Technical Report

Nr.: FIN-010-2009

Reusable Visualizations and Animations for Surgery Planning

Konrad Mühler, Bernhard Preim

Arbeitsgruppe Visualisierung

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Nr.: FIN-010-2009

Reusable Visualizations and Animations for Surgery Planning

Konrad Mühler, Bernhard Preim

Arbeitsgruppe Visualisierung

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Impressum (§ 5 TMG):

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Der Dekan

Verantwortlich für diese Ausgabe:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Konrad Mühler
Postfach 4120
39016 Magdeburg
E-Mail: muehler@isg.cs.uni-magdeburg.de

http://www.cs.uni-magdeburg.de/Preprints.html

Auflage: 61

Redaktionsschluss: 22.06.2009

Herstellung: Dezernat Allgemeine Angelegenheiten,
Sachgebiet Reproduktion

Bezug: Universitätsbibliothek/Hochschulschriften- und
Tauschstelle

Reusable Visualizations and Animations for Surgery Planning

Konrad Mühler and Bernhard Preim

Abstract— For surgical planning, the exploration of segmented structures in 3D visualizations and 2D slice views is essential. However, the generation of visualizations which support the specific treatment decisions is very tedious. Therefore, the reuse of once designed visualizations for similar cases can strongly accelerate the process of surgical planning. We present a new technique that enables the easy reuse of both medical visualization types: 3D scenes and 2D slice views. We introduce the keystates as a concept to describe the state of a visualization in a general manner. They can be easily applied to new datasets to create similar visualizations. Keystates can be shared between surgeons to reproduce and document the planning process for collaborative work.
Furthermore, animations can support the surgeon on individual exploration (e.g. by emphasizing critical anatomical structures) and are also useful in collaborative environments and discussions, where complex issues must be presented in a short time. Therefore, we provide a framework, where animations can be visually designed by surgeons during their exploration process without any programming or authoring skills. We discuss several transitions between different visualization within the medical environment and present example applications from clinical routine.

Index Terms—Visualization reuse, animation authoring, medical visualization.

1 Introduction

The exploration of 3D visualizations plays a growing role in surgical planning, since they provide a good spatial impression and a three-dimensional overview of complex organs. Especially for difficult cases, anatomical structures and pathologies are segmented to provide more quantitative information like distances and volumes and to separate different tissues that can be hardly distinguished due to similar values in the 2D slices. The surgical planning process consists of two main types: the individual exploration of the data by the surgeon and the collaborative discussion and presentation. Due to the time constraints in clinical routine, it is crucial to make both types of surgical planning highly efficient.

Exploring a dataset means to examine different structures in 2D as well as in 3D. A significant effort is still necessary to create expressive visualizations. There must be still an effort invested in creating single meaningful visualizations by the surgeon. Therefore, we provide a new technique to enable the **reuse of visualizations** for other datasets where similar anatomic and pathologic structures are present. For example, the planning of an oncological tumor resection in the liver is based on the patient *individual* data. But *all* cases have in common, that one or more tumors must be resected with a safety margin and a minimum of destruction of healthy tissue. Therefore, the visualizations that must be created are strongly similar. The reuse of once created visualizations therefore accelerates the exploration process.

For collaborative planning or presentation of cases in an environment like the tumor board (that must also be accomplished in a short time) it is crucial to generate summaries of the longer individual planning process. The surgeon must present the main aspects and critical points he revealed in the individual planning to discuss them with colleagues of different specialties. Since a presentation of the original 3D data in an interactive exploration is too tedious for a short presentation, pre-rendered animations are a good option to give an insight in the 3D data preserving the three-dimensional spatial cues on the one hand and on the other hand reducing the presentation time to a minimum. Therefore, we present a new intuitive technique to **generate animations visually and automatically**.

Outline

In Section 2, we discuss related work with the focus on approaches to support explorations, the reuse of visualizations, and the efficient generation of animations. In Section 3, we give an insight into typical surgical workflows and present example case scenarios. In Section 4, we introduce a new concept for reuse of visualizations and animations – the keystates – and explain, what information is gathered and how the keystates are used to create similar visualizations for many datasets. In Section 5, we describe, how the keystates are used to enhance the interaction process of individual intervention planning. The efficient authoring of reusable animations using the keystates is presented in Section 6. We present example applications for the keystates from the area of liver surgery planning and neck surgery planning in Section 7 and close with a discussion of our results and an outlook on future work in Section 8.

- *Konrad Mühler and Bernhard Preim are with the Department of Simulation and Graphics, University of Magdeburg, 39106 Magdeburg, Germany, E-mail: muehler@isg.cs.uni-magdeburg.de*

2 Related Work

In the past, different groups developed methods to **support the process of visualization generation and exploration**. Ma [8] presented a framework called *Image Graphs* that visualizes the changes made in a visualization during the exploration. The nodes of the graph are snapshots that are connected by edges. Each edge represents the change of a single rendering parameter like rotation or color. Adjacent nodes differ in exactly one parameter and may not be generated in temporal sequence. *Image Graphs* can be shared for collaborative visualizations and animations can be created by selecting snapshots from the graph as keyframes. Jankun-Kelly et al. [6] presented a model to formally describe the exploration process of a visualization. In their model, the changes of parameters are stored to share explorations and results and to process information between different visualization interfaces. They provide several steps of an exploration in a graph comparable to the *Image Graph* of [8] and thus providing reconstructions of earlier results. Marks et al. [9] introduced the *Design Galleries*, where a set of randomly generated input parameters like lightning is used to create visualizations and animation transitions. The results are presented to the user hoping the expected visualization is among the results. To create a sufficiently large set of proposals, a long computation time is needed.

Koop et al. [7] presented for VisTrails, a pipeline-based visualization framework, an approach called *VisComplete* that provides several proposals during the visualization pipeline buildup that the user may use to complete the pipeline. The proposals are based on an analysis of previously created pipelines with similar modules and parameterization stored in a database. *VisComplete* addresses researchers with a certain background of computational knowledge and the aim to create single visualizations. To support the reproducibility of a visualization, Scheidegger et al. [13] employed a history management in VisTrails stored in a version tree.

To select good viewpoints on anatomical structures in a medical scene, Mühler et al. [12] introduced a new technique that is based

on multiple parameters, like visible surface and preferred region. For each parameter, a parameter map is generated that stores the parameter value of each camera position on a discretised surrounding sphere. The parameter maps for the structure of interest are weighted summed up and the maximum is taken as a good viewpoint. In this framework the user can select a structure from a list and an automatic camera flight to a good viewpoint is automatically performed in real-time. A drawback of the approach is that the user has no ability to integrate his personal comprehension of a goof viewpoint, e.g. by defining own good views.

