and interaction techniques for the direct manipulation of those data. In our review of the state of the art, we focus on the latter only. We first discuss spreadsheet-related interaction techniques regardless of the input modalities involved, and then give an overview of related work on pen + touch interaction at large.

2.1 Interaction with Spreadsheets

Early research work about user interfaces for tabular data manipulation investigated ways to support users in making sense of large tables. The Table Lens [47] adopts a focus+context strategy [10] to embed visual representations of the data into the grid-based representation of the table, letting users sort and filter, get detailed symbolic representations of rows and columns of particular interest and identify values that characterize the distribution [45]. Interaction with the table involves keyboard shortcuts and a pointer, but includes basic flick gestures performed with the mouse as well.

With the advent of interactive surfaces, researchers have started investigating how to use the pen and touch modalities for spreadsheet manipulation. Tableur [70] aims to support the quick creation of relatively simple spreadsheets away from the office workstation. The pen is used to sketch the table by hand on a blank canvas and then input cell values. Gestures trigger commands such as the recognition of the digital ink, the erasure of content, and the propagation of a formula over a range of cells. It focuses on small, informal spreadsheets that can reasonably be drawn by hand. The WritLarge [65] early-stage design system for electronic whiteboards also enables users to create simple tables by sketching among many other things. The digital ink that represents the table can be selected with the non-preferred hand and semantically *"elevated"* [65] thanks to a simple heuristic that recognizes the hand-drawn grid and turns it into an actual table structure. A very interesting property of this approach is that users can go back and forth between ink and formatted table, editing strokes at the ink level to merge some cells, for instance. But again, this approach primarily supports the creation of small tables drawn by hand.

While the above works essentially focus on sketching simple spreadsheets on a blank 2D canvas, other related work has investigated the use of pen and touch to manipulate data in conventional spreadsheets programs. As part of their solution to interactively repair tables extracted from documents on mobile devices, Hoffswell and Liu [25] describe a small set of gestures that users can perform to insert and delete cells, as well as split or merge cells that were incorrectly recognized by the automatic extraction process. Although not a spreadsheet program but rather a tool for visual data exploration, TouchPivot [29] features a table view that users can interact with to perform some transformations on the data. The focus is specifically on filtering and pivoting data using simple pen and touch interactions. Perhaps most related to our work, Pfeuffer et al.'s investigation of thumb+pen interaction on handheld tablets [44] includes a spreadsheet application probe that enables users to perform a set of cell manipulations using the pen primarily as a selection tool, while the non-preferred hand complements that input by setting spring-loaded modes and adjusting parameters. As discussed later, this probe strongly influences our core division of labor. It essentially focuses on grid-level operations, however, and does not address friction between grid

and value-level operations, which we identify as a major usability issue with spreadsheet programs on interactive surfaces.

Finally, other modalities than pen + touch have been investigated to interact with spreadsheets. Gesslein et al. [16] use a virtual reality headset to extend the limited display space of a tablet in a mobile context of use. The additional display capacity is used to show multiple worksheets, to extend the current worksheet, and to superimpose additional information such as dependencies between cells using the third dimension. Many interactions are performed with the pen, though some actions are performed with mid-air and touch gestures. Takayama et al. [59] run an elicitation study to design a set of contactless gestures for mid-air spreadsheet manipulation in front of a regular desktop monitor. Finally, Perelman et al. [41] use a smartphone to perform grid-level selections on tablets, observing that touch interaction alone is limited to a few gestures, many of which are reserved for navigation actions.

2.2 Pen + Touch Interaction

Support for concurrent pen and touch input only came in the mid 2000s [66]. An early study by Brandl et al. [5] identified several benefits to performing bimanual interactions with a combination of pen and touch, grounded in Guiard's kinematic chain model [17]. They discussed some foundational principles such as using the pen for precise input with the preferred hand and coarser actions (mode selection, parameter adjustments) with the non-preferred hand. They also empirically compared pen + touch to pen + pen and touch + touch, finding the pen + touch combination to perform better on a task that involved simultaneous inking and navigation. Informed by observations of people manipulating physical paper and notebooks, Hinckley et al. soon after introduced their *"pen+touch=new tools"* division of labor where *"the pen writes, touch manipulates, and the combination of pen + touch yields new tools"* [24].

In this paper, Hinckley et al. illustrate the general idea on an application for note-taking and scrapbooking. But pen + touch input has been investigated in a variety of applications that follow the same division of labor for the most part: active reading [23] and annotating documents with systems such as RichReview [67] and SpaceInk [52]; handwritten text editing [60]; page-based document editing [37], working with maths [68]; and designing on a whiteboard [65].

Other research has investigated applications that depart from this general division of labor. In a laboratory experiment, Matulic and Norrie [36] found the pen to also be effective at performing some widget manipulation tasks, leading them to believe that the division of labor was *"not always clear-cut."* In their spreadsheet probe for thumb+pen interaction, Pfeuffer et al. make the pen select rather than write: *"the pen selects, touch manipulates"* [44] – acknowledging selection as one of the core transactions in the spreadsheet interaction model. Hamilton et al.'s pen + touch interactive system for a real-time strategy game [21] lets users make precise selections and invoke commands with the pen. The preferred hand is also used to navigate, with the pen stowed in the palm. The pen can also be used to select and manipulate existing vector shapes, for instance to edit node-link diagrams [14]. Graphies [49] – an expressive network visualization authoring environment – uses the pen primarily for selection and for parameter adjustments. In NEAT [15] – a set of pen + touch techniques for vector graphics layout – the pen is used not only to sketch shapes but to draw alignment guides as well. Overall, we observe that beyond writing, *selection* comes a close second as a role for the pen. It is indeed a particularly effective precision tool for delineating arbitrary regions, which will prove particularly useful for spreadsheet interaction as discussed in Section 5.

Pen + touch has also been investigated in a large number of data visualization systems where users manipulate many small interface elements and perform elaborate selections like they do with spreadsheet programs. Discussing them all is beyond the scope of this paper. We only mention interesting trends and refer the interested reader to Lee et al.'s comprehensive survey [33] for

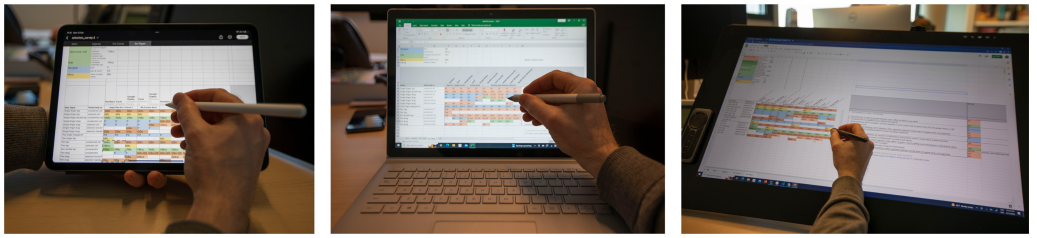

Fig. 2. Sample configurations used in the analysis of commercial spreadsheet programs: *Numbers* running on an Apple iPad Pro; *Excel* running on a Microsoft Surface Book; *Google Sheets* running on a Wacom Cintiq Pro.

details on this topic. As pen and touch control different pointer types, a general trend in many systems is to use them for different types of actions, helping disambiguate input without requiring an explicit mode switch (see, *e.g.*, [11, 69]). The pen, again, is often used for selection. While some systems such as SketchInsight [32] rely on touch for this action, a lot of systems rather rely on the pen (*e.g.*, [11, 51, 54, 69]), taking benefit from an input device whose *"key affordance [...] is the accuracy for direct pointing and dragging, providing the ability to compose precise selections by sketching complex shapes [...] while reducing the occlusion caused by touch"* [51].

Prior work discussed so far has involved regular digital pens in combination with regular, direct touch input. In the last few years, research projects have looked into expanding input capabilities, for instance with gaze [42, 43], speech [58], pen grip and hand posture [7, 35, 38] or even tablet grip [71]. We come back to these additional input capabilities in Section 7 where we consider going beyond regular pen + touch as future work.

3 WHAT USERS *CAN* DO WITH COMMERCIAL SPREADSHEET PROGRAMS

Microsoft Excel, Apple Numbers and Google Sheets – to name well-known commercial spreadsheet programs – are all available on tablet computers. Their cumulative download counts amount to billions. These spreadsheet programs typically have fewer features than their desktop counterparts, but their UI design and interaction model is actually very similar, which is one of the causes of friction between the grid, value and navigation layers. Table 1 summarizes our systematic analysis of input-to-action mappings over multiple *configurations*: different spreadsheet programs (Microsoft Excel, Apple Numbers, Google Sheets) and operating systems (Microsoft Windows, Apple iPadOS, Apple macOS), running on the three representative pen + touch devices illustrated in Figure 2.[1] The effect of each input action is categorized into: **grid-level selections (GS$_*$)** and **manipulations (GM$_*$)**; **value-level selections (VS$_*$)**; **panning & zooming the worksheet (P, PZ)**; and **invoking contextual widgets (MN)**. Looking at this table, we can make several observations.

Lack of consistency. One first observation is the inconsistency [40] between input-action mappings across configurations. Looking at individual rows, color variations reveal significant differences between them. There are variations among hardware configurations for a given spreadsheet program, and among programs for the same hardware configuration. These variations are symptomatic of the friction between the three interaction layers (grid, value, navigation), different spreadsheet program UI design teams addressing the problem in different ways.

[1]The raw table of all input events and actions triggered, per configuration, is provided as supplemental material, together with information about spreadsheet program version and hardware configuration.

Table 1. Summary of User Actions and their Effect, by Spreadsheet Program, by Operating System, by Display Surface Hardware

			Apple iPad Pro + Pencil 2			MS Surface Book		Desktop computer + Wacom Cintiq Pro				
			Numbers (iPadOS)	Excel (iPadOS)	Google Sheets (iPadOS / Safari)	Excel (Windows)	Google Sheets (Windows / Chrome)	Numbers (macOS)	Excel (macOS)	Google Sheets (macOS / Chrome)	Excel (Windows)	Google Sheets (Windows / Chrome)
	User input	**Performed on**										
l.01	Single-finger tap	unselected cell	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n
l.02	Single-finger tap	selected cell	MN	MN	MN	MN			MN			
l.03	Single-finger double tap	(un)selected cell	VS_{lc}	VS_{lc}	VS_{lc}	VS_{lc}	VS_{lc}	VS_{nc}	VS_{nc}	VS_{lc}	VS_{lc}	VS_{lc}
l.04	Single-finger drag	unselected cell	P	P	P	P	P	GS_n,GS_a	GS_n,GS_a	GS_n,GS_a	P	P
l.05	Single-finger drag	selected cell	P	P	P	P	GS_a	GM_{cp}	GS_n,GS_a	GS_n,GS_a	P	GS_a
l.06	Single-finger drag	selection border			GS_a	P	P		GM_{cp}	GM_{cp}		
l.07	Single-finger drag	selection handle	GS_a	GS_a	GS_a	GS_a					GS_a (*)	
l.08	Two-finger drag/pinch		PZ	PZ	PZ	PZ	P	P	P	P	PZ	P
l.09	Two-finger tap		MN					MN				MN
l.10	Pen tap	unselected cell	VS_{nc}	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n	GS_n
l.11	Pen tap	selected cell	VS_{nc}		MN							
l.12	Pen double tap	(un)selected cell	VS_{ac}	VS_{lc}	VS_{lc}	VS_{nc}	VS_{lc}	VS_{nc}	VS_{nc}	VS_{lc}	VS_{nc}	VS_{lc}
l.13	Pen drag	(un)selected cell		P	P	GS_n,GS_a	GS_n,GS_a	GS_n,GS_a	GS_n,GS_a	GS_n,GS_a	GS_n,GS_a	GS_n,GS_a
l.14	Pen drag	selection border			GS_a	GM_{cp}			GM_{cp}	GM_{cp}	GM_{cp}	
l.15	Pen drag	selection handle	GS_a	GS_a	GS_a			GS_a			(*)	

(*) The selection rectangle features handles only if it was created using finger touch (tap for single cell select).
Freeform inking available by switching to a dedicated annotation mode.

GS_n	grid-level selection *a*	VS_{ac}	value-level selection	P	pan sheet
GS_a	grid-level selection *(adjust cell selection)*	VS_{lc}	value-level selection *(value editing: position caret after last character)*	PZ	pan & zoom sheet
GM_{cp}	grid-level manipulation *(cut & paste selected cells)*	VS_{nc}	value-level selection *(value editing: position caret after nearest character)*	MN	invoke context menu

Actions are grouped in four categories, which are visually encoded with redundancy to help identify them based on the action code's first letter and background color: grid-level selection GS_* and manipulation GM_*; value-level selection VS_*; P, PZ panning & zooming the worksheet; MN invoking contextual widgets. Empty cells indicate that this input has no effect.

Friction between grid and navigation layers. A typical example of friction that results in a variety of mappings across configurations is the case of `drag` input events. Starting with the case of single-finger drags performed on already-selected cells, Table 1 lists four different actions (l.05),[2] illustrated in Figure 3. Considering all drag input events (l.04-07 + l.13-15), the same table shows that regardless of the modality (finger or pen), these are sometimes mapped to grid-level selection actions (GS_*, GM_*) and sometimes to navigation actions (P). There is broad agreement regarding two-finger drag/pinch input events however: these are systematically mapped to navigation (l.08).

