# Contextual Snapshots: Enriched Visualization with Interactive Spatial Annotations

Peter Mindek[*]
Institute of Computer Graphics and Algorithms
Vienna University of Technology

Stefan Bruckner[†]
Department of Informatics
University of Bergen

M. Eduard Gröller[‡]
Institute of Computer Graphics and Algorithms
Vienna University of Technology

## Abstract

Spatial selections are a ubiquitous concept in visualization. By localizing particular features, they can be analyzed and compared in different views. However, the semantics of such selections are often dependent on other parameter settings and it can be difficult to reconstruct them without additional information. In this paper, we present the concept of contextual snapshots as an effective means for managing spatial selections in visualized data. The selections are automatically associated with the context in which they have been created. Contextual snapshots can be also used as the basis for interactive integrated and linked views, which enable in-place investigation and comparison of multiple visual representations of data. Our approach is implemented as a flexible toolkit with well-defined interfaces for integration into existing systems. We demonstrate the power and generality of our techniques by applying them to several distinct scenarios such as the visualization of simulation data and the analysis of historical documents.

**CR Categories:** I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques

**Keywords:** spatial selections, annotations, linked views, provenance

## 1 Introduction

For the purpose of visual analysis of large datasets, it is often necessary to display various subsets of the examined data using different visualization techniques. Linked views and integrated views are very helpful for the visual analysis in case the examined data are multivariate or otherwise complex. In such cases, the investigation of the data may require a specification of spatial regions which are to be examined individually using integrated or linked views. Creating spatial selections, or brushing, is a widely used method for specifying such regions. Brushing techniques are also often employed to specify a *degree of interest* (DOI) function for focus+context visualization as described by Furnas [Furnas 1986]. Smooth brushing is a term used to describe techniques allowing the specification of a non-binary DOI function, which defines a continuous transition between focus and context data.

[*]mindek@cg.tuwien.ac.at
[†]stefan.bruckner@uib.no
[‡]groeller@cg.tuwien.ac.at

Another aspect of the visualization of complex datasets is a frequent need for creating annotations in the rendered images. The annotations can serve as a mechanism for assigning semantics to parts of the visualization image, for providing additional insight into the data, or for keeping provenance information. Usually, the annotations are meaningful only if the data are displayed in the same way as when the annotations have been created. For instance, if a structure is annotated in a volume dataset, the annotation loses its meaning when the transfer function or the viewing angle changes in such a way that the structure in question is no longer visible. In such cases, it is necessary to keep track of the visualization settings together with the annotations.

In this paper, we propose a method for managing arbitrary spatial selections in the image space of visualizations. We define several terms for the purpose of describing the proposed method. A *selection* is a non-binary DOI function in image space. A *visualization snapshot* is a set of parameter values describing the state of the visualization system at a particular point in time. Finally, we introduce the concept of *contextual snapshots*. A contextual snapshot is an entity which holds multiple selections together with a visualization snapshot. The visualization snapshot provides context for the associated selections. By keeping the selections in contextual snapshots, it is possible to reconstruct the states of the visualization system in which the selections have been created.

Contextual snapshots manage selections in such a way that the algorithm for transforming user input to the DOI function in image space is independent from the underlying application. Just by replacing this algorithm, it is possible to handle different types of user interactions in a common way using contextual snapshots.

To demonstrate the proposed concept, we implemented the contextual snapshots in a volume visualization application. The application displays a multivariate 4D hurricane dataset. The user can create selections in image space by using a lasso metaphor. The selected data can be further analysed using linked views. We use this example to explain which parts contextual snapshots consist of and how they work together.

The contributions of this work are: a concept of contextual snapshots for managing multiple spatial selections in different steps during the visualization session; an implementation of contextual snapshots which provides means for linking additional views to individual selections; a technical evaluation of the implementation using two distinct use cases. In applications such as 4D data visualization, it is sometimes necessary to select and annotate the visualized data multiple times while the visualization parameters are changing. Contextual snapshots provide a basis for keeping track of these interactions. In current systems, selections made in image space have to be processed before the image changes, otherwise the selections would become invalid in the new context. Contextual snapshots record the description of the context together with the selection, so that it can be processed even after the context has changed.

## 2 Related Work

We propose a method for managing spatial selections in image space. There are various scenarios where multiple selections are made in order to achieve a certain goal. The goal might be to select subsets or features of a dataset. Furnas [Furnas 1986] presented DOI functions as a concept for the specification of focus data. Doleisch and Hauser [Doleisch and Hauser 2001] used a DOI function obtained by smooth brushing to modify the visualization mapping in 3D flow visualization. Doleisch et al. [Doleisch et al. 2003] presented a framework for the specification of data features visualized in several linked views. Ulinski et al. [Ulinski et al. 2007] proposed two-handed methods for creating selections in volume rendering. Unger et al. [Unger et al. 2008] used smooth brushing in the visualization of statistical characteristics in subsets of large datasets. Various methods for increasing the usefulness of 3D scatterplots incorporating brushing have been developed [Kosara et al. 2004][Piringer et al. 2004]. Streit et al. [Streit et al. 2012] proposed a model-driven design process for exploring multiple linked datasets. Yu et al. [Yu et al. 2012] proposed methods for selecting data in large 3D point clouds by screen-space interaction. In visualization applications which employ brushing or similar techniques, the user interaction is usually limited to the common context. Contextual snapshots remove this limitation by providing means for keeping the context for each individual interaction instance.

Direct user interaction with the visualization results can also be used for other goals than specifying data subsets. Gerl et al. [Gerl et al. 2012] incorporated brushing on renderings of data attributes for the specification of semantics in volume visualization. Guo et al. [Guo et al. 2011] introduced a sketch-based interface for direct volume rendering which replaces the traditional way of transfer function design. Wei et al. [Wei et al. 2010] proposed a sketch-based interface for an interactive 3D vector field exploration. The concept of contextual snapshots manages spatial selections as entities created by user interaction in image space. However, the concept is designed in such a way that the spatial selections could be employed as metaphors for the mentioned types of user interaction as well. Contextual snapshots provide a potential to increase the scalability of such interaction methods by allowing the management of multiple interaction instances simultaneously, while each instance can be meaningful in different contexts.