The **reuse of visualizations** or parts of them was picked up by a few papers. Hamel et al. [5] analyzed non-photorealistic illustrations for the used rendering styles to reuse them in new illustrations. Parameters like line style or shading are extracted and can be reused for other models to achieve a similar visual appearance. Svakhine et al. [14] provided illustration motifs to support the efficient generation of medical volume illustrations. The motifs contain several effects for a zone of interest, e.g. to remove occluding material or to draw edges in a specific style. Different interfaces are provided for authors and users to define the visualization style of a zone and its surrounding. They target on the exploration of a single dataset for a wide variety of complex representations. Groth and Streefkerk [4] presented a system to store the provenance of a visualization. They want to restore the sequence of steps, a researcher performed, to gain the insight in a molecular visualization. Each action (e.g. rotation or zoom) is stored in a history graph and can be annotated manually by the researcher. The approach does not provide any adaption of visualization for further datasets but enables the reproducibility for one dataset. An approach to specify the layout of textual annotations of a visualization by an example illustration was presented by Vollick et al. [15]. They used annotations layouts that were created manually by illustrators and tried to apply these on new images to achieve a similar layout. As a drawback, the system is rather slow (it took four minutes to create a layout with about 20 annotations).

There are several attempts to support the process of efficiently generating **animations**. Obviously, classical animation tools like Maya®[2] or Blender [3] are build to create single expressive animations with a high effort by highly specialized experts. Those tools are not targeted at mass production as well as inexperienced users like surgeons. Wohlfart and Hauser [17] presented a system to generate animations of medical volume visualizations. Their system enables an author to create a story as a sequence of single visualizations. They propagate that the user gets a better insight into the presented data, if he has the ability of interactive excursions. Animations can be paused for an individual exploration and resumed afterwards. The author creates the different states of an animation as nodes that can be rearranged and edited manually. Nevertheless, it is still a tedious process to create animations. Furthermore, the audience of such animations is not clear, but for clinical routine and surgical planning the proposed authoring process of [17] is not viable. An approach where existing animations are combined and applied to new scenes was presented by Wang et al. [16]. They extract single parts from different surface animations and try to find compatible sets of triangles in a new scene where the surface mesh animations can be applied automatically.

In Mühler et al. [11] scripts are used to describe the behavior of an animation for medical intervention planning. They used abstract descriptions for structures and parameters that enable a reuse of scripts for similar datasets. Even if the used script language is easy to learn, their approach is only usable for experienced developers.

Nearly all the presented approaches target at the careful exploration of single datasets or the generation of single animations. In contrast, we aim at inexperienced users without any programming or visualization background, namely surgeons. For these users, it is essential to explore a lot of patient data in short time to come to a treatment decision. Therefore, we provide a technique for the reuse of visualizations and animations for a class of datasets.

3 Medical Background

In surgery planning for most surgical interventions there is a common workflow. Nevertheless, there are substantial differences due to individual differences which influence, e.g., the difficulty of the intervention and thus the necessity of technical support and image-guidance. Therefore, we present two case scenarios with respect to requirements for the exploration process in detail.

3.1 Surgical Workflow

The surgeon explores and reviews the data, consisting of 2D slices and 3D polygonal meshes of the segmented structures, in detail. He fades structures in or out, changes the visualizations style of structures (e.g. the color or transparency) and changes the position of the camera in a complex and tedious process. Especially the handling of the virtual camera is very time-consuming, since surgeons are mostly unexercised in interaction with 3D visualizations. The 3D exploration is always combined with a detailed inspection of the 2D slices. Depending on the surgeon's preferences, this inspection is performed a) parallel, where 2D and 3D visualizations are shown side by side, b) integrated, where the slice is shown in the 3D scene, or c) separated, where either the 3D or the 2D view is visible.

After an individual exploration of the data that may take from about 10 to 40 minutes often colleagues are consulted to discuss the case and the initial treatment decision. In many hospitals, a 'tumor board' is established, where all cases are discussed in a larger community of surgeons, radiologists and medical doctors of other specialties. In current surgery planning applications snapshots may be generated and discussed with colleagues.

After planning the intervention, the segmented data needs to be provided in the operation room. The surgeon often needs to look up details or assure by comparing the discovered intra-operative situation with the planned data. Therefore, a transfer of the planned results into the operation room is required. It is obvious, that in critical situations, that were not foreseen, the surgeons needs a real view on the planned data (3D scenes and 2D slices), e.g. to change the viewpoint of the camera or to enable a specific set of structures. Thus, a selection of printed snapshots are not enough and the presentation and exploration of the 3D data in the operation is crucial. On the other hand, an intervention is a highly critical situation where no time is available for elaborative explorations of the 3D scenes and 2D slices. Thus a set of different visualizations that were generated during the planning process can be a very good starting point for short explorations. Selecting a visualization, that is very similar to the visualization the surgeon has in mind during the intra-operative situation leads to only a very few interactions to change for example the viewpoint only a little bit or enabling only one structure instead of changing the visibility and appearance of many structures.

Beside the treatment decisions and the real interventions, which are doubtless in the focus of a surgeon, the documentation of the decisions made during the whole workflow is an important aspect. Therefore, the necessary effort should be kept at a minimum. The surgeon should be as little as possible confronted with an elaborative process of generating snapshots and animations, that protocol his exploration and decision process and made it reproducible, e.g., in case of complications or for an analysis in a study. Therefore, techniques for an automatic documentation are highly recommended.

3.2 Case Scenarios

Surgery planning is usually based on tomographic images (e.g. MRI or CT data). The actual planning process is often based on segmented structures in particular if a high density of soft tissue structures with overlapping image density values occurs. Surgical planning in the neck region, the abdominal region or orthopedic interventions are preferably performed with a combination of 2D slices and segmented structures in a 3D polygonal rendering. Our work focuses on visualizations of segmented structures, although there are areas like emergency cases, where segmentation is less important and a direct volume rendering is preferred. The segmentation is normally not performed by the surgeon himself. Some radiological workstations provide semi-automatic techniques and advanced segmentations are performed by external services [10].

