A Flexible, Low-Complexity Device Adaptation Approach for Data Presentation

René Rosenbaum[a], Alfredo Gimenez[a], Heidrun Schumann[b], and Bernd Hamann[a]

Institute for Data Analysis and Visualization, Dept. of Computer Science, [a]University of California, Davis 95616, CA, U.S.A.;
[b]University of Rostock, Rostock, Germany;

ABSTRACT

Visual data presentations require adaptation for appropriate display on a viewing device that is limited in resources such as computing power, screen estate, and/or bandwidth. Due to the complexity of suitable adaptation, the few proposed solutions available are either too resource-intensive or inflexible to be applied broadly. Effective use and acceptance of data visualization on constrained viewing devices require adaptation approaches that are tailored to the requirements of the user and the capabilities of the viewing device.

We propose a predictive device adaptation approach that takes advantage of progressive data refinement. The approach relies on hierarchical data structures that are created once and used multiple times. By incrementally reconstructing the visual presentation on the client with increasing levels of detail and resource utilization, we can determine when to truncate the refinement of detail so as to use the resources of the device to their full capacities. To determine when to finish the refinement for a particular device, we introduce a profile-based strategy which also considers user preferences. We discuss the whole adaptation process from the storage of the data into a scalable structure to the presentation on the respective viewing device. This particular implementation is shown for two common data visualization methods, and empirical results we obtained from our experiments are presented and discussed.

Keywords: Device adaptation, Progressive refinement, Multi-resolution, Data presentation.

1. INTRODUCTION

Many modern techniques that allow for effective visual presentation rely on modern desktop computers, but as the need to access data at *any time* and from *any location* increases, so does the need for data visualization on devices that are mobile but constrained in their capabilities due to resource limitations. Because of these constraints, the presentation will be subject to long response rates and an overloaded visual representation. This in turn dramatically decreases user acceptance.[1] Appropriate adaptation is a crucial requirement for the use of constrained devices as valid and effective visualization tools.

Although approaches related to device adaptation have already been proposed[2–4] they usually lack flexibility or efficiency. Typical approaches store many different representations of a dataset or apply on-demand processing in order to satisfy device constraints. Though effective for some applications, this is highly resource-intensive and application-specific, especially if the device properties and user demands are heterogeneous or if the data is to be displayed many times on multiple devices. Progressive content transmission and refinement overcomes such resource limitations[5] and has a long tradition in image communication[6] and other application fields that reach far beyond its primary purpose.

We propose a device adaptation approach for data presentations that takes advantage of the inherent scalability of progressive data streams. Progressive presentations are created once, centrally stored, and used multiple

Further author information: (Send correspondence to René Rosenbaum)
René Rosenbaum: E-mail: rosenbaum@ieee.org
Alfredo Gimenez: E-mail: alfredo.gimenez@gmail.com
Heidrun Schumann: E-mail: schumann@informatik.uni-rostock.de
Bernd Hamann: E-mail: hamann@cs.ucdavis.edu

times. For application to multiple devices, centralized storage greatly reduces data redundancy and required resources. It allows for a presentation to be shown at successively increasing levels of detail (LODs), as long as the demands associated with processing, transfer, and display are met for the device. We also present a method to determine when the data to be handled uses the provided resources up to capacity. With this method, instantaneous system response and appropriately adapted presentation can be achieved for nearly any user and device. The practical relevance of this approach will be demonstrated for treemap and scatterplot displays (Figure 1).

After reviewing relevant work in the area of device adaptation and progressive refinement related to visualization (Section 2), we introduce the main idea of the proposed approach (Section 3). Section 4 is dedicated to discussing the options required to turn the data into a progressive presentation. We describe the adaptation strategy in Section 5. Section 6 presents results obtained from our experiments and summarizes the properties of the approach. Section 7 provides conclusions and remarks concerning future research directions.

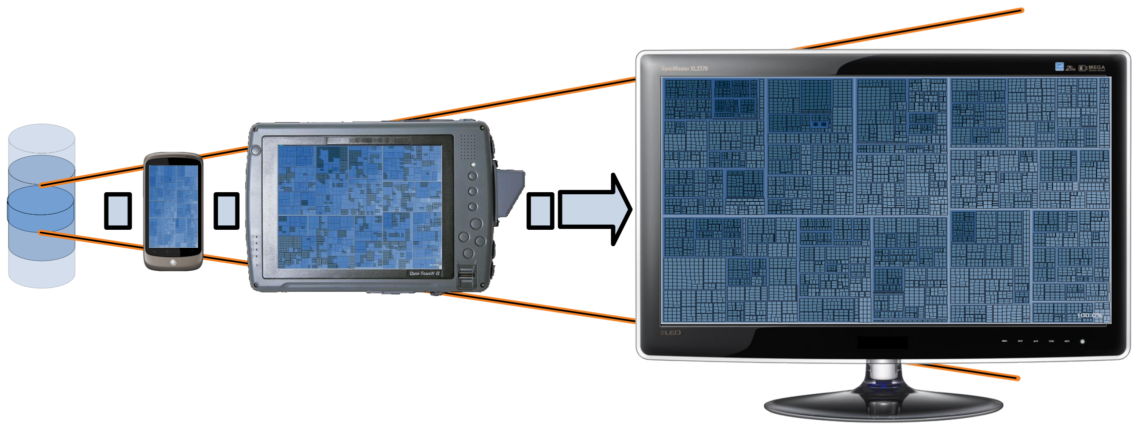

Figure 1. Representation of a treemap display on different viewing devices using the proposed adaptation approach.