In addition to the concept of contextual snapshots, we propose a method for combining them with various views of the visualized data. The integration and linking of multiple views has been extensively explored [Balabanian 2010][Tory 2004]. Bier et al. [Bier et al. 1993] proposed a see-through interface as a natural way of displaying additional data. Balabanian et al. [Balabanian et al. 2008] introduced a framework for the specification of visualization parameters for time-varying data. Using the framework, visualizations of several time steps integrated into one image can be created.

The concept of contextual snapshots can also be used for preserving user-created provenance information for a visualization. Various techniques keeping provenance or describing the visualization process have been developed. Bavoil et al. [Bavoil et al. 2005] proposed VisTrails. It is a system for creating and maintaining visualization pipelines with the possibility to execute them and to record their provenance information. Our method differs form VisTrails in that the user can create spatial selections of the explored data in a specific context and annotate them to store the visualization provenance information. This provides a strong link between the provenance information and the underlying data. Heer et al. [Heer et al. 2008] presented a design space analysis of history keeping systems. Kreuseler et al. [Kreuseler et al. 2004] proposed an approach to include a history mechanism into a visual data mining framework. Groth and Streefkerk [Groth and Streefkerk 2006] presented a method for capturing the history of the knowledge discovery process using a visualization system with an ability to create annotations for provenance information. Ellkvist et al. [Ellkvist et al. 2009] presented an architecture for provenance interoperability between multiple sources. In contrast to these systems, contextual snapshots provide means to insert annotations to particular spatial data in a particular stage of the visualization session. As the annotations are automatically linked with the current context, they store provenance information besides their actual content.

Santos et al. [Santos et al. 2009] proposed VisMashup, a framework for simplifying the creation of custom visualization applications. The authors of VisMashup focused on the ability to combine various visualization pipelines to create a new visualization application. In our work, we aim at extending existing visualization pipelines with interaction possibilities, such as creating selections or interactive annotations.

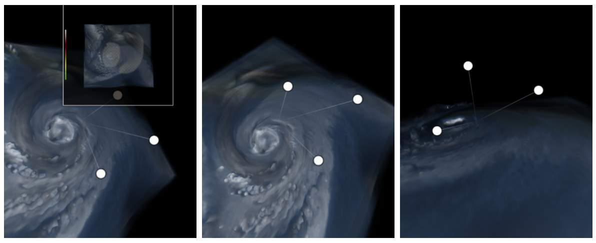
## 3 Overview of Contextual Snapshots

Many visualization systems use brushing, selections, and linked views to provide means for exploration of complex datasets. There are various tools for specifying the selections and they usually serve only one specific purpose. The idea of contextual snapshots is to harness a single mechanism of 2D spatial selections for different tasks, such as data selection and manipulation, data annotation, or specification of DOI functions. In contextual snapshots, this is achieved by the following concepts: multiple selections can be stored and each of them can be created in a different context; an algorithm of transforming the user input to the DOI functions of the selections is interchangeable; each selection managed by contextual snapshots can be linked with a number of additional views which we refer to as *embedded visualizations*. It is possible to display them directly in the visualization image, hence the name embedded visualizations. Embedded visualizations are interactive and they can display arbitrary GUI elements or visualize data specified by a respective selection.

In the example application, the selections can specify a spatial region in image space. An embedded view which displays the histogram of the data in the specified region is linked with each selection. Contextual snapshots provide data affected by the selection to the embedded visualization. It depends on the implementation of individual embedded visualizations how these data are used. Another embedded visualization is a simple text field. It does not display the selected data region, but it allows users to type in arbitrary text-based annotations. Such an embedded visualization is linked with every selection, thus providing a possibility to annotate selected data subsets.

### 3.1 Concept of Contextual Snapshots

A visualization system may apply various parameters to modify the visual mapping. In terms of contextual snapshots, the values of a chosen subset of said parameters, the *visualization snapshot*, define a state of the visualization system. For any state of the system, a user can create several selections in the rendered images. A contextual snapshot stores a visualization snapshot together with all selections created when the state of the visualization system corresponded to this visualization snapshot. In the example application, we use the position and the orientation of the virtual 3D camera as the description of the visualization snapshot. Each image-space se-



(a) (b) (c)

Figure 1: Interactive anchors (white circles) representing individual contextual snapshots. For better 3D orientation, the anchors are connected with the coordinate origin by a thin line. (a) shows how the thumbnail of an anchor can be displayed. (b) and (c) show the anchors from different camera views.

lection is bound to one 3D camera view. The selection is valid only if the volume is rendered using this camera view.

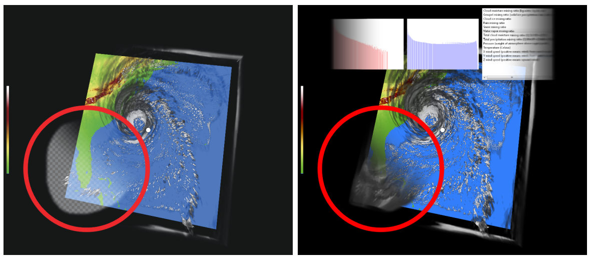
The idea of contextual snapshots is based on the fact that the semantics of the user-made selections depends on what is currently displayed. When the user selects a particular feature in the image, the selection is meaningful only until the way how the feature is displayed changes. Therefore, we extract a visualization snapshot every time a new selection is created. The visualization snapshot is linked to the selection to create a contextual snapshot. The contextual snapshot then provides a reproducible spatial selection related to what was displayed when the selection was made. It stores the appropriate visualization context in the form of the values of the visualization-system parameters.

The strength of contextual snapshots is that they can maintain multiple selections created during different steps of the visualization session. Each contextual snapshot can potentially contain multiple selections created in a common state of the visualization system. The information stored within a contextual snapshot can be used to restore this state, so that the selections can be displayed and actively used. By restoring the state of the visualization system according to the individual contextual snapshots, it is possible to browse all selections created within a visualization session.