Surgical scenario 1: In oncological liver surgery, the resection of tumors and metastases from the liver is a common surgical intervention. The tumors need to be resected with a specific safety margin. Therefore, each tumor must be inspected in 3D and 2D in detail to preserve the safety margin. The predicted volume of remaining liver tissue is crucial to decide on a resection strategy. The vascular structures in the liver (veins and arteries) are essential in the planning process. They must be cut at the correct points to ensure the full supply and drainage of the remaining liver tissue. Therefore, several proposals of resection planes must be carefully analyzed. During the exploration a lot of structures must be displayed and hidden to visualize the large number of relations (e.g. the tumor in relation to several vascular structures or different resection proposals with respect to the safety margins). Additionally, for nearly each visualization the viewpoint must be changed completely and the surgeon often must switch between 3D and 2D to verify his decisions on the original data. At the end, several resection proposals are chosen for a discussion with colleagues to come to a final decision.

Surgical scenario 2: For patients with a malignant tumor in the neck or head region, a neck dissection must be carried out to remove the tumor as well as lymph node metastases. Due to the high concentration of vital structures in a narrow space, an intervention must be planned very carefully. Depending on the distances of lymph nodes to the neck muscles, these must be also resected if they are infiltrated by lymph nodes. If the distances between critical lymph nodes and vital structures of risk like the arteries are too small, the patient is inoperable. In individual planning the surgeon inspects all structures of risk with respect to the lymph nodes. He enables and disables lymph nodes by size and inspects their spatial relation to the muscles, veins, etc. For each structure of risk he must perform a full rotation around the structure to ensure that all lymph nodes are covered. The surgeon must change the transparency of many structures to reveal occluded structures. Thus, the individual planning process is characterized by a large number of recurrent tasks. Furthermore, for a tumor board, the surgeon must prepare a set of animations and snapshots to present his results for a collaborative discussion.

4 Concept of Keystates

To enable the reusing of visualizations and the automatically authoring of animations, we developed **keystates**. A keystate is an abstract description of the current visualization (no matter if it is 2D or 3D) and contains all information that is necessary to reuse it for similar datasets. Furthermore, multiple keystates can describe an animation. The aim is to apply a keystate on other datasets to get as a result a visualization containing the same information presented in the initial visualization. In the next sections, we describe what information is stored in a keystate (Section 4.1), how the information is gathered (Section 4.2), and how a keystate is adapted to further datasets (Section 4.3).

4.1 What information is stored in a keystate?

A keystate must contain all information that is necessary to rebuild similar visualizations. For both, 2D slice views and 3D scenes of segmented and artificial structures like needles, these are:

1. Information about the visibility of each structure.
   This includes structures that are enabled but currently hidden by other structures.

2. Information about the style parameters of each structure.
   These are: color, transparency, silhouette width and color.

3. Structure of interest.
   What structure lies in the focus of the surgeon?

4. Information about the viewpoint or the slice.
   Where is the camera located in a 3D scene and where is it looking at? What slice is visible in the 2D slice view?

Especially the 3rd aspect requires knowledge about what the surgeon wants to see in the current visualization. The visualization goal is either to assess the morphology of an anatomic structure, a group of structures or the spatial relation between structures (e.g. a distance). Knowing the focus of the surgeon is a prerequisite to interpret the current viewpoint in a 3D scene in a way that it enables the generation of similar viewpoints for other datasets. For 2D slice views, the structure of interest is important to present a similar slice in other datasets.

Therefore, we define a keystate for 3D scenes as a 3-tuple K_{3D} with

$$K_{3D} = (S, V, I) \tag{1}$$

where S is a set of style parameters for each structure or structure group, V is the viewpoint information and I the structure and type of interest. For 2D slice views, a keystate is a 3-tuple K_{2D} with

$$K_{2D} = (S, C, I) \tag{2}$$

where S and I are the same as for K_{3D} and C represents the context information visible in the slice. The design of S, V, I and C will be explained in detail in the next section.

4.2 How is the information gathered?

All information for a keystate should be gathered automatically from the visualization. We will see that this is not possible in every case. The **information of each structure's visibility, color, transparency** etc. can easily be derived from the scene. However, to support a reuse of a keystate for other datasets, this information must be generalized. It is obvious that *'lymph node 27 is visible and opaque'* is not applicable, if a dataset contains only 15 lymph nodes. Therefore, conclusions like *'all lymph nodes are visible and opaque'* are aspired.

We analyze the scene with respect to different types of groups. Some of these groups are defined in the implicit information of each case, e.g. anatomical belongings like *vessel*, *bone*, or *muscle*. Other groups are derived automatically from the underlying data and application dependent knowledge. These are for example *'Structures on the left/right side of the neck'* or *'Lymph nodes with a size larger than 2cm'*. As a result for each determined group we get information for the used common style parameters. If there are no common styles for all members of a group, e.g. some lymph nodes are red while others are yellow, this specific group is not gathered in the keystate. The style information is stored in S as a set of styles s with

$$s = (\text{groupname}, \text{visibility}, \text{color}, \text{transparency}, \text{silhouette}) \tag{3}$$

To obtain the **viewpoint information** for 3D scenes as well as the slice number for 2D views, it is essential to know the visualization goals of the surgeon. These can be for example the evaluation of a possible infiltration of anatomic structures by tumors, the access planning for interventions or the precise understanding of abnormal structures. If no user interaction is desired, e.g., if the keystate must be generated automatically (see Section 5.1), this information must be gathered automatically. In case that the surgeon selected a structure from a list or in the scene during the exploration, this structure is taken. For 3D scenes where the camera position was manually changed, we search for the structure that is most centered in the image space that is preferably unoccluded by other structures and that has probably a high importance given by the underlying data (e.g. for neck surgery lymph nodes have a high importance while bones are less important). For each structure i, we compute a value of its importance in the current view SOI with

$$SOI_i = (IC_i + A_i) * IMP_i \tag{4}$$

where IC is the inverse mean distance of all pixels of a structure to the center of the image space, A the visible portion of the surface of the structure in comparison to the size of the image space, and IMP the importance of the structure. The image center value IC is 1 when the structure consists of exactly one pixel in the center of image (and no other pixels) and near 0 if many pixels of the structure lies at the border of the image space. IC is calculated as follows:

$$IC = 1 - \frac{\sum_{j=1}^{n} x_j + y_j}{n * (is_w + is_h)} \quad (5)$$

where x and y are the coordinates of each pixel j of the structure, n is the number of visible pixels and is_w respectively is_h the width and the height of the image space. We reuse the approach introduced by Mühler et al. [12] to determine the viewpoint parameters for all camera positions on a surrounding sphere. The nearest camera position in this viewpoint data is taken and its data of the image center parameter of each structure as well as all occlusions are taken into account. As a result we get one structure with the highest value of SOI that might be in the focus of the surgeon.

