Visualization of off-screen data on tablets using context-providing bar graphs and scatter plots

Peter S. Games[a] and Alark Joshi, Ph.D.[a]

[a]Boise State University, 1910 University Drive, Boise, USA;

ABSTRACT

Visualizing data on tablets is challenging due to the relatively small screen size and limited user interaction capabilities. Standard data visualization apps provide support for pinch-and-zoom and scrolling operations, but do not provide context for data that is off-screen. When exploring data on tablets, the user must be able to focus on a region of interest and quickly find interesting patterns in the data. We present visualization techniques that facilitate seamless interaction with the region of interest on a tablet using context-providing bar graphs and scatter plots. Through aggregation, fisheye-style, and overview+detail representations, we provide context to the users as they explore a region of interest. We evaluated the efficacy of our techniques with the standard, interactive bar graph and scatter plot applications on a tablet, and found that one of our bargraph visualizations - Fisheye-style Focus+Context visualization (BG2) resulted in the fewest errors, least frustration and took the least amount of time. Similarly, one of our scatter plot visualizations - User Driven Overview+Detail (SP3) - resulted in the fewest errors, least frustration and took the least amount of time. Overall, users preferred the context-providing techniques over traditional bar graphs and scatter plots, that include pinch-and-zoom and fling-based scrolling capabilities.

Keywords: Visualization, mobile, focus + context, touch screen

1. INTRODUCTION

Worldwide sales of tablets totaled 119 million in 2012 and are projected to reach 370 million by 2017 .[1] Displaying and interacting with data on these tablets can be challenging due to the constrained screen size and limited user interface capabilities. Mobile visualization systems must present data in a way that allows users to quickly recognize and easily interpret visualizations while also addressing these unique constraints of mobile devices.[2] This includes smart techniques to present an appropriate amount of data that will promote effective user interaction[3] and exploration of data.

In this paper, we present techniques for representing off-screen data on a tablet using context-providing bar graphs and scatter plots. The techniques allow users to focus on the data in a region of interest, but also provide context regarding the data distribution and overall trends in the data. This context is provided through aggregation and fish-eye style representations. Through the conducted user evaluation, we found that some of our techniques were better than the traditional bar graphs and scatter plots, that include pinch-and-zoom and scrolling capabilities.

We believe that these techniques provide a better understanding of the applicability and usability of aggregation, fisheye, overview first-details on demand type techniques in the context of a mobile device. The techniques presented can be applied to visualize univariate and bivariate data in a wide variety of domains such as financial management,[4] quantified self tasks such as visualizing the number of steps taken over time,[5] and other domain specific applications[6] that allow data exploration on a tablet-sized device.

To evaluate our techniques, we developed three bar graph visualization apps and three scatter plot visualization apps, all of which provide context in varying ways. We also developed a standard control bar graph

Further author information: (Send correspondence to Alark Joshi)
Alark Joshi: E-mail: alark1@gmail.com
Peter Games: E-mail: petergames@u.boisestate.edu

visualization app and a standard control scatter plot visualization app against which we compared our context-providing visualizations. Based on the user evaluation, we found that among the bar graph visualizations, the Fisheye-Style Focus + Context Visualization (BG2) resulted in the highest user accuracy, as well as required the least time and caused least frustration. On evaluating the scatter plot techniques, we found that the User-Driven Overview + Detail (SP3) resulted in the highest user accuracy (tied with the standard scatter plot), but required the least time, and caused the least frustration.

2. RELATED WORK

Creating effective visualizations for mobile devices requires additional planning and preparation, as one cannot simply port visualizations intended for desktop computers to mobile devices.[2] A variety of mobile device characteristics must be considered, including limited display size, reduced computational power, different user input methods, slower connection speeds, increased likelihood of user distractions,[7] and varying work environment conditions. There are a variety of published approaches and strategies that relate to presenting and interacting with visualizations on small, *phone-sized* devices. Karstens et al.[8] discuss the importance of avoiding excessive scrolling and panning on small devices. Instead, they present an approach to visualize hierarchies using *Rectangular View*, which consists of a radial layout where the levels are arranged as squares, in order to more efficiently utilize the limited screen space. Zhang et al.[9] explore personalization as an approach to preventing information overload with mobile information systems. By customizing the services provided to a user, only the desirable information is presented, resulting in greater efficiency. The authors also utilize a zoom-in/zoom-out approach for line charts and a hierarchical clustering technique for scatterplots.

2.1 Addressing Mobile Device Constraints

Chittaro[10] presents an approach to showing multiple visualizations on a small screen in an overview fashion either by displaying the visualizations in different areas of the same screen or by allowing the user to rearrange the visualizations to possibly overlap. Additionally, an approach is presented to explore the time series by "zooming" in or out temporally. Schmeiß et al.[11] present an approach to support time series exploration by the use of a time slider widget. With this functionality, mobile visualizations display events and points of interest at a user-specified time. These techniques allows a user to focus on a region of interest (zoom/slider), but end up sacrificing context in the process.

By virtue of the fact that mobile device users are mobile, there is a greater likelihood of user interruption when using a mobile device. To compensate for the user's decreased attention, *user interaction tasks should be minimized and simplified.* A prototype for a context aware mobile application requiring minimal user interaction is presented by Hertzog and Torrens.[13] This prototype allows a user to manage schedule details and travel plans based upon current contextual information.

2.2 Representation of Off-Screen Objects

With limited screen size, there is a greater need to effectively represent *off-screen objects* in mobile visualizations. There have been several proposed approaches to conveying the presence of off-screen objects to the user. Burigat et al.[14] present three different approaches for displaying off-screen objects: a halo/arc technique and two arrow-based approaches. The halo approach surrounds off-screen objects with circles that reach the display area. Both arrow approaches utilize arrows which point at off-screen objects, and represent the distance to an off-screen object by arrow size or length. The authors stress the importance of ensuring that users can easily interpret the visual encoding of off-screen objects.

Gustafson et al.[15] present *Wedge*, an approach to representing off-screen objects with acute isosceles triangles. Two corners of the triangle are displayed on the screen and the third corner coincides with the off-screen target. Burigat and Chittaro[16] present the results of a user evaluation which compares three different approaches for representing off-screen content. Two approaches involve using proxies, scaled arrows and *Wedge*, and the third approach utilizes an overview + detail technique. The authors state that none of these approaches are applicable for very large numbers of off-screen objects. Regarding the overview + detail approach, the authors discuss the drawbacks of small overviews on mobile device screens, including difficulty relating the overview and detail view. In this study, the overview + detail technique in the test *application did not allow the user to increase the size of* *the overview display.* Burigat et al.[17] compare two techniques for representing off-screen content, an overview + detail technique and a contextual cues technique. In this experiment, the number of off-screen objects is larger than the number in similar, previously conducted experiments.[16] The results show that users are faster and more accurate with the overview + detail technique, most likely because of the increased mental workload required by the wedge technique.