2. DEVICE ADAPTATION AND PROGRESSIVE REFINEMENT IN VISUALIZATION

In the past, adaptation of visual contents to the capabilities of the viewing setup and user demands has mainly been accomplished by multi-resolution representations and the interactive or semi-automatic selection of an appropriate level of detail. Self-acting adaptation to the capabilities of the viewing device is considered to be an open research question and is not widely covered in literature. This problem becomes complex because of the multiple factors influencing a meaningful adaptation. Most of the related approaches published focus on the problems imposed by constrained display size,[7–10] but do not aim at a general solution considering other system constraints. One of the examples addressing adaptation to screen estate is the *output-oriented* rendering approach proposed by Novotny and Hauser.[8] Other technical capabilities such as the provided processing power or bandwidth are neglected. A proposal for adaptation based on the provided bandwidth has been presented by Mao et al.[3] They consider different transmission issues and conclude that one of the various factors that must be taken into account for adaptation is the amount of data to be transferred. Besides the amount of data does our approach consider the screen space, bandwidth, and computing power available on client side, and thus provides a more general solution.

Most of the adaptation strategies mentioned are server-based in order to rely on a-priori knowledge of the viewing device to create a representation that can be appropriately viewed with the client[2,4,10]). A typical example proposed for the adaptation of three-dimensional content based on the MPEG-21 framework has been introduced by Kim et al.[4] Although this is a meaningful strategy for selected application areas, the associated frequent creation and storage of many different representations of the same source is resource-intensive. Furthermore, this approach does not allow for interactive changes in the data or transmission order once it has been created. Our approach uses a predefined multi-purpose data structure that is created once and used for all device adaptations, which allows for user interaction and non-redundant data transmission, even when parts of the presentation have already been sent.

New research directions that apply principles from scalable image coding recently led to proposals founded on progressive content. The adaptation strategy proposed by THIEDE ET AL.[5] is fully client-based and provides a number of benefits as it has low complexity and does not require any prior knowledge of data, device, and user. This makes it a meaningful approach if the client conditions are not known in advance. However, the adaptation strategy is *reactive*[11] and requires additional computing power on client side. Our approach also makes use of progressive contents, but is *predictive* and server-based, and thus allows the client to dedicate all of its capabilities to decoding and display of the presentation.

Progressive refinement has been primarily developed to overcome bandwidth limitations in networked environments and tightly combines the different stages required for networked data displays – compression, transmission, and display. The general idea is to store the data in a single scalable representation. Whenever required, this resource is streamed and transmitted by traversing it in such a way that the decoding of a truncated stream leads to a reconstruction of the data with less detail.[6] Thus, content previews can be provided, and first conclusions can already be drawn with little data received. Progression also provides semantical benefits, such as the inherent support for the information-seeking mantra[12] and regions of interest (ROIs). Based on this, animated tours-through-the-data can be provided or transmission can be tailored to the requirements of the user.[13] However, little is known about the application of progressive refinement in the visualization of abstract information. Our prior work started to fill this gap[13, 14] and showed that wherever progressive refinement can be applied, it opens up a new way for efficient data handling and presentation. In the research we present here we apply the inherent principles of progressive refinement to the predictive adaptation of data displays for abstract data to the capabilities of the viewing device.

3. MAIN IDEA: CREATE ONCE, USE MANY TIMES

The adaptation approach proposed makes use of the scalability that is inherent in progressive data streams and can be literally stated as: "Create once, use many times". The progressively refinable data structure is *created once* and stored on central server space. If a user requests a visual presentation, the associated geometry is progressively streamed to the client device, and then displayed using the client's capabilities. When the client's resources are used up to their full capacities, the server truncates the progressive stream. This approach makes possible the *multiple use* of a single data source and a flexible, low-complexity adaptation.

The criteria we use to determine when refinement should be completed include aspects crucial for appropriate data presentation, such as response rate and maximum data volume to be transmitted. These requirements are constrained by the user preferences and device resources, in order to determine a truncation point that is specific for the respective user and device. This is achieved with a profile-based adaptation strategy applied on server side before transmission.

4. FROM DATA TO PROGRESSIVE STREAM

In order to apply progressive refinement, the original data must first be transferred into an appropriate internal representation before being streamed to the viewing device. This representation must be highly flexible in order to serve the requests of many different viewing devices and users, for any kind of data input. Such *progressive data structures* must satisfy two technical requirements: (1) *scalability* and (2) *compression* of the data. Besides stating general information to the satisfaction of those requirements, we show how they can be implemented for two data visualization methods of abstract data. More general information to the implementation of both requirements can be found in.[5, 13, 14]

Scalability Progressive refinement allows for constant previews of the data. These previews are abstractions of the data represented by a less detailed visualization. Scalability in this case refers to the ability to generate such previews at increasing levels of detail. For device adaptation, higher levels of scalability are preferable, since they allow for a broader range of detail levels, and therefore a broader range of devices. A progressive data structure is basically a *LOD-hierarchy* with leaf nodes representing the original *data items* and the inner nodes the *aggregate items* of all the items associated with the respective subtree.[15]

There are three main approaches for progressive refinement of presentations in networked environments, based on: the original data, the associated geometry, or the rendered image. While progressive refinement of image data is widely addressed in related literature, our application focuses on progressive refinement of the geometry used to render the image that is presented to the user. This requires scalability in data or geometry that, if not inherent, must be introduced. This is a still open issue of many ongoing research activities, though different meaningful solutions have previously been proposed. By demonstrating implementation of progressive refinement for the treemap[13] and three-dimensional scatterplot data display, we address both aspects of scalability: the refinement of (1) data that is already hierarchically organized and (2) data that requires the explicit generation of a hierarchy. Both displays make use of progressive data structures and hierarchical abstractions.

The *treemap display*[16] is a space-filling visual representation of hierarchical data. Each node of the hierarchy is represented by a single rectangle. These rectangles are shown as nested sets such that a certain node bounds all the nodes of its children, and thus each node represents a valid aggregation (cf. Figure 1). In this case, the data to be presented is already hierarchical, so the LOD-structure required for scalability is given for the data as well as geometry.

