Visualization in Radiation Oncology: Towards Replacing the Laboratory Notebook

Erik W. Anderson [1] Steven P. Callahan [2] George T. Y. Chen [3,4] Juliana Freire [1]
Emanuele Santos [2] Carlos E. Scheidegger [2] Cláudio T. Silva [2] Huy T. Vo [2]

[1] School of Computing, University of Utah
[2] Scientific Computing and Imaging Institute, University of Utah
[3] Harvard Medical School
[4] Department of Radiation Oncology, Massachusetts General Hospital

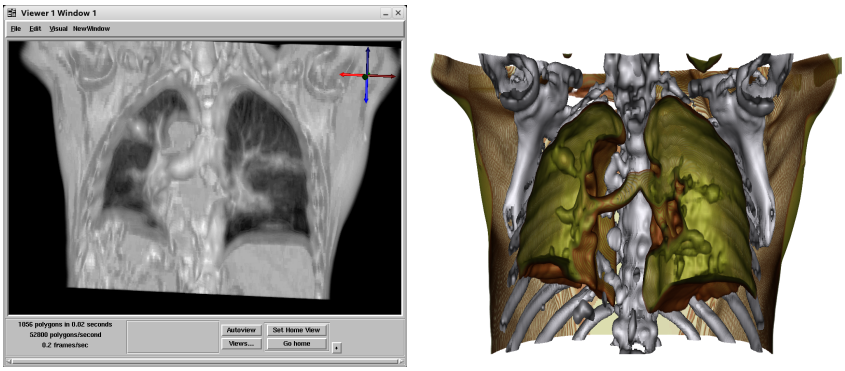

Figure 1: Importance of advanced visualization techniques in radiation oncology. The figure on the left is representative of the type of slice-based visualization used currently in radiation oncology. Because this technique is not able to clearly distinguish different kinds of tissues and structures or represent three-dimensional motion, it makes diagnosis and treatment harder. In contrast, the image on the right was generated using volume rendering with a transfer function that clearly differentiates between pathological and normal tissues.

Abstract

Data exploration in radiation oncology requires the creation of a large number of visualizations. For treatment planning, detailed information about the processes used to manipulate data collected and to create visualizations is needed for assessing the quality of the results. Current visualization systems allow the interactive creation and manipulation of complex visualizations. However, they lack the ability to manage the data involved in the visualization process, and in particular, they lack mechanisms to capture the provenance of both the visualization process and associated data. Consequently, they do not provide adequate support for the creation and exploration of a large number of visualizations. VisTrails is a visualization management system that manages both the process and meta-data associated with visualizations. A novel feature of VisTrails is an action-based mechanism which uniformly captures the provenance of data as well as of the visualization process. The detailed information about the history of visualization process, together with an intuitive interface for comparative visualization, enables a number of features that streamline the task of exploration through visualization and greatly simplify the scientific discovery process. In this paper, we explore the benefits of using VisTrails for large-scale data exploration in radiation oncology.

CR Categories: K.6.1 [Management of Computing and Information Systems]: Project and People Management—Life Cycle; K.7.m [The Computing Profession]: Miscellaneous—Ethics

Keywords: Visualization systems

1 Introduction

Visualization has been used by oncologists for many years to aid in the process of cancer diagnosis as well as treatment planning. Among other uses, modern medical imaging devices such as Computed Tomography (CT) scanners have facilitated the diagnosis process by providing the means to distinguish between pathological tissue (such as a tumor) from normal tissue. Typically, a clinician navigates through a series of cross-sections of the resulting 3D scan data to find the problematic areas. More advanced tools have been developed by the visualization community for extracting features and facilitating 3D exploration (*e.g.* direct volume rendering with transfer function manipulation [7]). Although these advanced tools have yet to be approved for clinical use, they are often used in research hospitals to quickly explore the data and act as a guide for interventional procedures [12, 9]. The usefulness of advanced visualization techniques can be seen in Figure 1—volume rendering derives a much more detailed image in which it is much easier to identify pathological tissues and their three-dimensional (3-D) movement than traditional techniques. The Radiation Physics Division of the Department of Radiation Oncology at the Massachusetts General Hospital (MGH) is an example of a group using advanced visualization tools. Radiation oncologists at MGH have been using recent volume rendering techniques to locate tumors in preparation for radiation therapy treatment. However, the visualization process as currently deployed is very complex and time-consuming. Whereas a scanner can create a new dataset in minutes, using these advanced tools, it takes from several hours to days to create appropriate visualizations.

