Expanding Selection for Information Visualization Systems on Tablet Devices

Ramik Sadana
Georgia Institute of Technology
ramik@gatech.edu

John Stasko
Georgia Institute of Technology
stasko@cc.gatech.edu

ABSTRACT

Selection is a fundamental operation in interactive visualization applications. Although techniques such as clicking and lassoing items of interest are sufficient for basic selections, a more sophisticated interaction mechanism is required for expressing complex queries to modify or generalize existing selections. The ability to perform these advanced selections is critical for effective analysis within visualization systems. On touch-based devices such as tablets, however, expressing advanced selections is difficult due to the absence of a cursor and modifier keys. In this work, we address this limitation by presenting new interaction techniques that leverage a person's non-dominant hand. We use these techniques for advanced selection operations such as expanding, modifying, and replicating existing selections. Further, we introduce a method for performing generalized selection on tablet devices that provides a fluid mechanism to control the attributes and parameters of selection.

Author Keywords
Information visualization, selection, multi-touch, tablets

ACM Classification Keywords
H.5.m. Info Interfaces and Presentation (e.g. HCI): Misc.

INTRODUCTION

Selection is a fundamental operation in interactive visualization applications [36]. It can be used for finding details about an item or as an initial step for subsequent operations such as filtering, brushing, and deletion. While basic techniques for selection such as clicking/tapping and lassoing are robust and employed across domains, performing an advanced selection (e.g., adding to an existing selection) often requires expressing more complex and nuanced queries. This complexity is a direct consequence of the modes associated with the selection operation, including add-to-selection, remove-from-selection, intersect-with-selection, replace-selection, and toggle-selection [32].

In commercial operating systems, modifier keys (Cmd, Ctrl and Shift) are used to augment the basic selection by enabling two additional modes — add-to-selection and remove-from-selection. For example, a control click on a file icon in MS Windows adds that file to the existing selection set. Graphics applications such as Photoshop extend support for all five modes, with each using an explicit button for activation. Advanced selection behavior is also evident within desktop visualization systems. Applications such as Tableau and Spotfire support three selection modes – replace, add, and remove.

While advanced selection behavior is common across desktop-based visualization systems, the same is not true for visualization applications on touch devices. In fact, currently there are inconsistencies in the way basic selection itself is supported in applications. For example, Kinetica [26] supports a limited form of selection, employing it only for grouping tasks. Vizable [2] supports selection for details-on-demand on linecharts, but does not provide selection on barcharts. Sadana and Stasko employ selection more extensively in their system [27], using it for brushing, filtering, and details-on-demand. However, their system only supports the basic (replace) selection mode.

Since advanced selection is important for visualization applications, there is a clear need to explore and understand it in more detail. However, several challenges must be addressed to bring advanced selection to visualizations on touch-devices. First, it is not immediately clear which modes should be supported. Further, identifying touch gestures to use for these operations is also challenging. The predominant trend in touch-based visualization systems has been to adopt a standard vocabulary of gestures, i.e., a combination of tap, pan, pinch, and rotate [4, 26, 27]. Since these interactions are typically already employed for other critical operations, identifying alternate interactions becomes even more difficult.

In this design-focused research, we introduce a new approach for providing powerful selection capabilities in touch-based visualization systems. First, by leveraging a person's non-dominant hand, we extend the standard vocabulary of gestures available within these systems. Next, we use the new interactions to support advanced selection operations such as expanding, modifying, and duplicating existing selections. Finally, we present a novel interaction technique to perform a special type of advanced selection — generalized selection.

BACKGROUND

Advanced Selection on Multitouch Devices

On touch-devices with finger-based input, a person typically selects items by tapping on the intended targets or by lassoing around them. Usually, this is less precise than on desk-

ISS'16, November 06 - 09, 2016, Niagara Falls, ON, Canada
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-4248-3/16/11$15.00
DOI: http://dx.doi.org/10.1145/2992154.2992157

top computers because of people's inaccuracy in touching the screen with their finger(s) [19] and the occlusion that results from it [24]. These issues of imprecision have been well documented in the past, and researchers have presented several techniques to mitigate them. Previous work includes *take-off* [24] that placed the cursor at a fixed offset from the touch location, and *zoom-pointing* [5] and *sliding-widgets* [22] that both enhanced the touch areas to allow precise selection of the target.

Enhanced selection techniques have also been proposed for improving selection of an item from a dense cluster of items, a task that is central to visualizations. In LinearDragger [3], users begin by initiating a drag gesture at the position of the target object. This reveals a zoomed-in view of the area, with the object nearest to touch location selected. As the user drags the finger away, the system sequentially scrubs through all the neighboring objects. Lifting the finger commits the selection.

The other way of countering imprecision is by modifying erroneous selections. Several advanced selection techniques for modifying selections on touch-screens have been proposed. For example, Wu et al. [35] described a Pile-n-Browse gesture for selecting objects by scooping them between both hands. Wilson et al. [33] presented an approach to select multiple objects by placing multiple fingers on them. We considered these techniques for our work, but due to their focus on large surfaces and the use of bimanual interactions, their applicability to tablet-based visualizations is limited.

Benko et al. [6] highlighted the value in making selection simple, since the ability to directly touch an object to select it is a very appealing aspect of touch screens. This was further confirmed by the guessability study conducted by Wobbrock et al. [34] where participants preferred the use of tap and lasso gestures for both single and group selection tasks. However, so far the adoption of advanced selection on tablet devices has been limited. In the space of commercial applications for tablets, only those in the graphics domain, such as Photoshop and Paper, provide support for advanced selection modes such as add-to-selection and remove-from-selection.

Advanced Selection in Information Visualization

Selection is a cornerstone of interaction within visualization systems [32, 36]. The key role of selection is in assigning *reference*: indicating data item(s) of interest that can subsequently be operated upon. If there are multiple coordinated views, selected items can also be highlighted and brushed across views [10]. Alternatively, selecting an item can reveal specific details about the item in a details-on-demand view.

Advanced selection is also critical to visualizations and advanced operations are common across desktop visualization systems, such as Tableau and Spotfire. Typically, these systems extend the behavior that the operating system provides and support modes such as add-to-selection and remove-from-selection. These modes are often operated with modifier keys (Ctrl, Alt, Shift) or mode-switches. However, as ubiquitous as these selections are on desktop visualization systems, they are noticeably absent across touch-based visualization systems such as Kinetica [26], Vizable [2], and TouchWave [4].

