Beyond-the-Desktop Interactive Visualizations

Steffen Wenz

Abstract— There are many established information visualizations on desktop computers that rely on a regular screen and the combination of mouse and keyboard as input devices. Mobile devices, however, are becoming more and more widespread. Also, tabletop computers may soon find their way into everyday life. Existing visualizations cannot be ported to these devices due to different screen sizes and input modalities. In this paper, nine exemplary interactive visualizations across different devices are discussed, covering three different areas: photo collections, maps and scatterplots. Then, four different criteria are applied to them. The examples are categorized by data type, screen size, input bandwidth and supported tasks and techniques to reduce screen clutter. The resulting classification leads to some discoveries concerning how the examples deal with the limitations and opportunities of new devices. Photo collection tools, for example, are forced to abandon their typical file browser interface on mobile devices. Maps, on the other hand, profit from innovative methods of input. Altogether, interactive visualizations on devices beyond the desktop have the potential to develop new input and output concepts that increase usability.

Index Terms—Interactive visualizations, mobile devices, tabletop computers, taxonomy

1 Introduction

Information visualizations have been a subject of research for many years by now. Computers are becoming more and more powerful and enable complex scientific visualizations. But moreover, information visualizations have found their way into many everyday application domains. People increasingly use visual tools such as zoomable maps, charts and diagrams to navigate large data sets or visualize data themselves. For example, map visualizations such as Google Maps have found widespread use in recent years. Also, Microsoft Excel and other spreadsheet applications allow users to generate many kinds of diagrams with just a few clicks.

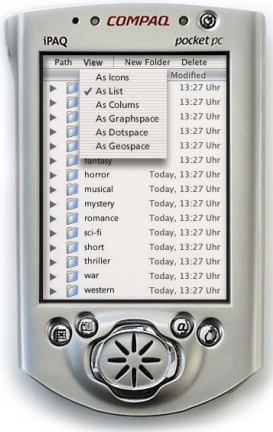

Fig. 1. Small screen space on a PDA [13]

However, these applications usually run on desktop computers and therefore rely on a regular computer screen for output, and a keyboard and mouse for input. Furthermore, they are tailored to the usage behavior associated with desktop computers, meaning that applications can assume that the user is seated at a desk. But nowadays, computers come in all shapes and sizes! Surface computers and large wall-mounted displays offer more screen space than desktop computers and enable new forms of collaboration in applications, but require radically different input devices and methods of usage. Also, mobile devices are becoming ubiquitous and are slowly catching up to desktop computers in terms of processing power and storage capacity, but have limited methods of input and little screen space. This issue is demonstrated in figure 1, where a file browser view, though very compact, can only display 15 data items on a PDA.

Visualizations that are designed for desktop computers cannot simply be ported to other device types. The differences in screen size, methods of input and general usage behavior have to be taken into account [6]. These can either be limiting factors, or they can enable interactive visualizations beyond what is possible on a desktop computer. In this paper, a selection of example interactive visualizations across different devices is discussed. The purpose of this is to find out how common visualizations are adapted to the characteristics of certain devices. Since this discussion should be structured in some way, a set of properties or criteria will be applied to all examples. These criteria, which are introduced in the following section, will make it possible to quickly see differences and similarities between visualizations on various devices.

- *Steffen Wenz is studying Media Informatics at the University of Munich, Germany, E-mail: steffen.wenz@campus.lmu.de*
- *This research paper was written for the Media Informatics Advanced Seminar on Information Visualization, 2008/2009*

2 Properties of Interactive Visualizations

There are some requirements for the criteria chosen in this paper. Firstly, they should reflect the properties of the device the visualization was designed for. Visualizations may depend on certain methods of input or output that are not available on other devices. Secondly, the criteria should contain information on how the visualization is adapted to the characteristics of its device. For example, an application on a mobile phone has to deal with the limited screen space, while an application on a surface computer may have to deal with the lack of a keyboard for text entry. The criteria introduced in this section build on previous research in the field of visualization taxonomies, which is discussed in the following subsection.

2.1 Related Work

A relatively early attempt to categorize visualizations is the task by data type taxonomy by Ben Shneiderman [12]. It assumes users are navigating a large set of structured data in search of certain information. Shneiderman proposes two criteria for classifying visualizations: data type and task. The data type describes the attributes of the data set that is to be visualized and can either be 1-, 2-, 3- or multidimensional, or temporal, network or tree data. The tasks are actions which the user may perform within the visualization tool, and can be any combination of the following: Overview, zoom, filter, details-on-demand, relate, history and extract. These tasks may seem familiar as they are based on the information-seeking mantra coined by Shneiderman: “Overview first, zoom and filter, then details-on-demand” [12].