Visualization systems such as SCIRun [11] and ParaView [4] allow the interactive creation and manipulation of complex visualizations. These systems are based on the notion of dataflows [8], and they provide a visual interface to produce visualizations by as-

sembling *pipelines* out of modules connected in a network. These systems provide easy-to-use visual interfaces for creating and modifying visualization pipelines. However, these systems have important limitations that greatly hamper their usability in large-scale data exploration tasks. Notably, they lack the infrastructure to manage the visualization process and the associated data—input data, meta-data, as well as derived data products. In particular, they provide no mechanism for systematically capturing information about visualization provenance. In addition, they lack mechanisms for scalable exploration of the parameter-space of visualizations and for interactively comparing the different results.

To generate visualizations that are useful for the oncologists, often, a visualization expert needs to go through a lengthy, trial-and-error process in which she creates a large number of images. Due to the high sensitivity of the visualization process for radiation oncology, many pages of hand-written notes are required that describe the visualization history in detail (*i.e.* all modifications to the data and/or pipeline). And in order to allow reproducibility, besides the hand-written notes, a very large number of files needs to be saved, including datasets, derived images, animations, and visualization pipelines. Furthermore, the notes, pipelines, and file-name conventions, are usually only understood by their creator. This greatly hinders collaboration and also hampers the useful lifetime of data. For example, a post-doctoral researcher working on visualizations may keep detailed laboratory notebooks and large file structures of saved data—when she leaves, it is hard and sometimes impossible for other team members to continue (or re-use) her work.

VisTrails [1, 2] is a visualization management system that streamlines the process of data exploration through visualization. An important feature of VisTrails is a history management mechanism that keeps detailed provenance of both visualization pipelines and associated data. The provenance information not only enables reproducibility, but also, when combined with a multi-view interface for comparative visualization, it allows users to efficiently and effectively explore data through visualization: they can return to previous versions of a visualization pipeline (aka dataflow); apply a pipeline to different data sets; systematically explore the parameter space of the pipeline; query the visualization history; and comparatively visualize different results. Note that VisTrails is not intended as a replacement for systems such as SCIRun [11], VTK [5], or ParaView [4], instead, it provides infrastructure that can be combined with and enhance these systems.

In this paper, we explore an application of visualization in radiation oncology and demonstrate the benefit of using a system that provides a data exploration infrastructure for generating and interacting with large-scale visualizations. Because VisTrails unobtrusively captures detailed provenance information and also manages visualization data, it removes the need for manually creating and maintaining large directory structures of resulting datasets and images as well as the need for a detailed, hand-written laboratory notebook. In addition, the provenance information enables many features that simplify the generation and analysis of a large number of visualizations, including support for collaboration and a mechanism for creating visualizations in bulk.

The paper is organized as follows. Section 2 describes the current process of visualization in radiation oncology. In Section 3, we give a brief overview of VisTrails and some of its key features. We revisit the visualization of radiation oncology in Section 4 and show how VisTrails simplifies the process. In Section 5 we discuss the merits of a visualization management system and the impact it can have in radiation oncology.

2 Visualization in Radiation Oncology

Visualization in Radiation Oncology is used not only for diagnosis and data exploration, but for treatment planning and analysis as well. In order to provide researchers and clinicians with visualizations used for treatment planning and research, a detailed log of the exact process used to create a visualization is necessary. This helps the specialists understand the resulting image and ascertain its accuracy. For this reason, a record must be maintained both of all changes to the visualization process and of the parameters controlling the various sub-processes.

Often, the visualization produced by a pipeline is only the result of the visualization process applied to a single dataset out of a potentially large ensemble of related data. For example, a set of CT scans over several timesteps during radiation treatment. As we describe below, each dataset in an ensemble may require a re-parameterization of modules in the dataflows that produce the visualizations. Not only do these parameter changes need to be recorded, but the re-parameterized visualization pipelines must be saved to capture the changes made to them.