Drucker et al.[18] present a comparison of two interfaces on a touch tablet: a traditional desktop interface (including Windows, Icons, Menus, and Pointer-style Interfaces) and an interface that affords more direct manipulation of the data visualization itself (referred to as a FLUID interface). Based on the conducted user study, they found that users were faster and more accurate using the FLUID interface, and users also preferred the FLUID interface. The authors conclude that when using a touch device, there are benefits to converting a conventional desktop interface to a gesture-based interface.

3. APPROACH

Our approach to effective visualization and interaction of data on a tablet device was inspired by aggregation[19] and fisheye techniques[20] popular in the information visualization literature. Given the small screen size of these tablet devices, we want to explore the applicability of these techniques for effective data representation and exploration in the mobile environment.

To this effect, we developed three bar graph visualization apps and three scatter plot visualization apps, all of which provide context in varying ways. For evaluation purposes, we developed a traditional bar graph visualization app and a scatter plot visualization app that allow pinch-and-zoom and fling-based scrolling. One of the primary goals for the six presented visualizations is to effectively *represent all of the data* points, either in focus or context. The user can interactively explore the data through built in interaction paradigms to easily find the “sweet spot” for visualizing the specific data set. In subsequent sections, we introduce the three bar graph visualization techniques first followed by the three scatter plot techniques.

4. VISUALIZATION OF UNIVARIATE DATA USING BAR GRAPHS

To facilitate explanation and maintain consistency in describing the univariate visualization techniques using bar graph, we will assume that the bar graph visualization is *oriented horizontally*, such that the individual bar values are displayed vertically. Thus, when we discuss the bar height, we are referring to the dimension of the bar which represents the data value and when we discuss the bar width, we are referring to the dimension of the bar which is not intended to convey meaningful information to the observer.

Bar graphs are an effective technique for visualizing univariate data values. However, with large data sets, allowing a user to visualize some of the data values in detail while visualizing the remaining data contextually requires thoughtful design. If all data values are displayed similarly, increasing the detail, or width of a bar, would lead to an increase in the overall width of the visualization. As the visualization width increases, displaying context, or an overview of the data, on a constrained display becomes difficult. Thus, viewing some of the data values in detail while maintaining context presents challenges.

4.1 Focus + Context-Based Visualization Using Aggregation (BG1)

The first of the three bar graphs involves aggregating all of the contextual data values into a small area. This is done by representing the value (vertical height) of multiple data items within a common aggregation area. This approach is shown in Figure 1. The aggregation area, represented in blue, contains a representation of multiple context data values (off-screen data).

If we employ this approach with a data set of about 200 values, the resulting visualization will look like the sample visualizations shown in Figure 2 (which have differing focus bar widths). This approach provides focus in the desired area, represented by bars of user-defined width, while simultaneously representing the remaining data values contextually. The aggregation area on the left represents context data values to the left of the focus area and the aggregation area on the right represents context data values to the right of the focus area. This approach to displaying contextual data provides the user with a visual representation of the distribution of the off-screen data values.

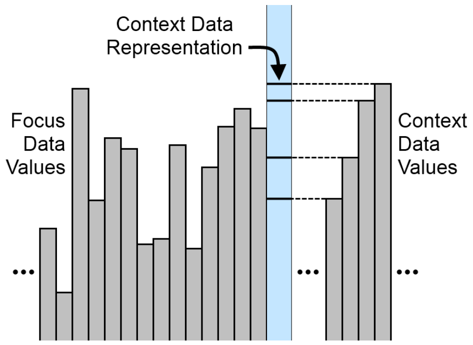
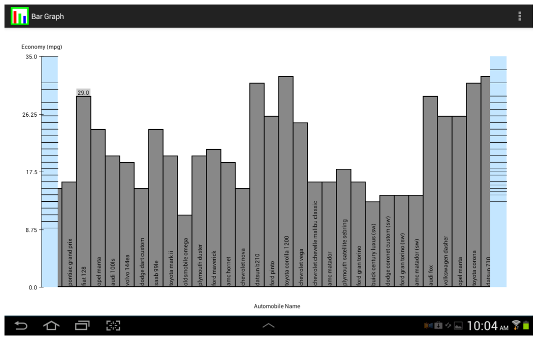
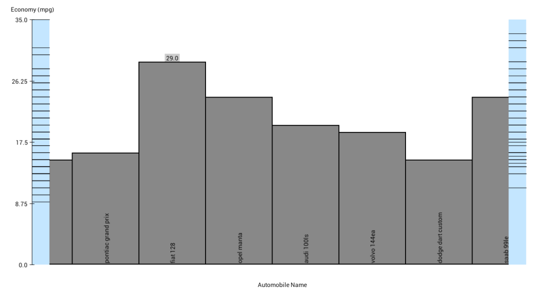


Figure 1. The aggregation area, represented in blue, contains a representation of multiple context data values.



(a) Default focus bar width.



(b) The focus bar width has been increased by a spread gesture, moving some data items from the focus region to the context region.

Figure 2. A visualization providing focus in the desired area while simultaneously representing the remaining data contextually, in the two blue regions. Even though Figure 2(b) and subsequent figures are screenshots of the application running on a tablet, the top & bottom margins have been cropped to focus on the visualization techniques.

4.1.1 Interactive Functionality

Supported interactive functionality includes scrolling through the data values by sliding (or flinging) a finger horizontally in either direction. This functionality moves some of the detail data values to the context aggregation area and some of the context data values to the detailed display. Therefore, both detail and overview data values change as the user scrolls through the data set. The user may also perform spread or pinch gestures to increase or decrease the level of detail (width of bars) in the detailed display. Figure 2(a) shows a visualization with the default focus bar width and Figure 2(b) shows the resulting visualization after the user has increased the focus bar width via a spread gesture. To display a particular data value, the user may tap on a bar within the focus area and the corresponding data value is displayed on top of the bar, as can be seen in both Figure 2(a) and Figure 2(b).

4.1.2 Implementation

In implementing this visualization, we began by considering a hypothetical visualization consisting of all data values displayed in focus, which we will refer to as the full focus visualization. In most circumstances, it is not possible to display the full focus visualization, as the full focus visualization width would exceed the available tablet display area. It is important to draw the distinction between the full focus visualization and the displayed focus visualization, which is the portion of the full focus visualization which is actually displayed on the tablet at any given moment. We also keep track of the current width of the focus bars, which the user can change via the pinch and spread gestures.

As the user navigates, we keep track of the pixel offset into the full focus visualization of the displayed focus visualization. In other words, if the visualization offset is presently 1000, we would draw the portion of the full focus visualization which starts at horizontal pixel location 1000 (0 based) and extends (left to right) to the other end of the visible display area. Thus, provided we properly draw the visualization based upon the current value of the offset and the focus bar width, we need only increment or decrement the offset as the user scrolls and flings through the visualization.