Contextual snapshots are represented by icons which we refer to as anchors. The anchors are embedded in the original visualization as interactive graphical annotations. The position of an anchor represents the visualization context stored within the respective contextual snapshot. For instance, in a volume rendering application, the anchor can be positioned in 3D space to represent the camera position when the respective contextual snapshots has been taken. The anchor can also display a thumbnail of how the visualization looked like when the respective contextual snapshots has been created. In addition to an informative purpose, the anchors are also interactive and they can be used to restore the visualization-system state to the respective contextual snapshot. They serve as a user interface for browsing through the contextual snapshots created during the visualization session. Figure 1 shows the graphical representation of anchors.

## 3.2 Embedded Visualizations as Linked Views

To broaden the possibilities of using context-aware selections, we introduce a method for linking interactive embedded visualizations for each selection. The additional visualizations can show different aspects of the selected data, or they can display comparisons of various selected areas. To demonstrate different ways how the selections can be used, we implemented the following embedded visualizations for the hurricane visualization application: a histogram of selected data values; a text-based annotation; a variable picker.



(a) (b)

Figure 2: (a) A selection (marked with the red circle). (b) An integrated view of two variables using the selection after its activation. Histograms are shown for both variables as embedded visualizations.

The embedded visualization displaying the histogram of selected data values can be linked to multiple selections at once. In this case, histograms for individual selections are displayed in overlays so that they can be easily compared. The text-based annotation is linked to each selection and it provides for the user the possibility to type in a short description of the selected data subset. The variable picker is a GUI element which displays a list of all variables present in the dataset. The chosen variable is displayed in those parts of the image, where the selection has been created. Figure 2 shows how the picked variable is integrated with the rest of the visualization by a smooth transition. The smooth transition is due to the non-binary DOI function of the selection.

In our method, each spatial selection can be linked with multiple embedded visualizations. Embedded visualizations are interactive visualization pipelines with access to the data subsets specified by selections with which they are linked. A single embedded visualization can be reused to display aspects of different subsets of the explored data by simply linking it with different spatial selections.

For the purpose of displaying the embedded visualizations, we propose a mechanism for activating individual selections. One or multiple selections can be activated by the user at once. In this case, only those embedded visualizations are displayed which are linked with every activated selection. The rationale for this mechanism is that the embedded visualizations can show different aspects of the data specified by multiple selections at once. This way, the selections and their embedded visualizations can be used to compare several data subsets.

A sketch-based interface is used for activating selections. The user activates selections by making a stroke in image space. The selections which are crossed by the stroke are activated and subsequently their linked embedded visualizations are displayed. The embedded visualizations are grouped together in a sliding bar, which is displayed either on the top, the bottom, the left, or the right side of the visualization image.

The position of the sliding bar is determined by the direction at the end of the stroke which was used to activate the selections. The interaction method of using a stroke was chosen so that an arbitrary subset of the selections can be activated, which might be difficult with various standard selection mechanisms such as draggable rectangles. By creating the stroke which activates the selections, the users can also immediately point the system to where they would like to have the embedded visualizations displayed.

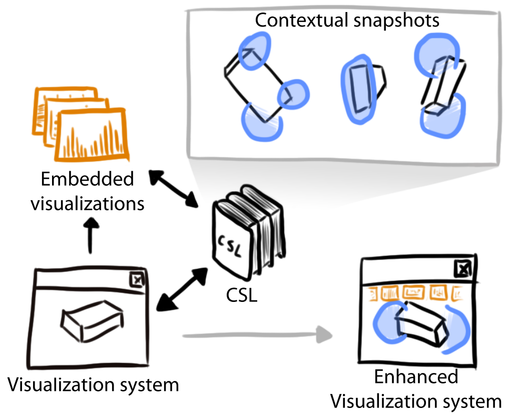


Figure 3: Overview of the system. The black arrows represent the data flow between the visualization system, the embedded visualizations and the Contextual Snapshot Library (CSL). The gray arrow denotes the transition from the original rendering to the rendering of the enhanced visualization.

The sliding bar is capable of showing several embedded visualizations at once. In case there are more embedded visualizations for the activated selections than could actually fit on the screen, the sliding bar enables scrolling of its content. The scrolling is executed by an animated transition, so that the users have a visual feedback on the direction of the scrolling. The sliding bar fades to the original visualization on both sides for better integration. A gradual blurring filter was used on both sides of the sliding bar, so that the attention of the users is guided to the embedded visualizations currently shown in the middle of the bar.

Showing more embedded visualizations at once provides users with an overview of the additional data displayed for the activated selections. However, for the interaction it might be sometimes necessary to enlarge individual embedded visualizations. For this purpose, the embedded visualization displayed in the middle of the sliding bar can be switched to a so called maximized view. If the view is maximized, the embedded visualization is displayed on the whole screen rather than just in the sliding bar.

### 3.3 Visualization System Enrichment

Contextual snapshots are a concept meant to be implemented in an underlying visualization system. All interaction and feedback related to the contextual snapshots is realized through the visualization output of the underlying system. Contextual snapshots are visually represented by the anchors which are placed in the visualization result produced by the system. With the implementation of the concept of contextual snapshots, various views of the data and the interactive interface become integrated. This approach is particularly well-suited for the rapidly growing area of mobile devices such as tablets where the display also serves as the input device.

We have implemented the contextual snapshots as a library which can be included into an existing visualization system. We call it Contextual Snapshot Library (CSL). The CSL is responsible for rendering the anchors, the selections, and the embedded visualizations into the original visualization image. Additionally, it provides an interface for the data transfer between the selections and the embedded visualizations and it mediates user input so that the anchors and embedded visualizations are interactive.