The three-dimensional *scatterplot display* is a visual representation technique for low-dimensional data. Three selected quantitative dimensions of the data are displayed as a collection of points in a Cartesian coordinate system. The data does not have inherent hierarchical structure in this case, so it must be introduced. Elmquist and Fekete[15] recently proposed an example for achieving this goal in the context of hierarchical data displays. Due to their efficiency and applicability, however, we adopted R-Tree[17] data structures for this purpose. R-Trees[17] are a LOD-hierarchy that cluster n-dimensional data components using spatial indexes. Data is organized into bounded clusters such that the root of the r-tree bounds all values in the dataset, and each hierarchy level bounds the clusters of its respective lower levels. We use these bounds as aggregate items within the tree (cf. Figure 2, right and 4). Each lower level of the tree provides more detail, and the number of levels is entirely dependent on the maximum allowed size of the clusters. The cluster size can also be dynamic or user-defined, in which case, the smaller the cluster size, the more levels within the tree, and the more granular the device adaptation. The trade-off is a larger amount of aggregate items.

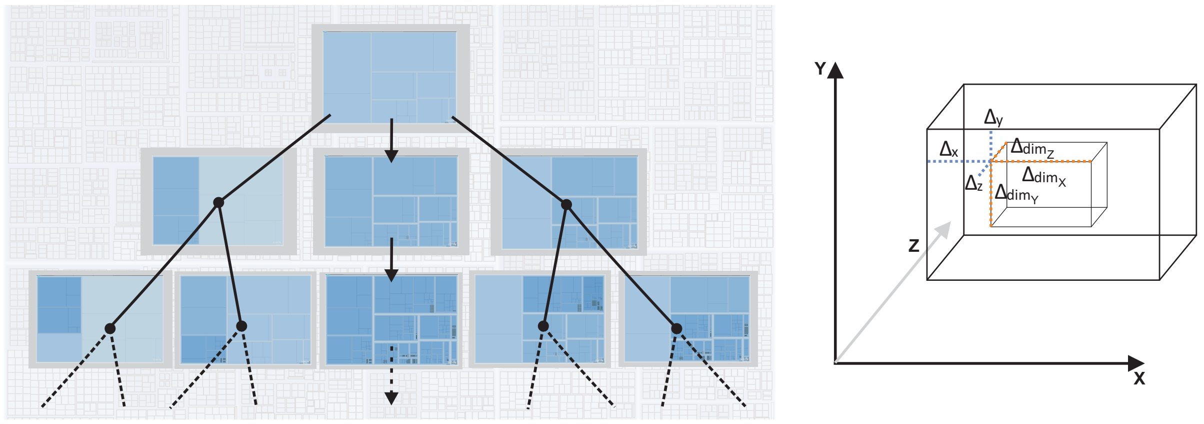

Figure 2. Illustration of a progressive data structure for the treemap display (left). Inner nodes represent aggregations associated with the corresponding subtree. The presentations shown in the center represent the contributions of all nodes of the hierarchy at the particular level. The display gains accuracy with each incremental refinement stage. The fact that the shown refinement is founded on recursive nesting can be used to encode the structure efficiently. The principle of delta-coding applied in our work is shown for a single parent and child box of a three-dimensional scatterplot (right).

Compression The use of an LOD hierarchy and aggregate items usually increases storage requirements. In order to decrease resource consumption and overcome bandwidth limitations, we propose compression of the centralized data structure. By compressing the progressive data structure and streaming the still compressed information, we achieve this for storage and transmission simultaneously. In order to avoid costly transcoding of the compressed data for different streaming orders, random access within the compressed data is required. The compression scheme most suitable for progressive refinement provides random access to all nodes of the LOD-hierarchy with the least consumption of system resources and transmission capacities. We also require the compression to be lossless in order to allow for high accuracy of the presentation.

For the compression of the progressive data structure, we take advantage of its inherent nature as a LOD-hierarchy. Due to the nesting property of each level, there are many redundancies that can be meaningfully removed by using incremental relative values instead of absolute values. This applies for progressive *treemaps* and *scatterplots* alike. To achieve this, we apply a compression approach similar to delta-coding.[18] Delta-coding offers a great trade-off in complexity, compression efficiency, and random access, and does not limit the scalability of the data structure, nor is it lossy. As the respective primitives can be defined relative to the parent primitive, a much smaller value range is required. This in turn can be used to reduce the number of bits needed to encode position and dimensions. Less bits are required with each successive progression level. As long as all related parent nodes are available, an individual primitive is independently decoded and accessible. We do not apply additional coding of the resulting individual codewords as decoding would further intensify the effects of computing constraints existing at client side.

5. PROFILE-BASED DEVICE ADAPTATION BASED ON PROGRESSIVE CONTENT

The proposed adaptation strategy is founded on two main concepts: (1) progressive refinement successively increasing detail and complexity of the presentation, and (2) profiles to determine the device's resource constraints serving as the foundation when to finish refinement. Our device adaptation approach also considers user variability, by combining the user and device perspectives into a single approach. After stating the *considered user requirements and associated system resources* discussed in this publication, we will provide details to the *profiles used for adaptation* as well as the *technical and semantical aspects of adaptation*. We will also propose a *a system framework for profile-based device adaptation.*

5.1 Considered user requirements and associated systems resources

User preferences are stated by a requirements tuple $lim(t, vol, size)$ which specifies three crucial aspects of data presentation. The attribute t refers to the *response time* of the system ($0 < t < \infty$) and is an important factor for usability. The attribute vol stands for the *maximum data volume* that can be transmitted ($0 < vol < \infty$) and is often used to limit transmission time or costs. $size$ expresses the *minimal primitive size* ($0 \leq size \leq \infty$) that must be met by all geometry shown on screen. The respective values can be interactively stated by the user, synthesized from prior or default user preferences, or can be based on empirical research, as the "web stress index".[1] The refinement is completed when a single requirement can no longer be satisfied by the system's capabilities. Providing the user with means to further refine the data, however, should always be a property of adaptation technology.

In order to consider the requirements tuple for adaptation, its attributes are mapped to the main resources of a viewing device as follows:

Computing power The provided computing power corresponds to the *response time*, t, of the system. It is assessed as the time required for updating the current visualization after new incremental data has arrived, and constrained by the device's ability to encode and visualize the incoming data interactively.