Table 1. Data type criteria [12]

Data type	Typical example
1-dimensional	Textual data, lists
2-dimensional	Images, geographic data
3-dimensional	Architectural models
Temporal	Timelines with (overlapping) events
Multi-dimensional	Database records with more than three attributes
Tree	Hierarchically organized data
Network	Data sets with complex relationships

Table 2. Screen size criteria

Screen size	Typical devices
Small	Mobile phone, PDA
Medium	Laptop, desktop computer
Large	Tabletop, surface computer, wall-mounted-display

Table 3. Task criteria [12]

Task	Explanation
Overview	Gain an overview over the entire data set
Zoom	Zoom in on interesting data subsets
Filter	Filter out uninteresting data items
Details-on-demand	Show additional attributes
Relate	Show relationships with other data items
History	Keep a history of actions to support undo and refine
Extract	Extract interesting data subsets or query parameters

The tasks proposed by Shneiderman are generic and independent of their technical implementation on different devices. However, technical limitations of input devices are a critical factor for visualizations. In [5], the design space for input devices is analyzed. Devices are modeled as combinations of sensors which measure their position in one of three linear or rotary axes. The model also allows for discrete sensors, such as buttons. Individual sensors are then combined to form devices, using different composition methods. The authors prefer this parametrical analysis over a taxonomy, as the criteria one chooses are not guaranteed to be complete and on the same logical level. The paper also deals with different bandwidths of human muscle groups to be used with input devices, and of course also of the devices itself. Based on this concept, [14] proposes an abstraction layer to allow for substitution of input devices for other, equivalent devices. The authors specifically mention the need to emulate mouse and keyboard on mobile devices. One given example is text entry on mobile phones. The phone's number keypad has to replace a keyboard - the number of keys is of course much smaller, and the model accurately predicts that multiple key presses are needed to type a single letter.

Applications developers for handheld devices have to take great care to utilize the limited screen space efficiently. For this purpose, a number of techniques have emerged. [7] attempts to categorize these techniques for clutter reduction, as the authors call it, in a taxonomy. On the top level, the authors distinguish between techniques that affect appearance of data items, spatial distortion or temporal appearance (meaning animation). Sampling, filtering, changing point size or opacity and clustering are examples for appearance clutter reduction techniques. Point/line displacement, topological distortion, space-filling, pixel-plotting and dimensional reordering on the other hand are techniques working with spatial distortion. Finally, animation can also reduce clutter. The authors also compare these techniques against each other using a set of criteria, for example if they avoid overlap or if they keep spatial information intact.

2.2 Introduction of Criteria

While the taxonomies and models explained in the previous section seem very suitable for this paper, they cannot be applied to visualizations across devices as-is. Therefore, a combination of the criteria is proposed in this subsection.

Data The first criterion is the data type, in accordance with [12]. The data type not only tells what kind of data can be visualized, but is also characteristic of the task the user is trying to solve. Also, it may indicate what kinds of data sets visualizations are compatible with. For example, tools for viewing maps, which are essentially 2-dimensional data, may also be suitable for other types of images. The possible data types are defined in table 1. It should be noted that not all visualizations fit neatly into these categories. As this is a qualitative analysis, combinations of different data types shall simply be identified as such.

Screen size Screen size differs greatly among computers and is thus the second criterion. It is directly correlated with the amount of information that can be displayed at once. But different screen sizes are usually also associated with certain user behaviors. For example, mobile phones are not only characterized by their small screen, but also by their mobility context. Users may be outside in the sun (and low-contrast text thus be hard to read) and have a limited attention span. For this reason, the general device types will be noted alongside the screen size. This criterion will make it possible to identify types of visualizations which have not (yet) been adapted to certain screen sizes and device types. For simplicity, screen sizes are grouped into three categories (see table 2), and resolution is not taken into account. This list is imprecise and by no means complete. It is designed to fit the examples discussed later in the paper and needs to be expanded to include other devices.

Input device Applications may also heavily depend on certain input devices. The characteristics of input methods are the third criterion. To quickly characterize an input device, some hints are taken from [5]. A mouse would be described as a combination of two linear sensors (2D), a discrete rotary sensor (the mouse wheel, 0.5D) and three binary linear sensors (the buttons). The authors of the referenced design space analysis include much more detail in their model. However, since this paper constitutes only a qualitative analysis, it can be allowed to be less precise. Given these criteria, it can be determined if a visualization can in theory be ported to a different device which offers compatible methods of input.

Task/Technique Finally, it is interesting to see how applications deal with the characteristics of the device they were designed for. The fourth criterion is a combination of the tasks proposed in [12] and the clutter reduction techniques described in [7]. The tasks are defined in table 3.