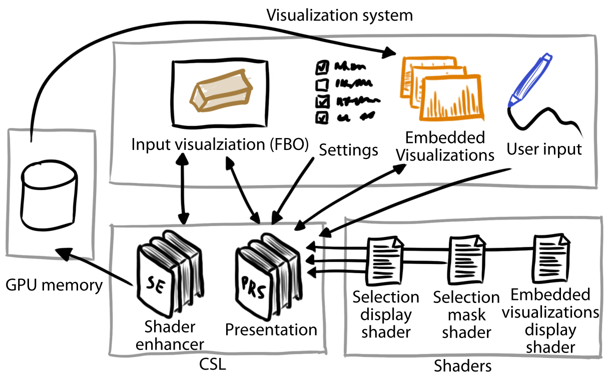


Figure 4: Architecture of the visualization system integrating the CSL. The arrows denote the data flow.

Figure 3 shows the data flow between the visualization system, the embedded visualizations, and the CSL. The visualization system gathers user input and transmits it to the CSL. The CSL stores contextual snapshots generated from the user input and it renders the selections, the anchors, and the embedded visualizations into the original visualization image. The result is an enhanced visualization system.

## 4 Architecture of the CSL

The goal of the CSL is to provide means for including the concept of the contextual snapshots into existing visualization pipelines. This integration is carried out on the source-code level through an application programming interface (API).

The input to the CSL is an image of the visualization - the *input image*. It is continuously provided in the standard rendering loop of the system. The output is an image of the visualization, in which the graphical annotations are included - the *output image*. The visualization system can display the output image instead of the original image of the visualization. The annotations inserted into the input image are the anchors of the contextual snapshots, spatial selections, and the embedded visualizations, as illustrated in figures 1 and 2.

Figure 4 shows the overall architecture of an existing visualization system using the CSL. The library itself is split into two parts. The part responsible for managing the contextual snapshots, interaction, and rendering of the annotations, is called Presentation (PRS). To exploit capabilities of modern GPUs, it uses several shaders for rendering of the annotations. These are: the selection mask shader, which transforms user input into the DOI function of the selection; the selection display shader, which renders the selection on the screen; the embedded visualizations display shader, which renders the sliding bar. Each of these shaders can be exchanged to modify how selections are treated.

The functionality of the PRS can be extended by the Shader Enhancer (SE). The SE is an auxiliary tool for the data transfer between individual modules of the visualization system. It stores data specific to individual selections of the contextual snapshots in the GPU memory so that it can be used in different visualization pipelines of the system. The SE simplifies the utilization of the selections in the visualization system by providing access to all data subsets specified by their DOI functions. The motivation of storing the data in the GPU memory is that the GPU implementations

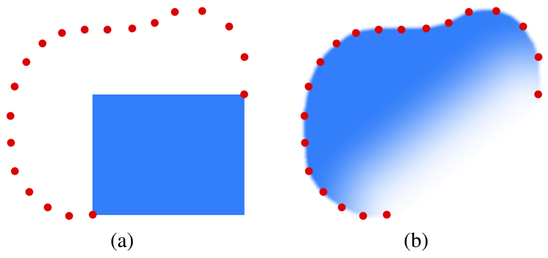


Figure 5: Two different selection mask shaders. The red dots illustrate the selection stroke, the blue area is the generated selection mask. Blue color means fully selected, white color means not selected at all. Notice the gradual change in the level of selection, which enables smooth brushing.

of visualization algorithms can access the data without having to transfer them to to CPU memory.

## 5 Implementation

The CSL is implemented in C++, using the Qt library. It uses OpenGL for rendering of the annotations and for the data transfer with the visualization system via textures. The input image is provided as a framebuffer object (FBO). The CSL creates its own OpenGL context and uses it to render the input FBO. Subsequently, it renders all the annotations on top of it. The output is also an FBO, which the CSL uses as a rendering target.

The CSL utilizes Qt's signal/slot mechanism. If the visualization system hosting the CSL uses Qt as well, it can implement actions reacting to various events of the CSL employing this mechanism. The signal/slot mechanism together with the API are the means of transferring data between the visualization system and the CSL.

The contextual snapshots can be stored on the hard drive in the XML format. The anchor screenshots and the selection masks are stored in PNG files. CSL also provides functionality to load this information and recreate all the annotations for the currently visualized data.

### 5.1 Contextual Snapshots

The anchor is an interactive element meant for changing the current state of the visualization system to what is represented by the respective contextual snapshot. Anchors can be perceived as clickable bookmarks for the contextual snapshots. The location of an anchor is an important visual cue. It can be positioned with respect to a feature of the visualization to potentially reveal the semantics or content of the contextual snapshot.

The underlying visualization system can make a request to select an anchor displayed on a certain position in image space. The contextual snapshot represented by the selected anchor becomes active. The state of the visualization system is changed so that it corresponds to the active contextual snapshot. All selections belonging to this contextual snapshot can now be displayed.

The activation of the contextual snapshot is accompanied by a time-varying interpolation between the values of the parameters it stores and the current values in the present state of the visualization system. For individual parameters, different interpolation functions can be used in order to achieve an appropriate interpolation for a

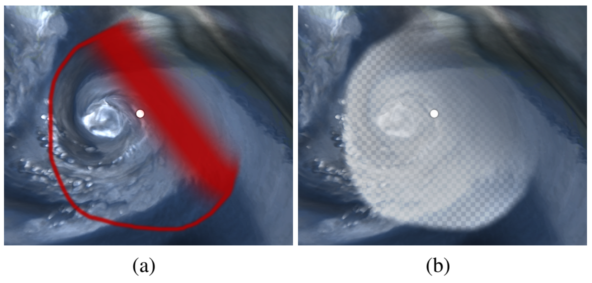


Figure 6: Two different selection display shaders. Both shaders are able to show fuzziness of the selection. Each of them might be suitable for different applications.

specific data type (e.g., Slerp for rotation matrices). This enables smooth transitions between system states while switching between them. In our example application, the state of the visualization system is defined by the camera position and orientation. Therefore, the activation of a contextual snapshot causes the visualization system to smoothly change the 3D camera to the view in which the contextual snapshot was recorded.