2.1 The Visualization Process in Radiation Oncology

Creating adequate visualizations for researchers and clinicians is a lengthy process requiring many iterations of visualization formation, feedback, and refinement *per dataset*. This iterative process begins with data collection and distribution. After receiving a dataset from a radiologist, the data must be pre-processed in order to be readable by the visualization tool being used. Fortunately, this process is often transparent as many visualization packages are capable of reading the industry-standard DICOM [10] format. An initial visualization is then created, which is subsequently refined until a suitable visualization is derived. Below, we illustrate this process through a concrete example.

Initial Visualization. After the raw data is read, a visualization dataflow must be formed and executed to create a visualization of this data. Initially, a simple dataflow is created to ascertain whether substantial modifications are required. Typically, modifications are required to rectify discrepancies in the raw data. Some of these are dependent on the hardware profiles of the data collection device used to generate the dataset being inspected. For example, to accommodate scanners with different resolutions in different dimensions, the initial pipeline may need to be modified to rescale the grid or voxel sizes in a single axis. It is not uncommon for this scaling parameters to change based on the scanning device used to generate the dataset being visualized. Since a patient may be scanned with different equipment over the course of a treatment, different scaling may be needed for each of the patient's scans. Each change must be recorded in a laboratory notebook on a per dataset basis, and each pipeline derived in this process must also be saved in a file.

After the parameter changes required to capture differences in data collection mechanisms are recorded and taken into account, the data itself must further manipulated to form a meaningful visualization. Operations may be needed to transform the dataset into a format that is required by the visualization system. Some systems require volumes of data to be of a certain size or have a specific range of scalar values associated with each sample requiring resampling and requantizing of the data. After this resampling and quantization, the data is substantially changed and this change can lead to misleading visualizations. Therefore, similar to the parameter changes described above, any manipulation must be documented. This ensures that not only re-parameterizations can be associated with a specific subset of visualizations, but that also that specialists are aware of these changes and the potential alterations they may have caused to the images.

Finally, the visualization is refined by re-parameterizing the transfer function portions of the visualization pipeline. This re-parameterization represents the exploratory aspect of visualization in radiation oncology. Each change in the transfer function yields a new visualization that may help identifying a particular type of

tissue or structure. When a meaningful visualization is found, the parameters defining the transfer function is recorded in a laboratory notebook and the pipeline saved in order to ensure the reproducibility of the visualization.

Because radiation oncologists are interested in the irradiation of small regions of moving tissues, it is important to be able to generate meaningful visualizations of not only the structure surrounding an area of interest, but also of the movement in three dimensions of these structures. The most intuitive way of forming such a visualization is to create an animation of these data as they vary in time. An example of this visualization process as it applies to a time-varying dataset representing a breathing cycle is discussed in detail below.

Example 2.1 (Re-parameterization of Lung CT Scans)
When data is received from a radiologist, it typically consists of a set of volumes representing a breathing cycle for an individual patient. After an initial pipeline is constructed that derives some meaningful representation of the data, we need to discover the voxel scaling as it applies to the corresponding hardware collector. In order to begin a parameter space exploration, an initial visualization of the unmodified parameter is generated. This first visualization represents a voxel scaling of 1.0 in all axes. The pipeline for this visualization is saved and connected to the data set being explored. This connection of the pipeline to the corresponding dataset can be made in several ways: assigning a representative name to the pipeline file; copying the file to a different directory; or recording the change in the laboratory notebook modification. In this scenario, a file was created by concatenating the unique identifier of the dataset, the value of the time-slice being analyzed, and a short description of the visualization created. This file was then placed in a directory named by the globally unique data set identifier, based on the anonymized patient identifier, and notes were entered into a laboratory notebook section specific to that dataset regarding the time, date, and filenames producing a visualization.

After a visualization of the unscaled data is generated, an inspection of the image can produce the next guess for the proper parameter value required for the data set in question. At this point, the parameter value is set and a new visualization is created. This process is repeated until an acceptable visualization is found. In each iteration, files are created in the directory mentioned above with the filenames being containing the name of the parameter being modified followed by the parameter value for that file. In the case of a large parameter space being explored, this single exploration can create literally *hundreds* of individual files and many pages of handwritten notes documenting the procedure and files associated with the formation of the visualizations. Once an appropriate value is found, the change is *promoted* to the original visualization and another note is made in the laboratory notebook detailing the parameter and the value selected.