Friction between grid and value layers. Table 1 also shows that selections tend to be cumbersome no matter the level considered (grid or value). For instance, selecting a range of cells (a grid-level action) typically involves two steps: first, a cell is selected (GS_n) with a tap (l.01+l.10), followed by a drag on a small, difficult-to-acquire element such as the selection border or lower-right corner handle (l.06-07 + l.14-15) to adjust the selection (GS_a). Value-level selections are cumbersome as well, requiring a double tap to get through the grid layer and interact with values on the layer below, one at a time. In addition, while this double tap is mapped consistently to a value-editing mode-switch across configurations, the actual selection state resulting from this switch varies (l.03, l.12). As shown in Figure 4, depending on the configuration considered, a pen double tap can either automatically select the entire string, position the text insertion caret at the end of the string, or position it between the two characters nearest to the double-tap location. In all cases, selecting a

[2]Notation **l.nn** refers to line numbers in Table 1.

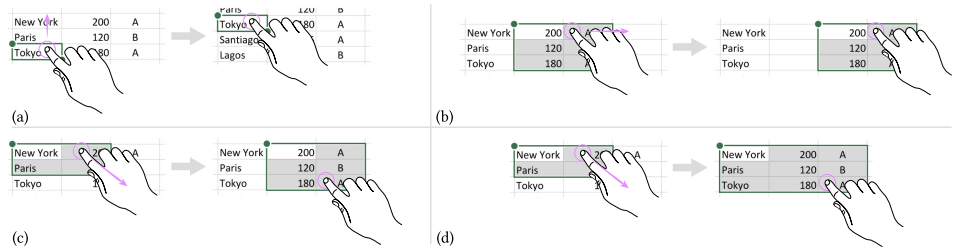

Fig. 3. Four different actions triggered when dragging with a single finger on an already-selected cell: (a) pan the sheet P; (b) cut & paste the content of the cells from the current selection GM_{cp}; (c) reset the selection and then adjust it GS_n,GS_a; (d) adjust the current selection without resetting it GS_a.

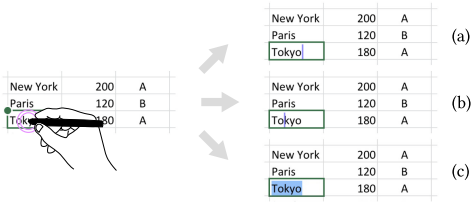

Fig. 4. Three different behaviors when performing a pen double-tap between characters o and k in cell Tokyo: enter value-edit mode, and either (a) position the caret at the end of the string (VS_{lc}); (b) position the caret in the string, precisely between those two characters (VS_{nc}); or (c) select the whole string (VS_{ac}).

specific substring or moving the caret to another position requires additional input actions. On the desktop, this is facilitated by the keyboard and the mouse's multiple buttons. But none of these is available – or easily accessed when available – on interactive surfaces.

In summary, spreadsheet programs for interactive surfaces lack in clarity when it comes to disambiguating grid-level interactions (cell selection and manipulation), value-level interactions (string editing), and navigation interactions (panning and zooming the worksheet). There are multiple inconsistencies across programs, and many actions – including some very basic ones – are cumbersome. The only consensual design choice we observe is that *two-finger interactions are consistently dedicated to the navigation layer* (**Obs$_1$**). Apart from that, there is no clear model for the fundamental action of selecting elements, which is key to any direct manipulation technique. We argue that the difficulty in selection is mainly due to spreadsheet programs following *"a legacy of designs that have treated pen or touch interchangeably"* [24], missing opportunities to leverage the specific capabilities of pen input.

4 WHAT USERS *COULD* DO WITH SPREADSHEET PROGRAMS

The above analysis gives us an overview of what actions users *can* do with current commercial spreadsheet programs operated with pen and touch. We now start considering what actions users *could* do with those input modalities. We first report on what strategy they would adopt to select different types of elements in a spreadsheet, gathering data through an elicitation study in which users demonstrated what they would do to perform different types of selections.

As in typical elicitation studies [64], participants were presented with the effect of an action (*referent*) and asked to demonstrate the interaction steps (*signs*) they would expect to perform to trigger that effect.

4.1 Participants

Sixteen volunteers (6 women, 10 men), all right-handed, aged 20 to 46 year-old (average 26.6, median 25), participated in the experiment. All of them reported using a spreadsheet program regularly on the desktop: Microsoft Excel (11), Apple Numbers (2), Google Sheets (5), or Open/Libre Office Calc (5). One of the participants also reported using Apple Numbers on an iPad Pro tablet. Seven participants reported regularly using a digital pen for drawing on a tablet; three for taking notes or annotating documents; one for browsing the Web. Two participants reported infrequent usage of a digital pen for gaming on a Nintendo Console, and four reported never using one.

4.2 Apparatus

The experiment ran on a Windows 10 Pro desktop workstation (Intel Xeon CPU/32GB RAM) connected to a Wacom Cintiq Pro display (24", 3840 x 2160 pixels) equipped with a Wacom Pro Pen 2. The software was implemented as a Web application (Figure 5), collecting input events using the W3C Pointer Events API [6]. The application ran in the Chromium (v96) Web browser.

4.3 Task and Procedure

To avoid considering selections in an overly idealistic and non-ecological context, our study takes into account not only grid-level and value-level actions from Table 1, but also includes several additional actions. These actions involve manipulating elements once they have been selected. The complete set of interactions presented to study participants is detailed in Appendix A.1.

Participants entered the room, read and signed a consent form that explained the overall goal of the experiment, then filled out a demographic form. They were explicitly told that there was not one unique answer or universal truth to the different questions they would be asked. We then collected their answers for all 28 referents listed in Appendix A.1, as described below.

Participants received the following instructions: *"This tablet is tactile. Imagine that you only have your fingers and the stylus (both tip and eraser) to interact with it. Use your fingers and/or pen directly on the left picture to show us the action(s) that you would do to trigger that effect. It is up to you whether you perform the gesture using one hand, two hands, the pen only, etc."*

In line with studies such as, *e.g.*, [46, 53], that give participants an opportunity to revise some of their signs after having seen the whole set of referents, we divided the experiment into two phases: *Initial* and *Revision*. In each phase, participants completed 28 trials during which they performed actions to invoke one of the referents. Before starting with the *Initial* phase, participants were told that if they ran into conflicts – *i.e.*, if they wanted to perform the same actions (signs) they had performed for an earlier referent – they would have the opportunity to address those conflicts later on if they wanted to and should not worry about this. Once they had become familiar with the complete set of referents in the *Initial* phase, they were given the opportunity to revise any of the signs in the *Revision* phase.

Figure 5 illustrates a typical trial in our study. The referent is expressed as a question displayed in the top part of the interface and is illustrated by two pictures of a spreadsheet. The picture on the left shows the spreadsheet's current state and the picture on the right shows the spreadsheet's state after the referent has been invoked. Participants performed the sequence of actions directly on the left picture. As in any elicitation study, the interface was passive and did not implement any spreadsheet-related functionality. It only left digital ink showing when and where input was performed, what type of pointer was used (pen or touch) and what type of actions were performed (tap, dwell, trace). For instance, in Figure 5, the participant used the pen to enclose substring `setosa`.

As in Wobbrock et al.'s study of user defined gestures for surface computing [64], participants rated how much the signs they proposed were (1) a good match for the referent and (2) easy

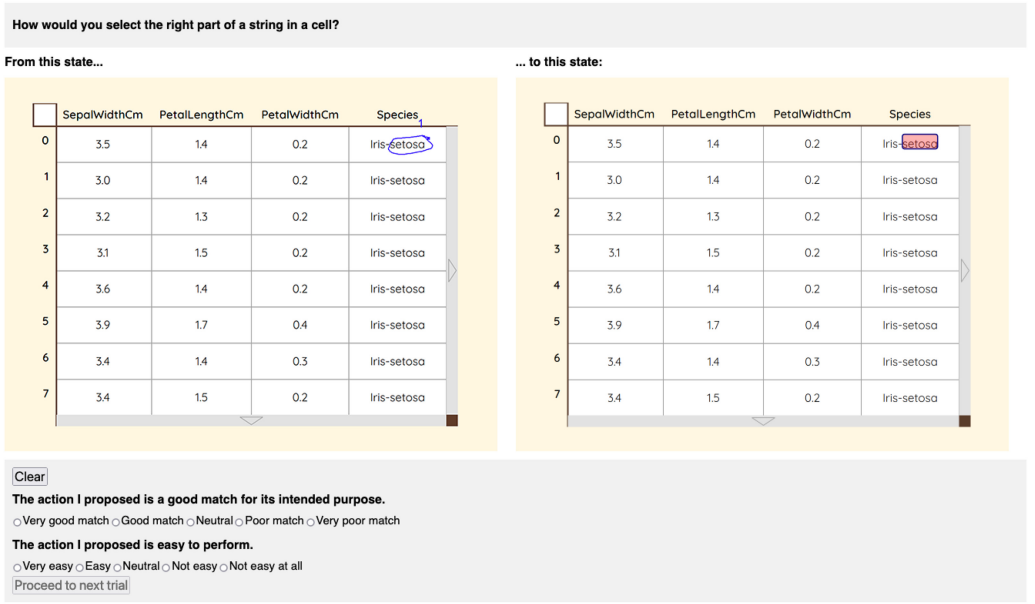

Fig. 5. A trial in the study. The *referent* is presented as a question in the top panel. The left and right spreadsheets respectively show the state *before* and *after* the referent's effect is applied. Participants use their fingers and pen directly on the left panel to make their *sign* proposal. They then evaluate their proposal with a *Match* and *Easy* score using the 5-pt Likert scales displayed below the spreadsheets.

to perform. They did so thanks to the two 5-point Likert scales at the bottom of the interface (Figure 5).

Within each phase (*Initial* and *Revision*), trials were grouped by Scope and Action Type (Appendix A.1). The presentation order of groups was counterbalanced across participants using a Latin Square. Within a group, referents were presented in a random order. For each participant, the overall presentation order of the 28 referents was the same across the two phases.

4.4 Results

For each trial, the experimental software made a screen capture of the left picture annotated with input traces, and recorded the *Match* and *Easy* 5-pt scores that participants gave to their signs. In case participants iterated over a given trial in the *Revision* phase, the experiment software recorded a revision action and overwrote the data collected in the *Initial* phase with the revised data. In addition, participants were video-recorded and the operator took notes to collect their feedback.

4.4.1 Classification into Sign and Modality Categories. Following recommendations by Tsandilas [61], we rely on Fleiss' Kappa coefficient (κ) to assess the level of agreement between participants regarding their proposals for the different referents.

Before computing κ values per referent, we organize participants' proposals into different categories. κ then gives an indication of how much participants agree about the proposed category for a referent. κ's computation takes into account the chance bias to output a normalized value ($\in [-1, 1]$) that can be interpreted as follows: a positive value means agreement beyond chance (+1 meaning perfect agreement), and a negative value means disagreement beyond chance.

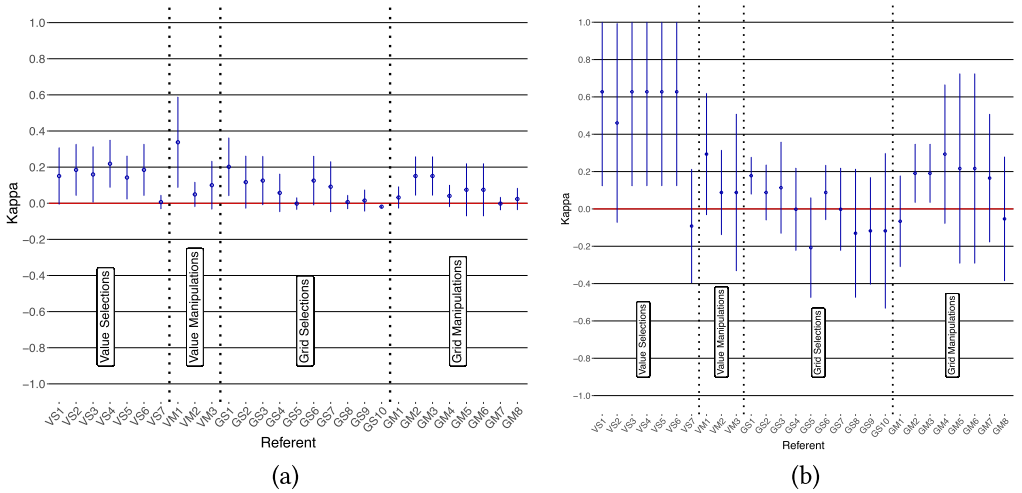

Fig. 6. (a) Agreement about *sign* proposals per referent. (b) Agreement about *modality* proposals per referent.

We analyze agreement at two different levels: at a fine level by classifying participants' proposals into *sign* categories (*i.e.*, specific series of input actions); and at a coarse level by classifying them into *modality* categories.[3]

4.4.2 Level of Agreement per Referent. The average level of agreement regarding the proposed *sign* per referent (Figure 6(a)) is low: $\kappa = 0.1 \pm 0.08$ (median=0.1, min=-0.02, max=0.34). The average level of agreement regarding the proposed *modality* (Figure 6(b)) is only slightly better: $\kappa = 0.18 \pm 0.26$ (median=0.14, min=-0.2, max=0.6). While agreement about modality is low on average, we observe a level of agreement that significantly varies across referents. In particular, a comparison based on confidence intervals regarding the proposed modality shows some difference in κ between groups of referents VS and GS: 0.512 with 95% CI [0.117, 0.906]. Figure 6(b) illustrates this, showing that κ is much higher for value-level selections (VS) than for grid-level selections (GS).[4] Below we report the general trends observed for value-level and grid-level interaction, by group of referents.