Bandwidth Transmission via low-bandwidth channels also strongly influences the *response time* of the system. This especially applies if large indivisible data chunks must be transferred. It is assessed via the amount of time necessary to transfer a constant amount of data. Another requirement related to communication via low-bandwidth channels is the *maximum data volume*, vol, that can be transferred to the client. It is assessed as the data volume that arrives on client side.

Display space Down-scaling of visual content in order to fit its presentation to limited screen estate causes many undesirable effects for presentation. The requirement of a *minimum primitive size*, $size$, represents the demands of a user to clearly recognize and distinguish individual primitives within a data presentation. It is assessed as pixels per dimension.

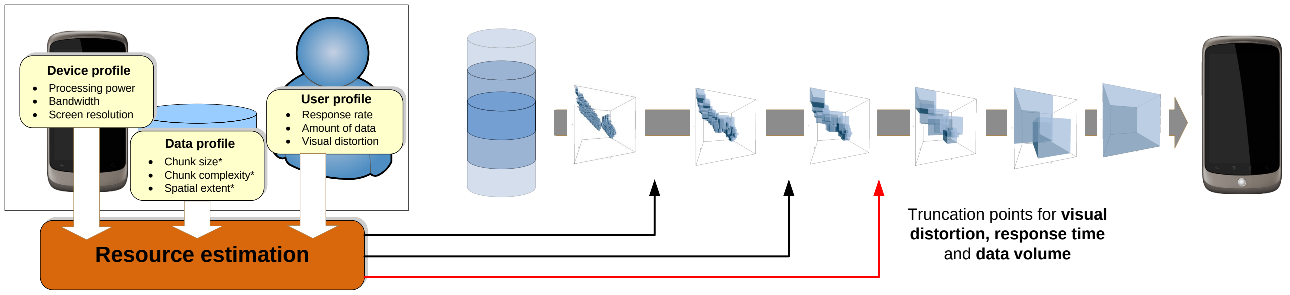

Figure 3. Profiles and attributes describing properties of *device* and *data* as well as *user* requirements. Resource estimation determines appropriate truncation of the progressive data stream for device adaptation (red arrow).

5.2 Profiles used for adaptation

In order to estimate when user requirements can no longer be met during refinement, the approach relies on a-priori knowledge stored in profiles. We divide into three different kinds of profiles: (1) device, (2) data, and (3) user profile (see Figure 3). The *device profile* describes the main resources of the viewing device that correspond to the stated user requirements and thus is crucial to achieve meaningful adaptation. The *data profile* serves as a means to consider properties of the data as well as particular implementations. The *user profile* has been introduced to allow for seamless integration of the user preferences into the approach.

Device profile This profile captures the main resource constraints of the device. It states the *processing power* of the device, the *bandwidth* of the devices' communication channel, and the *screen resolution* of the device. Due to the sheer variety of specifications and device types, we categorized devices into classes and used aggregated instead of detailed values for processing power and bandwidth. To select the value representing the capabilities of a specific class, we propose to conduct a series of empirical assessments and use the resulting average values. Screen resolution is defined by the number of pixels in each direction.

Data profile This kind of profile mainly describes and stores the properties of all individual nodes of the progressive data structure. *Chunk size* represents the amount of data associated with a certain node. *Chunk complexity* is stated individually for each displaying technique and specifies the efforts needed to decode and integrate the encoded primitive in the respective visualization. In order to determine these values, we assessed their decoding and update times empirically for each class of devices and visualization. In order to allow for multiple use, the *spatial extent* of a primitive is calculated for a fixed screen resolution that is later transferred into display screen space and measured in pixels.

We are aware of the fact that the introduced profiles do not represent all factors influencing an appropriate data presentation and strictly focused on attributes that are essential to illustrate the adaptation procedure. The dependencies between the different attributes and profiles are crucial for the success of the adaptation approach and explained as part of the adaptation procedure in the following section.

5.3 Technical aspects of adaptation

To overcome the various problems imposed by the complex creation[11] or storage of many similar presentations, we propose to progressively refine the data on client side until its system resources are utilized up to capacity. The appropriate amount and detail of data to deliver is predicted based on the stated profiles as follows:

The actual presentation and adaptation process starts after receiving the initial request for a data display from the client. The server selects the corresponding progressive data structure and traverses the hierarchy. It then sends the data to the client, which decodes the received chunks and incrementally restores the presentation. Device adaptation is accomplished by estimating the point within the incremental data stream when consumption exceeds the provided capacities (cf. Figure 3). In order to achieve this, profile matching and compliance checking is crucial and must be done for each attribute of the requirements tuple separately. Always starting from the beginning of the stream, each match leads to a different truncation point, so the point leading to the shortest data stream (see red arrow in Figure 3) is chosen to ensure compliance with all specified user requirements. In order to avoid early truncation caused by single geometry items, we propose to deal with compliance checking for *minimum primitive size* in a slightly different way; we omit all primitives that would violate the constraints from further compliance checking and profile matching.

Minimum primitive size The matching for minimum primitive size is based on the *spatial extent* of the primitive. As the spatial extent is based on a preassigned resolution, it must first be transferred to the domain of the respective viewing device as specified by its *screen resolution*. If the resulting extent is smaller than the primitive size allowed by the user, the primitive is omitted from transmission. This leads virtually to a smaller stream. The truncation point is the end of the current stream.

Response time Considering response time requires knowledge of the *processing power* of the viewing device and the *chunk complexity* of the data items as well as *chunk size* and provided *bandwidth*. The first geometrical primitive that is too complex to be decoded and integrated in the presentation within the required response time marks the truncation point with regard to processing power. The same strategy applies regarding bandwidth. The corresponding truncation point is found as soon as a data chunk is too large to be received within response time.