In order to draw the visualization, we begin by determining the index of the first visible bar in the focus region, i_f, by simply performing integer division: $i_f = \frac{o}{w_{fb}}$, where o is the pixel offset into the full focus visualization, and w_{fb} is the width of a focus bar. Next, we determine the pixel offset not of the visualization, but of *the first visible bar.* It is determined by a simple modulus operation: $o_b = o \ mod \ w_{fb}$, where o_b is the pixel offset of the first displayed bar, o is the pixel offset into the full focus visualization, and w_{fb} is the width of a focus bar. Thus the bar offset will be a value between 0 and $w_{fb}-1$. Next, we loop through the data values beginning with the index of the first visible bar in the focus region (i_f) and draw an appropriate rectangle for each value. The four coordinates for a bar are determined as follows: $l_i = left + ((i - i_f)w_{fb}) - o_b, r_i = l + w_{fb}, t_i = top + h\left(1 - \frac{v_i}{max}\right)$, and $b_i = top + h$ where l_i is the left border of the current bar, $left$ is the left margin width, i is the index of the current bar, i_f is the index of the first visible bar, w_{fb} is the width of a focus bar, o_b is the pixel offset of the first displayed bar, r_i is the right border of the current bar, t_i is the top border of the current bar, top is the top margin height, h is the available display height (minus top and bottom margins), v_i is the data value for the current bar, max is the maximum data value, and b_i is the bottom border of the current bar.

We continue looping while r_i is less than the sum of the left margin and the available display width (minus left and right margins). Therefore, as soon as the visualization extends beyond the available display width, we discontinue drawing focus bars. In this way, we only draw the bars which are visible at the present moment, avoiding unnecessary operations (drawing bars located outside of the display area). In order to improve the aesthetic appearance, we set a clipping rectangle bounded on the left by the left margin width and bounded on the right by the sum of the left margin and the available display width (minus left and right margins). This makes the disappearance of focus bars from and the introduction of focus bars to the focus region smooth and crisp. Two additional loops are utilized to draw all of the line segments in the two aggregation regions.

The fling functionality was implemented to facilitate smooth, fluid navigation through a data set. When the user first initiates a fling action, the fling duration (and subsequently the end time of the fling action) is calculated. The total fling displacement, which is proportional to the fling gesture velocity, is also calculated at this time. We then recursively call a method which, based upon the current time, determines the percentage of the fling duration that has elapsed. This time percentage is then interpolated using a decelerate interpolator, resulting in the current percentage of the total fling displacement. This fling process continues to its completion time, unless it is prematurely aborted by a user tapping on the screen at any moment during the fling duration.

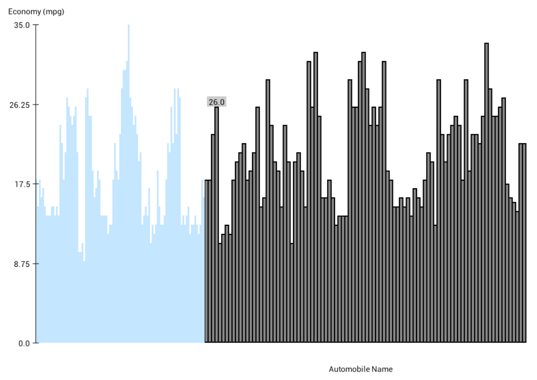
One disadvantage of this first bar graph visualization approach is the fact that although a user can visualize the data distribution of the contextual data values, the user is unable to visualize the ordering of the contextual data values. It may be the case that the user wishes to see how the data values change along the x-direction. For example, with time series data, the user may wish to see how the data values change over time rather than simply see all of the data values in a non-temporal context.

4.2 Fisheye-Style Focus + Context Visualization (BG2)

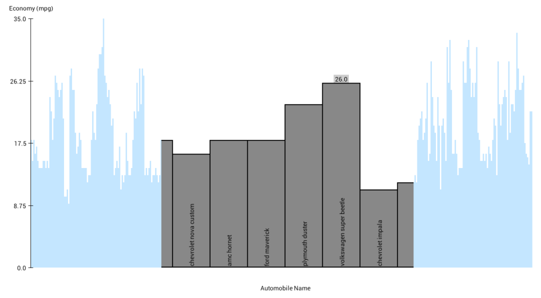
The second bar graph approach allows visualization of the ordering of the contextual data values. In this approach, focused data values are represented at a greater width than contextual data values. This approach is shown in Figure 3, where the contextual data is again represented in blue. With this approach, we see a sharp demarcation between the focused and contextual data, evidenced by the change in width and color of the bars.

4.2.1 Interactive Functionality

Supported interactive functionality for this second approach is similar to the interactive functionality for the previous visualization (BG1). Additionally, the relative horizontal location of the detailed view within the entire visualization will change as the user scrolls through the data set. To display a particular data value, the user may tap on a bar and the corresponding data value is displayed on top of the bar. For example, the user has selected the same data item in Figures 3(a) and 3(b) and its corresponding data value (26.0) is displayed.



(a) Focus bars are too narrow to display identifier text within each bar.



(b) The focus bar width has been increased, moving some data items from focus to context.

Figure 3. Focused data values (gray) are represented at a greater width than contextual data values (blue). The user has tapped the same data item in the two visualizations and the data value is displayed on top of the corresponding bar.

4.2.2 Implementation

In implementing this visualization, we decided to devote a constant percentage (we chose 50%) of the display width to the focus region. Thus, as the user increases the level of detail (width of bars in focus region), some data points (or portions of data points) will move from the focus region to the context region. Conversely, as the user decreases the level of detail, some data points will be "pulled" from the context region to the focus region. One goal for this visualization was to ensure that as a user interacts, a bar in the bar graph which spans the transition between focus and context regions is faithfully represented in *both* focus and context regions. We also stressed that the transition from one region to another should be smooth and even. In other words, as a bar moves from one region to another (due to user interaction), at each instant it maintains an appropriate percentage in each region. For example, when a bar is 75% in the context region and 25% in the focus region, its visual representation must be comprised of two parts: one part being 75% of the context bar width (and colored with the context bar color) and the other part being 25% of the focus bar width (and colored with the focus bar color).

Although the previous visualization (BG1) represented off screen bars as line segments within the aggregation area, this visualization differs in that all bars in the bar graph must be visible at all times. In order to ensure that all bars fit in the display area, some bars will need to be sufficiently narrow. Thus, we recognize that there are upper limits on the focus and context bar widths. The user may resize the focus and context bar widths *independently*, but neither width can be so great as to cause the visual display to exceed the available horizontal display area. Both widths also have a lower bound of one pixel. The results of increasing focus bar width by a spread gesture can be seen in Figure 3.