The clutter reduction techniques identified in [7] are listed in table 4. The authors explicitly leave out some techniques, such as changing color as an appearance technique. The list of criteria will have to be expanded here.

These two sets of criteria are on slightly different semantic levels. Shneiderman's tasks can be actively performed by the user. The clutter reduction techniques, on the other hand, are generally techniques used by the application to enhance usability. However, there is some overlap: Zooming is both a task and a form of topological distortion, and also, filtering can be found in both taxonomies. Also, the clutter reduction techniques are meant to deal with limited screen space. But these techniques can be universally applied to visualizations, as clutter of information is also a problem on large screens due to cognitive limitations. The application of these criteria is expected to show whether certain devices only allow for visualizations with few supported tasks. Also, applications will be comparable in what techniques they apply to deal with device limitations.

Table 4. Clutter reduction technique criteria [7]

Technique	Explanation
Appearance	
Sampling	Show a random data subset
Filtering	Show a data subset based on query parameters
Change point size	Change size of item representation
Change opacity	Change opacity of item representation
Clustering	Merge items into a cluster
Spatial distortion	
Point/line displacement	Change position of data items
Topological distortion	Distort the background, either uniformly (zoom) or non-uniformly (fisheye)
Space-filling	Arrange items as non-overlapping rectangles (tree map)
Pixel-plotting	Show data items as single pixels
Dimensional reordering	Change attribute axes
Temporal	
Animation	Animate item representation

3 Examples

In this section, nine example visualizations from three different areas are introduced. The criteria chosen in the previous section are then applied to them.

3.1 Photo Collections

Digital cameras are becoming ubiquitous, and many modern mobile phones are capable of taking high quality photos with built-in cameras. As such, more and more people carry photo collections on their mobile devices. Typical tasks when working with photo collections include finding pictures from a certain time or event, but also organizing the pictures in folders and annotating them with keywords. The data type of photo collections is not immediately clear. Photos are 2-dimensional data, but since the focus for the following applications is on navigating the entire collection, the dimensions of the visualized metadata shall be considered the data type. For example, a tool that organizes photos by their average brightness would be considered to have 1-dimensional data.

Pocket PhotoMesa The first example is Pocket PhotoMesa, a zoomable image browser for PDAs [10]. To gain an overview over a collection of photos, a lot of screen space is usually needed. The authors of Pocket PhotoMesa avoid the need for any scrolling by displaying the entire photo collection in a tree map. The photos are organized in folders, each occupying a rectangular area on the screen which is filled with small thumbnails. The user then interacts with the application using a stylus. By tapping into the whitespace inside a folder, the application zooms in to this folder. By tapping a picture in any zoom stage, it is enlarged and brought to the foreground. Users can then pan and zoom the picture, and return to the collection view by tapping the white space surrounding the picture. Interestingly, Pocket PhotoMesa is an adaptation of an application on desktop computers. The authors mention the difficulties of dealing with the small screen space and the stylus input, which offers fewer input sensors than a mouse.

TiDi Browser TiDi Browser is also an image browser for PDAs, but employs different techniques to efficiently display many pictures on a small screen [3]. It takes advantage of metadata embedded in picture files, specifically time and location information. Users usually group pictures by events which are bound to a certain time and place. TiDi Browser does not require its users to sort pictures in folders themselves. Instead, two small histograms are displayed at the sides of the screen. One of them plots photo frequency over time, the other encodes the distance of each photo to a specified home zone (thus reducing location information to one dimension). Users can then identify events where many photos were taken at the same time in the same place and quickly jump there by tapping with a stylus. The center of the screen is reserved for viewing photos. The current photo takes up about a fourth of the screen area; its file name, time and distance to the home zone are also displayed. Below the currently selected picture, a small number of thumbnails is displayed, showing the temporal context. Users can drag along the time line and bring other pictures into view. Figure 2 shows a screenshot of TiDi Browser in action.

Fig. 2. TiDi Browser plots photo metadata in two histograms to provide an overview [3]

Flux Flux is an application for photo collections that runs on a surface/tabletop computer [2]. Unlike the previous two examples, it is mainly intended for organizing photo collections, rather than just browsing through them in a given structure. Flux makes use of the large available screen area to display many thumbnails at once. It sports a tangible user interface using real-world physical interaction, meaning that users manipulate screen objects directly using their fingers or two pens. Photos can be dragged, resized and rotated at will. They react in a physically plausible way by simulating inertia and friction. Using a circular gesture, photos can be hierarchically organized in clusters or "workspaces". These are visualized as white rectangles and can themselves be manipulated by touch gestures. The contained photos then behave as they were attached to the workspace. Photos can also be annotated using actual handwriting. In addition to this, Flux supports automatically arranging all photos by time, quality or similarity.