North et al. [23] earlier explored selection operations on large touchscreen displays. Through a user study, they compared the use of one and two-handed postures for group selection tasks within a glyph-based visualization technique. More recently, Willett et al. [31] gathered user-elicited gestures for selection in interactive graphics on large vertical displays. Their results indicated a preference for one-finger and one-handed interactions over two-handed interactions. However, the tasks they employed were referential in nature ('Select all the peaks' or 'Select three lowest data points'), and differed from the advanced selection operations that are the focus of our work, such as modify or duplicate selection.

ADVANCED SELECTION FOR TABLET VISUALIZATIONS

In this work, we focus on tablets: portable, handheld devices with a touchscreen display ranging from 7 to 13-inches. Several research and commercial data visualization applications have been developed for such devices recently [2, 4, 26, 27]. Our goal was to design support for advanced selection operations within these applications. Our design process involved two main steps. First, we identified the selection modes that are applicable to tablet-based visualization systems. Second, we identified appropriate touch interactions for enabling and operating these modes.

Modes for Advanced Selection

Because selection modes are not mutually exclusive and a combination of a few modes can be sufficiently powerful and expressive [32], not all modes need to necessarily be supported in a system. Identifying relevant selection modes is crucial, however. One option is to replicate the functionality provided within comparable systems on desktops, such as Tableau and Spotfire. This would help address the typical use case for advanced selection in visualizations, wherein users want to perform a selection task, but are unable to express the query using the standard selection operations. However, on touch-devices, another unique set of use-cases also emerge as a consequence of the input being (imprecise) finger-based instead of cursor-based. Below, we describe these diverse use-cases for advanced selection on touch-based tablets. For each case, we identify the selection mode(s) that best represents the use-case.

View-driven Selections

V1. Compound filter - A filter query may be expressed over multiple attributes (e.g., for movies: $1998 < \text{Year} < 2008$ AND Genre = Comedy OR Drama). This filter operation can be executed by clicking on or lassoing around data items in the view that match the filter criteria, and activating "keep only" or "exclude" options. However, advanced selection techniques are needed here since target items may be placed at distant locations, on different axes, or in entirely different views. Relevant selection modes: *add-to-selection* and *intersect-with-selection*.

V2. Brushing & Linking - Selecting items in one view to track them in a different view is a key operation in visualization applications. Often this operation is one of creating a selection window and subsequently modifying it to observe the effect in other views. Modifications that are useful are resizing (or adding and removing items), translation, and duplication of selection windows. Relevant selection modes: *add-to-selection*, *remove-from-selection*, and *duplicate-selection*.

Challenges of Touch Input

T1. Imprecision of touch - Touch input is inherently imprecise as the finger is many times larger than a target pixel on the display [19]. This imprecision affects selections made with drag or lasso operations, such as range selections made by dragging a finger on an axis [27]. Imprecision causes the fingers to overshoot or undershoot the intended boundaries of selection. Thus, the user may want to resize the selection window. Relevant selection modes: *add-to-selection* and *remove-from-selection*.

T2. Occlusion by hand - The hand manipulating the display often covers the majority of the visualization. This issue is particularly notable on multi-view configurations where selections are dependent on observing data items across views (or brushing). Thus, frequent hand lift-offs are required, which subsequently imply features for readjusting the selection window. Relevant selection modes: *add-to-selection* and *remove-from-selection*.

The above use cases highlight the need for four selection modes—add-to-selection, remove-from-selection, duplicate-selection, and intersect-with-selection. In the following section, we design interaction techniques to operate these modes.

Interaction Techniques for Advanced Selection

The advanced selection interactions typically consist of the user enabling a mode and then performing a selection, or performing a selection first and then picking a mode. In both implementations, communicating the mode to the system is required. On touchscreens, this can be done in a number of ways. The three main approaches are *multitouch menus*, *touch overloading*, and *quasimodes*.

Multitouch Menus

As the number of gestures employed in a system increases, the cognitive overhead of remembering the gestures also increases. Using menus can help reduce this overhead by encouraging recognition over recall. A common example on touch-devices are the marking menus. These menus can be activated with a gesture and typically appear around the touch point. The options in the menu can be accessed either by tapping, using a stroke [21], or through a chorded action of the fingers [20].

The use of menus for advanced selection is common in desktop applications such as Photoshop and Sketch. Similar behaviors can easily be replicated in tablets – a menu can either be persistent or can appear automatically once selection is made. However, the drawback of the first approach is that persistent menus take up valuable space on the screen. The second case, on-demand marking-menus, may occlude sections of the view that contain information needed to select a mode. Further, introducing marking-menus within a system that does not already use them has the potential to impose unneeded complexity. If the behavior is unique, the discoverability of the menu, at least initially, is low. Conversely, once users gain experience with the menu, they are likely to expect menus for other operations as well.

Touch Overloading

A second approach involves *touch overloading*: differentiating between touch events, based on their properties, to enable distinct actions. Commonly used properties include location on the screen, input type (finger vs. stylus), duration, and direction of movement. More recently, other properties have also been shown to be effective for differentiating touches, such as velocity, shear [14], and pressure [17].

Any of these properties can be adapted to create interactions for advanced selection operations. For example, tapping the screen with pressure could activate the add-to-selection mode. However, the disadvantage of touch overloading is similar to that of marking-menus. Novel interactions that are considerably different from ones already used in the system require significant effort from the user to learn and remember. Further, since the touch properties themselves do not have any direct relationship with advanced selection (akin to, say, pressure and thickness), the learning effort is further amplified.

Quasimodes

Modes are states that a system enters wherein all user actions pertain to a specific category of tasks. While the use of modes is common in desktop computers, they are also popular on mobile devices. A common example is lists—tap and drag gestures in a list map to selection and scrolling, but by enabling the edit mode, the same gestures map to operations such as removing and reordering of items.

Modes can operate in one of two manners—*persistent* and *transient*. Persistent modes, also known as "latching" modes [13], stay in effect until cancelled or changed. For instance, the edit list operation is an example of a latching mode. Research has revealed that these modes tend to lead to more errors [28] since users often forget their state and struggle to identify how to cancel the mode if they realize they have made an error.