The position of the anchor in image space can be set manually through the API of the CSL. We also provide the possibility to assign a transformation matrix to every contextual snapshot. The anchor is then automatically positioned in the image using this transformation. In case of a 3D visualization, the anchor can be assigned the same transformation matrix as the visualized object. In this case, the relative position of the anchor to the visualized object would be constant even if the camera view changes. This gives the user an idea from which angle the contextual snapshot represented by the anchor was taken.

### 5.2 Selections

Selections are created by calling functions of the CSL's API. If a contextual snapshot is active, the created selection is automatically assigned to it. Otherwise a new contextual snapshot is created from the current values of the visualization-system parameters and the selection is assigned to it. This mechanism enables multiple selections to be assigned to a single contextual snapshot.

The process of creating a selection consists of three steps: recording of the user input, transforming the input to a DOI function in image space, displaying the DOI function on the screen as the selection. In our approach, we made a clear separation between these steps. Each of them is implemented as a stand alone shader program with clearly defined input and output. Because of this separation, it is possible to customize the process of creating selections for various applications, such as selections using the lasso metaphor or rectangular selections. In both cases, the only component of the CSL that is exchanged is the shader realizing the transformation of the user input to the DOI function.

Internally, a selection is composed of a series of points in the image space of the visualization. We refer to this series as a selection stroke. The selection stroke can be sketched by the users employing mouse, tablet, or a similar input device. The visualization system can continuously provide the coordinates of input points until the creation of the selection is finished. The CSL then generates an image of a grey-scale mask representing the non-binary DOI function, where the pixel luminosity encodes the degree of interest in the point. We refer to this image as selection mask. The selection mask is generated by a selection mask shader. The input of the shader program is the selection stroke encoded in a one-dimensional texture. The output of the shader program is the image of the selection mask.

Depending on the shader program used to generate the selection masks, the selections can enable the visualization system to realize smooth brushing abilities. We provide two different shader programs for generating the selection masks. Figure 5 shows how the selection mask generated by these two shaders would look like for the same selection stroke. The shader illustrated in Figure 5(a) creates a simple rectangle based on the first and the last point of the stroke. This constitutes a common way how the user input is transformed into a rectangular spatial selection. The shader shown in Figure 5(b) uses so called lasso metaphor. The points of the stroke define a closed polygon whose interior is filled. We enhance the lasso metaphor to account for the selection uncertainty by introducing smoothing of the edges. The smoothing between the first and the last point of the stroke is stronger if these two points are further away. The rest of the edges is smoothed by a constant factor. This enables users to control the amount of smoothing as well as its spatial location. The smoothing corresponds to the uncertainty in the specification of the area of interest.

The selections are displayed on the screen using the selection display shader. The way how the selections are displayed is important for specific applications. Figure 6 shows two different ways how the selections can be displayed. In Figure 6(a), an example of outline drawing is shown. The red color marks borders between selected and unselected areas. The thickness of the border denotes the fuzziness of the selection in that particular area. In Figure 6(b), the selection is displayed using an overlay texture. This method shows the selected area in a clear way, but it also partially occludes displayed data underneath.

### 5.3 Data transfer

The selection masks are stored in the GPU memory as a texture array so that the visualization system can access them at all times and employ them in the visual mapping. We used this ability in the hurricane visualization example to create a smooth transition between the two visualized variables.

The SE can be used to further increase possibilities of the selections. It allows the visualization algorithm to extract processed data samples to the GPU memory. Every extracted data sample is automatically weighted by the degree of interest specified by the activated selections. The extracted data can be then accessed in the embedded visualizations.

Currently, we provide an implementation of the SE for GLSL shaders. The SE inserts GLSL code for the extraction of data samples and their weighting by the selections' DOI functions in the visualization shader before it is compiled. The inserted code uses atomic operations and the GL_EXT_shader_image_load_store extension to output desired data. After the extraction, the data are available to the embedded visualizations as a texture stored in the GPU memory. We plan to extend the SE for other languages in the future.

## 6 Application Example - Historical Document Analysis

In addition to the example application introduced in section 1, we present another use case for contextual snapshots. The visualization system taken in this use case is a simple book reader application. A page spread consisting of two pages of a manuscript is displayed. A bar showing the current page within the whole manuscript is located below the pages. For a better user experience, a simple page turning animation is realized whenever the current page changes.

We combine our method with the described book reader application in order to add functionality enabling the users to employ it as an advanced manuscript analysis tool. We used the CSL to implement spatial selections on the displayed pages. The CSL automatically binds every new selection with the current page, so many selections on different pages can be created.

### 6.1 Manuscript Visualization

The dataset taken in this example consists of 723 high resolution photographs of pages of the Venetus A, a tenth century (AD) manuscript of the Iliad catalogued as Marcianus Graecus Z. 454, now 822. In addition to natural light photographs, some of the pages were recorded using UV photography as well. The UV light photographs were taken in order to reveal some details of the manuscript which were hardly visible with natural light. Together with the photographs, the transcript of the Iliad in ancient Greek was available as well.

For demonstration purposes, we manually preprocessed the acquired data. Few of the UV photographs were registered with the natural light photographs so that they could be easily used in the application. Additionally, appropriate passages of the transcript were matched with some of the photographed pages.

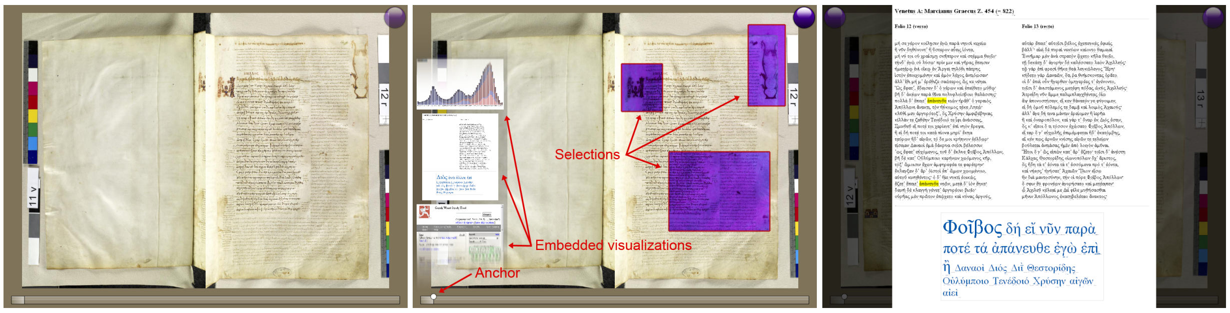
The described book reader application is only capable of displaying the natural light photographs. The book reader shows an icon for those pages where the UV data is available. The goal in this example is to implement the ability to display parts of the UV photographs on selected regions of interest. The regions of interest are rectangular areas selected for the pages with the available UV data. For this purpose, we integrated the CSL into the book reader application. In this way, the desired functionality can be realized by the possibilities offered by the CSL.