This automatic approach is possible but in many cases just a rough guess since the mental activities of the surgeon are not incorporated. Furthermore, for 2D slice views with multiple visible segmented structures as overlays (see Figure 3), we cannot reliably predict the structure of interest of the surgeon. Therefore, the surgeon is asked to indicate this information. If the surgeon wants to create a keystate he is asked to select a structure or relation from a list. Depending on the specific application, there are a small number of most recent options that is presented as thumbnails for a fast access. Nevertheless, an extended list of all visible structure can be expanded. The structure or relation of interest is stored in I.

As we know the structure of interest, we can gather the values for the viewpoint in the 3D scene. Since the viewpoint was determined manually, it will never be the optimal viewpoint computed by the viewpoint selection technique with the structure of interest as input. Therefore, it is necessary to store additional data about the viewpoint that enables its reconstruction for other datasets. We compute for every parameter the scalar value at the current camera position - for pre-calculated values like the projected surface size, the nearest camera position in the pre-calculated data is chosen. The parameter values p_i are stored in the vector P with $P = (p_1, p_2, ..., p_n)$ where n is the number of parameters. The weights for each parameter w_i are stored in W with $W = (w_1, w_2, ..., w_n)$. The default for the weights is the same for all keystates in an application. In a configuration stage, the surgeons together with an IT specialist determined a small set of presets the surgeon can select from, e.g. for his personal preference or for different planning tasks.[1] Therefore, we decided to store the weights in the keystate to guarantee a preferably exact reproduction of the manually created viewpoint. The viewpoint values are stored in the 2-tuple V with

$$V = (W, P) \quad (6)$$

For 2D slice views it is not sufficient to store the absolute slice number in the keystate as it is not appropriate to store the absolute camera coordinates for 3D viewpoints. For small structures that are only visible in a very few slices it would be adequate to find a slice with this structure visible in it. For large, especially elongated structures (e.g. vessels) running across many slices, this would not be appropriate. Therefore, we store the context structures, that are visible in the current slice as context information in the vector cs with $cs = (cs_1, cs_2, ..., cs_m)$ where m is the number of visible context structures.

Obviously, there are context structures with a higher importance than others. For example, examining a muscle in the neck region the tumor has a higher importance than the skull. Thus, besides the context structures, we store an importance value for every visible structure in the slice in the vector cs_{imp} with $cs_{imp} = (cs_{imp_1}, cs_{imp_2}, ..., cs_{imp_m})$.

Therefore, C is a 2-tuple with

$$C = (cs, cs_{imp}) \quad (7)$$

[1] The surgeons himself never catch sight of the weights but gets literally descriptions.

As we know how the required information for reuse and reconstruction is gathered, we will describe how keystates are applied to similar datasets.

4.3 Reuse of Keystates

The main aim of keystates is the reuse of a large set of visualization parameters for other but similar datasets. Applying a keystate to a new dataset, a visualization similar to the visualization, the keystate was created on, should be generated. To achieve this, three steps have to be taken:

1. The visual styles like color or transparency stored in the keystate must be applied for the specific structures occurring in the new dataset.

2. For 3D visualizations, a similar viewpoint must be generated using the stored viewpoint parameters and information about the structure of interest in the keystate.
   For 2D slice views, a slice must be found that contains the structure of interest as well as its context structures.

3. The structure of interest stored in the keystate must be analyzed with respect to its occurrence in the new dataset. Keystates might be deleted or more instances must be created. (e.g., if the structure of interest is a tumor and in the new dataset occur multiple tumors.)

Applying styles. The application of style parameters is fostered by the grouping of the structures at the generation of the keystate. Therefore, several style parameters must be applied to entire groups, e.g. *"lymph nodes on the left side"* or *"muscles"*, no matter, if there are a different number of structures - i.e. it does not matter if there are 10 lymph nodes in the new dataset while it were 25 in the original dataset. If there are new structures in the dataset that do not match any group, they are rendered with default values for their type.

Viewpoint selection. To generate an appropriate viewpoint for 3D scenes, that is similar to the one stored in the keystate, the stored viewpoint parameter vector P and the parameter weights W are used. For every possible viewpoint i in the scene the parameter vector p_{new_i} is generated using the viewpoint information for the new scene with $p_{new_i} = (p_{new_1}, p_{new_2}, ..., p_{new_n})$. The parameter vectors, P from the keystate and p_{new_i} for every viewpoint in the new scene, are weighted with the weights stored in the keystate:

$$p_w = (w_1 * p_1, w_2 * p_2, ..., w_n * p_n) \quad (8)$$

$$p_{new-w} = (w_1 * p_{new_1}, w_2 * p_{new_2}, ..., w_n * p_{new_n}) \quad (9)$$

The viewpoint thats weighted parameter vector with the shortest Euclidean distance in the n-dimensional parameter space (where n is the number of parameters) to the weighted parameter vector from the keystate p_w is chosen as the most similar viewpoint for the new scene.

Determine slice. To determine a matching slice for 2D slice views, all slices with an occurrence of the structure of interest are investigated. The visible context structures on each slice are examined. The sum of the importance values of the context structures is calculated and the slice with the highest value is taken as the most similar one. We have to consider two special cases:

1. If there are multiple adjacent slices with exactly the same context structures visible (and therefore the same sum of importance values) the slice that is most centered in the axial direction is chosen.

2. If there are multiple slices with different context structures but the same highest importance value, additional instances of the keystate are created and all these slices are taken into account.

Multiple instances of keystates. Besides the special case for 2D slices explained above, there are further cases that necessitate a change of the number of keystates. Even if the datasets are similar with respect to the medical question and anatomical region there are often differences in the number and existence of segmented structures. For example:

- The initial dataset contained one tumor that was explored and stored in a keystate. A new dataset might contain three tumors.
- In the initial dataset, three enlarged lymph nodes where inspected and three different keystates were created. A new dataset might contain only one enlarged lymph node.

Therefore, the structure of interest of each keystate is analyzed by existence in the new dataset. If the structure does not occur in the current dataset, the keystate is omitted. Are there less structures in the current dataset, than were stored as keystates, the excessive keystates are omitted. Are there more structures of the structure of interest's type in the current dataset than stored in the keystates multiple instances of those keystates are created under retention of the parameters for style and viewpoint.