Refinement of the Visualization. After an acceptable visualization is created, a set of images from various orientations is captured and sent to the researchers and clinicians to gather feedback. The feedback received from the specialists is used to modify the visualization pipeline, parameters, and possibly the data, to *enhance the quality* of the final visualization. Again, for each and every change made detailed records must be added to the laboratory notebook and the corresponding pipelines are saved.

The visualization is refined through a sequence of re-parameterizations. The most common of these re-parameterizations are changes to transfer functions. In this case, the parameters reflect only a mapping of scalar values to color values and do not manipulate the data in any way. Since the data or its underlying representation is not being manipulated, a change to these values needs to be recorded only when a parameterization is found that produces a meaningful visualization. However, since in many systems such as SCIRun and Paraview, there is no separation between a pipeline specification and its parameters, a whole pipeline needs to be saved to a file. Once an appropriate transfer function is discovered that clearly identifies the structures of interest to the specialists, the corresponding visualization pipeline must be saved, in this case, by appending some textual description of the visualization to the original filename (*e.g.* anon4877_axial_lesion_20060331.srn highlights a *lesion* in the axial view of patient 4877). Additionally, the parameter values must be recorded in a laboratory notebook in such a way that the visualization can be reproduced exactly. Unfortunately, complex transfer functions can have many points defining the opacity portion of the function and even more tuples representing the red, green, and blue color channel points. The sheer amount of data produced as a direct result of these sorts of re-parameterizations of the visualization pipeline causes not only an immense number of files to be generated inside a confusing directory structure, but also results in the writing by hand of many pages of notes in various laboratory notebooks associated with the specific dataset being represented.

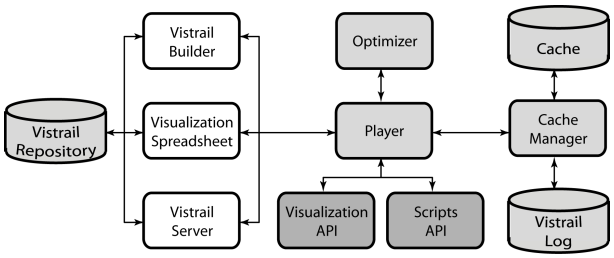

Figure 2: VisTrails Architecture.

2.2 Discussion

The need to manually record changes to a notebooks and to save a large number of pipeline files is not only time consuming—requiring logging of seemingly minor parameter changes, but also leads to an explosion in the volume of visualization metadata—massive numbers of individual files representing the different visualization pipelines created in the exploration process, which often differ in a single re-parameterization per file. The sheer number of files and notes taken often leads to confusion when attempting to recreate a specific visualization. One reason for this is the fact that provenance is captured in an incomplete and non-uniform way. Part of it is encoded in an unstructured form in filenames. In addition, different people use different naming conventions and thus the information cannot be easily queried. Consequently, it can be challenging to locate the correct saved pipeline to reproduce a visualization. Besides, since information about the relationship among pipelines is not systematically stored (except in the handwritten notes), identifying the differences in the parameterizations of two similar visualization pipelines is a laborious and difficult, yet necessary, task.

3 The VisTrails System: An Overview

VisTrails is a visualization management system that manages both the process and metadata associated with visualizations. With VisTrails, we aim to give scientists a dramatically improved and simplified process to analyze and visualize large ensembles of simulations and observed phenomena. The high-level architecture of the system is shown in Figure 2. We only sketch the main features of the system here, for further details see [1, 2]. Users create and edit dataflows using the *Vistrail Builder* user interface. The