Transient modes, or "quasimodes" [25], stay in effect only as long as the user maintains the action required to activate the mode. These modes provide an easy and reliable way to return to the default application state and have been shown to significantly reduce mode errors compared to persistent modes [28]. On desktops, most examples of quasimodes make use of modifier keys, such as Shift, Command, Ctrl, and Alt. A similar idea can be adopted for tablets, but a replacement for those modifier keys must be found.

ADVANCED SELECTION INTERACTION DESIGN

Based upon the considerations discussed above, we used quasimodes to provide advanced selection on tablets. In the absence of keyboards with shift and control keys, a different modifier was needed to trigger the mode. To address this need, we leveraged the use of a person's non-dominant hand. The complementary roles of dominant and non-dominant hands have been studied extensively [12]. In HCI research, two-handed interactions have been employed for drawing and selection tasks, and for techniques such as ToolGlasses [7].

Most of these applications, though, assume stable configurations of use, where both hands are free to provide input to the system. On tablet devices, however, the non-dominant hand is often restricted to holding the tablet. As a result, the range of movement available to the non-dominant hand is severely constrained. Based on the way the tablet is held, people can only perform basic tap, hold, and drag gestures at the edges of the screen using either the thumb or the fingers.

Wagner et al. [29] initially explored this idea for handheld tablet devices. By observing how people typically held tablets, they introduced three categories of bimanual interactions — *bimanual taps*, *bimanual chords*, and *bimanual gestures*. Each of these assigned specific operations to the non-dominant hand, e.g., in the case of chords and gestures, finger(s) manipulated specific widgets on the screen, such as menus and lists. While these interactions are certainly more complex than we require, the bimanual tap configuration is more relevant for our use. The authors employ these gestures, which consist of tapping or holding on the edge of the screen, for actions such as activating a marking menu or displaying the *keypad* mode on the soft keyboard.

We build on these results, but simplify the interaction to match the role of a modifier. Modifiers only act as precursors to the primary action. Thus, we only require a single bit of information—whether the modifier was performed or not. Since a basic hold gesture provides this information, we employ it as a modifier.

The Clutch Modifier Technique

We define *Clutch*[1] as the action of the non-dominant hand performing a hold (or long-press) gesture on the screen. We utilize this Clutch action as a modifier in the system. The behavior of our implementation differs from modifiers on desktops. There, a modifier typically does not align to a unique category of tasks. For example, the tasks performed with a Shift key cannot be classified together into a single category. We employ Clutch to introduce interactions that all map to the same category of tasks—advanced selection.

To introduce Clutch on tablets, we had to define the locations on the screen where the action would be permissible. This choice of locations depends on the way people hold tablets. Wagner et al. [29] conducted an initial study to observe people's grips when holding a tablet and found five configurations: ThumbBottom, ThumbCorner, ThumbSide, FingersTop, and FingersSide. In their observation of 8 participants, they identified FingersSide as the most preferred option.

While their results are applicable to our work, they are notably different from our informal observations of people using tablets under both regular and controlled conditions today. The predominant posture we observe is a combination of ThumbSide and ThumbCorner. These observations are also consistent with the patterns that are found in the results from a Google Image Search for terms *tablet*, *grip*, *ipad*, and *holding*. The difference in observations can partly be explained by the significant drop in the weight of tablets since the original study was conducted in 2012. For instance, the weight of an Apple iPad, the dominant tablet in the market, has declined by over 30% since 2012. The weight of a device has a strong bearing on how people hold it, as acknowledged by the authors of the original study.

Ultimately, our design choice was to relax the location requirement and support Clutch on all four edges of the screen. We provide a 100 pixels wide zone that runs along the edges. Since Clutch is a single discrete action, the effort required to operate it is small. Enabling it on all edges further reduces this effort. Moreover, it better supports the division of labor model since Clutch is not restricted to any single hand. Depending on where the primary action needs to be performed, either hand can be used to activate Clutch. When a Clutch is detected, a bright blue halo image appears beneath the clutching finger to provide visual feedback to the user.

To demonstrate the utility of Clutch gestures for advanced selection, we add them to a system providing multi-coordinated visualizations on tablets [27]. The system supports several methods for selecting glyphs in charts such as scatterplots and bar graphs. Individual glyphs can be selected by tapping on them, while multiple glyphs can be selected by lassoing around them. Swiping on the axes selects glyphs between a range of values. Finally, via brushing, glyphs selected in one chart are also highlighted in the other chart.

Mapping Clutch to Selection Modes

Using Clutch as a modifier, we are able to augment and reuse all the primary gestures, such as tap, drag, and pinch. In mapping these modified gestures to advanced selection tasks, we sought a correspondence between the roles of the original gesture A and the modified gesture Clutch + A. We achieved this in the following manner: gestures that create a new selection in the original state do the same in the modified state, but with a different selection mode enabled. Similarly, gestures that modify an existing selection in the original state modify a copy of the selection in the new state. Enabling the Clutch behavior that is consistent with the behavior of modifiers on desktops should help promote learnability of the gestures. In the following section, we describe the role of each modified gesture in detail. The accompanying video illustrates the dynamics of each operation as well.

Clutch + Tap

The Clutch + tap gesture is performed by activating Clutch and tapping with a finger anywhere on the screen (Figure 1). This gesture augments the standard tap behavior that selects the glyph placed beneath the finger. Whereas the non-clutched tap deselects previously selected glyphs, Clutch + tap simply adds the new glyph to the existing selection. If the tapped glyph is already selected, Clutch + tap deselects the glyph and removes it from existing selection.

Clutch + Drag

Similar to Clutch + tap, Clutch + drag is performed by activating a Clutch and dragging a finger on the screen. We use

[1]We use the term "Clutch" much like a car clutch, different from that as used in a thread of existing touch interaction research [9].