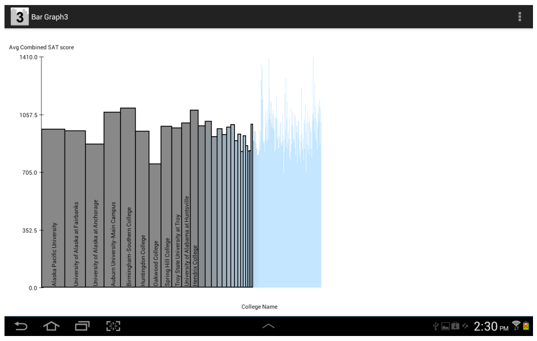
Allowing a user to resize both focus and context bar width independently via a pinch/spread gesture provides tremendous flexibility, but in this situation it also presents challenges. When a user performs a pinch/spread gesture, we determine which set of bars (focus or context) to modify based upon the gesture's focal point. The focal point of a pinch/spread gesture is the point midway between the two fingers forming the gesture. As the bars in the visualization are resized, the visualization is redrawn in real time in accordance with the new bar width. However, when the visualization is redrawn, the focal point of the ongoing gesture may now correspond to a different region of the visualization (focus or context). Therefore, a long, continuous pinch or spread gesture may begin by resizing focus bars, only to result in a visualization redraw such that the focal point of the gesture is now in the context region, and then subsequently resize context bars. To reduce the frequency with which this problem occurs, focus bars are resized when the focal point of the gesture is within the focus region and *all context bars* (in both left and right context regions) are resized when the focal point of the gesture is *within the left context region only*.

To display the visualization, we found it helpful to conceptually divide the display area into three regions: the focus region and the two context regions (to the left and right of the focus region). Before drawing each of the three regions, we set a clipping rectangle in order to make the transitions between focus and context smooth. The clipping rectangle for the left context region is bounded on the right by c_1, where c_1 is calculated as $c_1 = left + (i_f)(w_{cb}) + (l_c)(w_{cb})$, where $left$ is the left margin width, i_f is the index of the first focus bar, w_{cb} is the width of a context bar, and (l_c) is the fraction of the left transition bar located in the context region. Once this clipping region is established, we draw all of the left context region bars. The clipping rectangle for the focus region is bounded on the left by c_1 and on the right by $c_2 = c_1 + w_{fr}$, where c_1 is the left bound of the focus region, and w_{fr} is the width of the focus region, which cannot be greater than a constant percentage (50% in our case) of the display width, although it can be less than 50% of the display width. Once this clipping region is established, we draw all of the focus region bars. The clipping rectangle for the right context region is bounded on the left by the c_2 (the right side of the clipping rectangle for the focus region). Once this clipping region is established, we draw all of the remaining context bars. It is important to note that the transition bars need to be drawn in *both context and focus regions* (which have the appropriate clipping region set) so that the bar appears to be in a state of transition between the two regions.

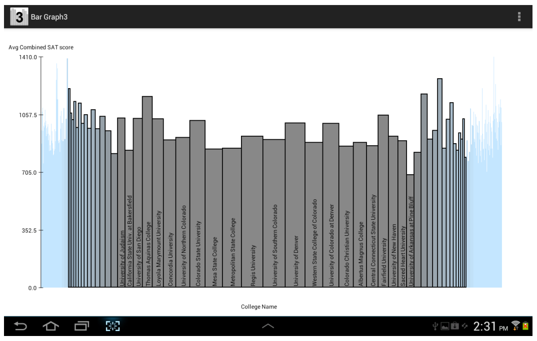
Therefore, the total width of the displayed visualization is equal to the sum of (1) the context bar width multiplied by the number of full bars in the left context region, (2) the portion of the left transition bar that is located in the left context region, (3) the portion of the left transition bar that is located in the focus region, (4) the focus bar width multiplied by the number of full bars in the focus region, (5) the portion of the right transition bar that is located in the focus region, (6) the portion of the right transition bar that is located in the right context region, and (7) the context bar width multiplied by the number of full bars in the right context region. It is necessary to ensure that this sum never exceeds the available display area so that all data points are represented in the visual display at all times.

4.3 Weighted Transitions for Fisheye-Style Focus + Context Visualization (BG3)

To avoid the sharp demarcation between the focused and contextual data, we implemented a third approach whereby the bar width changes gradually. As before, the bars will be thicker in the desired area of focus and thinner in the contextual region. However, with this third approach, the width of the bars will not change abruptly at one data value, but rather will change gradually over many data values. This approach is shown in Figure 4. It is worth noting that, although fisheye approaches can result in distortion, in this case, *data values are not being distorted*, as the data values are represented by bar height. The bar width in a bar graph (with horizontal orientation) is not intended to convey meaningful information to the observer.



(a) The bar of maximum width has been moved to the extreme left of the visualization.



(b) The bar of maximum width has been moved to the center of the visualization.

Figure 4. Focused data values are represented at a greater width than contextual data values, but bar width changes gradually.

4.3.1 Interactive Functionality

Supported interactive functionality for this third approach is similar to that of the first (BG1) and second (BG2) visualization. This functionality will move some of the detail data values to the context region and some of the context data values to the detailed region. Also, the relative horizontal location of the detailed view within the entire visualization will change as the user scrolls through the data set.

4.3.2 Implementation

In implementing this visualization, we maintain a maximum bar width for the one bar which is displayed at maximum width and a minimum bar width for potentially many of the contextual data value bars. The minimum bar width is one pixel, as it is our goal to faithfully represent *all* data values. The maximum bar width can be modified by the user via the pinch/spread gesture. To draw the bar graph, we loop through each of the data points and draw a corresponding bar, the width of which is determined based upon the distance from the bar of maximum width. Additionally, the color of the corresponding bar is also interpolated between the color of the bar of maximum width and color of the context bars.

As with the second bar graph visualization, all bars in the bar graph must be visible at all times. However, unlike the second univariate visualization, the user cannot resize focus and contextual bars independently. The user simply resizes the bar of maximum width, and all other bar widths are calculated based upon that value. As before, we need to ensure that all bars fit in the available display area width. Thus, we recognize that there are upper limits on the maximum bar width. The user may resize the maximum bar width, but it can never be so great as to cause the visual display to exceed the horizontal display area.

It is important to note that the overall width of the entire visualization is dependent upon the location of the bar of maximum width. If the bar of maximum width has been moved to the extreme left, as in Figure 4(a), the overall width of the entire visualization will be at a minimum. If the bar of maximum width has been moved to the center, as in Figure 4(b), the overall width of the entire visualization will be at a maximum. In the former case, there will only be one bar which is a distance of 1 from the bar of maximum width. Similarly, there will only be one bar which is a distance of 2 from the bar of maximum width. The fact that there are fewer of these "thicker" bars and more of the "thinner" bars, results in a narrower overall visualization width. As the user modifies the maximum bar width, we calculate the maximum value for the maximum bar width. However, in order to do this, we need to perform the calculation *assuming that the bar of maximum width is positioned in the center of the visualization* (even though it may not be). This allows us to cap the maximum bar width value in such a way that as the user interacts with the visualization, the entire visualization width will never exceed the display area, regardless of the location of the bar of maximum width.