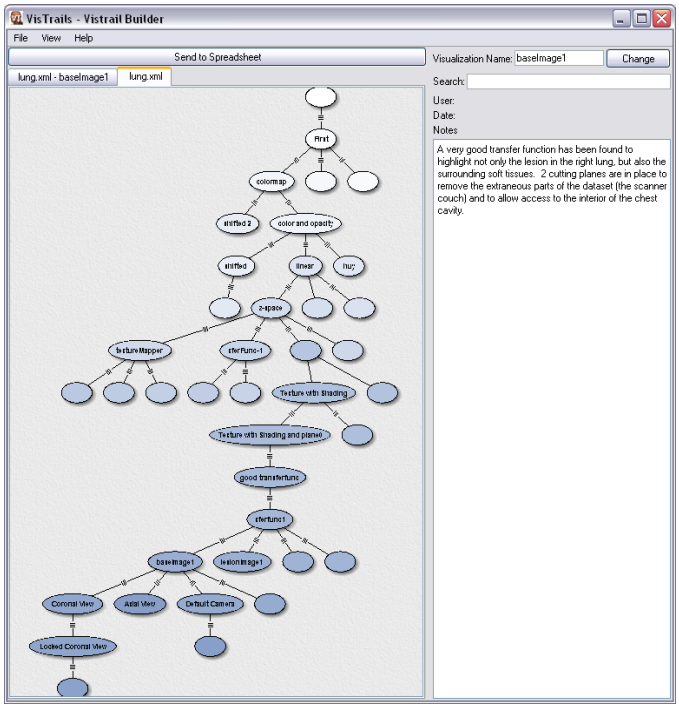

Figure 3: A snapshot of the VisTrails history management interface. Each node in the history is a separate dataflow that differs from its parent by changes to parameters or modules. This tree represents the trial-and-error process followed to generate the images shown in Figure 4.

dataflow specifications are saved in the *Vistrail Repository*. Users may also interact with saved dataflows by invoking them through the *Vistrail Server* (*e.g.* through a Web-based interface) or by importing them into the *Visualization Spreadsheet*. Each cell in the spreadsheet represents a view that corresponds to a dataflow instance; users can modify the parameters of a dataflow as well as synchronize parameters across different cells. Dataflow execution is controlled by the *Vistrail Cache Manager*, which keeps track of operations that are invoked and their respective parameters. Only new combinations of operations and parameters are requested from the *Vistrail Player*, which executes the operations by invoking the appropriate functions from the Visualization and Script APIs. The Player also interacts with the *Optimizer* module, which analyzes and optimizes the dataflow specifications. A log of the vistrail execution is kept in the *Vistrail Log*. The different components of the system are described below.

Vistrail Specification. A dataflow is a sequence of operations used to generate a visualization. A vistrail captures the notion of an *evolving dataflow*—it consists of several versions of a dataflow. The information in a vistrail serves both as a log of the steps followed to generate a series of visualizations, a record of the visualization provenance, and as a recipe to automatically re-generate the visualizations at a later time. The steps can be replayed exactly as they were first executed, and they can also be used as templates—they can be parameterized. In order to handle the variability in the structure of operations, and to easily support the addition of new operations, we represent vistrails using XML (for more details, see [2]). An important benefit of using an open, self-describing, specification is the ability to query, share, and publish vistrails. This allows a scientist to locate dataflows suitable for a particular task or data products generated by a given sequence of operations, as well as to publish an image along with its associated vistrail so that others can easily reproduce the results.

History Management. As discussed above, a vistrail captures information about the evolution of a dataflow or collection of related dataflows—it behaves as a versioning system for dataflows. A vistrail consists of a tree where each node corresponds to a dataflow (see Figure 3). But instead of storing the dataflows themselves, we store the operations that take one dataflow to another. An edge between a parent and child nodes in a vistrail tree represents a set of change actions applied to the parent to obtain the dataflow for the child node. The action-based provenance mechanism of VisTrails is reminiscent of DARCS[1]. This structure allows scientists to easily navigate through the space of dataflows created for a given exploration task. In particular, they have the ability to return to previous versions of a dataflow and compare their results. To simplify the retrieval of particularly interesting versions, a vistrail node can optionally have a name. At any point in time, the scientist can choose to view the entire history of changes, or only the dataflows important enough to be given a name.

Note that the vistrail tree structure only shows the dependencies among the versions. To convey the chronological order in which the versions were created, we use different saturation levels to indicate the age of the various dataflows: darker nodes are the ones created more recently. There are other possible visualizations: collaborations can be seen by distinguished users through different colors; the tree might be selectively pruned to only show results of a query on the annotations, etc.