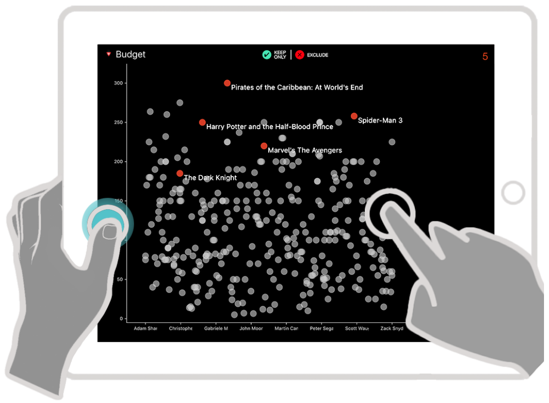
Budget
KEEP ONLY
EXCLUDE
5
Pirates of the Caribbean: At World's End
Harry Potter and the Half-Blood Prince
Spider-Man 3
Marvel's The Avengers
The Dark Knight

Figure 1: Clutch + tap gesture to select multiple glyphs.

this gesture to enable different selection modes depending on the type of selection currently active in the system.

1. Single glyph selected – A single glyph is selected by either tapping on it or lassoing around it. Since identifying a glyph in a dense region is fairly imprecise [19], we use Clutch + drag gesture to assist in selecting a target glyph.

The technique works in the following manner. When a Clutch + drag is initiated with one glyph selected, the system projects the drag movement further in that direction to identify the glyph placed closest to the selected glyph. Once identified, the system holds for the user to move the finger by the same amount as the distance between the two glyphs. For target glyphs placed further away, we use a control-display gain to expand the magnitude of movement. When the movement exceeds the distance between the glyphs, the selection snaps to the other glyph, deselecting the original glyph (Figure 2). This also reveals a label depicting the primary key of the selected glyph for additional feedback. The overall effect of the operation resembles that of using a trackpad where finger's movement maps to the movement of a cursor. Similar to BubbleCursor [11], the selection is snapped to a glyph at all times, i.e. there is no transitional state in the middle when no glyph is selected.

Our design is also similar to the LinearDragger technique for selecting an item from a dense cluster of items [3], with three key differences. First, the user does not need to place their finger near the dense area of points and can perform the action from a distance. Second, instead of mapping the selection to one-dimensional movement, we use both dimensions of the movement to allow more precise access to the neighboring glyphs. Finally, we provide details for each glyph, helping users differentiate them from one another.

2. Multiple glyphs selected – Dragging on the view draws a lasso path and selects all points that lie within the path. Using a Clutch augments this behavior by enabling the add-to-selection mode. Thus, new glyphs are selected without deselecting previously selected glyphs. This is similar to Clutch + tap. However, Clutch + lasso does not support remove-from-selection. If the lasso region contains glyphs that are already selected, their selection state is *not* toggled.

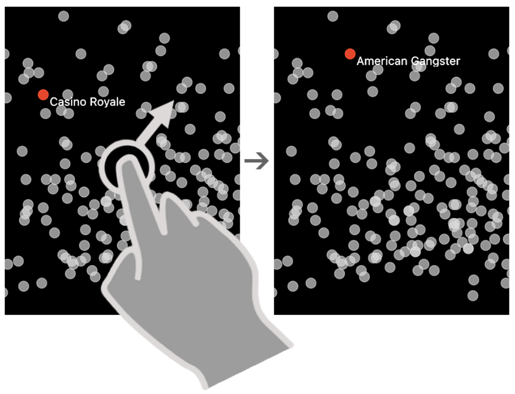

Figure 2: When a single glyph is selected, Clutch + drag translates the selection to a neighboring glyph that is located in the direction of the drag.

3. Rectangular selection – If a view contains an active selection rectangle on any axis, the Clutch + drag gesture responds based on the position where the drag is initiated.

A. Clutch + drag on the axis: Clutch + drag in this case creates a new rectangular selection without dismissing existing selections (Figure 3). However, the selection mode employed depends on whether the existing selections and the new selection are on the same axis or on different axes. In case of the former, the add-to-selection mode is activated and all glyphs that lie within any of the selection rectangles are selected. For the latter, we considered both union and intersection as potential options. We debated providing both as a toggle option, but ultimately concluded that the need for union of selections on independent axes was fairly small. Thus, we solely use an intersection operation on selections belonging to different axes. We use color variation to differentiate the selected region from the unselected region (Figure 3).

B. Clutch + drag inside selection: Dragging inside a selection without Clutch moves the selection along the axis. Dragging with Clutch results in a similar effect, but instead of moving the original rectangle, it moves a copy of the selection, by activating the duplicate-selection mode (Figure 4). The glyphs that lie within either of the rectangles are selected, i.e. we use union of the rectangles for selection.

Clutch + Pinch

The Clutch + pinch gesture is performed by activating a Clutch and executing a two-finger pinch gesture on the view. We utilize this gesture for modifying existing selections by increasing and decreasing the size of the selection area.

1. Clutch + pinch for multiple selected glyphs – If the view contains multiple selected glyphs, the Clutch + pinch gesture can be used for scaling the size of this selection (Figure 5). When the gesture is initiated, the system creates a convex hull that encloses all the selected glyphs. The pinch gesture then controls the size of the hull. We utilize a flexible aspect ratio pinch on the region, i.e. the horizontal and vertical scaling of the hull individually depend on the movement of the fingers in the x and y direction respectively.

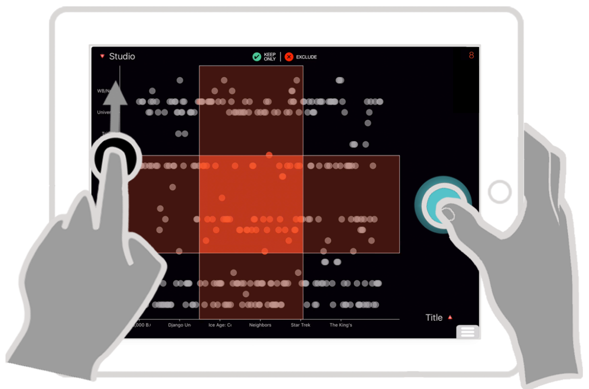
Figure 3: Clutch + drag on the axis creates a new rectangular selection that intersects with existing selections. Notice the color difference in selected and non-selected portions. Here, the right hand is performing the Clutch action.

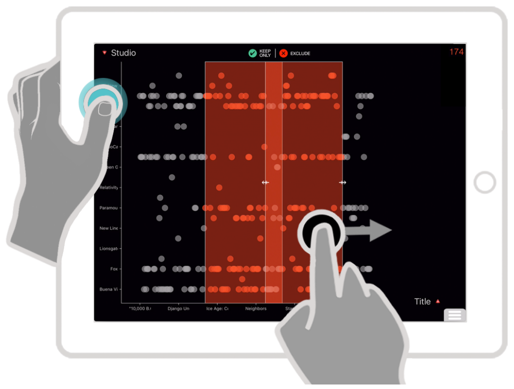
Figure 4: Clutch + drag inside a rectangular selection creates a duplicate.