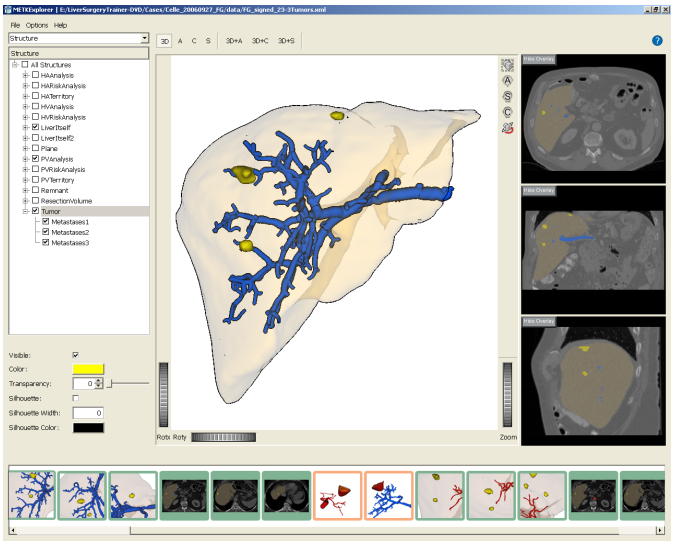

Fig. 1. **Keystates in an application.** The horizontal list at the bottom contains all keystates. Keystates can be loaded from older datasets or created new automatically (red framed) and manually (green border).

5 Enhance the Interaction Process

To unfold their full potential, the concept of keystates must be well integrated in the application workflow. Keystates can be created automatically and manually. For the automatic solution, several aspects must be taken into account, which will be discussed in Section 5.1. Areas of application are presented in Section 5.2.

5.1 Automatic Generation of Keystates

Automatically generated keystates may be useful for the documentation of workflows or to provide an undo history to the user. In contrast to keystates, generated on demand, automatically generated keystates lack in the determination of the structure of interest (see Section 4.2). Furthermore, constraints must be defined to describe **when** a keystate is generated automatically. One option is to generate a keystate after each single interaction, like changing the color of one structure, move the camera a little bit or disabling a structure. This fine granularity turned out to be too fine. Since too many keystates are generated, e.g. disabling five structures in short succession would leads to five keystates with only minor differences, but only one new visualization, the user may be interested in. Therefore, we allow a certain time range

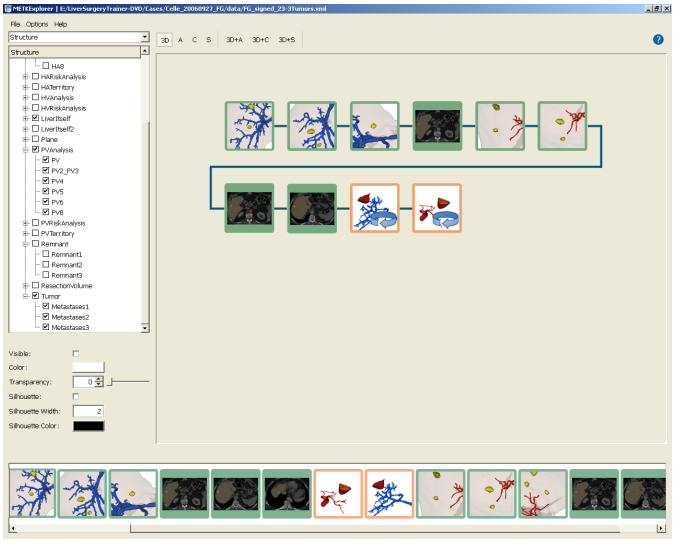

Fig. 2. **Storyboard to arrange keystates for an animation.** Keystates can be dragged in the storyboard and arbitrary arranged. If a rotation around one keystate must be performed this can be activated by a double click.

where all changes made in are combined in one keystate. To discretize the movements of the virtual camera, we use a rest time approach, where a new keystate is generated after a certain rest time of the camera. Thus, different movements that are performed in fast sequence are combined in one keystate. To visually separate automatic from manually generated keystates, they are prominent framed with different colors (see Figure 1).

5.2 Application Areas of Keystates as Interaction Support

We present three different applications of automatically generated keystates during the exploration process: a) Undo histories, b) the sharing of workflows, and c) the documentation.

Undo histories. Each automatically generated keystate can be used as an undo state for a visualization history. Clicking on a preliminary generated keystate, the represented visualization is restored. Continuing the exploration from an older keystate in the history, we neither branch the history (e.g. like [4]) nor overwrite all keystates created after the old state from which the exploration is continued (e.g. as Adobe Photoshop® [1] does). In discussions with our medical partners, an addition of all new keystates at the end of the history was preferred. Thus, the visual appealing of the history list remains linear.

Sharing workflows. Keystates generated automatically in the background enables an easy sharing of workflows between different surgeons or between a radiologist and a surgeon. The radiologist can make a first plan from his point of view and a surgeon can reproduce the performed exploration, reuse it and extend or adapt it for his own usage. This is especially useful for a remote planning, where the surgeon does not work in the near surrounding of the radiologist, or where the segmentation and intervention proposals are created by an external service.

Documentation. The documentation of the surgical planning process is still a lack in the new area of 3D surgical planning. Keystates can be stored as a direct representation of the exploration digital for a later reproduction of the planning process or can be printed as plain snapshots for the patient's file. Furthermore, animations that were created using the keystates ca be stored for a detailed insight in the surgeon's decisions.

6 Animation Authoring using Keystates

Besides the use of keystates as a support for the scene exploration in individual intervention planning, keystates may be used to create animations automatically. Each keystate is similar to a keyframe in

a classic animation, describing the state of a scene at a certain time. Between the keystates, an interpolation of style parameters and a transition of the viewpoint are performed.

Since the keystates can be applied automatically to a new dataset, also animations can be created automatically for new datasets without any interaction or additional effort for the user. Defining keystates the surgeon can concurrently define animations. How this is supported in detail is described in Section 6.1. For the transitions between different keystates, several aspects like the camera path must be considered (Section 6.2).