3.2 Maps

The visualization of geospatial data is a common application in many fields. Interactive geographic visualizations can convey more data than static maps, for example by using multiple layers of data. The examples in this section are limited to simple street map visualizations in the likes of Google Maps. In this case, the data type is 2-dimensional, as the maps are basically image data. Typical tasks when viewing a map are locating a certain place, judging distances or finding paths.

Halo + ZUI One common problem of map visualizations is that users lose their context when they zoom in on a point of interest. Several focus plus context visualizations exist to address this problem: Overview minimaps can always keep the entire map in sight, but take up some screen space and may to too small to be useful [6]. Fisheye visualizations make judging distances difficult. The authors of [1] take a different approach: Halo, which is a technique for visualizing off-screen locations. With Halo, the entire screen area is dedicated to the zoomable user interface (ZUI) of the map. It is assumed that points of interest (such as results of a location-based search) are marked with overlays on the map. If one of these points leaves the screen area due to panning or zooming, a circle is drawn around that location. The radius is calculated so that the arc of the circle is visible at the edge of the screen area. In addition to that, the opacity of the arc decreases as the point of interest moves further away. This way, users can quickly

and intuitively judge the distance to that point. If the program determines that too many circles would overlap in the same area, they are clustered to form a single halo with double line strength. Halo is a device-independent concept, but it is especially relevant for small screens. As such, for the sake of this paper it is assumed to run on a small screen device using a stylus as input.

PengYo Modern phones allow for interaction methods beyond button and stylus input. Apple's iPhone is a prime example: Its multitouch screen is the main means of interaction. Several sensors provide additional input: The iPhone has GPS support and also sports an acceleration sensor, allowing it to sense its orientation when at rest [1]. PengYo is an iPhone application for social interaction that takes advantage of these sensors [9]. It displays the position of nearby friends on a street map. (Facebook data is used for this purpose.) Friends can be "penged" simply by tapping their representative icons, upon which they receive a notification, much like Facebook's poke feature. The map is initially centered on the user's position, but can be panned by dragging the finger across the screen, and zoomed by touching the screen with two fingers and then varying their distance. This control scheme is a de facto standard for navigating large images on the iPhone, but PengYo employs another trick to enhance usability. The street maps images are loaded from Google Maps and are thus 2-dimensional, but they are displayed as a plane in 3-dimensional space in PengYo. The user can control the viewing angle by tilting the device: If the iPhone is held parallel to the earth's surface, the map is viewed straight from the top. But if the device is tilted upwards towards the horizon, or rotated, the view changes accordingly (see figure 3). The user interface serves as a metaphorical window to "hybrid space", meaning the enrichment of actual spatial data with abstract information. The user can thus examine his surroundings intuitively while preserving his position or context.

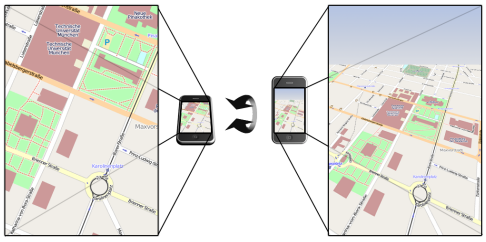

Fig. 3. The viewing angle is controlled by tilting the iPhone in PengYo [9]

DTLens DTLens is a map visualization tool for MERL DiamondTouch tabletop computers [8]. Its approach for providing a focus plus context interface is very different from the previous two mobile solutions. As the screen area of a tabletop computer is quite large, DTLens can afford to display the map in its entirety at all times. DTLens supports multiple users who interact with the tool using a multi-touch interface, as can be seen in figure 4. To view a point of interest in detail, the user creates a small rectangular fisheye lens by tapping once, or by opening and dragging the lens to the desired size with two fingers. The user has to press down both fingers firmly, or else the lens collapses when both fingers are released. These lenses serve as windows to a higher zoom level, while the information normally obstructed is preserved with fisheye distortion. DTLens offers some convenience functions: Users can change the size and zoom level of a lens, move it around on the map, and minimize it. Also, it is possible to draw overlays on objects in the lens view. If the lens is collapsed, the overlay is translated to the overview map. The global zoom level is constant, which greatly aides collaborative work. Also, the DiamondTouch screen is capable of distinguishing multiple users. The authors take advantage of this and allow each user only to manipulate the lenses he or she created.

[1] http://www.apple.com/iphone/features/

Fig. 4. DTLens supports collaboration of multiple users [8]

3.3 Scatterplots

Scatterplots are a quite universal visualization. They plot two (or three) variables of a given data set in a Cartesian coordinate system. Individual data items are displayed as points in the appropriate location. Additional data dimensions can be encoded in the appearance of the points, for example size and color. With scatterplots, it is possible to quickly see correlations between variables by the shape of the cloud made up of single data items.