Maximum data volume This is the simplest matching for all three attributes. The *chunk size* of all data items are accumulated until the maximum allowed amount is reached. This determines the associated truncation point.

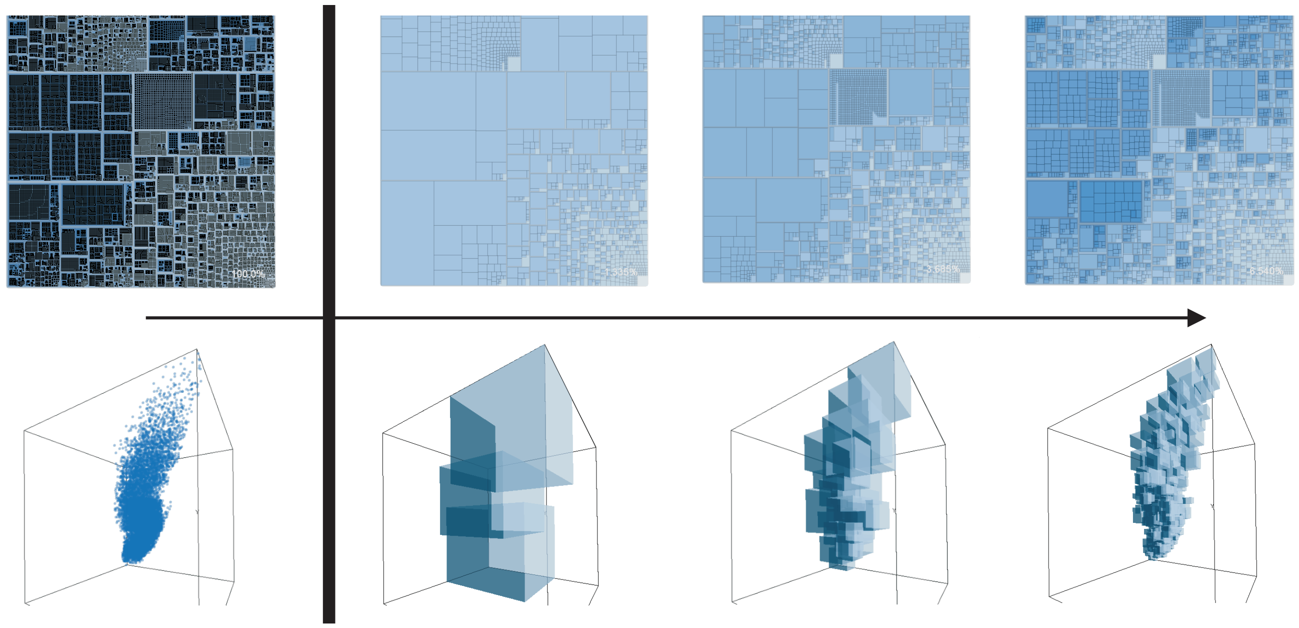

Figure 4. Examples for the visual representation of a dataset using the treemaps (top) and scatterplot display (bottom). The left column illustrates the displays showing *all* the data, the right column shows intermediate previews that are progressively refined until the final *adapted* presentations (right) are achieved.

5.4 Semantic aspects of adaptation

Removing parts of the data may lead to misinterpretations. We propose to provide one or both of the following kinds of visual clues in order to represent truncated data: (1) a refinement status display and (2) display-specific clues. The status display shows the relative amount of the currently displayed data. It is applicable to many different displays, but does not indicate neglected data within the display. Display-specific clues may be used to highlight omitted data, but are specific for each visualization approach. Status displays are well known and not further discussed in this publication.

The developed display-specific visual clues for the treemap display are based on color. An inherent property of progressive treemaps is that whenever inner nodes are visible it means that the node has further siblings. We take advantage of this property by assigning one distinct color to inner nodes and others to leaf nodes. In the case of the scatterplot, we distinguish aggregate items from data by shaping the former as rectangular prisms, and the latter as points. Both examples of display-specific visual clues are shown in Figure 4. It is worth noting that using different shapes does not violate our adaptation strategy regarding the requirements of constrained primitive size.

In order to allow for further investigation of the adapted presentation, we offer interactive means to refine those inner nodes that contain omitted data even if this violates the initially stated user requirements.