6.1 Interaction Techniques for Animation Authoring

The use of keystates for animation authoring is also an HCI problem. As we described, the keystates can be created automatically or on-demand by the user. All keystates are represented as thumbnails in a list. To define an animation, the user can drag arbitrary keystates from this list and drop them into a panel - the storyboard (see Figure 2). The keystates can be re-arranged there in their sequence. Additionally, the surgeon must only define two more parameters: the transition time between the keystates and the time of rest at each keystate. The input of the transition time is performed in an abstract manner using a slider providing values like “slow” and “fast”. We do not use numeric values, since the exact time between two keystates can vary depending on the distance between the two viewpoints: a short camera flight should not be as long as a long flight, e.g. only 1 second instead of 4 seconds for a long one. Even if a definition of the break time for every keystate is possible, on value for all keystates is used in clinical routine due to efficiency. A special transition is the rotation around a visualization. This is described by only one keystate that is marked by the user double-clicking on the keystate or using a context menu and specifying the rotation axis from a set of three options (axial, sagital, coronar). These keystates are indicated by an icon in the lower corner of the snapshot.

Sets of keystates describing an animation are also saved for later reuse for other datasets. Therefore, a once defined animation can be generated automatically without any user interaction for later datasets. The animations can differ in length and number of keystates. The length of transitions between two keystates depends on the length of the transition - namely the camera movement. Long camera paths lead to longer transitions. The number of keystates can differ if there a different number of structures must be inspected in the new datasets (e.g. if there are three metastases instead of one). To omit keystates or create multiple instances the same algorithm is used as for single keystates (see Section 4.3).

Our discussions with surgeons revealed a set of such animations that now can be generated automatically and presented to the surgeon before the intervention planning to get a first glance on the new dataset as well as for later collaborative discussions and for the documentation.

6.2 Animation Transitions

Besides the keystates, that define the important views and presented information, the transitions between the keystates are a second major aspect that we consider to come to holistically animations. In general, each transition is an interpolation between two keystates. Each parameter of the visualizations, e.g. color or camera position, must be interpolated separately. Depending on the type of parameters, we distinguish three major types of transitions:

1. The interpolation of style parameters
2. The movement of the camera
3. The transition between 3D scenes and 2D slice views and between different slices in 2D views.

We will present our solutions for the three transition types in the following.

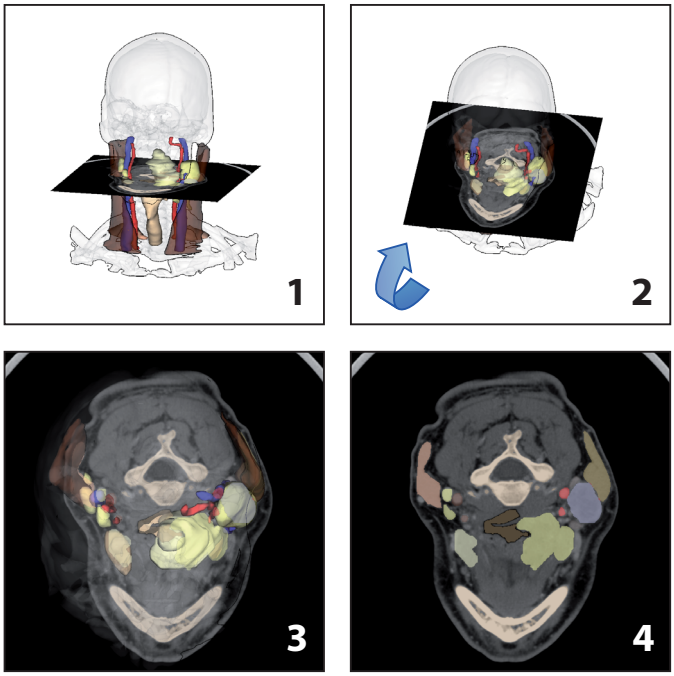

Fig. 3. **Animated transition between a 3D scene and a 2D slice view.** The slice is blended in the 3D scene, the camera is moved above the slice and the 3D structures are cross-faded into 2D overlays.

6.2.1 Interpolation of style parameters

To interpolate style parameters like the color or transparency of structures, the simplest approach is to linearly interpolate each value over the whole time of the transition. But as the camera is also often moved synchronously, the structures that are changed in their appearance are not visible over the whole time of transition. Therefore, we perform all interpolations of style parameters in the first 50% of a transition. Thus, the user can focus in the second part of the transition on the new target structure, the movement of the camera and therewith keeps a better orientation. To interpolate boolean values like visibility or silhouettes, these parameters are converted into numeric representations to change them continuously. Thus, appearing or disappearing structures can be gradually faded in or out.

6.2.2 Camera path planning

The movement of the camera between two different viewpoints (respectively two keystates) is subject to several constraints. The most important aspect is the preservation of the orientation of the user. The user must not loose the orientation during the complete camera movement and should always know where the camera is located and where it is looking at. A further constraint is the length of a camera movement and the visual appealing – a camera movement neither should be too long nor too fast. Moreover, the whole movement should be pleasant to the user. Since we are working primarily with compact scenes, all camera positions are located at a surrounding sphere. The simplest path for a camera between a start and target viewpoint is the shortest path on the surrounding sphere. However, an orientation of the user is not guaranteed for all paths. For example if the camera is moved above a pole of the sphere, it must be flipped, what is a severe disturbance of the orientation. Furthermore, discussions with surgeons revealed that there are preferred regions for the camera as they are for single viewpoints [12]. Therefore, we prefer camera paths that are probably longer but show the scene always from familiar positions. We compute a camera path between two points as a bicubic spline with a control point in the preferred region. Thus, the camera is adducted by familiar regions and camera paths over the poles of the scene are avoided.

A second important aspect is the zooming of the camera. If it is too large during the whole movement, the orientation is lost. Therefore,

we zoom out in the first part of a camera movement, looking at the structure of interest at the start point and zoom in on the target structure in the second part of the movement. The amount of zooming as well as the temporal length of a camera movement is affected by the distance between the start and the target viewpoint. If the camera is moved to a structure in the nearer surrounding, no full zoom out on the complete scene is performed, while a full zoom is mandatory to preserve the orientation if the target structure is located at the opposite side of the scene. For a more pleasant movement the camera is accelerated at the beginning and slowed down at the end.