Scatterplots with geometric-semantic zoom Scatterplots are generally used to visualize large data sets and as such seem unsuitable for small screens. It is vital for a scatterplot visualization to support the overview task as this is required to see trends and correlations. But also, users may be interested in details on single data items. [4] attempts to bring these features to mobile devices. The authors implemented a prototype that visualizes a book database with 7,500 items on a PDA. Two approaches to managing the limited screen size are then compared. The first approach uses geometric-semantic zoom. The user starts out with an overview of the entire collection. The date of publication and sales price are plotted on the two axes. Interaction is realized with a stylus. The user can zoom into a region of interest by tapping and holding near an item he or she wants more details on. At this point, some items may move off the screen. Thus, context is not preserved; only the labels on the axes provide some orientation. As the user zooms further by holding the stylus, the few data items still in sight transform from single pixels to white rectangles. They now contain more details on the books they represent, at first only the book title, then a picture of the cover and eventually other metadata. At the highest zoom level, a single item takes up most of the screen, with only the edges of neighboring items visible. As an alternative to this interface, the authors also developed a scatterplot visualization that uses fisheye distortion to preserve context at all zoom levels. Data item representations remain single pixels up to the highest zoom level, where they take up the entire screen.

Mobile Liquid 2D Scatter Space Another approach for a scatterplot interface on mobile devices is introduced in [13]. The authors created a scatterplot tool named Mobile Liquid 2D Scatter Space, or ML2DSS. The main feature of this visualization is the "liquid browsing" technique. One common problem of scatterplots is that items might overlap. The previous example solved this by letting the user zoom in until individual items were distinguishable. ML2DSS takes a different approach. Data items, in this case entries of a movie database, are represented as circles of different sizes. Instead of zooming, the user taps and holds a stylus near a single item or a cluster of items. The immediate area around the stylus is then magnified using a sort of fisheye distortion. The strength of this effect is controlled by the amount of force on the stylus. But instead of distorting the appearance of the data items themselves, only the distances between the circles are changed. As a result, the selected item stands out as neighboring items move to the side in a smooth animation. Users can receive details on items in a popup window. Also, they have full control over the configuration of the axes. Users can assign different attributes to

axes using drop-down list, and also adjust the range using text entry. Since these are quite disruptive changes to the visualization, the transition between two different states is always animated. Figure 5 shows a screenshot of ML2DSS where the user has selected a number of items (marked blue) and is currently holding the stylus near a data item to see additional details.

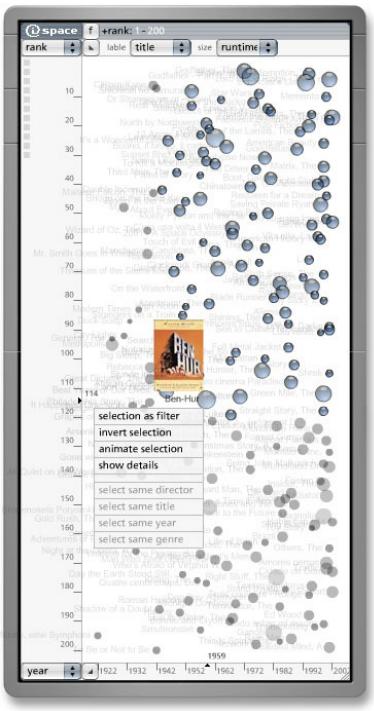

Fig. 5. Mobile Liquid 2D Scatter Space (selected items marked blue) [13]

3D scatterplots 3-dimensional scatterplots solve some of the shortcomings of 2-dimensional ones: A third dimension is added, and as such, yet more information can be visualized, and items that would be overlapping in a 2-dimensional projection are now distinguishable. However, user interaction is quite problematic. The three data dimensions still need to be reduced to two for output on a screen. The user needs to be able to navigate within the "cloud" made up by the data, and be able to select single or multiple data items. [11] shows a prototype of a 3D scatterplot on desktop computers. It allows users to load multidimensional data sets and then visualize up to five selected attributes (three axes as well as color and opacity). The information is then visualized in four linked views, each of which the user can rotate and zoom using a mouse. The prototype allows the selection of data items through brushing. Users can paint on any 3D scatterplot view; the data items that are painted over are highlighted in all views. Since this prototype was designed with Shneiderman's information-seeking mantra in mind, it supports many of the proposed tasks: It is possible to extract details-on-demand by plotting selected items in a separate histogram. Also, users can deduce relations/correlations between data items. For convenience, the prototype keeps a history of performed actions, and allows the extraction of selected data sets.