As we have done with previous visualizations, we keep track of a view offset, which represents the precise pixel offset into the full focus visualization of the horizontal location of maximum width. We then calculate a *floating point index* identifying the bar (integer portion) and specifically the horizontal location within the bar (decimal portion) of maximum width: $f_{max} = \frac{o}{w_{max}}$, where f_{max} is the floating point bar index corresponding to maximum width, o is the pixel offset into the full focus visualization, and w_{max} is the maximum bar width. We then calculate the distance between a bar with index i and the point of maximum width as: $d_i = |i - f_{max}|$, where d_i is the floating point distance between the bar with index i and the point of maximum width. Lastly, we calculate the width, w_i, of a bar with index i as: $w_i = min(1.0, (0.9^{d_i})(w_{max}))$. To simplify matters, no clipping is required for this approach. All bars are drawn appropriately in one pass through the data set.

5. VISUALIZATION OF BIVARIATE DATA USING SCATTER PLOTS

Scatter plots typically display values for two variables, but may sometimes depict more than two variables by varying the size, color, or shape of the plotted points. Although the proposed approaches to follow could be applicable for more than two variables, we will focus our effort on two variable scatter plots.

Allowing a user to visualize some of the data values in detail while visualizing the remaining data contextually is a challenge associated with scatter plot visualizations. If all data values are displayed similarly, increasing the detail would lead to an increase in the visualization width and height. As the visualization grows in both dimensions, displaying an overview of the data on a tablet display becomes difficult. Thus, viewing some of the data values in detail while maintaining context is challenging.

5.1 Aggregation-Based Visualization Using a Circular Layout (SP1)

The first approach to displaying bivariate data using scatter plots involves displaying proxies. In this approach, the region of interest (ROI) is surrounded by a ring, which represents the contextual data. We decided upon a circular ring because the represented data values could be located anywhere in a 2D area. If we draw a line from a contextual data point to the center of the ring and clip that line such that it only appears within the ring, we have a mechanism to represent the contextual data point. This approach is shown in Figure 5, whereby a contextual data point is represented as a line segment within the ring. Further, the lines could also indicate relative distance between the data point and the focus area. We vary the saturation of the line based on the distance of the point from the region of interest (closest - black lines, farthest - white lines).

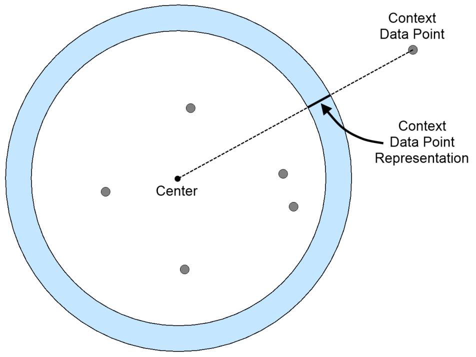
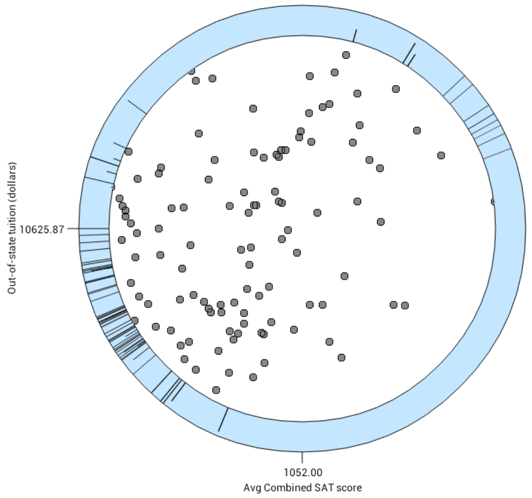
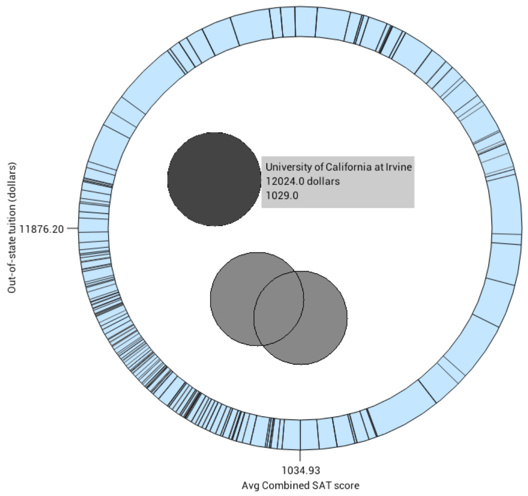


Figure 5. A context data point is represented by the intersection of the ring and a line drawn from the data point to the center of the ring.

If we employ this approach with a data set of about 200 values, the resulting visualization will look like the two sample visualizations (with differing levels of detail) shown in Figure 6. This approach provides focus in the desired circular area while simultaneously representing the remaining data contextually. This approach to displaying contextual data provides the user with a visual representation of the distribution of the context data values. One disadvantage to this approach is the fact that the ROI is circular and mobile device displays are not circular. This results in unused screen area.



(a) Default display.


Out-of-state tuition (dollars)
11876.20
University of California at Irvine
12024.0 dollars
1029.0
1034.93
Avg Combined SAT score

(b) The user has zoomed in by performing a spread gesture and selected a data item by performing a tap gesture.

Figure 6. A visualization providing focus in the desired circular area while simultaneously representing the remaining data contextually, in the blue ring.

5.1.1 Interactive Functionality

Supported interactive functionality includes scrolling (and flinging) through the data values by sliding a finger in any direction. Both detail and overview data values will change as the user scrolls through the data set. The user may also perform spread or pinch gestures to zoom into the ROI. To display the identifier and the x and y data values for a data point, the user may tap on the corresponding point within the ROI. We also interactively update the x- and y-axis scales corresponding to the current zoom-level of the region of interest.

5.1.2 Implementation

In implementing this visualization, we begin by creating a clipping region to display the contextual data values. This clipping region is set to be the difference between the outer radius circle region and the inner radius circle region. Once this clipping region is set, only visual elements within the contextual ring will be displayed. We then loop through the data values, calculating x and y coordinates and a distance from the display center. Based upon this distance, we calculate an appropriate context proxy color. Lastly, we draw the line for each of the contextual data values. We then replace the clipping region with the inner circle clipping region to allow display of focus data values. Once this focus clipping region is set, only visual elements within the focus circle will be displayed. We then loop through the data values, calculating x and y coordinates and drawing a circle centered on those coordinates.

In order to accommodate user scrolling and zooming functionality, we maintain a multiplier value, an x-offset value, and a y-offset value. The multiplier is initialized to 1 and the offsets are initialized to 0. As the user interacts via pinch/spread gestures, the multiplier is modified accordingly. In order to avoid distortion, all zooming changes scale equally in both x and y directions. That is, the x and y scales are “locked” in sync. As the user interacts via scroll and fling gestures, the x and y offset values are modified accordingly. Thus, the formula to calculate the x coordinate of a bivariate data value is: $x = \frac{(v_x - min_x)}{(max_x - min_x)}(w)(m) + left - o_x$ where x is the x coordinate, v_x is the x-dimension data value for the current bar, min_x is the minimum x-dimension data value, max_x is the maximum x-dimension data value, w is the available display width (minus left and right margins), m is the multiplier, $left$ is the left margin width, and o_x is the x-dimension offset. The y-coordinate calculation differs only slightly, because the y-coordinate value of 0 represents the top of the display and we use the common convention of y-coordinate values increasing upward and decreasing downward. Thus, the y coordinate is calculated as: $y = \left(1.0 - \frac{(v_y - min_y)}{(max_y - min_y)}\right)(h)(m) + top - o_y$ where y is the y coordinate, v_y is the y-dimension data value for the current bar, min_y is the minimum y-dimension data value, max_y is the maximum y-dimension data value, h is the available display height (minus top and bottom margins), m is the multiplier, top is the top margin height, and o_y is the y-dimension offset.