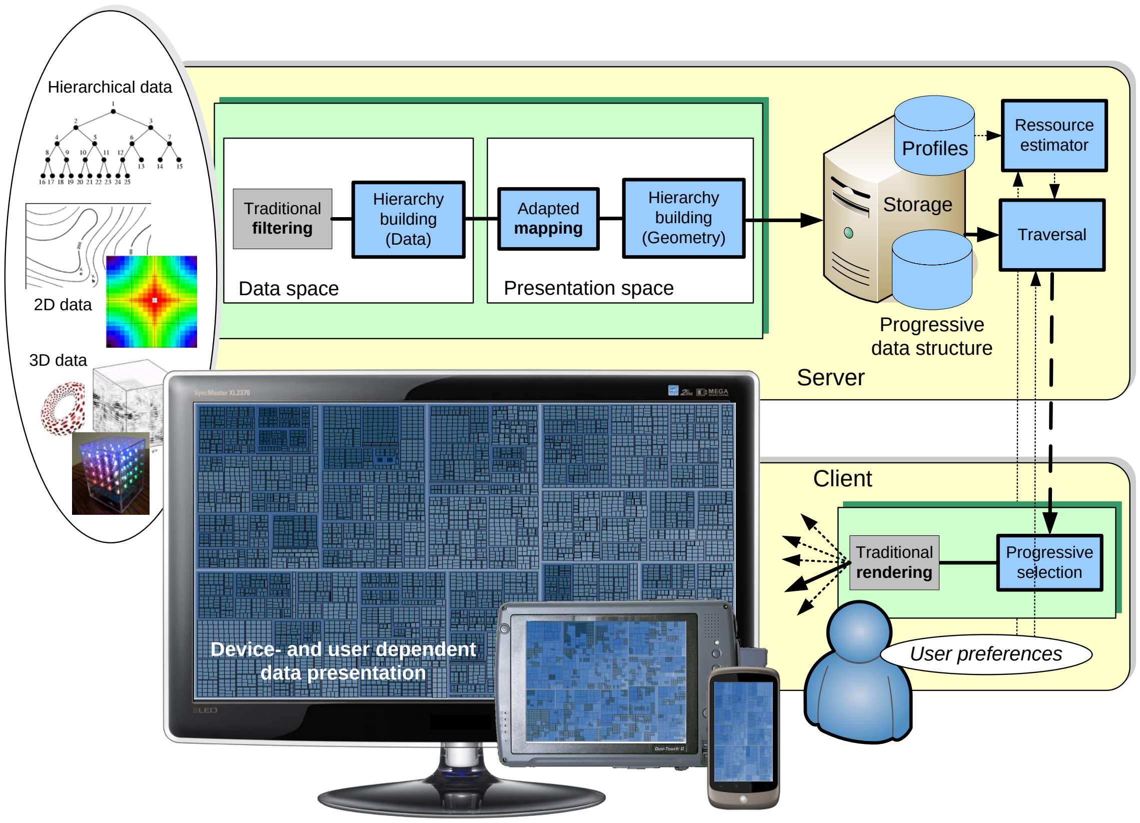

Figure 5. A system framework for profile-based device adaptation adopting the traditional visualization pipeline.

5.5 A system framework for profile-based device adaptation

Due to the variety of different demands and presentation systems, it is difficult to state in a rigorous manner where to implement the proposed concept for device adaptation into a specific visualization system. Thus, we propose a system framework for profile-based adaptation (Figure 5) that is based on a general progressive presentation framework[14] abstracting from the specifics of existing presentation systems. The framework is able to describe progressive treemaps and scatterplots as well as other data displays and serve as an aid for their suitable implementations. The following statements focus on the components required for progressive refinement and device adaptation only (blue boxes in Figure 5). For a more comprehensive explanation of the framework see[14] or.[19]

Server-side components The core of our approach is the compressed *progressive data structure* that describes the data visualization and provides lower detail aggregations. This structure is achieved by introducing scalability (hierarchy-building) in either data or geometry space. The following *mapping* stage must be able to assign meaningful geometry to the aggregations. A *traversal* component sequences the hierarchy dependent on current demands–the user preferences in this case. This sequence is validated by a *resource estimator* which determines a truncation point most appropriate for adaptation based on *profiles*.

Client-side components The incremental refinement data is received by a *progressive selection* component. This component successively decodes and reconstructs the geometry of the visualization and passes it to the rendering component in order to provide the next successive data view.

6. RESULTS AND DISCUSSION

We implemented the proposed device adaptation approach for the treemap and scatterplot displays. This section provides the *empirical results* obtained as well as a *discussion* of its general properties. The belonging visual representations produced by our approach for different viewing devices are illustrated in Figures 1 and 4.

6.1 Empirical results

We estimated the performance of the adaptation approach for different spatial resolutions and layouts. The dataset used for all assessments is *logs_A_03-01-01.xml*, part of the 2003 *InfoVis contest* concerning hierarchical data. Due to space constraints, we will focus in the presentation on utilizing progressive treemaps. The results obtained for the three-dimensional scatterplot, however, are meaningfully comparable. The increased system requirements imposed by handling the additional dimension, however, lead to less primitives that can be displayed in resource-constrained setups.

The presented results focus on two main points: (1) compression performance of delta-coding, in order to justify the proposed compression strategy of the progressive data structure, and (2) technical aspects of adaptation to demonstrate the achieved degree of adaptation to user-defined requirements.

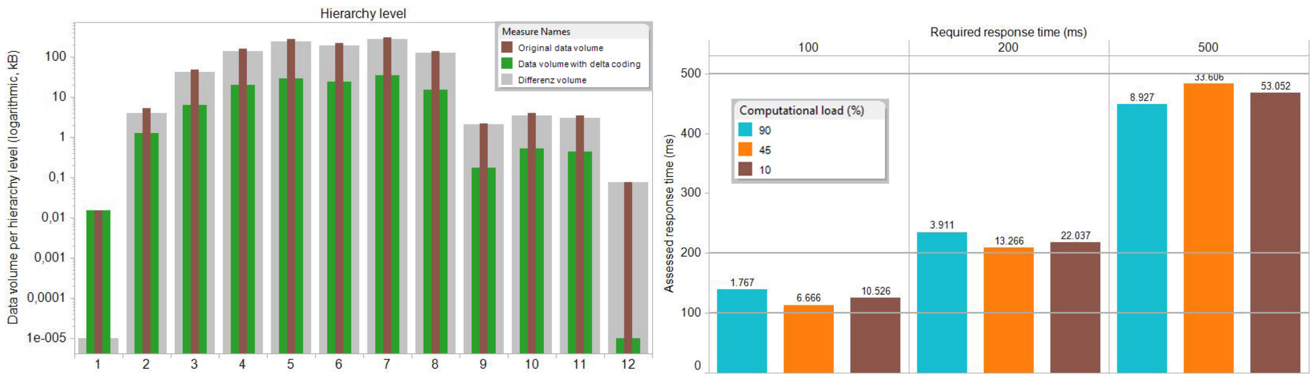

Figure 6. Data volume resulting from delta-coding vs. the original volume of primitives (left) and the adaptation quality of our approach regarding response time (right). The reference lines indicate the desired response times and the labels the numbers of transferred primitives. We used a constant screen resolution for all assessments.