### 6.2 Manuscript Visualization Enhancement

Figure 7(a) shows the book reading application without the enhancements. Figure 7(b) shows the application enhanced using the CSL. The contextual snapshots were used to manage the selections. The selections are employed to show parts of the UV photograph of the page. The transcript of the page as well as color histograms of the selected parts of the natural light and UV photographs are displayed on the left side of the image. These additional visualizations are linked with the activated selections. The activated selections are concurrently used to display parts of the registered UV photograph of the selected page. Any of the additional visualizations can be maximized, as shown in figure 7(c).

The only parameter with an impact on the manuscript visualization is the index of the current page. This parameter determines which pages of the manuscript are displayed. As the turning of the pages is animated, we allow the current page index to be a real number. The fractional part of the index is used for the animation of the page turning.

Parts of the displayed pages can be selected. The selection is meaningful only for the page in which it was created. The current page index constitutes the visualization snapshot for this application, because it alone fully describes the context for the selections. The



**Venetus A: Marcianus Graecus Z. 454 (= 822)**

**Folio 12 (verso)**

μή σε γέρον κοίλῃσιν ἐγὼ παρὰ νηυσὶ κιχείω
ἢ νῦν δηθύνοντ᾽ ἢ ὕστερον αὖτις ἰόντα,
μή νύ τοι οὐ χραίσμῃ σκῆπτρον καὶ στέμμα θεοῖο·
τὴν δ᾽ ἐγὼ οὐ λύσω· πρίν μιν καὶ γῆρας ἔπεισιν
ἡμετέρῳ ἐνὶ οἴκῳ ἐν Ἄργεϊ τηλόθι πάτρης
ἱστὸν ἐποιχομένην καὶ ἐμὸν λέχος ἀντιόωσαν·
ἀλλ᾽ ἴθι μή μ᾽ ἐρέθιζε σαώτερος ὥς κε νέηαι.
Ὣς ἔφατ᾽, ἔδεισεν δ᾽ ὁ γέρων καὶ ἐπείθετο μύθῳ·
βῆ δ᾽ ἀκέων παρὰ θῖνα πολυφλοίσβοιο θαλάσσης·
πολλὰ δ᾽ ἔπειτ᾽ ἀπάνευθε κιὼν ἠρᾶθ᾽ ὁ γεραιὸς
Ἀπόλλωνι ἄνακτι, τὸν ἠΰκομος τέκε Λητώ·
κλῦθί μευ ἀργυρότοξ᾽, ὃς Χρύσην ἀμφιβέβηκας
κίλλάν τε ζαθέην Τενέδοιό τε ἶφι ἀνάσσεις,
Σμινθεῦ εἴ ποτέ τοι χαρίεντ᾽ ἐπὶ νηὸν ἔρεψα,
ἢ εἰ δή ποτέ τοι κατὰ πίονα μηρί᾽ ἔκηα
ταύρων ἠδ᾽ αἰγῶν, τό δε μοι κρήηνον ἐέλδωρ·
τίσειαν Δαναοὶ ἐμὰ δάκρυα σοῖσι βέλεσσιν.
ὡς ἔφατ᾽ εὐχόμενος, τοῦ δ᾽ ἔκλυε Φοῖβος Ἀπόλλων,
βῆ δὲ κατ᾽ Οὐλύμποιο καρήνων χωόμενος κῆρ,
τόξ᾽ ὤμοισιν ἔχων ἀμφηρεφέα τε φαρέτρην·
ἔκλαγξαν δ᾽ ἄρ᾽ ὀϊστοὶ ἐπ᾽ ὤμων χωομένοιο,
αὐτοῦ κινηθέντος· ὁ δ᾽ ἤϊε νυκτὶ ἐοικώς.
ἕζετ᾽ ἔπειτ᾽ ἀπάνευθε νεῶν, μετὰ δ᾽ ἰὸν ἕηκε·
δεινὴ δὲ κλαγγὴ γένετ᾽ ἀργυρέοιο βιοῖο·
οὐρῆας μὲν πρῶτον ἐπῴχετο καὶ κύνας ἀργούς,