Value-level selections (VS): participants heavily used the pen tip to delimit the scope of a selection within a cell. However, the level of agreement for these referents remains quite low primarily because *i)* some participants (9) delimited the scope using a Horizontal Line while some others (6) delimited it by circling it with an Enclose mark; and *ii)* other participants – likely influenced by their experience with current spreadsheet programs – first performed a double tap or dwelled in the cell before inking inside.

Value-level manipulations (VM): moving the selected substring (VM_1) reaches a relatively good level of consensus. All participants but one relied on a Drag-based event. However, they used different modalities: 9 participants used the pen tip, and 6 used their finger. Interestingly, finger drags were usually preceded by a Dwell event. This was likely to make it clear that this drag applied to the value level rather than the grid level. To delete a value selection (VM_2), participants heavily relied on the pen (15) but were split between using the pen tip (7) or the eraser (8). They also used

[3] Appendix A.2 provides the detailed definition of a *sign* in our study.
[4] Readers interested in the systematic comparison between referents can run analysis scripts provided as supplemental material, which perform such comparisons between pairs of referents and pairs of referent groups.

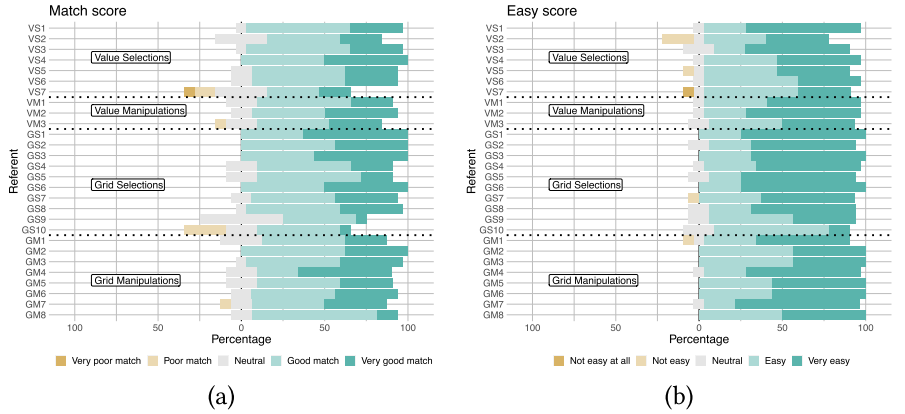

Fig. 7. (a) Distribution of Match scores per referent. (b) Distribution of Easiness scores per referent.

different event types. For example, 2 participants simply dragged the value selection away by dropping it in the background, while others preferred a custom mark to strike through the selection using a Horizontal Line (4) or a ZigZag custom mark (5).

Grid-level selections (GS): the level of agreement is particularly low regarding both the proposed sign and the proposed modality. When selecting a single cell (GS_1), a column (GS_3) or a row (GS_6), the simplest proposals consisted of a Tap event either on the cell, column header or row header. 8 participants consistently made such a proposal but were split regarding the modality: 4 used the pen tip, 3 used their finger while the others used either the pen tip for cell and column selections or their finger for row selections. We observed other proposals such as an Enclose mark around the whole value of a cell to select it (3) or a Flick gesture to select a row or a column. To select a continuous range of cells (GS_2), columns (GS_4) or rows (GS_7), we observed a similar kind of variability but involving a Drag event this time, proposed by 8 participants.

Grid-level manipulations (GM): the level of agreement is low as well. There seems to be stronger agreement regarding the use of the pen modality in comparison with grid-level selections although a statistical comparison is not really conclusive (difference in κ between groups *GM* and *GS* regarding modality is -0.155 with 95% CI [-0.405, 0.095]).

4.4.3 Match and Easy Scores. Match and Easiness scores given by participants to their proposals are reported in Figure 7. We use the ARTool package [13, 63] to analyze the effect of *Referent* on Match and Easiness scores. We observe that participants are particularly unhappy with a couple of proposals. VS_7 (*generalizing the selected substring of a cell value to all values in the column*), GS_9 (*select all cells holding a given value in a column*), and GS_{10} (*select all rows holding the same value in a given column*) are given match scores that are overall lower than the other referents. These three referents have match scores that are significantly lower than that of VS_4, GS_2 and GS_3 ($p < 0.05$). This is not particularly surprising as those three operations are typically not doable with direct manipulations in existing spreadsheet programs. It seems that participants are not able to identify a satisfying way of invoking them even when they are free to choose how to do so. Participants were satisfied with the rest of their proposals overall.

As illustrated in Figure 7(b), participants found all their proposals easy to perform. The only exception was GS_{10} (select all rows holding the same value in a given column) whose score is significantly lower than that of many other referents. The difference is significant ($p < 0.05$) when compared to VS_1, VM_2, GS_1, GS_3, GS_5, GS_6, GM_4, and GM_7.

4.4.4 Summary of Findings. While our participants demonstrated a lot of variability regarding the sign proposals they made for the selections and manipulations considered, our study still yields insights that can inform the design of pen + touch spreadsheet interaction, complementing the first observation drawn from the analysis of commercial spreadsheet programs (**Obs$_1$**, Section 3).

— **Obs$_2$**: *The choice of input modality does not disambiguate between grid and value layers.* Participants' proposals do not elicit a clear difference between the pen and touch modalities. The modality alone cannot be used to distinguish between interactions aimed at the grid layer and interactions aimed at the value layer. Seven of our participants actually stated explicitly that they would use one or the other interchangeably for most actions.
— **Obs$_3$**: *The pen is often used for value-level selections and deletion.* Two notable exceptions to the above observation are value-level selections and delete manipulations, which participants systematically performed with the pen. This was likely because of affordances specific to the pen such as its higher precision (useful to point or drag between two characters) and/or the use of pens on paper to circle or underline text. Participants also leveraged their experience with erasers that are often found at the other end of those pens. Most participants turned the pen upside-down and put it on the surface to erase values or grid elements.
— **Obs$_4$**: *Only a few simple custom marks are used.* A few participants proposed to use a custom mark to activate a command. However, no two participants ever proposed the same mark for a given referent (except the sorting manipulation GS_7 for which two participants used an arrow mark). Furthermore, the set of marks that participants proposed was limited to a small set of simple marks. While such command marks can act as efficient shortcuts for expert users [1], participants did not spontaneously propose to use them.
— **Obs$_5$**: *Pen and touch are seldom used together, multi-touch is almost never used.* Interestingly, there were few proposals that involved pen and touch together. The few instances we observed are for grid-level selections that involve cell-value-matching criteria (GS_9 and GS_{10}), whose specification requires two distinct scopes such as, *e.g.*, a cell value selected with the finger and the header of a column with the pen for GS_9. Finally, the number of proposals that involve multi-touch gestures for selections or manipulations is almost null.

5 INTERACTION TECHNIQUES FOR SPREADSHEETS ON INTERACTIVE SURFACES

Taken together, our analysis of pen and touch input in commercial spreadsheet programs and the elicitation study results establish one thing: there is little agreement about how to use pen and touch to interact with spreadsheets. This is true among interface designers (Section 3), among end-users (Section 4), as well as between designers and end-users.

In this section, we describe the set of interaction techniques that we designed to mitigate frictions between layers. We showcase these techniques using a prototype Web-based application (whole interface illustrated in Figure 15, see Appendix B.1 for implementation details). Most figures show data from the Titanic dataset [27]. The companion video also demonstrates most of the interactions described here.

5.1 Mitigating Friction between the Grid and Value Layers

5.1.1 Selection. While we did not observe a clear difference between pen and touch for disambiguating between grid-level and value-level selections in the elicitation study (**Obs$_2$**), there was a strong tendency to use the pen for value-level selections inside cells (**Obs$_3$**). Striving for consistency, we make the choice to have all selections performed with the pen. In order to differentiate a value-level selection from a grid-level selection, we adopt a strategy that takes the graphical context into account [14, 53] to implicitly disambiguate user intent by differentiating input performed *within a cell* from input performed *across cells*, as detailed below:

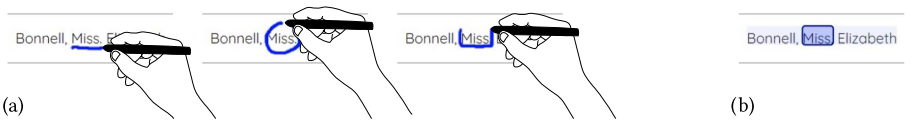

Fig. 8. Performing a subcell selection by drawing an arbitrary mark: (a) the pen leaves ink, providing an indication of what substring will be selected; (b) once the pen lifted, the ink is replaced by a beautified representation of the selection, consisting of the substring actually delineated (dark blue) and its complements on either or both sides (light blue).

- *Grid-level Selection.* Basic grid-level selections are performed as follows: a pen tap selects the underlying cell; a pen drag selects the corresponding range of cells; tapping a column or row header selects the corresponding cells. In each case the previous selection is cancelled.
- *Value-level (Subcell) Selection.* In our elicitation study, participants performed different gestures to select only part of a value inside a cell:[5] underline it, circle it, or draw a roughly rectangular shape around it. To support a variety of marks, we adopt a relaxed strategy for substring selection that only involves the inked mark's bounding box: 1) differentiate between value-level and grid-level selection simply based on the position and size of the bounding box (fully inside a cell *vs.* spanning more than one cell); and 2) select the substring of characters that fall within the horizontal range defined by the mark's bounding box. Users do not have to learn any specific mark, as they can use any simple gesture (**Obs$_4$**) to delineate the substring. Figure 8(a) shows three different ways to select a substring. Once the pen lifted from the surface, the system beautifies the selection (Figure 8(b)): the raw ink is erased and replaced by a set of rectangles that precisely delineate the selection. The most visually-salient rectangle corresponds to the subcell selection. Two low-contrast rectangles, one on each side, correspond to the complements to the main selection and afford some manipulations. Tapping any of these complement rectangles with the pen makes it the main subcell selection, the original one becoming part of the complement. Usage examples of complements are given in Section 5.1.2.
  The only mark that is confined within a cell and that does not select a substring is a short vertical drag. This specific mark, which has no horizontal range, positions the text caret in-between the two closest characters and enters value-level editing mode.

Once a selection has been made, users are able to extend it in two ways: multiple selection and semantic selection. The scope of these extensions differs depending on whether they are performed at grid level or value level, as detailed next.

- *Grid-level Multiple Selection.* To avoid cancelling the previous selection but rather modify it, users can touch anywhere in the current selection with the non-preferred hand and perform pen taps or drags with the preferred hand. This will add or remove cells to the existing selection rather than reset it. Such incremental selections are typically useful when performing table reshaping operations [20], for instance to extract only a few columns from tables featuring many dimensions. They are available in commercial spreadsheet programs on the desktop but not on interactive surfaces [41].
- *Value-level Multiple Selection.* On the one hand, a *grid-level* selection lets users select multiple cells, but the entire string value gets implicitly selected in each of the cells. On the other hand, a *value-level* (subcell) selection lets users select only a subset of the characters representing a cell's value, but is confined to that specific cell. Our selection model includes a generalization

[5]In the remainder of the paper we refer to this as a *subcell selection.*

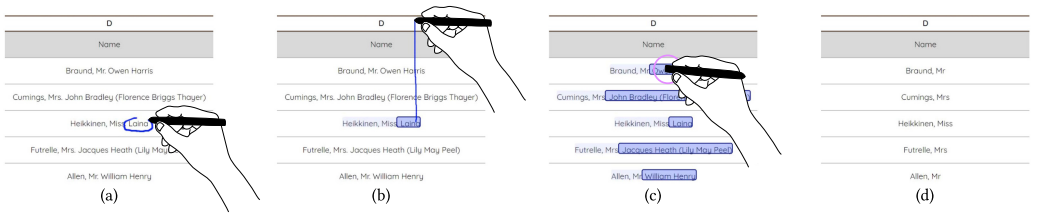

Fig. 9. Removing a substring across all cells of a column that contains the surname, title and first name of people embarked on the Titanic: (a) circling the end of one cell value using the pen, from the comma separating the title from the first name, to the end of the string; (b) dragging with the pen from the resulting subcell selection to the column header to generalize it to all values in the column; (c) tapping with the pen eraser on any of those subcell selections deletes them all, leaving only the surname and title (d).

mechanism that enables users to perform a subcell selection in a cell and then apply it across a range of cells. This is particularly useful to edit multiple cells at once, a manipulation that is typically impossible without resorting to scripting tools in any existing spreadsheet program. For example, value-level multiple selection can be used to remove the state code from a list of US cities at once. Users simply have to make a subcell selection in one of the cells, and then drag it over the target range of cells, or all the way to the column's header to apply it to all of its cells – the header representing an abstraction over cells holding data in the column [8]. All matching substrings in range get selected. The value of such a feature would be very limited if it only applied to cells that hold the exact same value. We thus implement an algorithm that infers a pattern from the initial subcell selection and applies it to the other cells. The corresponding generalization algorithm is described in Appendix B.2. In Figure 9, first names are removed from all cells in a column that contains full names.

- *Grid-level Semantic Selection.* In some desktop word processors, a single-click positions the text caret between the two closest characters; a double-click selects the entire word; a triple-click selects the entire paragraph. Similarly, in our selection model, multiple taps with the pen in the same place trigger semantic selections. A single-tap selects the cell; a double-tap selects all the cells that have the same value in the parent column; a triple-tap selects the lines that those cells belong to.
- *Value-level Semantic Selection.* In the same fashion, a double-tap on a subcell selection selects all matching substrings across all cells in the parent column, and a triple-tap selects the rows that hold matching cells. In cases where the substring to select is the exact same across the range, a double-tap on a subcell is functionally equivalent to dragging the subcell selection to the column header in order to generalize it to the entire column.