4 Discussion

In table 5, the results of the previous section are summarized. Altogether, the criteria are applicable for the chosen examples. However, in some cases, applying the criteria is a matter of interpretation and thus not deterministic. The visualizations are always designed specifically for small, medium or large screens and have well-defined methods of input. But the data type criterion is at times not easily applicable to the examples. Photo collections are hard to categorize. Not the pictures themselves, but the context in which they were taken is visualized. This is highly abstract information. In the end, however, all examples reduce this complexity by choosing only a few numerical dimensions to visualize. For example, Flux can sort pictures by their timestamp, quality and similarity, each of which is implemented as a 1-dimensional scalar value. This way, abstract information can be reduced to fit the data type criterion. Although intended for information visualizations by Shneiderman in [12], the data type criteria seem more suitable for scientific visualizations.

A taxonomy is "useful only if it facilitates discussion and leads to useful discoveries" [12]. A quick look at table 5 shows that all examples require at least 2-dimensional input. This is due to the fact that all examples let the user select or manipulate data item representations in 2-dimensional screen space. Examples that require only two dimensions of input are in principle portable to other devices that offer at least the same input bandwidth, for example a desktop computer in combination with a mouse. Pocket PhotoMesa and ML2DSS were ported from a desktop computer to PDAs and thus had to be adapted, since a stylus supports fewer modes of operation than a mouse [10]. Applications that run on tabletop computers, namely DTLens and Flux, take advantage of multi-touch and are not compatible with devices that do not offer this functionality.

The most interesting of the criteria are the tasks/techniques used by the visualizations. As mentioned before, the tasks as defined in [12] are actions the user can perform while seeking information, whereas the techniques as defined in [7] are mostly performed automatically by the application to reduce clutter. Visualizations for large or medium screens support 4 tasks on average, whereas small screen visualizations support only 2.7. On the other hand, small screen visualizations employ 4.3 clutter reduction techniques on average, compared to 3.7 for large or medium screens. This is not a proper statistical analysis, as the sample size is very limited and the criteria are not deterministic. But it is nevertheless an interesting observation. The medium/large screen applications all have more screen space and higher bandwidth methods of input. This seems to enable the support of more tasks. Especially relate, history and extract can be seen as convenience functions and are rarely implemented in the small screen examples. The additional controls needed to support these tasks would take up screen space and complicate the usage of the visualizations. Also, the clutter reduction techniques are intended for small screens by the authors. As such, it seems logical that they are used less frequently on medium and large screens.

Some tasks and techniques are common to all examples: All but one of the visualizations make use of animation of some sort. For example, DTLens animates the closing or minimizing of lenses, while ML2DSS animates selected data subsets to make them stand out from the rest of the data. Also, all examples with the exception of TiDi Browser support the overview task, and all but two support zooming. Some tasks and techniques are specific to certain types of applications. All map visualizations only support the overview and zoom tasks. Additional tasks would only be needed for more complex overlays. But since all examples focus on navigation within the map itself, zooming and panning suffice. Photo collections support more tasks, such as filter and relate. This makes sense, since users may want to filter large photo collections by their attributes, or see which photos relate to each other. The scatterplot examples all supported the details-on-demand task. The reason for this is that items in a scatterplot are only abstract representations of the original data. Once a user has navigated to an item of interest, it is necessary to provide additional attributes that were not visible in the overview visualization.

The visualization of photo collections is a very common task on desktop computers. Many tools for this purpose borrow heavily from the typical interface of file browsers. Figure 6 shows a photo collection viewed in the Microsoft Windows Vista Explorer. Google Picasa may arrange photos in "albums" instead of "folders", but the interface is still similar to that of a file browser, specialized for photo organization tasks [2]. The photo collection examples in this paper run on PDAs or tabletop computers and cannot use a mouse and keyboard for input.