Caching, Analysis and Optimization. Having a high-level specification allows the system to *analyze and optimize dataflows*. Executing a vistrail can take a long time, especially if large data sets and complex visualization operations are used. It is thus important to be able to analyze the specification and identify optimization opportunities. In the current VisTrails prototype, we leverage the vistrail specification to identify and avoid redundant op-

[1] http://abridgegame.org/darcs

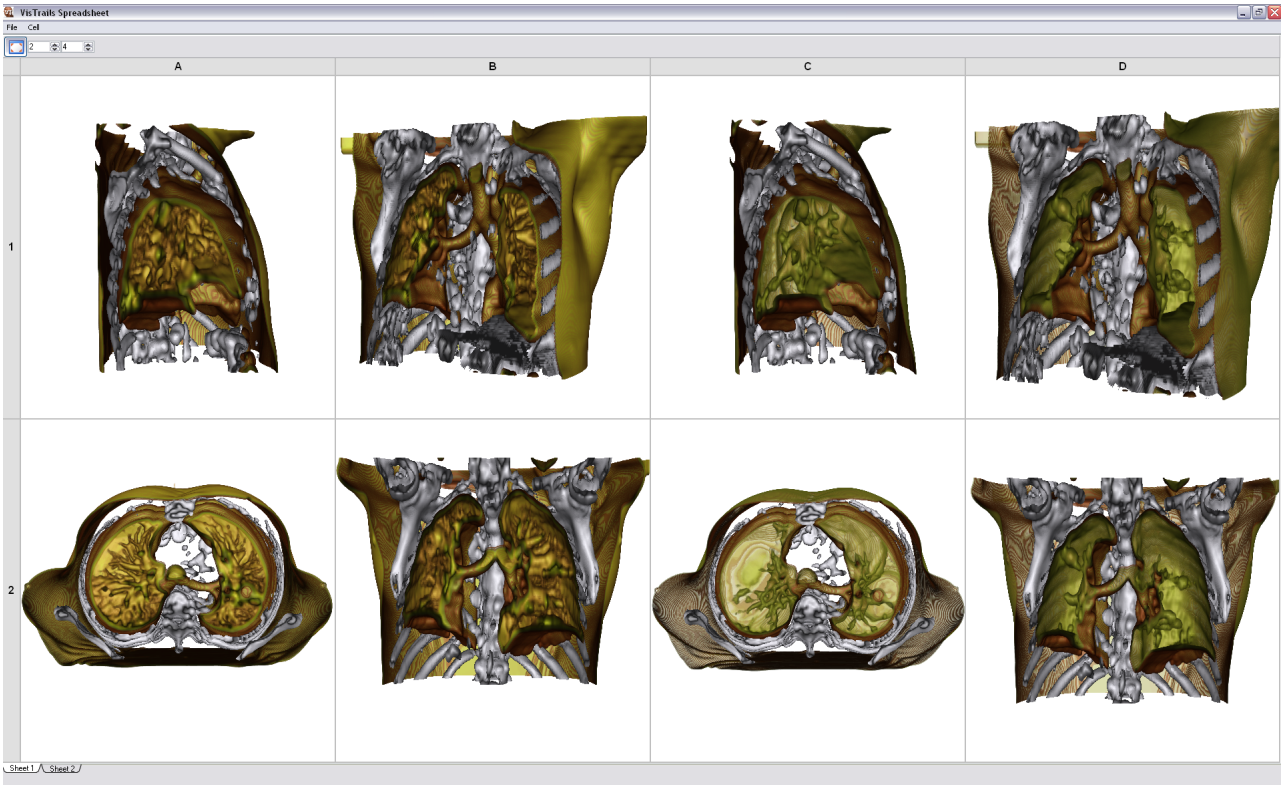

Figure 4: Multi-view visualization exploring different viewpoints, cross-sections and transfer functions. These sorts of visualizations allow specialists to easily track structural motion of both healthy and cancerous tissue in time while maintaining visual separation of the two.

erations. The Vistrail Cache Manager (VCM) is responsible for scheduling the execution of modules in vistrails by identifying previously computed subnetworks and performing constant-time cache lookups [1]. Caching is specially useful while exploring multiple visualizations. When variations of the same dataflow need to be executed, substantial speedups can be obtained by caching the results of overlapping subsequences of the dataflows.