5.1.2 Manipulation. Spreadsheets are used for a variety of purposes, and the proportion of *direct data* – as opposed to *derived data* computed using formulas – is significant [2, 12]. The ability to modify these direct data (insert and remove text, rearrange substrings) is a key interaction with spreadsheets, which is paradoxically poorly-supported on interactive surfaces. In this section, we describe the direct manipulations users can perform on selections.

Moving Selections. As all selections are performed with the pen, touch can be used for manipulations. Users can simply drag a selection with their finger to move it, removing the need for a dwell time. At the grid level, touch essentially performs layout transformations: dragging a grid-level selection with a single finger moves the corresponding cells to a new location on the grid. Entire rows or columns can be moved by dragging their header. Copy & paste is supported as well:

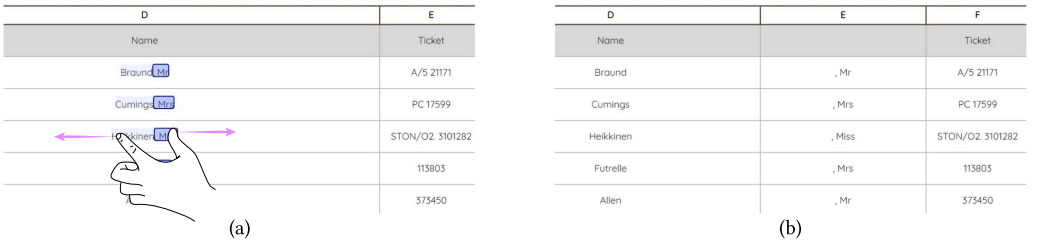

Fig. 10. Splitting title and surname in two separate columns. After generalization of the person's title selection (from comma to end of string), (a) performing an outward pinch gesture with one finger on the main subcell selection and another finger on the complement splits the two in separate columns (b).

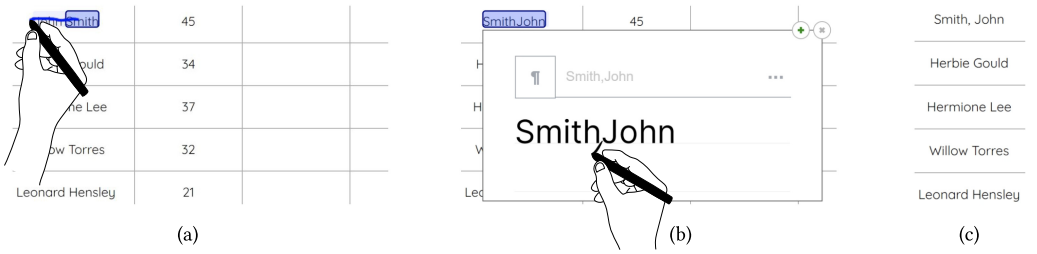

Fig. 11. Inverting first name and surname (`John Smith` ⟶ `Smith, John`) by direct value-level manipulation in one cell: (a) dragging the surname subcell selection to the left; (b-c) inserting a comma followed by a white space between surname and first name.

a single-finger double-tap on a selection copies it to the clipboard, and a dwell pastes it. Finally, users can also sort rows by performing a vertical flick gesture in the column they wish to sort by, as in TableLens [47].

Transforming Selections. The above manipulations involve only one finger. But other manipulations are best expressed using two fingers. An outward pinch gesture performed with one finger on a subcell selection and the other on its complement splits the two parts of the string in separate columns. Again, if the gesture is performed on a generalized subcell selection, all cells get split at once, as illustrated in Figure 10. This is useful for instance to split a formatted date (`dd/mm/yyyy`) in three columns [18], or make a landline prefix (`tel:` *vs.* `fax:`) its own dimension [20]. The converse layout transformation – merging two columns – is achieved using an inward pinch gesture.

The touch modality may not always be precise enough when performing value-level manipulations, however. We thus make the choice to enable moving subcell selections with pen drags. The disambiguation between a selection and a manipulation is simply based on the start location of the pen drag: if the movement starts on a selection, it moves that selection and relocates it at the index where it is dropped; otherwise, it initiates a novel selection. As illustrated in Figure 11, this makes it easy for instance to invert first name and last name, possibly adding syntactic elements in between, as discussed in Rigel [9].

Deleting Selections. We assign the delete manipulation to the other end of the pen throughout the interface, as we observed many elicitation-study participants turning the pen upside down to erase values (**Obs$_3$**). A tap with the eraser deletes the current selection. At the grid level, this will erase the cells' contents, leaving the grid untouched, unless the eraser taps a header, in which case the corresponding row or column gets deleted. At the value level, the eraser only deletes the

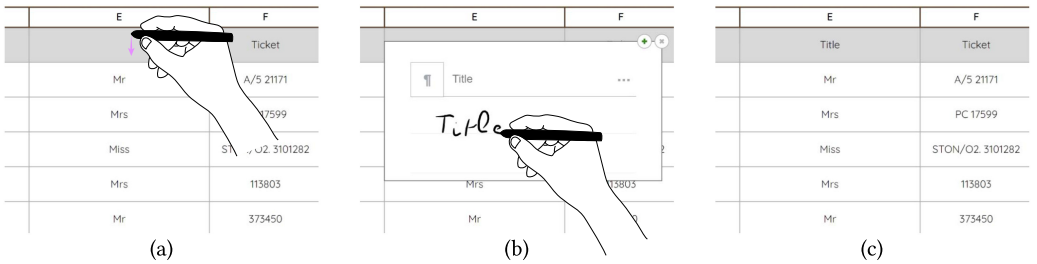

Fig. 12. Value-level editing. (a) Entering value-edit mode by performing a vertical pen drag gesture within the cell's boundaries; (b) handwriting the new cell value which gets recognized and (c) inserted in the cell.

selected substring, but possibly does this across rows if the subcell selection has been generalized to multiple cells (Figure 9(c) and (d)). The eraser can also remove foreign objects such as annotations (freeform ink and post-it notes, illustrated in Figure 15) when they are supported [44].

Editing Selections. Another important value-level manipulation consists of inserting new text or replacing existing text, for instance to fix wrong values in a table as illustrated in ActiveInk [51]. Making a short downward vertical mark in a cell or dwelling on a selection pops up a widget for handwritten text entry (Figure 12). Consistent with other manipulations, it is possible to edit multiple cells simultaneously by invoking the widget on a multiple cell selection. Our prototype uses an external library[6] to parse the ink input by users. It enables the input of new values and the editing of existing cell values as well. It also includes predefined gestures to, *e.g.*, remove a word by scratching it. Handwriting recognition libraries of this type have actually become good enough that they can parse mathematical expressions, opening the door to pen + touch spreadsheet formula authoring, which would be worth exploring as future work.

A few simple syntactic transformations [18] could also be performed directly with short vertical upward pen movements on a selection. For instance, in our current prototype, we map this event to letter case toggling, as this operation is often complementary to the value-level substring transformations introduced earlier.

Most value-level selection and manipulation actions presented in this section are not possible in current commercial spreadsheet programs, even on the desktop. The selection model and manipulation techniques presented here make all of this possible by direct manipulation, relying on simple input actions only. The companion video demonstrates how such capabilities enable advanced editing operations, which involve syntactic, semantic and layout transformations [19] with quick and simple interactions.

5.2 Mitigating Friction Between the Grid and Navigation Layers

Our analysis of commercial spreadsheet programs identified some friction not only between the grid and value layers, but between the grid and navigation layers as well (Section 3). At the same time, we observed that multitouch input was dedicated to navigation in all spreadsheet programs we examined (**Obs$_1$**).

In the selection model and manipulation techniques presented in the previous section, we were careful not to introduce any action that would have conflicted with the typical two-finger slide and pinch gestures used for navigation. The few two-finger manipulation gestures mapped to merging and splitting selections are easily disambiguated based on the specific elements they are

[6]MyScript iinkjs [39] submits the input ink to a Web service and gets the recognized string as a result, with very little latency, providing incremental feedback after every input stroke.

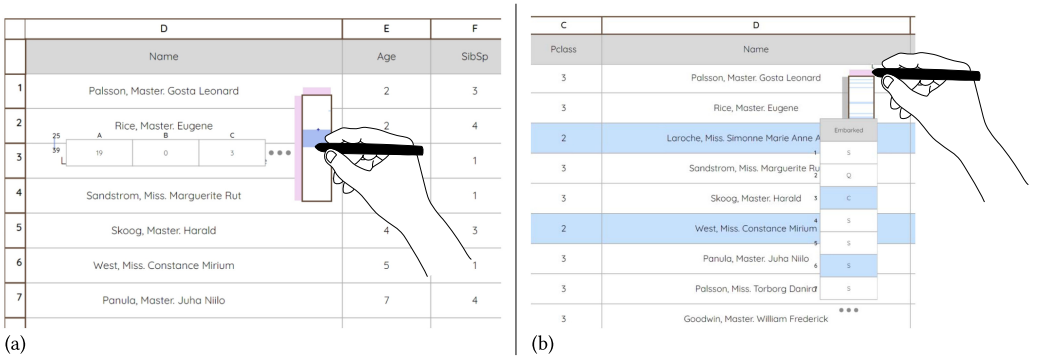

Fig. 13. Minitable widget for pen selections in large tables. (a) Selecting from rows 25 to 39 using the minitable. A preview of the row that falls below the cursor (row 39 here) is visible left of the pen tip. Row selection start and end indices are also displayed. (b) Brushing through columns in the gutter above the table shows a preview of the columns (column L here, as indicated next to the pen tip, showing where passengers embarked). Light blue lines in the minitable give an overview of which rows are currently selected (in this example, all rows with Pclass=2).

performed upon. By doing so, we can leverage the only interaction coherently implemented across spreadsheet programs, and minimize friction between the grid and navigation layers.

We aim to further reduce this friction by designing widgets that streamline grid-level selection and navigation. These are relevant primarily to cases where the spreadsheet holds large tabular datasets – as opposed to smaller, more informal spreadsheets with a looser structure. Observing that people often find it easier to utilize their fine motor skills with a pen rather than touch [48], we designed two types of pen-operated widgets, namely the Minitable and the Minivis, that enable advanced grid-level selections and make navigation in the spreadsheet faster.

Minitable. Mapping tables to a small overview, the *Minitable* (Figure 13) is a minimap that lets users brush through its rows and columns and select them without having to pan the main viewport. Figure 13(a) illustrates the selection of a range of rows from the minitable. When dragging vertically with the pen inside the minitable, the system displays a preview of the row that falls below the pen tip. The preview is placed next to the minitable to avoid problems of divided attention. Users can also get a preview of a row (resp. column)'s cells by dragging the pen in the gutter (painted pink) on the side (resp. top) of the minitable, without changing the current selection. Figure 13(b) illustrates this on columns. Similar to the minimap found in recent code editors, the minitable can be used to quickly scroll to a distant location in the worksheet: releasing the pen while inside the gutter (drag or tap) will automatically pan the worksheet to the corresponding location. Beyond navigation and selection, the minitable enables some manipulations as well. The simultaneous use of pen and touch enables users to move columns or rows to distant locations by direct manipulation: holding a column with the non-preferred hand, users select a distant location with the pen on the minitable and release the column once there. The combined use of pen and touch not only reduces friction between the grid and navigation layers, but actually facilitates a manipulation that was either tedious (on the desktop) or impossible (on interactive surfaces).

Minivis. The minitable provides means to interact with large tables, but essentially allows selections of contiguous blocks of rows or columns, regardless of their contents. It enables *grid-based* selections. Minivis plots (Figure 14) are complementary. They enable *value-based* selections. A minivis plot shows the distribution of values in a column depending on its type: a bar chart for

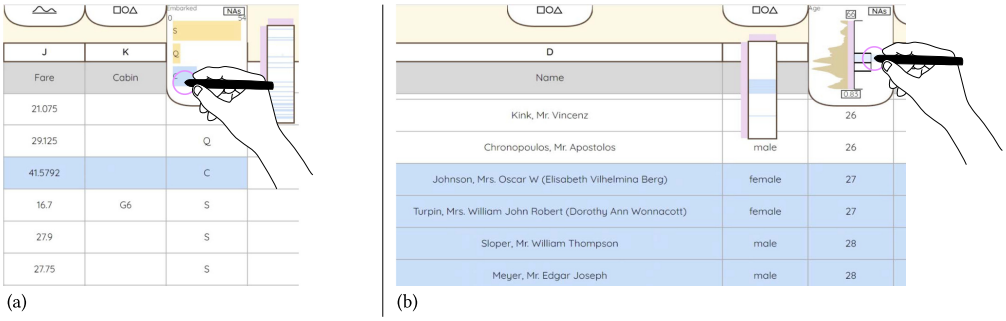

Fig. 14. Minivis plots. (a) A bar chart shows the distribution of values for categorical variables – here one of three ports where passengers could embark on the Titanic. The blue bar indicates that the user has selected rows featuring C (for *Cherbourg*) as the port of embarkation. (b) A density plot and a box plot are juxtaposed to visualize the distribution for quantitative variables – here the age of passengers. The blue area in the box plot indicates that the user has selected rows featuring an Age value that falls in the third quartile.

categorical columns (Figure 14(a)); a chart composed of a density plot on the left and a box plot on the right (Figure 14(b)) for quantitative columns. Minivis plots help characterize the distribution of values [47] and identify rows of particular interest (modes, clusters, outliers [57]). They are interactive, and here again the pen acts primarily as a precise selection tool. Users can perform a variety of value-based selections such as, *e.g.*, *"all rows of a given category"* by tapping on the corresponding bar (Figure 14(a)); *"values that fall in a given quartile"* by tapping the corresponding region in the box plot (Figure 14(b)); or a range of values by dragging vertically on the density plot. Minivis plots are similar in spirit to scented widgets [62], but applied to row selection in tables. Pink gutters similar to those of the minitable let users preview distribution values without actually changing the row selection or navigating the table.