[2] http://picasa.google.com/support/bin/answer.py?answer=93183#organize

Table 5. Visualizations and criteria

Visualization	Data type	Screen size	Input	Task/Technique
Pocket PhotoMesa	Photo collections (folder structure)	Small (handheld)	Stylus (2D)	Overview, zoom; Clustering, uniform topological distortion (zoom), space-filling (tree maps), animation
TiDi Browser	Photo collections (1D temporal, 1D distance)	Small (handheld)	Stylus (2D)	Filter, details-on-demand, relate; Filtering, pixel-plotting (histograms), animation
Flux	Photo collections (1D temporal, 1D quality, 1D similarity)	Large (surface computer)	Multi-touch (2*2D)	Overview, zoom, filter, relate; Filtering, change point size, clustering, point/line displacement (reordering), animation
Halo + ZUI	Maps (2D) + overlay (2D)	Small (handheld)	Stylus (2D)	Overview, zoom; Change point size, change opacity, clustering, uniform topological distortion (zoom), animation
PengYo	Maps (2D) + overlay (2D)	Small (handheld)	Tilt-sensor (3D), multi-touch (2*2D)	Overview, zoom; Change point size, topological distortion
DTLens	Maps (2D) + overlay (2D)	Large (surface computer)	Multi-Touch (2*2D), discrete touch strength (strong/normal)	Overview, zoom; Non-uniform topological distortion (fisheye), animation
Scatterplot with geometric-semantic zoom	2D	Small (handheld)	Stylus (2D)	Overview, zoom, details-on-demand; Change point size, topological distortion, space-filling, pixel-plotting, animation
Mobile Liquid 2D Scatter Space	2D + size/opacity coupled (1D)	Small (handheld)	Stylus (2D), continuous touch strength (1D)	Overview, filter, details-on-demand, relate; Filtering, change point size, change opacity, point/line displacement, non-uniform topological distortion (liquid effect), dimensional reordering (axes assignment), animation
3D scatterplot	3D + color (1D) + opacity (1D)	Medium (desktop computer)	Mouse (“2.5D”)	Overview, zoom, details-on-demand, relate, history, extract; Uniform topological distortion (zoom), pixel-plotting, dimensional reordering, animation

Therefore, they are forced to depart from the file browser metaphor and develop completely different interface concepts. For example, Flux simulates physical properties of the photos so that the user is reminded of sorting actual photos on a table. TiDi Browser uses metadata embedded in the photos to cluster them by events and locations, instead of relying on a given folder structure. These new interaction concepts can be more tailored to the specifics of photo collections as a data type and to the typical tasks performed with them. Therefore, the development of photo collection visualizations for small and large screens might result in better interfaces that will in the future influence the way photo collections are visualized on desktop computers.

Fig. 6. Photo collection in Microsoft Windows Vista Explorer [3]

The map visualization examples show that interaction with a map is possible on mobile devices through intuitive methods of input, such as dragging the map to pan. Each of the three examples focuses on a different aspect of map visualizations. Halo is concerned with the visualization of off-screen locations, which in turn is intended to cope with the limited screen space on small screen devices. PengYo uses acceleration sensors to intuitively adjust the 3D viewing angle of the map and thus concentrates on the input method rather than the visualization itself. DTLens enables collaboration through its multi-user, multi-touch interface. Therefore, each example aims to perfect a different component of interactive map visualizations. These could be combined to leverage all of the advantages. In contrast to the photo collection examples, which were mainly driven by input device limitations, map visualizations take advantage of new methods of input which increase usability. An example to support this hypothesis is the multi-touch capability of the iPhone. The iPhone uses multi-touch prominently to pan and zoom images and maps. Since this method of input turned out to be very successful, it has by now been ported to the new generation of MacBooks, which have multi-touch touchpads [4]. Also, Windows 7, the upcoming version of Microsoft’s operating system, will support multi-touch input [5].

Scatterplots are still a domain of desktop computers. They are used to visualize large data sets. Examples for scatterplots on mobile devices are still rare. But they show that scatterplots are capable of displaying relatively large data sets on small screens as a lot of information can be shown at once. Therefore, more scatterplot visualizations on small screen devices may be developed in the future. The 3D scatterplot example discussed in this paper was developed for desktop computers, but has the potential to be ported to tabletop computers. It would benefit from the higher screen area and resolution. Also, the main interaction method is brushing to select items, which could be realized with a touch interface.

[3] http://www.microsoft.com/windows/

[4] http://www.apple.com/macbook/

[5] http://www.microsoft.com/windows/windows-7/whats-new-possibilities.aspx

5 Conclusion

The criteria and the examples in this paper led to some interesting insights. Visualizing photo collections is a common application on desktop computers, but existing tools are inspired by file browsers and rely on a keyboard and mouse. This may not be the most effective concept to visualize photo collections, but people are simply used to working with file browsers. Photo collections on other devices are forced to abandon the file browser metaphor due to input and screen size limitations. As a consequence, this could introduce people to new concepts more suitable for photo collections, which might be ported to desktop computers in the long term. Interactive maps are also a very common application on desktop computers and mobile devices alike. Maps can take advantage of new methods of input offered by mobile devices. Multi-touch user interfaces for navigating 2-dimensional data such as images and maps have proven to be very successful on mobile devices. This realization has in turn started to influence desktop computers: Multi-touch will be supported by upcoming Apple and Microsoft operating systems, and has already found its way into some current laptops. Scatterplots are used to visualize large data sets in science and business, but are less common in consumer applications. However, they have potential to increase the usability of small screens since many data items can be visualized at once. Traditional visualizations for large data collections, such as column views, fill up small screens very quickly. In contrast, scatterplots visualize single data items as small circles or even single pixels, and show additional metadata only when an item of interest has been found.