2. *Clutch + pinch for selection rectangles* – For rectangular selections, the movement of each finger in the pinch gesture is mapped to the ends of the active rectangle. As the fingers move, the size of the active rectangle changes accordingly.

Clutch + Drag + Pinch
The Clutch + drag + pinch gesture is a special configuration of the Clutch + drag gesture, where the user begins by dragging a finger and, without lifting that finger, performs a pinch gesture using another finger. We currently utilize this gesture to further augment the Clutch + drag operation on single glyph selections. There, we utilized the gesture to scrub through the neighboring glyphs. In configurations when the glyphs are located in a densely packed region, executing the pinch-out gesture reveals a lens at that location with the region zoomed in (Figure 6). The user can now lift her second finger and continue dragging with the first finger. This cycles through the glyphs in the scaled up view in the lens. If the target glyph is situated outside the region in the lens, the lens appropriately shifts to the new location. Once the task is complete, the user can close the lens by either pinching-in or lifting all the fingers.

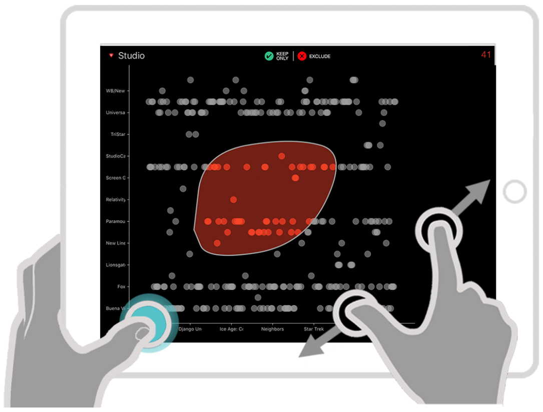
Figure 5: Clutch + pinch for multiple glyphs grows or shrinks the selection area.

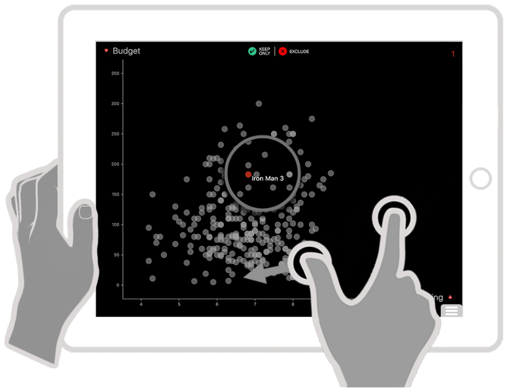
Figure 6: Clutch + drag + pinch for single selections reveals a lens with zoomed in region.

Low-Tension Clutch
The Clutch-modified gestures presented above can be described as *phrases*. Buxton [8] defines phrases as chunks of subtasks "glued together" into a routine that can be performed in a single continuous action. The physical tension that "chunks" or ties together the actions of a phrase is called phrase tension [18]. For the above gestures, the Clutch action behaves as the phrase tension.

Hinckley et al. [18] define three categories of phrase tension—full-, half-, and low-tension. These differ based on the duration for which the user keeps the physical tension active. In the current implementation of the Clutch-modified gestures, keeping the Clutch active for the entire duration is unnecessary since the Clutch info is only utilized at the start of the gestures. We thus configured the Clutch as a low-tension modifier—for all Clutch-modified gestures, users can lift the Clutch immediately after they begin the gesture with their primary hand (as demonstrated in Figure 6).

A low-tension Clutch helps in reducing discomfort that may occur in manipulating the screen with both hands simultaneously, while also supporting the tablet with one hand. It also further differentiates the behavior of the Clutch modifier from a desktop modifier. On desktops, when selecting text using Shift and arrow keys, the user cannot release the Shift key until the selection is complete, even though the arrow key is continuously pressed.

GENERALIZED SELECTION INTERACTION DESIGN

Conceptual Foundation

Selections can broadly be classified into two categories: view-driven and data-centric. View-driven selections are the “standard” selections where targets are identified based on the visual properties of the glyphs, such as position or size. Conversely, in data-centric selections, targets are identified based on a value or range of values of an attribute in the data. For example, data items with a specific value for a categorical attribute can be selected by mapping the attribute to a visual property such as color or shape, and using the legend.

Generalized selection is an advanced selection operation that combines these view-driven and data-centric selections. Here, the user identifies a target using a visual property, e.g. position or color, and subsequently intends to select other items that are similar to this item based on certain attributes of the data. For example, in a scatterplot of movies with *rating* and *profit* on x and y-axis, the user highlights the movie that is furthest towards the top right and wants to view other movies by the same director. This selection differs from the data-centric selection since the user may not be interested in knowing who the director is, but is only interested in the ratings and profit of the other movies made by him/her.

Heer et al. [16] described generalized selection as a “select objects like this” query, and presented a method to perform it in scatterplots on desktops. In their system, users invoke a context menu on a selected glyph and pick an attribute from a list. This selects all other glyphs in the view with the same value for the attribute as the originally selected glyph. If the attribute contains hierarchical information, selection can be cycled through different levels of the hierarchy by repeatedly clicking on the originally selected glyph. For example, if the attribute is a date, repeated clicks expand the selection to include items from the same day, same week, same month, and then the same year.

Generalizing a selection in the absence of a specialized method is fairly cumbersome, however. For instance, to perform the selection in the above example without a context menu, one needs the following steps: identify the director of the movie, switch the axis attribute to directors, lasso-select all movies by the specific director, and finally switch back to the original axis attributes to view the spread of the selected movies. An alternate method would be to add a second view, such as a barchart with directors on the x-axis, and use brushing between the views. Selecting the movie in the scatterplot would highlight the bar representing the director in the barchart. Subsequently, selecting this bar would select all connected movies in the scatterplot.

All these alternatives involve multiple steps, each of which alters users’ context by changing the views. Since such context switches are undesirable, we created a specialized method to perform generalized selection in a tablet visualization system. We designed a novel interaction technique that provides fluid access to attributes of data. Our implementation expands the operations originally presented in [16] by applying it to additional attribute types and adding methods for users to control the parameters of the selection.