Both the minitable and minivis plots leverage the specific affordances of the pen. The high motor precision and limited visual occlusion of the pen tip (compared to finger touch) make it possible to efficiently perform precise selections in spreadsheets holding large tables while keeping the visual footprint of those widgets reasonably small. They decrease the need for navigation actions, which is beneficial for different reasons depending on the interactive surface's dimensions: on a handheld tablet because few rows can be shown at once; on a larger digital drawing board because navigation often requires performing large movements. To further optimize screen real-estate use, minivis plots are only shown on-demand by pulling them from above the corresponding column header.

6 QUALITATIVE STUDY

In the previous section we have seen how we use the expressive power of pen and touch to design interaction techniques that reduce friction between the grid, value and navigation layers. We now report on a semi-structured qualitative study [3] aimed at gathering evidence about users' ability to appropriate those techniques. We developed a prototype (Figure 15) that implements all the techniques described in the previous section and supports handwritten annotations as found in [44]. Two configurations among those we examined are now offering this feature on interactive surfaces (dashed cells in Table 1). Those annotations typically reside on yet another layer, and we wanted to see if this might cause unanticipated friction. Our prototype thus lets users create annotations, both freeform ones using a side palette (Figure 15(d) & (e)) and post-it notes (Figure 15(f) & (g)).

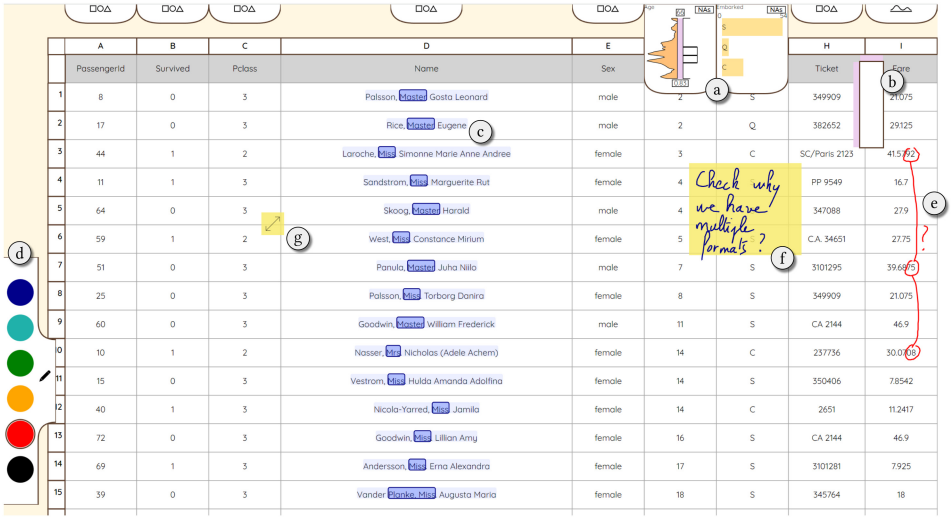

Fig. 15. The prototype Web-based application used in the qualitative study. Several features are illustrated in this screen capture: (a) minivis plots for row selection; (b) minitable for grid selection and worksheet navigation; (c) substring selection across cells; (d) annotation palette; (e) freeform ink annotation; (f) post-it annotation; (g) post-it annotation minimized.

There is much diversity in the expertise of people who use spreadsheets [8], for what purpose they use them, and how they use them [2], ranging from lay users who see them as a way to give some structure to their data to experts who master complex formulas and macros. But we were primarily interested in evaluating how users appropriate the new direct manipulations and how effective they are at reducing friction. We thus purposefully recruited participants who use spreadsheets to interactively explore and wrangle [30] their data. While such fairly advanced information workers are not the only ones who would benefit from the new techniques, we hypothesized that they were most likely to try them and give meaningful feedback in the relatively short time span (approximately 1 hour) of our semi-structured qualitative study.

6.1 Participants and Apparatus

Six volunteers (2 women, 4 men), all right-handed, aged 23 to 34 year-old (average 27.7, median 26.5), participated in the experiment. P1 is a musician; P2-P6 are computer scientists with a specialization in data visualization or HCI. All of them frequently manipulate tabular data but use different tools for this purpose. P1 prefers spreadsheet programs; P6 primarily uses scripting languages; P2-P5 use both spreadsheet programs (Apple Numbers, Microsoft Excel) and scripting languages (R, Python). None of them has used pen + touch to interact with spreadsheets before. P4 and P5 have accessed Google Sheets on a smartphone on a few occasions.

The apparatus used for the study is the same as that of the elicitation study (see Section 4.2), but with a display scale set to 200% to compensate for the very high pixel density (3840 × 2160 pixels on a 24" screen). The software is implemented as a Web application (Figure 15, see Appendix B.1 for details about the technical implementation).

6.2 Task and Procedure

Participants entered the room, read and signed a consent form that explained the overall goal and procedure of the experiment. They had been asked to provide a dataset of their own choosing –

that they were working on or had been working on recently – several days before the scheduled session in order to check that the dataset worked properly with our prototype. The study then consisted of the following steps.

- A short structured interview (≈ 10 minutes) during which participants were asked about their prior experience with pen + touch devices; experience with data manipulation and use of tabular data in general; the tools they were using and the challenges they were facing when working with data.
- Then the facilitator introduced the system (≈ 25 minutes), demonstrating[7] the possible actions to participants, who reproduced them right after the demonstration, one at a time. The introduction was carried out using the Titanic dataset [27] used in many figures in this paper. Similar selections and manipulations were grouped together, resulting in 30 short sequences. Throughout this hands-on introduction, the facilitator ensured that the participant had a good understanding of the possible manipulations so far and answered any question they may have had.
- Participants were then presented with their own data and could manipulate them freely. This second phase (34-to-55 minutes depending on the participant) was open-ended. There was no predefined task. Participants explored and manipulated their data at will following a think-aloud protocol. If a participant ran out of ideas too quickly, the facilitator asked questions about their dataset to foster new ideas and manipulations.
- Finally, participants filled out a questionnaire (≈ 10 minutes) to evaluate how useful and easy-to-use different features were. Data were collected using 5-point Likert scales, ranging from *"1. not useful at all"* to *"5. very useful"*; and from *"1. not easy to use at all"* to *"5. very easy to use"*.

6.3 Results

While some participants used the system without a clearly-defined goal, others managed to achieve several of their goals, and even made unexpected discoveries about their data, as discussed next.

Figure 16(a) gives an overview of the think-aloud phase. Color hue encodes the different types of interactions. The rectangle's position, above or below the participant's timeline, indicates that the interaction sequence is positive or negative overall. Sequences were coded as follows. An interaction sequence was considered positive if it led to the intended effect, or if it led the participant to orally formulate a hypothesis, a conclusion, or a positive comment related to it. An interaction was considered negative if it did not lead to the intended effect, or led the participant to orally formulate a negative comment about it. Positive and negative comments about the system as a whole, that cannot be linked to a specific interaction sequence, were coded as general comments.

Figure 16(b) visualizes the relative time spent performing manipulations of different types: grid-level basic operations; value-level direct manipulations; minitable and minivis selections; annotations. The legend also indicates the ratio of positive *vs.* negative sequences for each type.

6.3.1 Basic Operations. Unsurprisingly, basic operations (navigation, simple grid-level selection) amount to a significant proportion of all manipulations and are intermingled with other types of interactions. Participants were generally satisfied with them (85% positive).

All participants used two-finger gestures to navigate the worksheet, in line with **Obs$_1$**. Combined with selections to highlight particular values, panning the worksheet helped relate data points that were too distant to be shown simultaneously on screen. For instance, P5 was able to

[7]The script used for this introduction, as well as data analysis scripts and post-hoc questionnaires, are available as supplemental material.

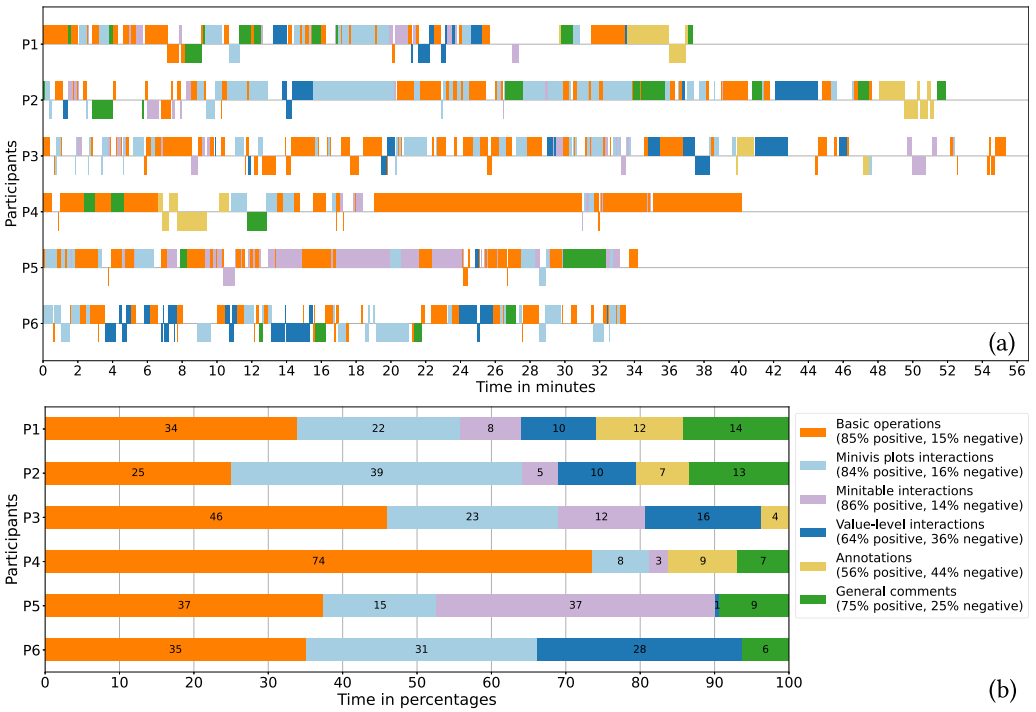

Fig. 16. (a) Interaction sequences of the think-aloud phase, with one timeline track per participant. (b) Relative time spent performing interactions of different types, per participant.

make sense of some data points that they had initially considered as outliers. They also identified patterns within selections to formulate hypotheses on their data. When navigation involved too much back and forth, some participants (P1, P2, P4 and P6) rather chose to perform a layout transformation, regrouping the columns of interest with basic single-finger drags. Bringing those columns together also enabled P2 and P6 to compare their distributions side-by-side thanks to the minivis plots.

P1 and P6 made use of two-finger pinch gestures to quickly merge or split columns. For instance, P1 fixed values that were wrongfully encoding different categories. P1 used column merge to create a new categorical column from two others, commenting that *"it enabled me to create new correlation data that I could then plot and compare."* The ability to sort values quickly with a single-finger flick on a column was also well-received by P3, P4 and P5 who used it many times to successively sort columns either for exploratory purposes (P3, P4) or for arranging their table according to a specific sorting they had in mind.

In line with **Obs$_3$**, participants also appropriated the pen eraser quickly, rating it 5 on the usefulness scale (Figure 17(a)). Using the pen for selection and then the eraser for deletion, P2 removed all rows but the ones with a specific value in one column, before deleting that column.

Semantic selections, which are difficult to achieve with regular spreadsheet programs, were found useful on multiple occasions. For instance, P3 wanted to know if a particular value they had found was unique, and performed a pen double-tap on it to look for other instances. Conversely, P1 used the same interaction to check that multiple values they had edited were indeed still equal. They also used it to replace missing values by zeros in an effective way: first, input a `'0'` with the pen in an empty cell and copy it with a finger double-tap, and second, perform a semantic selection

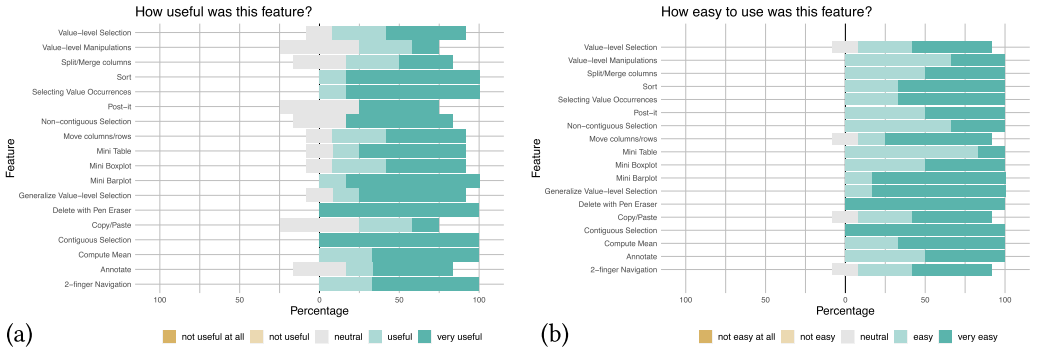

Fig. 17. Participants' evaluation of (a) usefulness and (b) ease-of-use for 18 interactive features using 5-point Likert scales.

with a pen triple tap on one of the remaining empty cells to select all of them, and then paste. P2 and P5 used the pen triple tap to select all rows with a matching value in a categorical column of interest. P5 observed that *"The triple tap is great, it allows to do things quickly."* Overall, participants rated these interactions very positively (Figure 17): 4.67 (easiness) and 4.83 (usefulness).