Playing a Vistrail. The Vistrail Player (VP) receives as input an XML file for a vistrail instance and executes it using the underlying Visualization or Script APIs. The semantics of each particular execution are defined by the underlying API. Currently, the VP is a very simple interpreter which supports VTK classes and external scripts. The VP needs the ability to create and execute arbitrary VTK modules from a dataflow. This requires mapping VTK descriptions, such as class and method names, to the appropriate modules in the dataflow. The wrapping mechanism is library-specific, and in our first version [1], we exploited VTK automatic wrapping mechanism to generate all required bindings directly from the VTK library headers. Our new implementation uses Python[2] to further simplify the process of wrapping external libraries, and to enable easy extensions to the system.

Creating and Interacting with Vistrails. The Vistrail Builder (VB) provides a graphical user interface for creating and editing dataflows. It writes (and also reads) dataflows in the same XML format as the other components of the system. It shares the familiar nodes-and-connections paradigm with dataflow systems. The VB also provides mechanisms to streamline the visualization process. As complex visualization pipelines contain many common tasks, a macro mechanism is provided for the re-use of pipelines or pipeline fragments. A bulk-update mechanism is also provided in to simplify the creation of a large number of visualizations of an n-dimensional slice of the parameter space of a dataflow (see Section 4 for details).

To allow users to compare the results of multiple vistrails, we built a Visualization Spreadsheet (VS). As shown in Figure 4, the VS provides the user a set of separate visualization windows arranged in a tabular view. This layout makes efficient use of screen space, and the row/column groupings can conceptually help the user explore the visualization parameter space [3]. The cells may execute different vistrails and they may also use different parameters for the same vistrail specification as a result of a bulk-update (see Figure 5). To ensure efficient execution, all cells share the same cache. Users can also create visualizations by analogy using the VS interface. For example, when finding a favorable set of parameters for one visualization, a user will likely need to change other related visualizations in the same way. Instead of having to identify the relevant operations, he can tell the system to automatically infer, *by way of analogy*, which changes are needed (see Figure 6). This makes it possible for non-experts to derive complex visualizations.

4 Using VisTrails: Visualization in Radiation Oncology Revisited

Below, we describe how VisTrails streamlines the visualization process presented in Section 2. Although the same manipulations and tasks are required, by providing an adequate data management and exploration infrastructure, VisTrails automates and greatly simplifies them.

[2] http://www.python.org

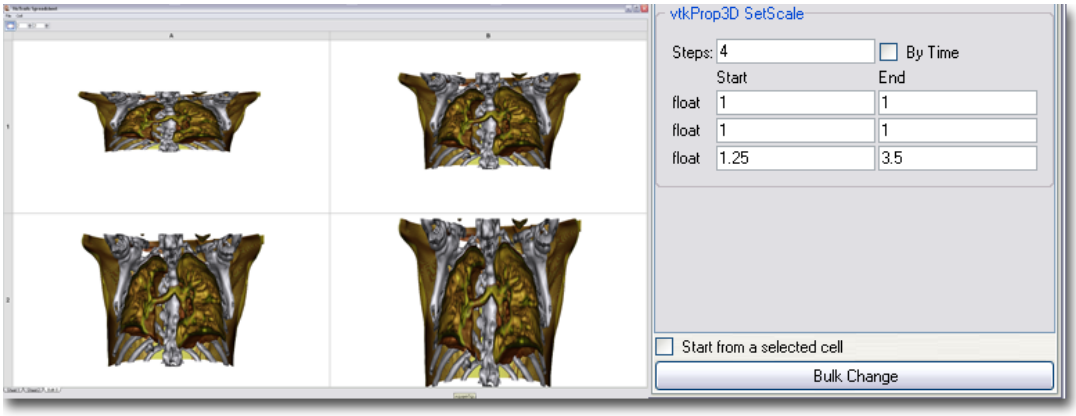

Figure 5: Results of a bulk update exploration voxel scaling in a single dimension. This exploration corrects for the non-uniform resolution of the data collection devices generating data.