Selection Variations — The outcome of generalizing a selection varies based on the size of the existing selection and the attribute used for generalizing it.

1. Single object selections: For a single selected glyph, outcome varies with the type of attribute as follows:
   (a) Categorical: For categorical attributes, the selection expands to include all data items that have the same value for the attribute as the selected glyph. (e.g., genre = ‘Action’)
   (b) Hierarchy: For attributes with hierarchical properties, selection expands to include the data items that match the value of the selected glyph at the lowest level of the hierarchy. For example, for the date attribute, we begin by matching the exact date of the data items (i.e., date = ‘07/13/2016’).
   (c) Quantitative: For quantitative attributes, the selection expands include to data items that fall within a neighborhood around the value of the selected glyph. (e.g., profit $\in$ [90, 110]).
2. Multiple object selections: Generalizing selections for multiple selected glyphs behaves similar to the single selected glyph, but expands the matching criteria to include values of the attribute from all the selected glyphs. For categorical attributes and hierarchical attributes, the union of all the values is used for selection, while for quantitative values, a range going from the minimum and maximum value of the attribute is used.

Interaction Design

The key component for the generalized selection operation is a list of attributes from which the user picks one. The necessity of displaying this list differentiates the operation from the other advanced selection operations we presented earlier.

Our interaction design consists of a widget that appears in the periphery of the selected data glyph(s). The widget is hidden by default, and can be introduced on demand. To circumvent the need for a persistent UI element to initialize the operation, we utilize a unique gesture — a *two-finger tap*. If one or more glyphs are selected, performing a two-finger tap gesture anywhere on the view presents the widget.

The widget consists of a single label that, initially, gives a description of the operation (Figure 7a). In the background, the widget contains a vertically scrollable list of options that remain hidden from the user. The widget operates in a manner emulating trackpad input with the entire screen behaving as a touchpad. The user can place her finger anywhere on the screen and drag vertically up or down to scroll through the options (Figure 7b). As an option becomes visible, the glyphs in the chart update to reflect the selection that is generalized with respect to the attribute the option represents.

To enable users to explore all available options, the widget maintains two separate selections. The primary selection maps to the originally selected glyphs and remains static throughout the operation. The secondary selection represents glyphs selected through generalization, and updates each time the user scrolls to a different option. By separating the selections, the user is able to explore the options over multiple drag gestures and finger lifts, and is not required to complete the entire selection in a single continuous stroke. To commit a selection, the user can tap outside the list. This merges the secondary selection into the primary selection. Alternatively, to cancel the operation and return to the original state, the user can either scroll the list to the top or perform a two-finger tap gesture to dismiss the widget.

The second step in generalizing a selection is controlling the parameters of the chosen attribute. As we discussed earlier, the parameters are dependent on the type of the attribute. Our design displays the options for modifying the parameters in a separate control that appears below the original widget (Figure 7c). This additional control only appears when the user halts on a particular option for more than 500 ms (i.e., scrolls to an option and lifts her finger).

For attributes with hierarchical properties, the secondary control presents the level of hierarchy being used for expanding the selection. Initially, this is the lowest level for that attribute (e.g. date). The other levels appear as columns of a horizontally scrollable list view contained within the control (Figure 7c). Users can switch to any other level by swiping left or right. Similar to vertical swiping in the attribute list, this horizontal swiping can be performed anywhere on the view.

Finally, for quantitative attributes, the secondary control contains a slider widget that depicts the range of values used for generalizing the selection (Figure 7d). When the initial selection is a single glyph, we use a 10 percent threshold around the value of the attribute. For multiple glyphs, the range extends from the minimum to the maximum value of the attribute for the selected glyphs. The slider widget extends from the minimum to the maximum across the entire dataset, with the selected range highlighted using handles. Since manipulating the small handles directly is difficult, their position can be controlled with a two-finger pinch operation (Figure 7d). The gesture can be performed anywhere on the screen. Dragging a finger causes the corresponding handle to move by an equivalent amount. Finally, we also embellish the widget with the distribution of the values for the attribute using the scented-widgets technique [30].

DISCUSSION

The Clutch technique we introduce in this paper extends the selection capabilities and features of touch-based visualization systems. However, the technique can also be applied to operations other than selection. In this section, we discuss a few extensions of the technique and reflect on the drawbacks and limitations of the current design.

Potential Extensions

We envision the Clutch gestures being applied for other operations such as layout-modification. For instance, in a system