6.3.2 Value-level Manipulations. Subcell selections and manipulations do not exist in commercial spreadsheet programs. Yet participants appropriated them quickly to perform elaborate transformations by direct manipulation. We first report on how they approached the selection step, and then detail the purpose of their value-level manipulations.

Subcell Selection. Participants delineated selections in different manners, confirming the need to support arbitrary marks for this purpose (Figure 8): P1, P2 and P5 systematically circled substrings, whereas P3 and P6 systematically underlined them. We did not observe any confusion between grid-level and value-level selection – both performed with the pen – again in line with **Obs$_3$** and suggesting that our disambiguation strategy (Section 5.1.1), while very simple, is effective. Participants always applied subcell selections to a group of cells, never to a single cell only. *Multiple subcell selections* and *semantic subcell selections* thus played a central role in value-level manipulations. P5 performed semantic subcell selections, using a triple tap in order to select entire rows with a cell matching the selected substring.

Purpose: Table Reshaping. Some subcell selections were used to split columns. For instance, P6 wanted to split a column holding `''name, date''` values in two columns. They selected the date substring in a cell and generalized that selection to the entire column, obtaining two selection groups: the dates and their complements. They then only had to pinch outward, one finger on each group, as illustrated in Figure 10. Realizing that there were trailing commas after the names, they selected one, generalized it, and used the pen eraser to delete them all. P1 also split a column, which was holding `''foo - bar''` values. But they removed the dashes thanks to a semantic subcell selection double-tap as they were confident that this was the sole delimiter used in all values.

Purpose: Data formatting. Multiple subcell selections were also used to perform syntactic transforms [18]. P2 and P3 used them to format numbers in quantitative columns. P3 knew that all values had the same number of fractional digits and wanted to keep the first one only. They performed a right-side generalization with an (implicitly-fixed) number of characters and erased it. P2 had a similar intent, but in their case the count of fractional digits varied from cell to cell and they

adopted a different strategy. Aiming to only keep the first three digits, they selected them in a cell, generalized that selection, and then inverted it by tapping on the complement, effectively selecting all trailing fractional digits beyond the first three (when any). They actually had to perform this transformation in several columns, and wished it were possible to generalize subcell selections to multiple columns at once, suggesting this might be achieved by dragging the pen across target column headers, explaining that *"It would do the same type of selection on the other columns, because I know that they share the same properties."* P2 also relied on a semantic subcell selection to replace decimal points by commas. Tapping on the selection of all points with the pen, they opened the ink text editor, scratched the point and wrote a comma instead, effectively replacing them all in a matter of seconds.

P1 and P6 combined subcell selections with simple gestures (**Obs$_4$**) to format columns containing string data. The small upward stroke gesture from a subcell to change letter case gave them *"the feeling to uplift the selection."* They found it easier to make this sort of edit than with other spreadsheet programs: *"Once I have formatted a cell with the proper casing, I copy/paste it to all other cells that need to be fixed. It's very tedious."* P6 went further with the manipulation of their selection: they dragged it from right to left, effectively swapping it with the complement. As the letter case was now wrong, they changed the first characters to uppercase, and the middle characters – formerly the first characters – to lowercase, all with a short series of swift pen gestures.

Purpose: Data editing. Finally, we observed multiple occurrences of raw data editing. For instance, P2 – who had inadvertently deleted the content of a cell – invoked the ink text editor with a short vertical drag in that cell and rewrote the string by hand. As mentioned earlier, P1 wanted to replace all missing values by zeros and achieved this by writing only one `'0'` and copying it at once in the remaining cells thanks to a semantic selection. P3 performed a similar syntactic transformation, replacing all second words in a range of cells by another word written with the pen.

Some participants tried to achieve more advanced editing on multiple subcell selections and highlighted one missing capability. The current interaction model allows erasing, moving, and replacing substrings. But in the latter case matched substrings can only be replaced by one unique new substring – written with the pen – across the range. For instance, coming back to the example in Figure 11, inverting first name and surname for all rows at once can be done with a multiple subcell selection (performed before the action depicted in Figure 11(a)). But inserting the comma and whitespace between surname and first name is not straightforward: invoking the ink text editor on surnames would then replace the surname in all cells with the same input value, irrespective of the original value. What is missing[8] is a conceptual equivalent to text caret generalization, which is somewhat similar to the multi-line editing cursors featured in modern code editors that enable inserting a string at possibly different positions on multiple lines at once.

We evaluated this issue as critical and iterated on our subcell manipulation techniques to address it. As illustrated in Figure 18, we added support for editing multiple *heterogeneous* subcell selections by visually representing subcell selections in the handwriting recognition widget. When invoked on a multi-subcell selection whose individual values differ, the widget symbolizes the block using a square glyph (Figure 18(b)) which act as placeholders,[9] leaving the user free to insert characters before and after it at will. The value manipulation described in Figure 11 for a single cell can now be performed on multiple cells at once, even if they hold different values.

[8]The absence of this capability, and the fact that the handwriting recognition algorithm often confused '0' with 'o', account for many (36%) of the value-level interactions categorized as negative (Figure 16).

[9]This glyph would ideally look similar to subcell selection rectangles (blue, rounded corners ▭) to better convey their role, but the iinkjs widget used for handwriting recognition does not support rendering arbitrary graphics out of the box.

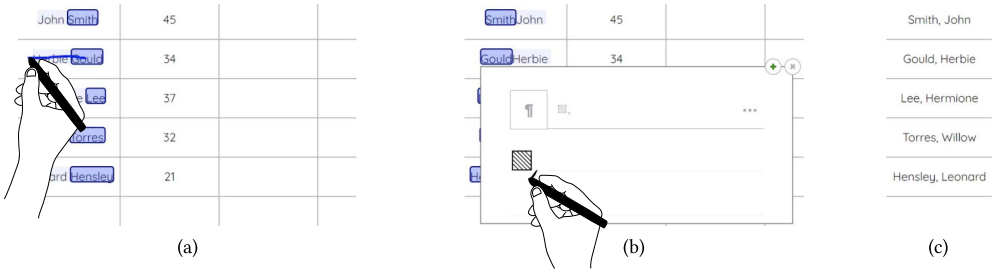

Fig. 18. Inverting first name and surname (`John Smith` → `Smith, John`) by direct value-level manipulation as in Figure 11, but for all cells at once: (a) dragging the generalized surname subcell selection to the left; (b-c) inserting a comma followed by a white space between all surnames and first names.

6.3.3 Minitable and Minivis Plots. As they use spreadsheets primarily for tabular data exploration and wrangling, study participants were naturally interested in the minitable and minivis plots (Figure 16(a)), except P4. Some participants used both the minitable and minivis plots (P1, P3), while other participants primarily used the minitable (P5) or the minivis plots (P2, P6). P2 and P6 actually interacted very often with minivis plots, performing as much or even more minivis interactions than basic operations (Figure 16(b)).

Participants used the minitable to navigate the worksheet, sometimes to reach a particular row or column quickly, but most often to move to a specific area regardless of its contents. The same type of interaction was used to quickly go back to the first row, first column, or both. The minitable was also used for some selections that would otherwise have been tedious because of grid *vs.* navigation friction. For instance, P2 brushed over the pink gutter with the pen to locate a precise row label, and continued dragging the pen inside the minitable's body to select a large number of rows from that point. The spontaneous use of pen + touch together was also observed on a few occasions (**Obs$_5$**). P5 used the minitable to drag & drop a row to a distant location in the worksheet by holding it with their non-preferred hand and quickly panning the worksheet with the pen on the minitable.

6.3.4 Annotations. Participants could annotate the spreadsheet by activating a dedicated side palette to make freeform annotations on a layer above the grid, or by drawing post-it notes that they could write upon and contract (Figure 15). However, participants did not use annotations much, either because it is not a regular part of their workflow, or because it was not particularly relevant in the context of a 30-to-45-minute analysis performed during a study.

Participants still provided feedback about this feature. They considered the freeform ink marks and post-it notes useful (rated 4.17, resp. 4.0) and easy to use (both rated 4.5) – see Figure 17. Figure 16-a shows a large number of annotation-centric interactions with a negative outcome, however (44%). This seems to be mostly due to the fact that annotations did not fully meet the expectations of participants who actually tried them. While most of these issues do not strongly relate to our interaction techniques, one seems worth mentioning. P4 wanted to have a place where annotations would always remain visible regardless of what part of the spreadsheet was visible – *"I want to have them always within reach"* (P4). They were in essence asking for a different annotation layer that would not be tied to the spreadsheet navigation layer. Simply adding another layer would likely cause even more friction, but as suggested by P4 this could be achieved by having a dedicated area in the spreadsheet's periphery that does not interfere with the grid – at the cost of the amount of screen real-estate dedicated to the display of the spreadsheet itself.

7 CONCLUSION AND FUTURE WORK

7.1 Summary

Pen + touch input has been demonstrated to offer interesting opportunities in terms of interaction design in a variety of contexts [24, 37, 52, 68]. In this article, we investigated its potential in the specific context of spreadsheets. Starting from the observation that even simple spreadsheet manipulations can be cumbersome when performed on interactive surfaces, we aimed to analyze the causes of the user experience deterioration and identified a particularly important one: *the grid, while being key to many interactions, acts as a layer that covers the entire workspace and hinders simple manipulations of the cell values that lie below it.* In addition, because interactive surfaces are primarily operated using direct input, spreadsheet programs that run on them actually feature a third layer dedicated to navigation actions that also covers the entire workspace. All these superimposed layers create ambiguity regarding the intent of many input actions, requiring users to perform cumbersome interaction sequences to reach the intended target elements. Having identified this key issue, we proposed a set of interaction techniques that leverage the expressive power of the pen to disambiguate between layers, letting users *break through the grid* and seamlessly select a variety of elements with simple pen-based actions.

We first examined a representative sample of commercial spreadsheet programs running on a range of operating systems and pen + touch hardware. This highlighted where inconsistencies and consensus across hardware and software configurations lay. We then conducted an elicitation study to better understand users' expectations when using pen + touch to interact with spreadsheets. Participants achieved little consensus, but the study yielded insights about the role of pen and touch as they perceive them, and the sort of gestures they would like to perform for direct manipulations. We then built upon those insights and our own experience as interaction designers to propose a new selection model. Building upon Pfeuffer et al.'s division of labor which states that *"the pen selects, touch manipulates"* [44], our model dedicates the pen to selections, but it also analyzes the spatial context of pen marks in order to implicitly disambiguate between the grid and value layers. We showcased how this model supports advanced spreadsheet operations with simple direct manipulation techniques. We then evaluated these techniques' usability by conducting a semi-structured qualitative study involving six information workers manipulating their own data. Results from this study suggest that people who use spreadsheets for data exploration and wrangling can easily appropriate the new direct manipulation techniques and find them useful.

The research question driving our investigation was how to improve direct manipulation in spreadsheets on interactive surfaces. We did not limit interactive surfaces to handheld tablets, but rather sought to design interaction techniques that can be used consistently across a range of interactive surfaces, from handheld tablets to large digital drawing boards. Pfeuffer et al. [44] designed *thumb*+pen interaction techniques where only the thumb of the non-preferred hand can be used as that hand holds the tablet. Contrary to their probe and to other prior work [29, 54], we do not assume that the non-preferred hand holds the tablet. It rather remains free to reach locations on the interactive surface other than an edge or a corner. We refrain from employing bi-manual interaction too much, however, reserving it for interactions of secondary importance for two reasons: situations where the non-preferred hand holds the device remain in scope of our approach; some prior work suggests that users are not necessarily inclined to use bi-manual pen + touch interaction frequently [68].

7.2 Future Work

The use of pen + touch for direct manipulation in spreadsheets opens multiple avenues for future work. Some are targeted at all users, others are rather targeted at information workers such as our

study participants, who frequently perform advanced data transformations. In that respect, our work can be seen as an early contribution to the emerging topic of post-WIMP interfaces for data manipulation, as discussed by Lee et al. [33].

One general avenue for future work is to investigate how to enable additional operations by direct manipulation without compromising usability. Chalhoub and Sarkar emphasize the key role of columns as a *"user-centric structure"* [8] beyond sorting and filtering, observing in their interviews with spreadsheet users that they can be appropriated for other operations including conditional formatting and formula authoring. Some of these *"column-based operations"* could lend themselves to specification by direct manipulation with pen and touch. Advanced transformations, including rearrangement and transposition [22] could be considered as well.

One particularly interesting issue to address is that of formula authoring, which is often tedious on interactive surfaces. As hinted at in Section 5.1.2, handwriting recognition libraries such as the one used in our prototype can now parse reasonably-complex mathematical formulas. This means that the pen can be used to manipulate not only plain values but spreadsheet formulas as well. But more interesting opportunities lie in the articulation between selection actions and formula authoring. As both grid-level selections and subcell selections are performed with the pen, the insertion of function calls and selection ranges could both be performed seamlessly with the pen. This would streamline an operation that often requires two devices even on the desktop, as cell selections are typically performed with the mouse while function calls are inserted with the keyboard.

Complementing this, subcell selections can be seen as transpositions to direct manipulation of spreadsheet text extraction functions (such as, *e.g.*, `LEFT()` and `RIGHT()`), combined with basic substring expression synthesis [18]. The visual reification of what essentially comes down to substring selection across cells does not have to be confined to the set of manipulations described in this paper, however. Subcell selections could be involved in formulas, as discussed above, but they could also be helpful in programmatic approaches to spreadsheet data manipulation [31, 55] where a variety of syntactic, semantic and layout transformations [56] get exposed as short programs.