6.2.3 Transitions for 2D slice views

The transition between different dimension of a visualization, i.e. between 3D and 2D and vice versa, should not be performed in an abrupt manner. Continuous transitions are perceived as more pleasant and it is much easier to interpret the changes correctly. The mapping between 2D and 3D information, e.g., where the slice is located and what structures are shown, is very important and must be preserved. Therefore, we perform a special transition between a 3D scene and 2D slice views (see Figure 3). First, the target slice is visualized in the 3D scene at its position in the volume. Afterwards, the camera is moved to an upright view on the slice, where the 3D scene is still kept visible. Not until the slice is completely shown the 3D structures are faded out and are replaced by 2D overlays of the slice. Since the visual styles of all structures are synchronized between 3D and 2D visualizations, the structures in the 2D slice have the same color as in the 3D scene, what supports the mapping process additionally. The transition from a 2D slice view to a 3D scene is performed conversely, where first the 3D structures are faded in the 2D slice view and afterwards the camera is moved to its final position in the 3D scene.

To animate the transition between two different slices in the 2D view, it is not enough to perform a simple blending. Therefore, we perform a real slicing from the source to the target slice. Thus, the user can keeps the orientation and gets important information, where exactly the slices are located, even if no 3D information is presented.

7 Example Applications of Keystates

The keystates were used for 21 cases of planning a liver tumor resection, for 7 cases of living liver donor transplantation and 14 cases of neck dissections. In this section, we present a few keystates that are used for oncologic liver surgery planning and for neck surgery planning.

7.1 Keystates for Liver Surgery Planning

First, the surgeon is looking for different types of vascular anomalies like atypical trifurcations at critical points of both important vascular systems portal vein (blue) and hepatic artery (red) (Keystate 1 in Figure 5). This aspect is difficult to judge for a surgeon using only 2D slices. Thus, the 3D view is chosen. Next, he inspects the liver and the existing metastases with respect to the infiltration of different vascular structures. He turns the portal vein on as well as the tumor and the liver as context information and chooses a viewpoint, where the minimal distance or infiltration is best visible (Keystate 2). He verifies his initial assessment with a view in the 2D slices (Keystate 3). The keystates were created on a dataset containing only one metastasis. In the liver of the patient of 'Case 3' three metastases were found. Therefore, additional instances of Keystate 2 and 3 were created automatically.[2] Following the surgeon repeats the inspection of infiltration for the second important vascular structures, the hepatic arteries (Keystate 4).[3] Afterwards, the surgeon inspects the planned resection plane with respect to the cutting points on the vessels. First, the remnant (green) is explored with the hepatic artery (Keystate 5) and the portal vein (keystate not shown). Afterwards, the resection volume (orange) is explored also with the hepatic artery (keystate not shown) and the portal vein (Keystate 6). The keystates were created on a dataset, where only one resection plane was planned. For 'Case 2' two different resection proposals were made. Therefore, additional instances of Keystate 5 and 6 were created automatically. For 'Case 3' no resection proposal was made. Thus, these keystates are omitted for 'Case 3'. Further explorations were performed to inspect the resection volume with the different vascular systems and to control in 2D slice views, if there is a minimal safety margin around the each tumor that must be maintained.

[2] We show two of the three created instances in Figure 5.

[3] The keystate for the 2D slice view is not shown in Figure 5 but it is similar to Keystate 3

7.2 Keystates for Planning a Neck Dissections

First, the surgeon inspects the tumor in the context of all segmented structures in the neck region (Keystate 1 in Figure 4). He repeats the inspection in the 2D slice view (Keystate 2). Afterwards, he judges the distances of all lymph nodes to the different structures of risk at the left and right side of the neck: muscles, arteries and veins. He inspects the *Musculus sternocleidomastoideus* on the left side (Keystate 3), and proceeded with the *Arteria carotis* (Keystate 4) and the *Vena jugularis* (Keystate 5). He finishes with the same structures at the right side of the neck (keystates not shown). For the tumor board, he create an animation, where a rotation around each structure of risk is performed.

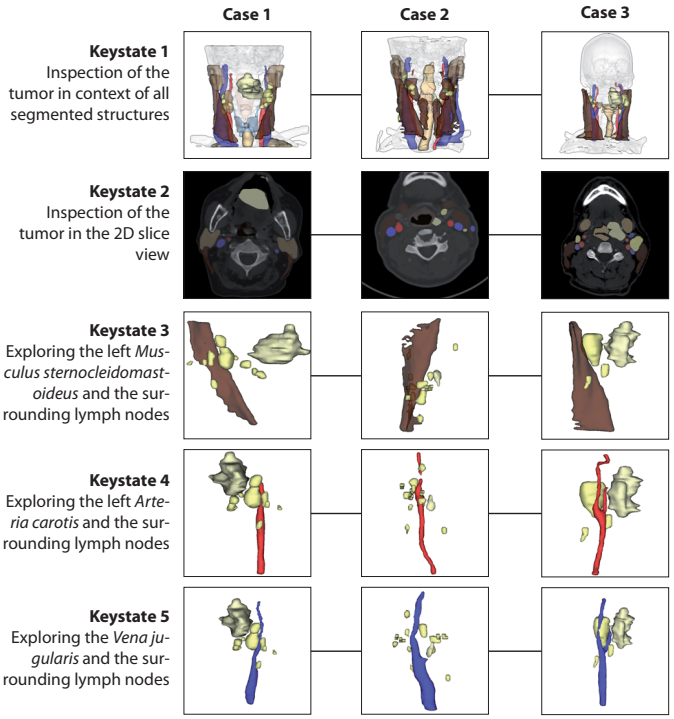

Fig. 4. **Example for reusing keystates for planning a neck dissection.** Each column represents a dataset. The surgeon inspected the tumor and the structures of risk with their surrounding lymph nodes to judge their infiltration and distances.

8 Discussion and Future Work

We have presented a new concept that enables the reuse of visualizations as well as the efficient authoring and reuse of animations for surgical planning. Using keystates, the surgeon can visually define, what he is interested in and how further visualizations for the surgical planning should look like. This enhances the individual planning process and supports the collaborative work, since for the first time surgeons are able to generate expressive animations by themselves. Our framework is widely usable for different modalities and independent of underlying image properties like the slice distance, since we use segmentation information. As the concept of keystates is currently used in several applications for surgical planning, more informal feedback is expected.

One aspect for future research is the problem of later changing keystates. The surgeon may want to modify a once created keystate, if aspects during the planning occur that were not obvious as the keystate was created. Instead of creating a new keystate (what would be nevertheless an efficient option), the modifications and explorations in new datasets may influence the original keystate. Another open question is the transfer of keystates between different surgical areas. Keystates created for liver surgery planning may be used in orthopedics. It seems to be an interesting option, if a surgeon can transfer a visual appealing visualization or animation from a colleague to his own visualizations.