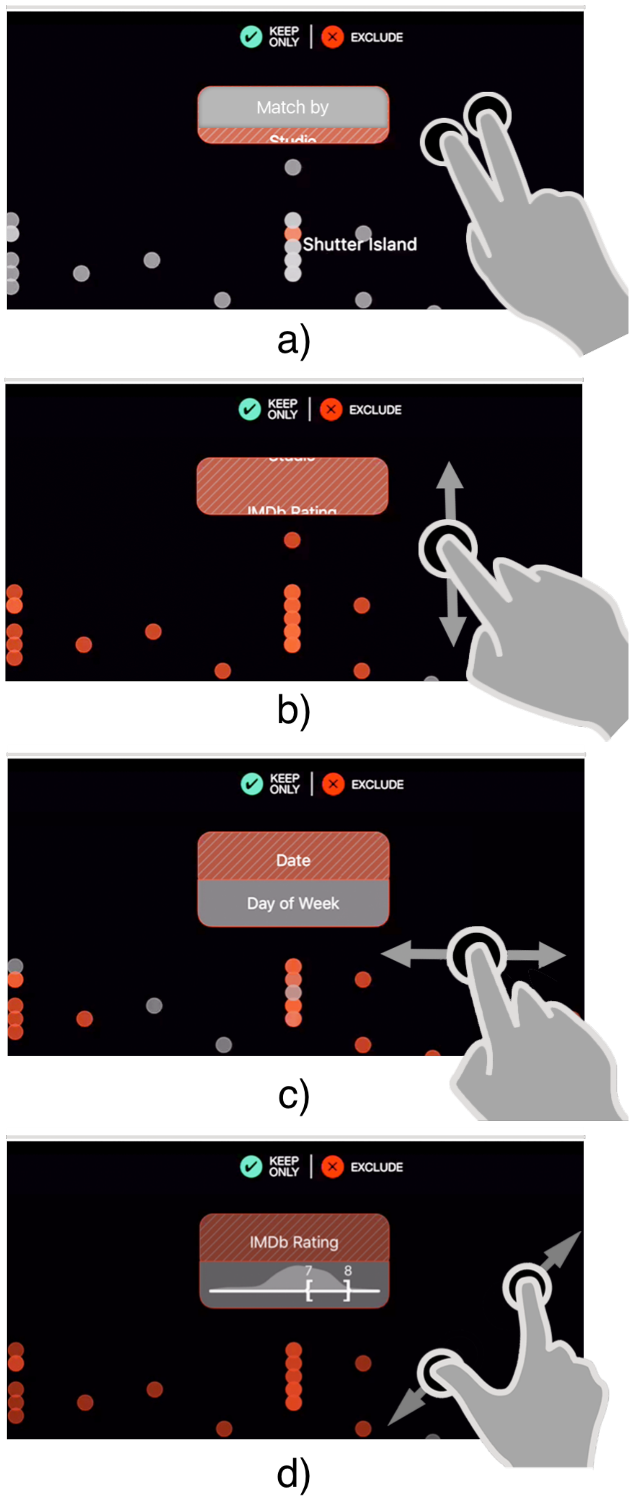

Figure 7: Generalized selection. a) User taps on the screen with two fingers to activate the generalized selection menu. b) To scroll through the list of attributes, user drags a finger vertically. c) For hierarchical attributes, user drags horizontally to access other levels of hierarchy. d) For quantitative attributes, the scented widget displays the selected and overall range of selection. Using a two finger pinch, user can modify the extents of the selection.

with multiple stacked views (e.g., [27]), Clutch + drag could be used for reordering or removing views from the screen. Clutch + pinch could be used for ad hoc scaling of certain views, in case the user wants more space for precise actions, or if views contain only a few glyphs. Finally, one could utilize Clutch + tap or double-tap to disconnect a view from other views. This would permit localized filtering of data for situations when users wish to compare views filtered differently. Alternatively, Clutch + tap also could be used for bookmarking the state of a view that users can revert to later.

While the Clutch gesture has applicability to a broad set of operations, the Clutch action itself is only one of many possible modifier techniques. With a modifier, one ultimately wants to supply a single unit of information to the system before performing a gesture. A non-preferred hand modifier has the benefit that it conforms with the division of labor model—the role of the non-dominant hand is that of framing the detailed action that the dominant hand performs. It also prevents the need for the preferred hand to perform a complex gesture. However, other properties of touch can be leveraged that permit the modifier to also be supplied by the preferred hand without adding difficulty. One potential option is to use pressure. Recently, 3D Touch [1] was introduced on Apple iPhones that adds pressure sensitivity to the screen. A modifier on this screen could be force-pressing the screen once before performing the gesture normally. Alternatively, a touch could be treated as a modifier if it is performed using the knuckle or nail of the finger [15]. In fact, many touch overloading techniques we presented earlier are suitable for use as modifier actions.

Drawbacks and Limitations

Although the Clutch modifier technique holds promise for touch-based visualization systems, it exhibits a number of potential limitations as well. First, the Clutch action is potentially disruptive and likely requires users to switch their grip. This can be particularly cumbersome if the user is mobile rather than stationary. Another limitation is the ergonomics of performing Clutch. It may be difficult to maintain the Clutch action for a continuous period of time. Although the low-tension mode we discussed earlier addresses this limitation to an extent, repeated Clutch actions over a short time period can still strain the hands and affect performance over time.

Perhaps the biggest limitation of the technique is discoverability and affordance. Discoverability is an issue both for the Clutch action itself and the mapping to the advanced selection operations. Since the Clutch action is not self revealing, users are less likely to discover or "guess" it on their own without guidance. Providing an affordance for the action is, thus, critical, although it is difficult to ascertain what the affordance should be. Further, if users discover the clutch action on their own, they still need some guidance for mapping the action to the advanced selection operations. The activation region of the Clutch action overlaps with the location of a few other operations, such as drag on axes and swipe-in from edges. This likely increases the probability of accidental Clutch activations. Nonetheless we believe that self-revelation affordances need to be integrated with the Clutch technique in the future.

Finally, user studies of the Clutch techniques compared to other approaches such as marking menus or alternative touch gestures are clearly needed to better understand each's strengths and weaknesses. The design-oriented research in this article has focused on introducing the idea of using Clutch gestures for advanced selections operations. Earlier, we presented the reasons that make us believe that the quasi-mode approach can succeed, but only user testing and actual trial usage will discover how people perceive and react to the different methods.

CONCLUSION

In this paper, we explore novel interactions for performing advanced selections on tablet-based visualization systems. We present the *Clutch* modifier technique and employ it for advanced selection tasks. The benefit of this approach is that it extends the vocabulary by augmenting existing gestures in place of introducing new and complex interactions. We also describe the design of interactions to provide generalized selection, and we extend the technique to include operations for parameterized control of the selections.

Much further work remains. For example, the generalized selection technique can be extended beyond selections made within charts, and into other components, such as the cells, rows or columns of the data table, or the filter badges that represent the active "keep only' or "exclude" filters. Opportunities also exist for providing compound generalized selections, such as generalization on multiple attributes (e.g., select all movies by same director OR same genre AND same release year). Similarly, the interaction techniques we introduce could be used for providing different types of operations, such as layout-modifications.

The potential to extend the interactions and feature-space of touch-based visualization tools is immense. The use of a Clutch operation augments everyday gestures that are familiar to people, and thus extends a command set (e.g., advanced selection) while helping to keep the interactions approachable and accessible. We hope that such design contributions can help spur further advances in touch-based visualizations, with the aim of achieving parity with desktops in the efficiency of data analysis.

ACKNOWLEDGEMENTS

This work was supported by a Google Faculty Research Award and by the National Science Foundation via award IIS-1320537.

REFERENCES

1. 3d Touch - iOS - Apple Developer. https://apple.com/ios/3d-touch/.

2. Vizable by Tableau. https://vizable.tableau.com/.

3. Au, O. K.-C., Su, X., and Lau, R. W. LinearDragger: A Linear Selector for One-finger Target Acquisition. In *Proc. of ACM CHI '14* (2014), 2607–2616.