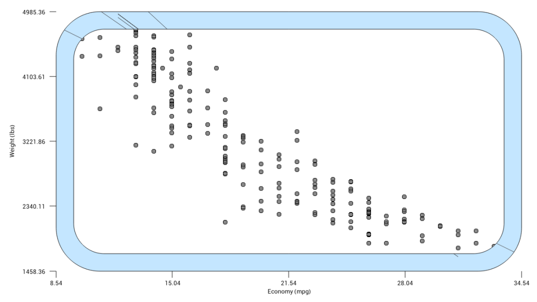
We also allow a user to tap upon a particular data value to see specific information, such as the identifier and the x and y dimension data values. When this happens, the selected data item is “highlighted” with a darker color and a small box containing the appropriate text appears next to the item (See Figure 6(b)). However, it is important to note that there may be multiple partial or complete overlapping data values. For this reason, we included functionality which allows a user to “cycle through” all data items which contain the tap coordinates by repeating the tap gesture in the same location.

5.2 Aggregation-Based Visualization Using a Rounded Rectangular Layout (SP2)

Our next approach involves modifying the previous technique by representing the “ring” as a rectangle with rounded corners. We choose to round the corners in order to facilitate the viewer’s visual extrapolation of any line segments located in the corners of the rectangle. This approach is shown in Figure 7. By utilizing a square with rounded corners, the **ROI is larger** than with the circular layout, and as a result we are capable of displaying more visual information within the focus region.

5.3 User-Driven Overview + Detail (SP3)

Our third scatter plot approach involves implementing an overview + detail technique whereby the user has the option to enlarge the overview visualization by selecting either overview or detail visualization for the full-screen display. A typical overview + detail visualization displays the overview in a small thumbnail at the bottom right corner of the detail view, as shown in Figure 8(a). There are drawbacks associated with small overviews on mobile device screens, including difficulty relating the overview and detail view.[16] As a result, we implemented functionality which *allows the user to increase the size of the overview display.* Thus, by tapping on the thumbnail overview display, the user is presented with the visualization seen in Figure 8(b), where the


(a) Default display.

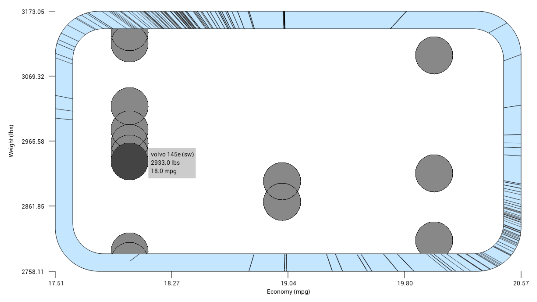
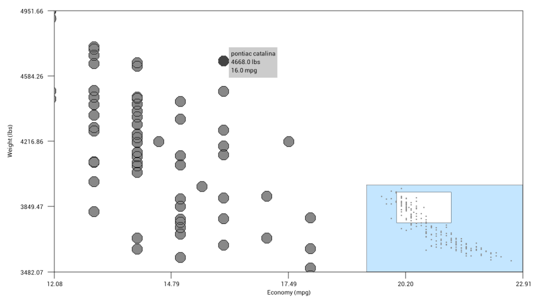

(b) The user has zoomed in by performing a spread gesture and selected a data item by performing a tap gesture.

Figure 7. A visualization providing focus in the desired rectangular area while simultaneously representing the remaining data contextually, in the blue region.

overview is displayed full-screen with highlighted detail. By tapping within this highlighted detail region, the user returns to the previous visualization (seen in Figure 8(a)). Therefore, the user can efficiently and seamlessly toggle between full-screen overview and full-screen detail techniques, both of which are interactive.


(a) Overview + detail visualization displaying the overview in a small thumbnail at the bottom right corner of the detail view.

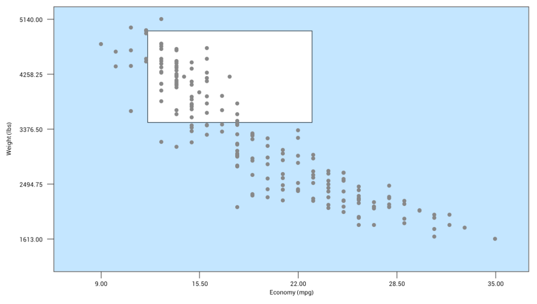

(b) Overview + detail visualization displaying the overview full-screen with highlighted detail.

Figure 8. An overview + detail technique where the user can toggle seamlessly between coupled full-screen overview and full-screen detail display.

5.3.1 Interactive Functionality

Supported interactive functionality varies depending upon whether the overview or detail view is displayed full-screen. If the detail view is displayed full-screen, the user can scroll through the data values by sliding a finger in any direction. This functionality will relocate the ROI within the information space. When this happens, some detail data values may disappear from the full-screen detail view while others may appear. Also, as the user scrolls, the highlighted detail region within the overview thumbnail will move to reflect the relative location of the current ROI within the information space. The user may also perform spread or pinch gestures to increase or decrease the level of detail within the full-screen detail display. This will have the effect of shrinking or enlarging the highlighted detail region in the overview thumbnail.

If the overview view is displayed full-screen, the user may move the highlighted detail region by sliding it to the desired location or resize the highlighted detail region by performing a pinch or zoom gesture. The user may also tap on the highlighted detail region in order to return to the full-screen detail view. The user could generate similar visualizations with spread and pinch gestures. However, spread and pinch gestures require time and accuracy to generate the desired visualization and one may lose context of where you are in the dataset. Our one-tap gesture seamlessly toggles between the two coupled visualizations - overview & detail.

5.3.2 Implementation

We draw the detail view in the same way that we have drawn the previous visualizations, however we exclude the aggregation area and instead include the thumbnail overview in the bottom right corner. In order to draw the data points in the thumbnail overview, we need to modify the multiplier and offset values appropriately and draw the data points again. Our thumbnail overview depicts the entire dataset (and includes margins) with a width of one-third of the available detail display width (minus left and right margins) and a height of one-third of the available detail display height (minus top and bottom margins). As a result, the multiplier for the thumbnail display is calculated as: $m_t = \frac{w}{(3)(w+left+right)}$ where m_t is the thumbnail multiplier, w is the available detail display width (minus left and right margins), $left$ is the left margin width, and $right$ is the right margin width. The horizontal offset and the left margin for the thumbnail view are: $o_{tx} = (m_t)(left), l_t = left + \frac{(2)(w)}{3}$ where o_{tx} is the thumbnail offset in the x direction, m_t is the thumbnail multiplier, $left$ is the left margin width, l_t is the thumbnail left margin width, and w is the available display width (minus left and right margins). Similar calculations are utilized to determine the vertical offset and the top margin for the thumbnail view. With the multiplier, margins, and offsets determined for the thumbnail view, we calculate the x and y coordinates for each of the data points using the previously presented equations and draw the data values.