Finally, mapping more operations to direct manipulation actions with regular pen and touch might prove challenging as few simple gestures remain available. Advanced operations – such as table pivoting, folding and unfolding to name a few – are arguably performed less frequently, mainly by expert users, and could be mapped to input capabilities that have only recently been investigated. These include the combination of pen, gaze and touch [42, 43]; the detection of hand posture [7, 35, 71], of tablet orientation [50]; the use of pen roll events, of passive surfaces to enable interactions outside the worksheet's bounds [34]; and the non-preferred hand's contact shape when working on a large interactive surface [38]. None of those capabilities have been considered in our interaction techniques as we purposefully limited ourselves to capabilities featured in off-the-shelf hardware. As some of them mature and eventually become widespread, it would be interesting to study how to integrate those newer capabilities, comparing them in terms of usability and efficiency to more conventional approaches such as exposing the same functionality *via* contextual menus.

APPENDICES

A ELICITATION STUDY

A.1 List of Referents

Table 2 lists questions for the 28 referents considered in the elicitation study. Some questions were actually a bit more detailed to give context (*e.g.*, the name of the columns to merge for GM_1). Study material is available as supplemental material.

Table 2. Referents Considered in the Elicitation Study

Scope	Action Type	Question
Value-level	Selection	(VS_1) How would you select the first character of a string in a cell? (VS_2) How would you select the comma (and only the comma character) in a cell? (VS_3) How would you select the last character of a string in a cell? (VS_4) How would you select the left part of a string in a cell? (VS_5) How would you select the sequence of characters ‘‘, NY’’ (and only that sequence) in a cell? (VS_6) How would you select the right part of a string in a cell? (VS_7) How would you generalize a sub-cell selection to its parent column?
	Manipulation	(VM_1) How would you move a selection within a cell? (VM_2) How would you delete part of the content of a cell? (VM_3) How would split a column into two columns?
Grid-level	Selection	(GS_1) How would you select a cell? (GS_2) How would you select a range of cells? (GS_3) How would you select a column? (GS_4) How would you select a range of columns? (GS_5) How would you select a set of columns? (GS_6) How would you select a row? (GS_7) How would you select a range of rows? (GS_8) How would you select a set of rows? (GS_9) How would you select the set of cells that have the same value in a column? (GS_{10}) How would you select the set of rows that have the same value for a specific cell?
	Manipulation	(GM_1) How would you merge two columns into one? (GM_2) How would you move a column? (GM_3) How would you move a row? (GM_4) How would you clear a cell? (GM_5) How would you delete a column? (GM_6) How would you delete a row? (GM_7) How would you sort a column? (GM_8) How would you fill up a column following the pattern of selected values?

A.2 Definition of a Sign

We define a *sign* as a series of events that is described along the following dimensions:

- The input modality, which can be Pen tip, Pen eraser, Single Touch, Multi-touch or Pen + Touch.
- The start and end locations of input, which can be a Column header, a Row header, a Cell, somewhere Inside-a-Cell, the Select-All button, a Column separator, the Background. We use Inside-a-Cell when the location within the cell itself carries information (*e.g.*, the participant draws a line between two specific characters of the value string).
- The input event type. We use four types of discrete events: Tap, Double Tap, Dwell and Flick. For continuous events, if the trace's trajectory does not bear meaningful information, we classify it as Drag. For other continuous events, we use the following five categories: Vertical Line, Horizontal Line, Diagonal Line, Enclose or ZigZag. A few traces do not fall in any of those categories and rather correspond to custom-shape gestures that we categorize into one of the following shapes: Circle, Arrow, Equal sign, Parallel sign, Less-than sign, V, Loops.

An event is defined as a combination of these dimensions, and a *sign* can be either a single event or a combination of atomic events. Our definition of a sign is quite specific not only regarding the description of an event but also regarding the transition between consecutive events. In particular, when a sign involves a couple of events that have the same modality, we make a distinction between the case where the input device remains in contact with the screen during the transition, and the case where it is lifted up between the two events. For example, a Dwell immediately followed by a Drag without lifting the pen up is different from a Dwell + Drag sequence where the user lifts the pen up after the Dwell. For the coarser *modality*-based classification, a participant's proposal is simply described as the combination of its events' modalities.

B IMPLEMENTATION DETAILS

B.1 Prototype Implementation

The Web-based prototype depicted in Figure 15 and used for the semi-structured qualitative study implements all interaction techniques from Section 5. It is developed entirely in JavaScript and D3 [4], and runs on the client side. Spreadsheet elements and interface widgets are all rendered in SVG. User pen and touch input events are handled with the W3C Pointer API [6].

The prototype is made available as supplemental material, and has been tested extensively with the Chromium Web browser on a Windows 10 PC connected to a Wacom Cintiq Pro. It also runs for instance on a Microsoft Surface Studio 2+, although some interactions that involve two simultaneous contact points are not supported so far because of input event API compatibility issues (the level of support for the W3C pointer API varies significantly across Web browsers and operating systems).

B.2 Generalizing Subcell Selections

Algorithm 1 below details how generalization works for subcell selections that include the cell's first character. Informally, priority is given to special characters such as dash, comma, and the like , falling back to different alphanumeric transitions (including juxtapositions of uppercase and lower case letters in either order) if no such character could be found. Other cases work similarly but are not detailed for the sake of conciseness: selections that include the last character use a mirror of the algorithm below; selections that include neither the first nor the last character use a combination of both algorithms; selections of the latter category consisting of a single character are generalized based on the transition from the previous character rather than the next one, consistent with the reading direction.

ALGORITHM 1: Generalization of a subcell selection that includes the first character of the source string. String indices start at 1. $s[i]$ returns the character at index i in string s. $s[i : j]$ returns all characters from string s between indices i and j included.

```
Def: 𝒟   // set of all delimiters, including special characters, arithmetic operators and currency symbols
Def: ℒ_l                                                    // set of all lowercase letters
Def: ℒ_u                                                    // set of all uppercase letters
Def: 𝒩                                                      // set of all digits
Def: enum U2L, L2U, A2N, N2A          // transitions from/to upper & lower case, from/to letter & number

Data: s                                                     // source cell (string)
Data: 𝒯                                                     // set of target cells (strings)
Data: i_s                                   // index of last char ∈ source subcell selection

ToI ← <pos: 0, type: None>                                  // transition of interest: <position, type>
if s[i_s] ∈ 𝒟 or s[i_s + 1] ∈ 𝒟 then
    ToI ← <pos: i_s, type: s[i_s]>                          // special character
else
    if s[i_s] ∈ ℒ_u ∪ ℒ_l then
        if s[i_s + 1] ∈ ℒ_u ∪ ℒ_l then
            if s[i_s] ∈ ℒ_u and s[i_s + 1] ∈ ℒ_l then
                ToI.type ← U2L                              // switch upper → lower case
                ToI.pos ← count(U2L, s[1 : i_s − 1])
            else if s[i_s] ∈ ℒ_l and s[i_s + 1] ∈ ℒ_u then
                ToI.type ← L2U                              // switch lower → upper case
                ToI.pos ← count(L2U, s[1 : i_s − 1])
        else
            ToI.type ← A2N                                  // switch letter → number
            ToI.pos ← count(A2N, s[1 : i_s − 1])
    else
        if s[i_s + 1] ∈ ℒ_u ∪ ℒ_l then
            ToI.type ← N2A                                  // switch number → letter
            ToI.pos ← count(N2A, s[1 : i_s − 1])
if ToI.type ≠ None then
    for t ∈ 𝒯 do
        i_t ← indexOf(ToI, t)          // get index of n^th occurrence of ToI.type where n = ToI.pos
        if i_t > 0 then
            select t[1 : i_t]                   // select up to index of n^th ToI occurrence in t
else
    for t ∈ 𝒯 do
        select t[1:ToI.pos]          // no delimiter identified, select based on original selection length
```

ACKNOWLEDGMENTS

We would like to thank all participants to our elicitation study, as well as the interviewees who participated in the qualitative study. We also thank Théo Imbert for his work on an early software prototype exploring pen + touch interaction with tabular data.

REFERENCES

[1] Caroline Appert and Shumin Zhai. 2009. Using strokes as command shortcuts: Cognitive benefits and toolkit support. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. Association for Computing Machinery, New York, NY, USA, 2289–2298. https://doi.org/10.1145/1518701.1519052

[2] Lyn Bartram, Michael Correll, and Melanie Tory. 2022. Untidy data: The unreasonable effectiveness of tables. *IEEE Transactions on Visualization and Computer Graphics* 28, 1 (2022), 686–696. https://doi.org/10.1109/TVCG.2021.3114830

[3] Ann Blandford. 2013. Semi-structured qualitative studies. In *The Encyclopedia of Human-Computer Interaction, 2nd edition*, Mads Soegaard and Rikke Friis Dam (Eds.). Interaction Design Foundation.

[4] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. 2011. D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics* 17, 12 (2011), 2301–2309. https://doi.org/10.1109/TVCG.2011.185

[5] Peter Brandl, Clifton Forlines, Daniel Wigdor, Michael Haller, and Chia Shen. 2008. Combining and measuring the benefits of bimanual pen and direct-touch interaction on horizontal interfaces. In *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI'08)*. Association for Computing Machinery, New York, NY, USA, 154–161. https://doi.org/10.1145/1385569.1385595

[6] Matt Brubeck, Rick Byers, Patrick H. Lauke, and Navid Zolghadr. 2019. Pointer Events Level 2 - W3C Recommendation. https://www.w3.org/TR/pointerevents2/. (April 2019).

[7] Drini Cami, Fabrice Matulic, Richard G. Calland, Brian Vogel, and Daniel Vogel. 2018. Unimanual pen+touch input using variations of precision grip postures. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST'18)*. Association for Computing Machinery, New York, NY, USA, 825–837. https://doi.org/10.1145/3242587.3242652

[8] George Chalhoub and Advait Sarkar. 2022. "It's freedom to put things where my mind wants": Understanding and improving the user experience of structuring data in spreadsheets. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems (CHI'22)*. Association for Computing Machinery, New York, NY, USA, Article 585, 24 pages. https://doi.org/10.1145/3491102.3501833

[9] Ran Chen, Di Weng, Yanwei Huang, Xinhuan Shu, Jiayi Zhou, Guodao Sun, and Yingcai Wu. 2023. Rigel: Transforming tabular data by declarative mapping. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 128–138. https://doi.org/10.1109/TVCG.2022.3209385

[10] Andy Cockburn, Amy Karlson, and Benjamin B. Bederson. 2009. A review of overview+detail, zooming, and focus+context interfaces. *ACM Comput. Surv.* 41, 1, Article 2 (Jan. 2009), 31 pages. https://doi.org/10.1145/1456650.1456652

[11] Andrew Crotty, Alex Galakatos, Emanuel Zgraggen, Carsten Binnig, and Tim Kraska. 2015. Vizdom: Interactive analytics through pen and touch. *Proc. VLDB Endow.* 8, 12 (Aug. 2015), 2024–2027. https://doi.org/10.14778/2824032.2824127

[12] Paul Dourish. 2017. Spreadsheets and spreadsheet events in organizational life. In *The Stuff of Bits: An Essay on the Materialities of Information*. The MIT Press. https://doi.org/10.7551/mitpress/10999.003.0005

[13] Lisa A. Elkin, Matthew Kay, James J. Higgins, and Jacob O. Wobbrock. 2021. An aligned rank transform procedure for multifactor contrast tests. In *Proceedings of the 34th Annual ACM Symposium on User Interface Software and Technology (UIST'21)*. Association for Computing Machinery, New York, NY, USA, 754–768. https://doi.org/10.1145/3472749.3474784

[14] Mathias Frisch, Jens Heydekorn, and Raimund Dachselt. 2010. Diagram editing on interactive displays using multi-touch and pen gestures. In *Diagrammatic Representation and Inference*. Springer Berlin, Berlin, 182–196.

[15] Mathias Frisch, Ricardo Langner, and Raimund Dachselt. 2011. NEAT: A set of flexible tools and gestures for layout tasks on interactive displays. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces (ITS'11)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2076354.2076356

[16] Travis Gesslein, Verena Biener, Philipp Gagel, Daniel Schneider, Per Ola Kristensson, Eyal Ofek, Michel Pahud, and Jens Grubert. 2020. Pen-based interaction with spreadsheets in mobile virtual reality. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 361–373. https://doi.org/10.1109/ISMAR50242.2020.00063

[17] Yves Guiard. 1987. Asymmetric division of labor in human skilled bimanual action. *Journal of Motor Behavior* 19, 4 (Dec. 1987), 486–517. https://doi.org/10.1080/00222895.1987.10735426

[18] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'11)*. Association for Computing Machinery, New York, NY, USA, 317–330. https://doi.org/10.1145/1926385.1926423

[19] Sumit Gulwani, William R. Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* 55, 8 (Aug. 2012), 97–105. https://doi.org/10.1145/2240236.2240260

[20] Philip J. Guo, Sean Kandel, Joseph M. Hellerstein, and Jeffrey Heer. 2011. Proactive wrangling: Mixed-initiative end-user programming of data transformation scripts. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST'11)*. Association for Computing Machinery, New York, NY, USA, 65–74. https://doi.org/10.1145/2047196.2047205

[21] William Hamilton, Andruid Kerne, and Tom Robbins. 2012. High-performance pen + touch modality interactions: A real-time strategy game esports context. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST'12)*. Association for Computing Machinery, New York, NY, USA, 309–318. https://doi.org/10.1145/2380116.2380156

[22] William R. Harris and Sumit Gulwani. 2011. Spreadsheet table transformations from examples. *SIGPLAN Not.* 46, 6 (Jun. 2011), 317–328. https://doi.org/10.1145/1993316.1993536

[23] Ken Hinckley, Xiaojun Bi, Michel Pahud, and Bill Buxton. 2012. Informal information gathering techniques for active reading. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'12)*. Association for Computing Machinery, New York, NY, USA, 1893–1896. https://doi.org/10.1145/2207676.2208327

[24] Ken Hinckley, Koji Yatani, Michel Pahud, Nicole Coddington, Jenny Rodenhouse, Andy Wilson, Hrvoje Benko, and Bill Buxton. 2010. Pen + touch = new tools. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST'10)*. Association for Computing Machinery, New York, NY, USA, 27–36. https://doi.org/10.1145/1866029.1866036

[25] Jane Hoffswell and Zhicheng Liu. 2019. Interactive repair of tables extracted from PDF documents on mobile devices. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI'19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300523

[26] T. Igarashi, J. D. Mackinlay, Bay-Wei Chang, and P. T. Zellweger. 1998. Fluid visualization of spreadsheet structures. In *Proceedings 1998 IEEE Symposium on Visual Languages*. 118–125. https://doi.org/10.1109/VL.1998.706154

[27] Will Cukierski and Jessica Li. 2012. Titanic - Machine Learning from Disaster. (2012). https://kaggle.com/competitions/titanic. Accessed 11-17-2023.