4. Baur, D., Lee, B., and Carpendale, S. TouchWave: Kinetic Multi-touch Manipulation for Hierarchical Stacked Graphs. In *Proc. of ITS '12* (2012), 255–264.

5. Bederson, B. B., and Hollan, J. D. Pad++: A Zoomable Graphical Interface System. In *ACM CHI Conference Companion* (1995), 23–24.

6. Benko, H., Wilson, A. D., and Baudisch, P. Precise Selection Techniques for Multi-touch Screens. In *Proc. of ACM CHI '06* (2006), 1263–1272.

7. Bier, E. A., Stone, M. C., Pier, K., Buxton, W., and DeRose, T. D. Toolglass and Magic Lenses: The See-through Interface. In *Proc. of ACM SIGGRAPH '93* (1993), 73–80.

8. Buxton, W. A. S. Chunking and phrasing and the design of human-computer dialogues. In *Human-computer Interaction*, R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg, Eds. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1995, 494–499.

9. Casiez, G., Vogel, D., Pan, Q., and Chaillou, C. Rubberedge: Reducing Clutching by Combining Position and Rate Control with Elastic Feedback. In *Proc. of ACM UIST '07* (2007), 129–138.

10. Chen, H. Compound brushing [dynamic data visualization]. In *Information Visualization, 2003. INFOVIS 2003. IEEE Symposium on*, IEEE (2003), 181–188.

11. Grossman, T., and Balakrishnan, R. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. In *Proc. of ACM CHI '05* (2005), 281–290.

12. Guiard, Y. Asymmetric division of labor in human skilled bimanual action. *Journal of Motor Behavior 19*, 4 (1987), 486–517.

13. Hackett, M., and Cox, P. T. Touchscreen Interfaces for Visual Languages. Tech. Rep. CS-2011-05, Dalhousie University, July 2011.

14. Harrison, C., and Hudson, S. Using Shear As a Supplemental Two-dimensional Input Channel for Rich Touchscreen Interaction. In *Proc. of ACM CHI '12* (2012), 3149–3152.

15. Harrison, C., Schwarz, J., and Hudson, S. E. TapSense: Enhancing Finger Interaction on Touch Surfaces. In *Proc. of ACM UIST '11* (2011), 627–636.

16. Heer, J., Agrawala, M., and Willett, W. Generalized Selection via Interactive Query Relaxation. In *Proc. of ACM CHI '08* (2008), 959–968.

17. Heo, S., and Lee, G. Force Gestures: Augmenting Touch Screen Gestures with Normal and Tangential Forces. In *Proc. of ACM UIST '11* (2011), 621–626.

18. Hinckley, K., Baudisch, P., Ramos, G., and Guimbretiere, F. Design and Analysis of Delimiters for Selection-action Pen Gesture Phrases in Scriboli. In *Proc. of ACM CHI '05* (2005), 451–460.

19. Holz, C., and Baudisch, P. Understanding Touch. In *Proc. of ACM CHI '11* (2011), 2501–2510.

20. Lepinski, G. J., Grossman, T., and Fitzmaurice, G. The Design and Evaluation of Multitouch Marking Menus. In *Proc. of ACM CHI '10* (2010), 2233–2242.

21. Luo, Y., and Vogel, D. Pin-and-Cross: A Unimanual Multitouch Technique Combining Static Touches with Crossing Selection. In *Proc. of ACM UIST '15* (2015), 323–332.

22. Moscovich, T. Contact Area Interaction with Sliding Widgets. In *Proc. of ACM UIST '09* (2009), 13–22.

23. North, C., Dwyer, T., Lee, B., Fisher, D., Isenberg, P., Robertson, G., and Inkpen, K. Understanding multi-touch manipulation for surface computing. In *Proc. of INTERACT '09*, Springer-Verlag (2009), 236–249.

24. Potter, R. L., Weldon, L. J., and Shneiderman, B. Improving the Accuracy of Touch Screens: An Experimental Evaluation of Three Strategies. In *Proc. of ACM CHI '88* (1988), 27–32. 00328.

25. Raskin, J. *The Humane Interface: New Directions for Designing Interactive Systems*. ACM Press/Addison-Wesley Publishing Co., 2000.

26. Rzeszotarski, J. M., and Kittur, A. Kinetica: Naturalistic Multi-touch Data Visualization. In *Proc. of ACM CHI '14* (2014), 897–906.

27. Sadana, R., and Stasko, J. Designing Multiple Coordinated Visualizations for Tablets. In *Computer Graphics Forum*, vol. 35 (2016), 261–270.

28. Sellen, A. J., Kurtenbach, G. P., and Buxton, W. A. S. The Prevention of Mode Errors Through Sensory Feedback. *Hum.-Comput. Interact. 7*, 2 (June 1992), 141–164.

29. Wagner, J., Huot, S., and Mackay, W. BiTouch and BiPad: Designing Bimanual Interaction for Hand-held Tablets. In *Proc. of ACM CHI '12* (2012), 2317–2326.

30. Willett, W., Heer, J., and Agrawala, M. Scented Widgets: Improving Navigation Cues with Embedded Visualizations. *IEEE TVCG 13*, 6 (Nov. 2007), 1129–1136.

31. Willett, W., Lan, Q., and Isenberg, P. Eliciting Multi-touch Selection Gestures for Interactive Data Graphics. In *EuroVis Short-Paper Proceedings* (2014).

32. Wills, G. J. Selection: 524,288 ways to say "this is interesting". In *Proc. of IEEE InfoVis '96* (1996), 54–60.

33. Wilson, A. D., Izadi, S., Hilliges, O., Garcia-Mendoza, A., and Kirk, D. Bringing Physics to the Surface. In *Proc. of UIST '08*, UIST '08 (2008), 67–76.

34. Wobbrock, J. O., Morris, M. R., and Wilson, A. D. User-defined Gestures for Surface Computing. In *Proc of ACM SIGCHI '09* (2009), 1083–1092.

35. Wu, M., Shen, C., Ryall, K., Forlines, C., and Balakrishnan, R. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. In *Proc. of IEEE TABLETOP '06* (2006), 185–192.

36. Yi, J. S., Kang, Y., and Stasko, J. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE TVCG 13*, 6 (Nov. 2007), 1224–1231.