Compression performance of delta-coding Applying an appropriate compression scheme to the LOD-primitive hierarchy leads to a significant reduction of its data volume. The results achieved by the proposed delta-coding of the rectangles are illustrated in Figure 6, left. The chart shows the reduction for each individual hierarchy level in logarithmic scale. Due to the fact that delta-coding cannot be applied on the first level, there is no compression gain. This changes with its application on every lower hierarchical level where less and less bits are required to encode the diminishing relative distances to parent primitives. This can be observed at level 2, where the encoding decreases the data volume by a factor of 4.2 and at level 12 where the factor is several orders of magnitude higher. This suggests that higher levels contribute more to data reduction. However, overall reduction also depends on the number of primitives to be encoded. Considering all primitives from the respective lower levels, the highest overall compression factor of 9.82 is achieved at level 7. Due to decreasing number of primitives, then, the achieved overall compression gain slightly decreases towards lower levels. The size of the whole delta-coded hierarchy is 11.28% of the size of the original hierarchy, resulting in an average compression factor of that is close to 9.

Similar results have been achieved for other datasets. It applies that *the more hierarchy levels and the more primitives at the lower levels* of the hierarchy *the higher the overall compression efficiency and gain.*

Response time In order to adapt the presentation to a specified response time, the client's processing power and the volume of each transmitted data chunk must be considered. The results presented in Figure 6, right, were obtained by observing the processing time required for primitive decoding and display. As their variety and hardware architectures are diverse and difficult to compare, we decided to simulate the computing power for different classes of viewing devices by using a mobile PC-based device at different degrees of capacity utilization (loads of 10%, 45%, and 90%). Assessing the volume of the transmitted data chunks is not required in our implementations because we sent each data item individually and thus, each transferred data chunk is small.

Figure 6 illustrates the degree of adaptation by the deflection of the measured values from the desired response time. The degree of adaptation is high, although no exact match could be achieved. It can be observed that our implementation overestimates the systems' capabilities for short response times. Contrarily, capabilities are underestimated for larger response times. This is due to the fact that the used treemap implementation handles larger numbers of primitives in an optimized fashion. We also observed that the largest deflection for response times occurs during high CPU loads. This may be due to the fact that the computing power in these cases was partially used at 100% capacity, which causes unforeseeable system behavior. However, the deflection was always less than 10% of the desired response time, which can be still considered as a close match.

Maximum data volume Figure 7, left illustrates the degree of adaptation for increasing user-defined maximum data volumes (10, 25, 75, and 150kB). The overall data volume was 155kB. All measurements are based on the same display resolution.

The chart clearly indicates that the maximum data volume determined by the proposed adaptation strategy is exact or very close to the stated values. The truncation points are optimal in three of four cases (10, 25, 75kB) and within a margin less than 0.1% in the other case (150kB).

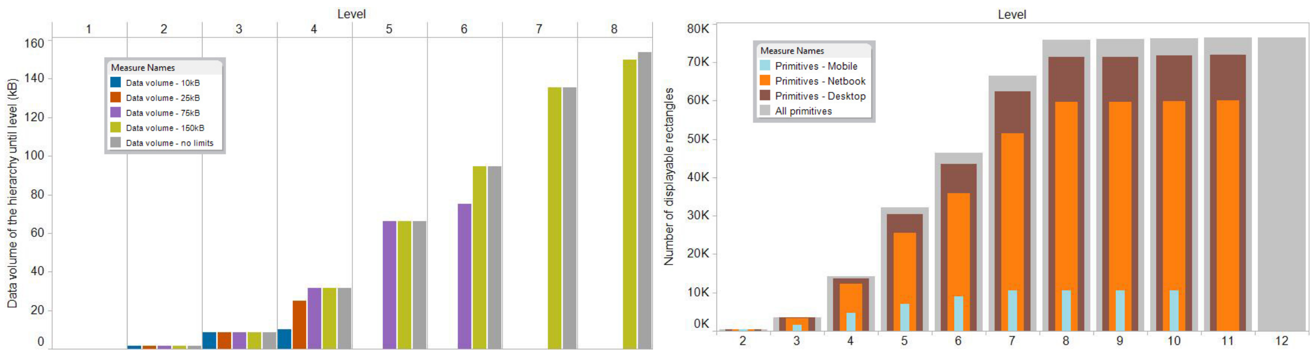

Figure 7. Adaptation regarding the maximum amount of data to be transmitted (left) and the screen resolutions of different viewing devices (right). The left chart does not show levels higher than 8 as the stated limits are reached before for all considered requirement values. The resolutions for the different viewing devices used for the right chart are 480x800 (mobile), 1024x768 (netbook), and 1440x900 (desktop). The right chart does not show the first level of the hierarchy due to the fact that it holds just a single very large primitive.

Minimal primitive size We estimated the degree of adaptation regarding primitive size for multiple resolutions common for modern viewing devices (cf. Figure 7, right). As the screen dimensions of the used mobile phones were sufficiently large, all primitives residing at the second hierarchy level are shown on all viewing devices. First adaptation takes place at the third level. This is due to the fact that the hierarchy contains a number of very small geometry at higher levels. Here, adaptation to the mobile device is strongest and more than half of the associated primitives are omitted. This continues with an increasing divergence until the seventh level, where only a few novel primitives can still be added to the display. On the eleventh and twelfth levels, there are no primitives that can be added without violation of the user requirements. With a less significant effect, this tendency also applies for the other two screen configurations. From level 8 onward, only a few new primitives can be displayed on the netbook or desktop device. No primitives residing at level 12 can be displayed on any of the considered viewing devices. The representations of the mobile device resulting from the applied adaptation are shown in Figure 4, top, for visual evaluation.

6.2 Discussion

Benefits Due to the use of a single predefined progressive data structure that is created once and used multiple times, many limitations of traditional adaptation approaches are overcome. Most of the complexity of the proposed approach is due to the creation of appropriate aggregations and the progressive data structure. As this is done in preprocessing on server side, we avoid the resource constraints common in traditional adaptation systems. It further provides non-redundant data transfer, and allows constraint of the transfer of data required for the adapted view.