[28] Zhongjun Jin, Michael R. Anderson, Michael Cafarella, and H. V. Jagadish. 2017. Foofah: Transforming data by example. In *Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD'17)*. Association for Computing Machinery, New York, NY, USA, 683–698. https://doi.org/10.1145/3035918.3064034

[29] Jaemin Jo, Sehi L'Yi, Bongshin Lee, and Jinwook Seo. 2017. TouchPivot: Blending WIMP & Post-WIMP interfaces for data exploration on tablet devices. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. Association for Computing Machinery, New York, NY, USA, 2660–2671. https://doi.org/10.1145/3025453.3025752

[30] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*. Association for Computing Machinery, New York, NY, USA, 3363–3372. https://doi.org/10.1145/1978942.1979444

[31] Sam Lau, Sruti Srinivasa Srinivasa Ragavan, Ken Milne, Titus Barik, and Advait Sarkar. 2021. TweakIt: Supporting end-user programmers who transmogrify code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI'21)*. Association for Computing Machinery, New York, NY, USA, Article 311, 12 pages. https://doi.org/10.1145/3411764.3445265

[32] Bongshin Lee, Greg Smith, Nathalie Henry Riche, Amy Karlson, and Sheelagh Carpendale. 2015. SketchInsight: Natural data exploration on interactive whiteboards leveraging pen and touch interaction. In *IEEE Pacific Visualization Symposium (PacificVis)*. 199–206. https://doi.org/10.1109/PACIFICVIS.2015.7156378

[33] Bongshin Lee, Arjun Srinivasan, Petra Isenberg, and John Stasko. 2021. Post-WIMP interaction for information visualization. *Foundations and Trends in Human–Computer Interaction* 14, 1 (2021), 1–95. https://doi.org/10.1561/1100000081

[34] Guy Lüthi, Andreas Rene Fender, and Christian Holz. 2022. DeltaPen: A device with integrated high-precision translation and rotation sensing on passive surfaces. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST'22)*. Association for Computing Machinery, New York, NY, USA, Article 57, 12 pages. https://doi.org/10.1145/3526113.3545655

[35] Fabrice Matulic, Riku Arakawa, Brian Vogel, and Daniel Vogel. 2020. PenSight: Enhanced interaction with a pen-top camera. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI'20)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376147

[36] Fabrice Matulic and Moira C. Norrie. 2012. Supporting active reading on pen and touch-operated tabletops. In *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI'12)*. Association for Computing Machinery, New York, NY, USA, 612–619. https://doi.org/10.1145/2254556.2254669

[37] Fabrice Matulic and Moira C. Norrie. 2013. Pen and touch gestural environment for document editing on interactive tabletops. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces (ITS'13)*. Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/2512349.2512802

[38] Fabrice Matulic, Daniel Vogel, and Raimund Dachselt. 2017. Hand contact shape recognition for posture-based tabletop widgets and interaction. In *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces (ISS'17)*. Association for Computing Machinery, New York, NY, USA, 3–11. https://doi.org/10.1145/3132272.3134126

[39] MyScript. 2022. Cross-platform handwriting recognition and interactive ink APIs. https://developer.myscript.com. (2022). Last accessed: 2023-09-28.

[40] Jakob Nielsen. 1994. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'94)*. Association for Computing Machinery, New York, NY, USA, 152–158. https://doi.org/10.1145/191666.191729

[41] Gary Perelman, Marcos Serrano, Christophe Bortolaso, Celia Picard, Mustapha Derras, and Emmanuel Dubois. 2019. Combining tablets with smartphones for data analytics. In *Human-Computer Interaction – INTERACT 2019*, David Lamas, Fernando Loizides, Lennart Nacke, Helen Petrie, Marco Winckler, and Panayiotis Zaphiris (Eds.). Springer International Publishing, Cham, 439–460. https://doi.org/10.1007/978-3-030-29390-1_24

[42] Ken Pfeuffer, Jason Alexander, Ming Ki Chong, Yanxia Zhang, and Hans Gellersen. 2015. Gaze-shifting: Direct-indirect input with pen and touch modulated by gaze. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology (UIST'15)*. Association for Computing Machinery, New York, NY, USA, 373–383. https://doi.org/10.1145/2807442.2807460

[43] Ken Pfeuffer, Jason Alexander, and Hans Gellersen. 2016. Partially-indirect bimanual input with gaze, pen, and touch for pan, zoom, and ink interaction. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI'16)*. Association for Computing Machinery, New York, NY, USA, 2845–2856. https://doi.org/10.1145/2858036.2858201

[44] Ken Pfeuffer, Ken Hinckley, Michel Pahud, and Bill Buxton. 2017. Thumb + pen interaction on tablets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. Association for Computing Machinery, New York, NY, USA, 3254–3266. https://doi.org/10.1145/3025453.3025567

[45] Peter Pirolli and Ramana Rao. 1996. Table lens as a tool for making sense of data. In *Proceedings of the Workshop on Advanced Visual Interfaces (AVI'96)*. Association for Computing Machinery, New York, NY, USA, 67–80. https://doi.org/10.1145/948449.948460

[46] Thammathip Piumsomboon, Adrian Clark, Mark Billinghurst, and Andy Cockburn. 2013. User-defined gestures for augmented reality. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems (CHI EA'13)*. Association for Computing Machinery, New York, NY, USA, 955–960. https://doi.org/10.1145/2468356.2468527

[47] Ramana Rao and Stuart K. Card. 1994. The table lens: Merging graphical and symbolic representations in an interactive focus + context visualization for tabular information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'94)*. Association for Computing Machinery, New York, NY, USA, 318–322. https://doi.org/10.1145/191666.191776

[48] Yann Riche, Nathalie Henry Riche, Ken Hinckley, Sheri Panabaker, Sarah Fuelling, and Sarah Williams. 2017. As we may ink? Learning from everyday analog pen use to improve digital ink experiences. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. Association for Computing Machinery, New York, NY, USA, 3241–3253. https://doi.org/10.1145/3025453.3025716

[49] Hugo Romat, Caroline Appert, and Emmanuel Pietriga. 2021. Expressive authoring of node-link diagrams with graphies. *IEEE Transactions on Visualization and Computer Graphics (TVCG)* 27, 4 (2021), 2329–2340. https://doi.org/10.1109/TVCG.2019.2950932

[50] Hugo Romat, Christopher Collins, Nathalie Henry Riche, Michel Pahud, Christian Holz, Adam Riddle, Bill Buxton, and Ken Hinckley. 2020. Tilt-responsive techniques for digital drawing boards. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST'20)*. Association for Computing Machinery, New York, NY, USA, 500–515. https://doi.org/10.1145/3379337.3415861

[51] Hugo Romat, Nathalie Henry Riche, Ken Hinckley, Bongshin Lee, Caroline Appert, Emmanuel Pietriga, and Christopher Collins. 2019. ActiveInk: (Th)inking with data. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI'19)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300272

[52] Hugo Romat, Emmanuel Pietriga, Nathalie Henry-Riche, Ken Hinckley, and Caroline Appert. 2019. SpaceInk: Making space for in-context annotations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST'19)*. Association for Computing Machinery, New York, NY, USA, 871–882. https://doi.org/10.1145/3332165.3347934

[53] Vít Rusnák, Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. 2018. Designing coherent gesture sets for multi-scale navigation on tabletops. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI'18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3173574.3173716

[54] Ramik Sadana and John Stasko. 2016. Expanding selection for information visualization systems on tablet devices. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces (ISS'16)*. Association for Computing Machinery, New York, NY, USA, 149–158. https://doi.org/10.1145/2992154.2992157

[55] Advait Sarkar, Andrew D. Gordon, Simon Peyton Jones, and Neil Toronto. 2018. Calculation view: Multiple-representation editing in spreadsheets. In *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 85–93. https://doi.org/10.1109/VLHCC.2018.8506584

[56] Rishabh Singh and Sumit Gulwani. 2012. Learning semantic string transformations from examples. *Proc. VLDB Endow.* 5, 8 (Apr. 2012), 740–751. https://doi.org/10.14778/2212351.2212356

[57] Awalin Sopan, Manuel Freier, Meirav Taieb-Maimon, Catherine Plaisant, Jennifer Golbeck, and Ben Shneiderman. 2013. Exploring data distributions: Visual design and evaluation. *International Journal of Human–Computer Interaction* 29, 2 (2013), 77–95. https://doi.org/10.1080/10447318.2012.687676

[58] Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M. Drucker, and Ken Hinckley. 2020. InChorus: Designing consistent multimodal interactions for data visualization on tablet devices. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI'20)*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376782

[59] Yuta Takayama, Yuu Ichikawa, Buntarou Shizuki, Ikkaku Kawaguchi, and Shin Takahashi. 2021. A user-based mid-air hand gesture set for spreadsheets. In *Asian CHI Symposium 2021 (Asian CHI Symposium 2021)*. Association for Computing Machinery, New York, NY, USA, 122–128. https://doi.org/10.1145/3429360.3468193

[60] Poorna Talkad Sukumar, Anqing Liu, and Ronald Metoyer. 2018. Replicating user-defined gestures for text editing. In *Proceedings of the 2018 ACM International Conference on Interactive Surfaces and Spaces (ISS'18)*. Association for Computing Machinery, New York, NY, USA, 97–106. https://doi.org/10.1145/3279778.3279793

[61] Theophanis Tsandilas. 2018. Fallacies of agreement: A critical review of consensus assessment methods for gesture elicitation. *ACM Trans. Comput.-Hum. Interact.* 25, 3, Article 18 (Jun. 2018), 49 pages. https://doi.org/10.1145/3182168

[62] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. 2007. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics* 13, 6 (2007), 1129–1136. https://doi.org/10.1109/TVCG.2007.70589

[63] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. 2011. The aligned rank transform for nonparametric factorial analyses using only ANOVA procedures. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'11)*. Association for Computing Machinery, New York, NY, USA, 143–146. https://doi.org/10.1145/1978942.1978963

[64] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'09)*. ACM, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866

[65] Haijun Xia, Ken Hinckley, Michel Pahud, Xiao Tu, and Bill Buxton. 2017. WritLarge: Ink unleashed by unified scope, action, & zoom. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI'17)*. Association for Computing Machinery, New York, NY, USA, 3227–3240. https://doi.org/10.1145/3025453.3025664

[66] Ka-Ping Yee. 2004. Two-handed interaction on a tablet display. In *CHI'04 Extended Abstracts on Human Factors in Computing Systems (CHI EA'04)*. Association for Computing Machinery, New York, NY, USA, 1493–1496. https://doi.org/10.1145/985921.986098

[67] Dongwook Yoon, Nicholas Chen, François Guimbretière, and Abigail Sellen. 2014. RichReview: Blending ink, speech, and gesture to support collaborative document review. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST'14)*. Association for Computing Machinery, New York, NY, USA, 481–490. https://doi.org/10.1145/2642918.2647390

[68] Robert Zeleznik, Andrew Bragdon, Ferdi Adeputra, and Hsu-Sheng Ko. 2010. Hands-on math: A page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology (UIST'10)*. Association for Computing Machinery, New York, NY, USA, 17–26. https://doi.org/10.1145/1866029.1866035

[69] Emanuel Zgraggen, Robert Zeleznik, and Steven M. Drucker. 2014. PanoramicData: Data analysis through pen and touch. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2112–2121. https://doi.org/10.1109/TVCG.2014.2346293

[70] Emanuel Zgraggen, Robert Zeleznik, and Philipp Eichmann. 2016. Tableur: Handwritten spreadsheets. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA'16)*. Association for Computing Machinery, New York, NY, USA, 2362–2368. https://doi.org/10.1145/2851581.2892326

[71] Yang Zhang, Michel Pahud, Christian Holz, Haijun Xia, Gierad Laput, Michael McGuffin, Xiao Tu, Andrew Mittereder, Fei Su, William Buxton, and Ken Hinckley. 2019. Sensing posture-aware pen+touch interaction on tablets. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI'19)*. Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3290605.3300285

Received 3 March 2023; revised 18 August 2023; accepted 26 September 2023