REFERENCES

[1] Adobe Systems. Adobe Photoshop. `http://www.adobe.com/products/photoshop/family/`, Mar 2009.

[2] Autodesk. Maya. `http://www.autodesk.com/maya`, Mar 2009.

[3] Blender Foundation. Blender. `www.blender.org`, Mar 2009.

[4] D. P. Groth and K. Streefkerk. Provenance and annotation for visual exploration systems. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1500–1510, 2006.

[5] J. Hamel and T. Strothotte. Capturing and re-using rendition styles for non-photorealistic rendering. In *Proceedings of EuroGraphics 1999*, number 3, pages 173–182, 1999.

[6] T. Jankun-Kelly, K.-L. Ma, and M. Gertz. A model and framework for visualization exploration. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):357–369, 2007.

[7] D. Koop, C. E. Scheidegger, S. P. Callahan, J. Freire, and C. T. Silva. Viscomplete: Automating suggestions for visualization pipelines. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1691–1698, 2008.

[8] K.-L. Ma. Image graphs - a novel approach to visual data exploration. In *IEEE Visualization*, pages 81–88, 1999.

[9] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: a general approach to setting parameters for computer graphics and animation. In *International Conference on Computer Graphics and Interactive Techniques*, pages 389–400, 1997.

[10] MeVis Medical Solutions - Distant Services. `http://mms.mevis.de/en/Distant_Services.html`, Mar 2008.

[11] K. Mühler, R. Bade, and B. Preim. Adaptive script based animations for intervention planning. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 478–485. Springer, 2006.

[12] K. Mühler, M. Neugebauer, C. Tietjen, and B. Preim. Viewpoint Selection for Intervention Planning. In *IEEE/Eurographics Symposium on Visualization (EuroVis)*, pages 267–274, 2007.

[13] C. Scheidegger, H. Vo, D. Koop, J. Freire, and C. Silva. Querying and creating visualizations by analogy. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1560–1567, Nov 2007.

[14] N. Svakhine, D. Ebert, and D. Stredney. Illustration motifs for effective medical volume illustration. *IEEE Computer Graphics and Applications*, 25(3):31–39, 2005.

[15] I. Vollick, D. Vogel, M. Agrawala, and A. Hertzmann. Specifying label layout styles by example. In *ACM Symposium on User Interface Software and Technology (UIST)*, pages 221–230, 2007.

[16] Y.-S. Wang and T.-Y. Lee. Example-driven animation synthesis. *Visual Computer*, 24(7):765–773, 2008.

[17] M. Wohlfart and H. Hauser. Story Telling for Presentation in Volume Visualization. In *IEEE/Eurographics Symposium on Visualization (EuroVis)*, pages 91–98, 2007.

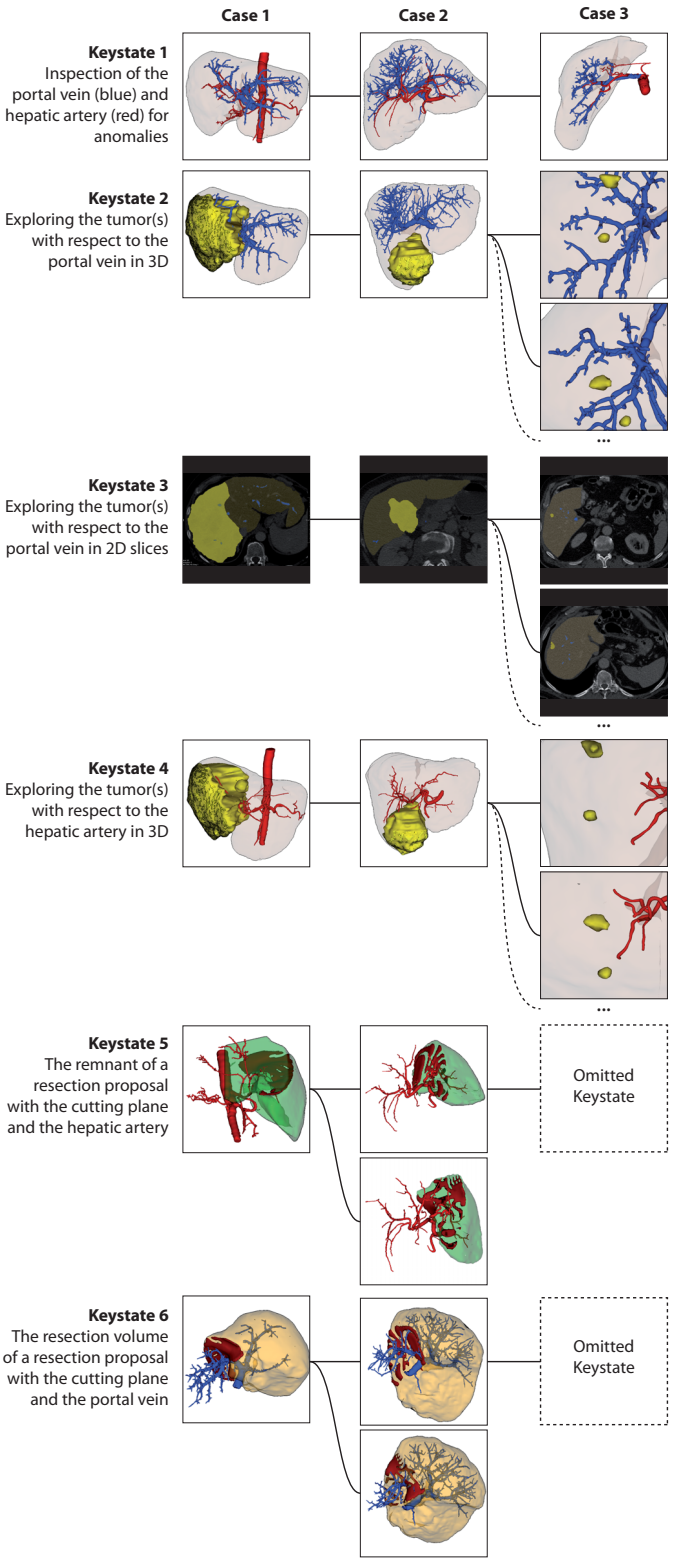

Fig. 5. **Example for reusing keystates for liver surgery planning.** Each column represents a dataset. For some keystates, multiple instances are created and some keystates are omitted, since the addressed structure of interest (the resection proposal) does not exist in 'Case 3'.