We represent the detail region as a white rectangle within the blue thumbnail overview (See Figure 8(a)). To calculate the coordinates for this highlighted detail region within the overview thumbnail, we first determine the four data values at the four edges of the full-screen detail display. The x-dimension data value at the left edge of the full-screen detail display is: $v_l = \frac{o_x}{(w)(m)}(max_x - min_x) + min_x$ where v_l is the x-dimension data value at the left edge of the full-screen detail display, o_x is the x-dimension offset, w is the available display width (minus left and right margins), m is the multiplier, max_x is the maximum x-dimension data value, and min_x is the minimum x-dimension data value. The x-dimension data value at the right edge of the full-screen detail display is: $v_r = \frac{w+o_x}{(w)(m)}(max_x - min_x) + min_x$ where v_r is the x-dimension data value at the right edge of the full-screen detail display, w is the available display width (minus left and right margins), o_x is the x-dimension offset, m is the multiplier, max_x is the maximum x-dimension data value, and min_x is the minimum x-dimension data value. We use similar calculations to determine the y-dimension data value at the top and bottom edge of the full-screen detail display. With knowledge of the four data values at the four edges of the full-screen detail display, we can then calculate the four coordinates for the thumbnail detail rectangle. The left border of this rectangle is: $l_{tf} = \frac{(v_l - min_x)}{(max_x - min_x)}(w)(m_t) + l_t + o_{tx}$where l_{tf} represents the left border of the white detail rectangle within the blue overview thumbnail, v_l is the x-dimension data value at the left edge of the full-screen detail display, min_x is the minimum x-dimension data value, max_x is the maximum x-dimension data value, w is the available display width (minus left and right margins), m_t is the thumbnail multiplier, l_t is the thumbnail left margin width, and o_{tx} is the x-dimension thumbnail offset. Similar calculations are performed to determine the remaining coordinates for the highlighted thumbnail detail rectangle.

By utilizing this approach, the detail and overview thumbnail remain synchronized at all times. As a user "zooms in" on the detail view, the corresponding detail data values enlarge while the thumbnail data values remain unchanged and the thumbnail detail rectangle *decreases* in size. Similarly, as the user performs a scroll gesture from right to left on the detail view, the detail view data values will also move from right to left while the thumbnail data values remain unchanged and the thumbnail detail rectangle moves *from left to right.*

The calculations to display the full-screen overview are very similar to the calculations to display the thumbnail overview, however we use different values for multiplier, margins, and offsets. Also, when in full-screen overview mode, in our scroll event handlers, the offset is incremented not by the distance scrolled, but by the *negative* distance scrolled. Similarly, when in full-screen overview mode, in our "zoom" event handlers, the multiplier is incremented not by a step value, but by the *negative* step value. This allows a user to interact directly with the highlighted detail rectangle within the full-screen overview. Thus, as the user "zooms in" on the full-screen overview, the detail rectangle increases in size and as the user performs a scroll gesture from right to left on the full-screen overview, the detail rectangle will also move from right to left.

6. USER EVALUATION

In addition to the design and development of the visualizations presented above, we also conducted a user evaluation. In order to facilitate the user evaluation, we created two additional standard control visualizations (one bar graph and one scatter plot) against which we compared our visualizations. The standard bar graph control visualization (referred to as BG4) provides no representation of contextual data, but it does provide the ability to zoom in, zoom out, scroll, fling, and tap on an individual data item to see its corresponding value. Similarly, the standard scatter plot control visualization (referred to as SP4) provides no representation of contextual data, but it provides the ability to zoom, scroll, fling, and tap on an individual data item. Therefore, we evaluated a total of eight visualizations.

There were 22 test participants in our user evaluation, consisting of 11 males and 11 females, all of which were between the ages of 18 and 49. All test participants answered multiple choice questions for each of the eight visualizations, in which they were asked to find: the highest, the lowest, the three highest, or the three lowest data items for the presented visualization (or for the specified dimension of the presented visualization for scatter plots). We identified several appropriate data sets of different types from the Carnegie Mellon data set archive[21] for the user evaluation.

The order of the eight visualizations tested was randomized, the ordering of the test questions were randomized, and the datasets differed between visualizations. Users were asked to find the best answer to the posed question and were then given an opportunity to provide qualitative feedback and evaluate the visualization. The user was required to rate her/his level of frustration in finding an answer to the test question. The level of frustration was measured on a Likert scale (1: not at all frustrating, 5: extremely frustrating). Lastly, the user could optionally enter a free-form text response to provide any other comments, thoughts, or ideas.

The test was administered using a testing program that obtains consent, collects demographic information, collects answers and records the time required to answer each question. The user evaluation results for the bar graph visualizations are show in Table 1 and the user evaluation results for the scatter plot visualizations are shown in Table 2.

Table 1. Bar Graph Visualization User Evaluation Results

Visualization	% Correct	Avg. Time (*sec.*)	Frustration Level
BG1	68.18%	117.50	2.18
BG2	**72.73%**	**80.20**	**2.09**
BG3	68.18%	99.96	2.86
BG4	63.64%	82.10	2.50

Table 2. Scatter Plot Visualization User Evaluation Results

Visualization	% Correct	Avg. Time (*sec.*)	Frustration Level
SP1	68.18%	95.97	2.86
SP2	68.18%	85.84	2.59
SP3	**77.27%**	**73.08**	**2.27**
SP4	**77.27%**	81.56	2.50

6.1 Analysis of User Evaluation

6.1.1 Bar Graph Visualization Analysis

Based upon the results, BG2 (Fisheye-Style Focus + Context Visualization) resulted in the greatest user accuracy, the lowest average time required, and the lowest frustration level of all of the bar graph visualizations (including the standard control bar graph visualization).

Although we envision the differences between BG2 and BG3 to be relatively minor, the results indicate that there exists a sizeable difference in user performance and satisfaction for these two visualizations. We expected that the abrupt transition between focus and context regions in BG2 might cause confusion or misinterpretation of the data and wondered whether BG3, with its gradual transition from focus to context, might be more comprehensible. Clearly, this was not the case. We received comments from participants about BG3 in which they said that they did not like the way the data “folded up” and that the variation in bar widths was “frustrating.”

We hypothesize that BG2 may be simpler in that there are only two bar widths to interpret, the focus bar width and the context bar width. BG3, on the other hand, contains bars of many different widths, all of which change gradually as the user scrolls through the dataset. Furthermore, with BG3, it is difficult to judge precisely where the focus region ends and the context region begins. Lastly, as mentioned earlier, the overall width of the BG3 visualization is dependent upon the location of the bar of maximum width. Thus, the width of the entire BG3 visualization changes as the user scrolls through the dataset, which could cause confusion or even appear erroneous.