The data type criterion was at times difficult to apply and seems more suitable for scientific visualizations that use numeric data. For example, photo collections are quite abstract data. The photos themselves may be 2-dimensional, but the examples in this paper focus on navigating within entire collections of photos, for example to find certain events. But in the end, this abstract task is accomplished by taking metadata such as a photo's timestamp or location into account - these additional attributes have a concrete data type.

The tasks and techniques supported by the examples were in some cases a matter of interpretation. Many tasks and techniques are somewhat abstract in nature and thus not deterministic. Still, the criteria chosen in this paper led to the desired results. One outcome was that applications on small screen devices seem to support fewer tasks, but apply more clutter reduction techniques. This makes sense since bigger devices have more screen space to position controls for additional functions, whereas applications on small devices have to use the available screen space as efficiently as possible and therefore apply clutter reduction techniques more aggressively.

The clutter reduction techniques proposed by [7] seemed somewhat arbitrary and overall incomplete. Using different colors to encode information was not included in the appearance criteria, but it was used by the 3D scatterplot prototype in [11]. The space-filling technique, described by the authors as a "non-overlapping rearrangement of large, rectangular points" [7], is mostly implemented in tree maps in practice, which is not reflected by its name. Also, it is unclear why the pixel-plotting technique is defined to encode information in single pixels only. The ML2DSS example displays data items as circles, i. e. identical geometric shapes, the only difference to single pixels being that items can vary in size and overlap partly.

These shortcomings could be addressed by conducting a more comprehensive survey of visualizations on various devices, including both scientific prototypes and commercial applications. The list of tasks and techniques could be expanded and refined in this way. Additionally, common techniques to compensate for input device limitations might be discovered. This paper was limited to only three application domains for visualizations. But of course there are many more types of visualizations which are appearing on small and large screen devices. This is an ongoing process, and as such it can be expected that entirely new concepts to visualize information on devices beyond the desktop will emerge in the future.

References

[1] P. Baudisch and R. Rosenholtz. Halo: a Technique for Visualizing Off-Screen Locations. In *CHI-CONFERENCE-*, pages 481–488. ASSOCIATION FOR COMPUTING MACHINERY INC, 2003.

[2] D. Baur, O. Hilliges, and A. Butz. Flux: Enhancing Photo Organization through Interaction and Automation.

[3] G. Bieber, C. Tominski, and B. Urban. TiDi browser: a novel photo browsing technique for mobile devices. In *Proceedings of SPIE*, volume 6507, page 65070O. SPIE, 2007.

[4] T. Büring, J. Gerken, and H. Reiterer. User Interaction with Scatterplots on Small ScreensA Comparative Evaluation of Geometric-Semantic Zoom and Fisheye Distortion. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, pages 829–836, 2006.

[5] S. Card, J. Mackinlay, and G. Robertson. A morphological analysis of the design space of input devices. *ACM Transactions on Information Systems (TOIS)*, 9(2):99–122, 1991.

[6] L. Chittaro. Visualizing Information on Mobile Devices. *COMPUTER*, pages 40–45, 2006.

[7] G. Ellis and A. Dix. A Taxonomy of Clutter Reduction for Information Visualisation. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, pages 1216–1223, 2007.

[8] C. Forlines and C. Shen. DTLens: multi-user tabletop spatial data exploration. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 119–122. ACM New York, NY, USA, 2005.

[9] M. Gross, H. Mangesius, D. Filonik, A. Hackel, and M. Bilandzic. Pengyo: A mobile application to support phatic communication in the hybrid space. In *Proceedings of the 6th International Conference on Information Technology: New Generations*. IEEE Computer Society Washington, DC, USA, 2009.

[10] A. Khella and B. Bederson. Pocket PhotoMesa: a Zoomable image browser for PDAs. In *Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, pages 19–24. ACM New York, NY, USA, 2004.

[11] G. SAHLING. *Interactive 3D Scatterplots–From High Dimensional Data to Insight*. PhD thesis, Masters Dissertation, Institute of Computer Graphics and Algorithms, 2002.

[12] B. Shneiderman. The eyes have it: a task by data type taxonomy for informationvisualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on*, pages 336–343, 1996.

[13] C. Waldeck, D. Balfanz, C. Center, and G. ZGDV. Mobile liquid 2D scatter space (ML2DSS). In *Information Visualisation, 2004. IV 2004. Proceedings. Eighth International Conference on*, pages 494–498, 2004.

[14] J. Wang and J. Mankoff. Theoretical and architectural support for input device adaptation. In *Proceedings of the 2003 conference on Universal usability*, pages 85–92. ACM Press New York, NY, USA, 2002.