The proposed adaptation strategy can be applied to a wide variety of data displays. Data and visualizations with an already defined LOD-hierarchy benefit immediately and require only minor extensions. Aggregate-based hierarchical data structures may provide the necessary progressive property for multi-variate data that is not inherently scalable.[14] Because it is possible to describe even more properties and profiles for a variety of visualization aspects, our approach offers flexibility for application-specific extensions. The proposed adaptation strategy may also be migrated in existing or future visualization systems as it makes use of the traditional visualization pipeline. It is most appropriate if potential viewing devices are known in advance, as the profiles and the progressive data structure can be created in such a way as to provide the most meaningful data aggregations.

Our approach can be broadly applied. As progressive refinement provides and combines data overviews and details-on-demand, it is especially suited for mobile devices. Other possible application areas are smart environments or crisis management centers characterized by central data storage and heterogeneous device ensembles.

Drawbacks The use of a scalable data structure also imposes drawbacks. Our approach strongly depends on many and meaningful levels of aggregation. If data or geometry cannot be scaled, the approach fails. When only a limited amount of hierarchy levels can be determined, the resulting device adaptation is coarse. Due to the fact that often a new progressive data structure must be created if there are changes in the data, the approach loses some of its benefits when applied to rapidly changing data sources. This also applies for client devices that are subject to strong temporal changes in the available capabilities.

7. CONCLUSIONS AND FUTURE RESEARCH

We have presented a novel approach for the adaptation of data presentations to resource constrained viewing devices and user preferences by taking advantage of progressive refinement. The adaptation approach is based on a progressive data structure that is traversed and truncated whenever profile-based resource estimation determines that the client's resources are used up to their full capacity. We introduced general requirements, proposed different options for appropriate creation of a progressive structure, and discussed the content adaptation strategy. We provided general as well as application-specific statements related to the progressive treemap and scatterplot displays. Empirical results we obtained from an implementation of the approach demonstrate that the data displays are still responsive and uncluttered even when dealing with large datasets and low-power devices. Thus, the proposed approach is a major step towards a general solution for device-dependent presentation in visualization.

The introduced adaptation approach considers the resources of the viewing device only and neglects the properties of specific visualization techniques. Thus, it is desirable to enhance the strategy by creating additional profiles to cover other related properties of the presentation task. Future work will also be directed to justify the potential of the general progressive refinement approach in visualization, and to develop more sophisticated scalable compression and transmission schemes as well as visualization techniques.

ACKNOWLEDGMENTS

We, the authors, gratefully acknowledge the support of Deutsche Forschungsgemeinschaft (DFG) for funding this research (#RO3755/1-1).

REFERENCES

[1] Korsah, K., "2009 ca web stress index," tech. rep., Computer Associates International (2009).

[2] Skorin-Kapov, L., Komericki, H., Matijasevic, M., Pandzic, I. S., and Mosmondor, M., "Muva: a flexible visualization architecture for multiple client platforms," *Journal on Mobile Multimedia* **1**(1), 3–17 (2005).

[3] Mao, S., Bushmitch, D., Narayanan, S., and Panwar, S., "Mrtp: A multi-flow real-time transport protocol for ad hoc networks," *IEEE Transactions on Multimedia* **8**, 356–369, (April 2006).

[4] Kim, H. S., Joslin, C., Di Giacomo, T., Garchery, S., and Magnenat-Thalmann, N., "Device-based decision-making for adaptation of three-dimensional content," *Vis. Comput.* **22**(5), 332–345 (2006).

[5] Thiede, C., Schumann, H., and Rosenbaum, R., "On-the-fly device adaptation using progressive contents," in [*Proceedings of Intelligent Interactive Assistance and Mobile Multimedia Computing 2009*], (November 2009).

[6] Tian, D. and Alregib, G., "Batex3: Bit allocation for progressive transmission of textured 3-d models," *IEEE Trans. Circuits Syst. Video Techn.* **18**(1), 23–35 (2008).

[7] Zhou, M. X., Chen, M., and Feng, Y., "Building a visual database for example-based graphics generation," *Information Visualization, IEEE Symposium on* , 23ff (2002).

[8] Novotny, M. and Hauser, H., "Outlier-Preserving Focus+Context visualization in parallel coordinates," *IEEE Transactions on Visualization and Computer Graphics* **12**(5), 893–900 (2006).

[9] Fuchs, G., Reichart, D., Schumann, H., and Forbrig, P., "Maintenance support - case study for a multimodal mobile user interface," in [*Proceedings of Electronic Imaging - Multimedia on Mobile Devices 2006*], (January 2006).

[10] Mkovec, Z., *Spatial data adaptation for interaction in special environments*, PhD thesis, Czech Technical University in Prague (2007).

[11] Funkhouser, T. A. and Squin, C. H., "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments," in [*Proceedings of the 20th annual conference on Computer graphics and interactive techniques*], 247–254, ACM, Anaheim, CA (1993).

[12] Shneiderman, B., "The eyes have it: A task by data type taxonomy for information visualizations," *Proceedings of the IEEE Symposium on Visual Languages* , 336–343 (1996).

[13] Rosenbaum, R. and Hamann, B., "Progressive presentation of large hierarchies using treemaps," in [*Proceedings of 5th International Symposium on Visual Computing 2009*], (November 2009).

[14] Rosenbaum, R. and Schumann, H., "Progressive refinement - more than a means to overcome limited bandwidth," in [*Proceedings of Electronic Imaging - Visualization and Data Analysis 2009*], (January 2009).

[15] Elmqvist, N. and Fekete, J.-D., "Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines," *IEEE Transactions on Visualization and Computer Graphics* **99** (2009).

[16] Shneiderman, B., "Tree visualization with tree-maps: 2-d space-filling approach," *ACM Trans. Graph.* **11**(1), 92–99 (1992).

[17] Guttman, A., "R-trees: a dynamic index structure for spatial searching," in [*Proceedings of the SIGMOD Conference*], 47–57 (June 1984).

[18] Deering, M., "Geometry compression," in [*SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*], 13–20, ACM, New York, NY, USA (1995).

[19] Haber, R. and McNabb, D. A., [*Visualization in Scientific Computing*], ch. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems, 74–93, IEEE (1990).