**Folio 13 (recto)**

αὐτὰρ ἔπειτ᾽ αὐτοῖσι βέλος ἐχεπευκὲς ἐφιεὶς
βάλλ᾽· αἰεὶ δὲ πυραὶ νεκύων καίοντο θαμειαί.
Ἐννῆμαρ μὲν ἀνὰ στρατὸν ᾤχετο κῆλα θεοῖο,
τῇ δεκάτῃ δ᾽ ἀγορὴν δὲ καλέσσατο λαὸν Ἀχιλλεύς·
τῷ γὰρ ἐπὶ φρεσὶ θῆκε θεὰ λευκώλενος Ἥρη·
κήδετο γὰρ Δαναῶν, ὅτι ῥα θνήσκοντας ὁρᾶτο.
οἱ δ᾽ ἐπεὶ οὖν ἤγερθεν ὁμηγερέες τ᾽ ἐγένοντο,
τοῖσι δ᾽ ἀνιστάμενος μετέφη πόδας ὠκὺς Ἀχιλλεύς·
Ἀτρεΐδη νῦν ἄμμε παλιμπλαγχθέντας οἴω
ἂψ ἀπονοστήσειν, εἴ κεν θάνατόν γε φύγοιμεν,
εἰ δὴ ὁμοῦ πόλεμός τε δαμᾷ καὶ λοιμὸς Ἀχαιούς·
ἀλλ᾽ ἄγε δή τινα μάντιν ἐρείομεν ἢ ἱερῆα
ἢ καὶ ὀνειροπόλον, καὶ γάρ τ᾽ ὄναρ ἐκ Διός ἐστιν,
ὅς κ᾽ εἴποι ὅ τι τόσσον ἐχώσατο Φοῖβος Ἀπόλλων,
εἴ ταρ ὅ γ᾽ εὐχωλῆς ἐπιμέμφεται ἠδ᾽ ἑκατόμβης,
αἴ κέν πως ἀρνῶν κνίσης αἰγῶν τε τελείων
βούλεται ἀντιάσας ἡμῖν ἀπὸ λοιγὸν ἀμῦναι.
Ἤτοι ὅ γ᾽ ὣς εἰπὼν κατ᾽ ἄρ᾽ ἕζετο· τοῖσι δ᾽ ἀνέστη
Κάλχας Θεστορίδης οἰωνοπόλων ὄχ᾽ ἄριστος,
ὃς ᾔδη τά τ᾽ ἐόντα τά τ᾽ ἐσσόμενα πρό τ᾽ ἐόντα,
καὶ νήεσσ᾽ ἡγήσατ᾽ Ἀχαιῶν Ἴλιον εἴσω
ἣν διὰ μαντοσύνην, τήν οἱ πόρε Φοῖβος Ἀπόλλων·
ὅ σφιν ἐῢ φρονέων ἀγορήσατο καὶ μετέειπεν·
ὦ Ἀχιλεῦ κέλεαί με Διῒ φίλε μυθήσασθαι
μῆνιν Ἀπόλλωνος ἑκατηβελέταο ἄνακτος·

Φοῖβος δή εἴ νῦν παρὰ ποτέ τά ἀπάνευθε ἐγὼ ἐπὶ ἤ Δαναοὶ Διός Διῒ Θεστορίδης Οὐλύμποιο Τενέδοιό Χρύσην αἰγῶν αἰεὶ

(a) (b) (c)

Figure 7: (a) The book reader application displaying the Venetus A manuscript. The icon on the top right corner indicates the availability of UV light data for this particular page. (b) The book reader application enhanced with the CSL to show additional data. An anchor of a contextual snapshot, selections, and embedded visualizations are displayed. (c) The transcript of the displayed pages with the tag cloud in the maximized view. A word selected in the tag cloud is highlighted in yellow.

current page index is visualized by the book reader as a slider displayed below the pages. The anchor of each contextual snapshot is placed on the slider according to the current page index. The anchor therefore visually represents the position of the displayed page spread and can be used as a bookmark.

For this use case, we employed rectangular selections which are displayed as boundaries of the selection masks. A simple edge detection filter was used as the selection display shader for this purpose. The selections can be taken to visually analyse interesting parts of the photographs. By clicking the UV icon, displaying of the UV light photographs in the selections can be enabled or disabled. As the UV and natural light photographs are co-registered, the selections create a comprehensible integrated view.

We have used three web-views displaying web pages as embedded visualizations. By using web pages, we demonstrate that the embedded visualizations managed by the CSL are interactive and they can contain arbitrary content. The first web-view shows color histograms from the selections. As multiple selections can be activated, we used a JavaScript library D3 [Bostock et al. 2011] capable of showing multiple histograms at once. For each selection, a color histogram is displayed. For the pages where the UV data are available, the histograms from the UV photographs are displayed as well. As all of the histograms are depicted in one view, they can be easily compared.

The second web-view shows the Greek transcript of the displayed pages. Contextual information of the selections, the current page index, is used to load appropriate pages from the transcript. A tag cloud of the most frequent words generated by the JavaScript is displayed below the text.

The third web-view shows the web page of the Perseus Word Study Tool [Mahoney 2001]. This web application provides an English translation of a specified Greek word, as well as further information. We have connected this view with the Greek transcript of the displayed pages. The user can double-click on any word in the transcript to automatically show its definition in the Perseus Word Study Tool.

The application of the CSL in the book reader example shows various ways how the contextual snapshots can be used to integrate different views of the visualized data. This use case demonstrates several types of annotations which can be helpful in analysing the historical manuscript. It shows that the integration of vastly distinct visualization techniques including online content and GPU-based rendering is easily possible with our approach.

## 7 Discussion

The goal of our work is to introduce a general concept for handling spatial selections created in different contexts during a visualization session. Instead of realizing a new standalone system, we have decided to implement this concept as a flexible toolkit that can be used in existing systems. We support this decision by showing how the proposed concept of the contextual snapshots can be implemented using the CSL in visualization systems from different areas.

In section 6 we show an example how the concept of contextual snapshots can be employed. State-of-the-art visualization systems usually treat selections in such a way that it is necessary to use several linked views to work with multiple selections simultaneously. To use selections as interactive annotations, each selection would have to be assigned a separate view, possibly in a separate window. Contextual snapshots allow to use multiple selections in the same view while the changes of the visualization are automatically tracked.

The selections can be displayed only if the parameter values of the visualization system are the same as when the particular selection was created. Even a small change to a parameter value can result in a substantial change of the visualization. In the future, ways how to treat parameter changes resulting in very small changes of the visualization could be explored. It is possible that such changes do not deactivate an active contextual snapshot.

By using the CSL to render the selections, the anchors, and the embedded visualizations, the performance of the rendering dropped from 60 FPS to 30 FPS. The performance drop of the whole system mainly depends on the temporal requirements of the embedded visualizations. This aspect can be improved in the future by parallelizing the rendering of individual embedded visualizations.

## 8 Conclusion

We proposed a method for enhancing existing visualization systems with interactive annotations. We have given examples to create interactive annotations which can aid users with various tasks, such as analysing a historical manuscript or multivariate weather simulation data. We introduced the novel concept of contextual snapshots, which provide a comprehensive means to manage spatial selections, to highlight interesting regions in visualizations, to display additional views for selected data, or to compare different regions.