The results also indicate the benefits associated with providing context in visualizations. BG4 contains similar features and functionality to the other visualizations *except* for the fact that it did not provide context in any way. BG4 resulted in the lowest accuracy and the second highest frustration level, leading us to conclude that effective mechanisms for providing context improve user performance and also user satisfaction. One user stated of the standard control bar graph visualization, “Knowing where you are on the graph is impossible.”

6.1.2 Scatter Plot Visualization Analysis

Based upon the results, SP3 (User Driven Overview + Detail) resulted in the greatest user accuracy (tie), the lowest average time required, and the lowest frustration level of all the scatter plot visualizations (including the standard control scatter plot visualization).

As the SP3 visualization toggles between one of two modes (full-screen detail and full-screen overview), it is not clear whether the ability to switch between full-screen detail and full-screen overview modes helped to improve user satisfaction and/or user performance. As user comments do not help to answer this question, additional testing could be helpful.

SP2 resulted in a lower average time required and a lower frustration level than SP1. Although the two visualizations are very similar, SP2 utilizes more of the available screen real estate to display the visual representation of the data. Several participants commented that they preferred SP2 to SP1 because it had a larger focus region and was capable of displaying a larger representation of the entire dataset within the focus region.

Overall, it appears that users preferred context provided by overview + detail to context provided by aggregation using the circle or rounded rectangle. However, several users did comment that after interacting with SP2 for a few minutes, they found the aggregation area to be very helpful in identifying minimum or maximum values. We believe that the standard overview + detail approach is a more familiar context-providing approach than aggregation approaches for the general public.

7. CONCLUSIONS AND FUTURE RESEARCH

We have presented and evaluated tablet-based visualization techniques that provide contextual cues about off-screen data objects. We developed three bar graph visualization apps and three scatter plot visualization apps, all of which provide context in varying ways. Through our user evaluation, we found that Fisheye-Style Focus + Context Visualization (BG2) to be the best bar graph visualization technique resulting in the highest user accuracy, the lowest average time required, and the lowest frustration level. We observed that users found BG2 to be relatively straightforward and simple, lacking the visual complexity associated with BG3 and providing more intuitive contextual data representation than BG1.

For scatterplots, we found the User Driven Overview + Detail (SP3) to be the best technique resulting in the highest user accuracy (tie), the lowest average time required, and the lowest frustration level. We hypothesize that the success of SP3, which is a modification of the standard overview + detail approach, may be due in part to the fact that overview + detail is a more familiar context-providing approach than the “competing” approaches.

Future user evaluations could also compare SP3 with a standard overview + detail approach to evaluate the value of toggling between full-screen detail and full-screen overview modes in SP3. The number of data items in BG2 and BG3, however is theoretically limited. As the minimum bar width is 1 pixel, the maximum number of data items is determined by the width (in pixels) of the available display area. We envision future research focused on novel approaches to displaying datasets larger than the limits imposed by the available display width.

REFERENCES

[1] "Gartner says worldwide media tablets sales to reach 119 million units in 2012." `http://www.gartner.com/it/page.jsp?id=1980115` (2012).

[2] Chittaro, L., "Visualizing information on mobile devices," *Computer* **39**(3), 40–45 (2006).

[3] Oviatt, S., "Human-centered design meets cognitive load theory: designing interfaces that help people think," in [*Proceedings of the 14th annual ACM international conference on Multimedia*], 871–880, ACM (2006).

[4] Inc., F., "Fidelity mobile apps." `https://www.fidelity.com/mobile/overview`.

[5] Fitbit, "Fitbit app gallery." `http://www.fitbit.com/apps`.

[6] Inc., K., "Ves/paraview mobile remote control." `http://vtk.org/Wiki/VES/ParaView_Mobile_Remote_Control`.

[7] Burigat, S. and Chittaro, L., "Geographic data visualization on mobile devices for user?s navigation and decision support activities," *Spatial Data on the Web–Modelling and Management* , 261–284 (2007).

[8] Karstens, B., Kreuseler, M., and Schumann, H., "Visualization of complex structures on mobile handhelds," in [*Proc. International Workshop on Mobile Computing*], (2003).

[9] Zhang, D., Karabatis, G., Chen, Z., Adipat, B., Dai, L., Zhang, Z., and Wang, Y., "Personalization and visualization on handheld devices," in [*Proceedings of the 2006 ACM symposium on Applied computing*], 1008–1012, ACM (2006).

[10] Chittaro, L., "Visualization of patient data at different temporal granularities on mobile devices," in [*Proceedings of the working conference on Advanced visual interfaces*], 484–487, ACM (2006).

[11] Schmeiß, D., Scherp, A., and Staab, S., "Integrated mobile visualization and interaction of events and pois," in [*Proceedings of the international conference on Multimedia*], 1567–1570, ACM (2010).

[12] Karstens, B., Rosenbaum, R., and Schumann, H., "Visual interfaces for mobile handhelds," *Proceedings of HCII2003,(Crete/Greece)* (2003).

[13] Hertzog, P. and Torrens, M., "Context-aware mobile assistants for optimal interaction: a prototype for supporting the business traveler," in [*Proceedings of the 9th international conference on Intelligent user interfaces*], 256–258, ACM (2004).

[14] Burigat, S., Chittaro, L., and Gabrielli, S., "Visualizing locations of off-screen objects on mobile devices: a comparative evaluation of three approaches," in [*Proceedings of the 8th conference on Human-computer interaction with mobile devices and services*], 239–246, ACM (2006).

[15] Gustafson, S., Baudisch, P., Gutwin, C., and Irani, P., "Wedge: clutter-free visualization of off-screen locations," in [*Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*], 787–796, ACM (2008).

[16] Burigat, S. and Chittaro, L., "Visualizing references to off-screen content on mobile devices: A comparison of arrows, wedge, and overview+ detail," *Interacting with Computers* **23**(2), 156–166 (2011).

[17] Burigat, S., Chittaro, L., and Vianello, A., "Dynamic visualization of large numbers of off-screen objects on mobile devices: an experimental comparison of wedge and overview+ detail," in [*Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*], 93–102, ACM (2012).

[18] Drucker, S. M., Fisher, D., Sadana, R., Herron, J., and schraefel, m., "Touchviz: a case study comparing two interfaces for data analytics on tablets," in [*Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*], *CHI '13*, 2301–2310, ACM, New York, NY, USA (2013).

[19] Jerding, D. F. and Stasko, J. T., "The information mural: A technique for displaying and navigating large information spaces," *Visualization and Computer Graphics, IEEE Transactions on* **4**(3), 257–271 (1998).

[20] Lamping, J., Rao, R., and Pirolli, P., "A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies," in [*Proceedings of the SIGCHI conference on Human factors in computing systems*], 401–408, ACM Press/Addison-Wesley Publishing Co. (1995).

[21] "Carnegie Mellon statlib datasets archive." `http://lib.stat.cmu.edu/datasets/`.