Our method is meant to be applied to visualization systems where the state changes over the period of the visualization session. Most of the interactive systems meet this description. In our method, the user-made selections in image space are linked with all necessary contextual information so that they remain meaningful during the whole visualization session. The contribution of our method is that it can be used to implement a variety of interactions involving spatial selections of the visualized data.

## Acknowledgments

The work presented in this paper has been partially supported by the ViMaL project (FWF - Austrian Research Fund, no. P21695) and by the Aktion OE/CZ grant number 64p11. The Venetus A dataset is courtesy of The Homer Multitext Project. The hurricane Isabel dataset is produced by the Weather Research and Forecast (WRF) model, courtesy of NCAR and the U.S. National Science Foundation (NSF).

## References

BALABANIAN, J.-P., VIOLA, I., MÖLLER, T., AND GRÖLLER, E. 2008. Temporal styles for time-varying volume data. In *Proceedings of 3DPVT'08 - the Fourth International Symposium on 3D Data Processing, Visualization and Transmission*, S. Gumhold, J. Kosecka, and O. Staadt, Eds., 81–89.

BALABANIAN, J.-P. 2010. *Multi-Aspect Visualization: Going from Linked Views to Integrated Views*. PhD thesis, Dept. of Informatics, Univ. of Bergen, Norway.

BAVOIL, L., CALLAHAN, S. P., CROSSNO, P. J., FREIRE, J., AND VO, H. T. 2005. Vistrails: Enabling interactive multiple-view visualizations. In *IEEE Visualization 2005*, 135–142.

BIER, E. A., STONE, M. C., PIER, K., BUXTON, W., AND DEROSE, T. D. 1993. Toolglass and magic lenses: the see-through interface. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '93, 73–80.

BOSTOCK, M., OGIEVETSKY, V., AND HEER, J. 2011. D3: Data-driven documents. *IEEE Transactions on Visualization and Computer Graphics 17*, 12, 2301–2309.

DOLEISCH, H., AND HAUSER, H. 2001. Smooth brushing for focus+context visualization of simulation data in 3d. In *Journal of WSCG*, 147–154.

DOLEISCH, H., GASSER, M., AND HAUSER, H. 2003. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the symposium on Data visualisation 2003*, Eurographics Association, Aire-la-Ville, Switzerland, VISSYM '03, 239–248.

ELLKVIST, T., KOOP, D., FREIRE, J., SILVA, C., AND STRÖMBÄCK, L. 2009. Using mediation to achieve provenance interoperability. In *Proceedings of the 2009 Congress on Services - I*, IEEE Computer Society, Washington, DC, USA, SERVICES '09, 291–298.

FURNAS, G. W. 1986. Generalized fisheye views. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, New York, NY, USA, CHI '86, 16–23.

GERL, M., RAUTEK, P., ISENBERG, T., AND GRÖLLER, E. 2012. Semantics by analogy for illustrative volume visualization. *Computers & Graphics 36*, 3, 201–213.

GROTH, D. P., AND STREEFKERK, K. 2006. Provenance and annotation for visual exploration systems. *IEEE Transactions on Visualization and Computer Graphics 12*, 6 (Nov.), 1500–1510.

GUO, H., MAO, N., AND YUAN, X. 2011. Wysiwyg (what you see is what you get) volume visualization. *IEEE Transactions on Visualization and Computer Graphics 17*, 2106–2114.

HEER, J., MACKINLAY, J., STOLTE, C., AND AGRAWALA, M. 2008. Graphical histories for visualization: Supporting analysis, communication, and evaluation. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov.), 1189–1196.

KOSARA, R., SAHLING, G. N., AND HAUSER, H. 2004. Linking scientific and information visualization with interactive 3d scatterplots. In *Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 133–140.

KREUSELER, M., NOCKE, T., AND SCHUMANN, H. 2004. A history mechanism for visual data mining. In *Proceedings of the IEEE Symposium on Information Visualization*, IEEE Computer Society, Washington, DC, USA, INFOVIS '04, 49–56.

MAHONEY, A. 2001. Studying the word study tool. *New England Classical Journal 28*, 3, 181–183.

PIRINGER, H., KOSARA, R., AND HAUSER, H. 2004. Interactive focus+context visualization with linked 2d/3d scatterplots. In *Proceedings of the Second International Conference on Coordinated & Multiple Views in Exploratory Visualization*, IEEE Computer Society, Washington, DC, USA, CMV '04, 49–60.

SANTOS, E., LINS, L., AHRENS, J., FREIRE, J., AND SILVA, C. 2009. Vismashup: Streamlining the creation of custom visualization applications. *IEEE Transactions on Visualization and Computer Graphics 15*, 1539–1546.

STREIT, M., SCHULZ, H.-J., LEX, A., SCHMALSTIEG, D., AND SCHUMANN, H. 2012. Model-driven design for the visual analysis of heterogeneous data. *IEEE Transactions on Visualization and Computer Graphics 18*, 998–1010.

TORY, M. 2004. *Combining two-dimensional and three-dimensional views for visualization of spatial data*. PhD thesis, Burnaby, BC, Canada.

ULINSKI, A. C., ZANBAKA, C. A., WARTELL, Z., GOOLKASIAN, P., AND HODGES, L. F. 2007. Two handed selection techniques for volumetric data. In *IEEE Symposium on 3D User Interfaces*, 26.

UNGER, A., MUIGG, P., DOLEISCH, H., AND SCHUMANN, H. 2008. Visualizing statistical properties of smoothly brushed data subsets. *Proceedings of the 12th Intern. Conference Information Visualization (IV 2008)*, 233–239.

WEI, J., WANG, C., YU, H., AND MA, K.-L. 2010. A sketch-based interface for classifying and visualizing vector fields. In *PacificVis*, IEEE, 129–136.

YU, L., EFSTATHIOU, K., ISENBERG, P., AND ISENBERG, T. 2012. Efficient Structure-Aware Selection Techniques for 3D Point Cloud Visualizations with 2DOF Input. *IEEE Transactions on Visualization and Computer Graphics 18*, 12, 2245–2254.