Bulk Updates and Comparative Visualization. After receiving a dataset from a radiologist, the data requires some pre-processing in order to discover any parameters that are dependent on the hardware scanning tools used to capture the data. With VisTrails, a set of images representing variation of voxel sizes and shapes can be generated in a single step, using the bulk-update facility. The process is as follows: the user selects, from the history tree (see Figure 3) the node which corresponds to the initial visualization. Then, the parameters corresponding to voxel size and shape are selected, and appropriate ranges for their values are defined. The system then generates a set of re-parameterized pipelines—corresponding to the different combination of these parameters, executes them, and displays the resulting visualizations in the spreadsheet. This is illustrated in Figure 5, which shows four visualizations of one time step of the lung data set varying the voxel size from 1.25 to 3.5 in four steps. By inspecting the spreadsheet, the visualization expert or specialist can easily select the most appropriate image. The sequence of parameter modifications corresponding to the selected image is then applied to initial visualization. This effectively creates a new branch in the tree which contains the pipeline for the selected image.

To perform comparative visualization, previously, it was necessary to run multiple instances of the visualization pipeline and do screen captures to extract the images so that they could be compared side-by-side. Obviously, using this approach, it was not possible to interact with the visualizations.

The Provenance Mechanism: An Electronic Lab Notebook. As illustrated above, the exploratory process is fully and automatically documented in the history tree—it is not necessary to record individual operations in a notebook, or save individual pipelines in separate files—all related pipelines are concisely stored in a single vistrails structure. The system systematically keeps detailed provenance information for each visualization created. In particular, the differences and relationships between pipelines is explicitly stored. Because this information is stored in a structured format (in XML), it can be easily and efficiently queried—no longer requiring users to manually inspect individual files and corresponding handwritten notes.

Another useful feature of VisTrails is the ability to annotate the nodes in the history tree. These nodes can be given descriptive names as well as be associated with notes that describe the visualization in more detail. These textual descriptions can be used, for example, to reflect the user's motivations for the choices made to create the visualization. This information is queryable and can help users to locate a desired visualization in a possibly large vistrail tree. Also note that, that changes to the history tree are tagged with the date and time of the change as well as the user who changed the tree. This allows VisTrails to provide visual cues of about the recency as well as the authors of a node. For the former, VisTrails uses use different saturation levels, and for the latter, different colors.

Comparative and Collaborative Visualization. To visualize the image generated by a pipeline P in a vistrail, a user needs just drag the node P and drop it into a spreadsheet cell. This allows users to quickly inspect different, related visualizations and using the provenance information, they can easily determine the parameterization that generated it. Furthermore, given two nodes in the tree, the system automatically derives the differences between them—including differences in parameter settings as well as changes in pipeline specification. Besides helping in the understanding of visualizations, it also enables users to explore data in a collaborative environment. Recall that in order to accomplish such an analysis with previous systems, a detailed examination of not only many pages of handwritten notes contained in a laboratory notebook is needed, but the myriad of saved visualization pipeline files must be examined to fully differentiate two visualizations.

Bulk Changes and Instant Animations. Regardless of the methodology used to generate a set of visualizations, it is often desirable to construct an animation over time in order to more thoroughly explore how a dataset evolves—in our example, such an animation can show how pathological tissues and tumors are affected by radiation treatment. This is particularly useful in terms of time-based CT scans where the output from the initial scan describes the motion of soft tissues during a complete breathing cycle. This common example makes effective use of both VisTrail's bulk-change capabilities and of the spreadsheet-based visualization system's ability to form animations from changes to the input parameter of the visualization pipeline. By performing a bulk-change to the dataset reader's input parameter, a series of visualizations can be derived that fully describe the breathing cycle of a patient. VisTrails can quickly composite the still images into an ordered animation that shows the motion during a breathing cycle.

Without this capability, animations are created from a sequence of screen-captured images which are input to a third-party tool. This process needs to be repeated for every parameter change from which an animation would be produced. In contrast, using VisTrail's analogy-based re-parameterization, actions required to replicate the parameter change can be easily identified and applied to the

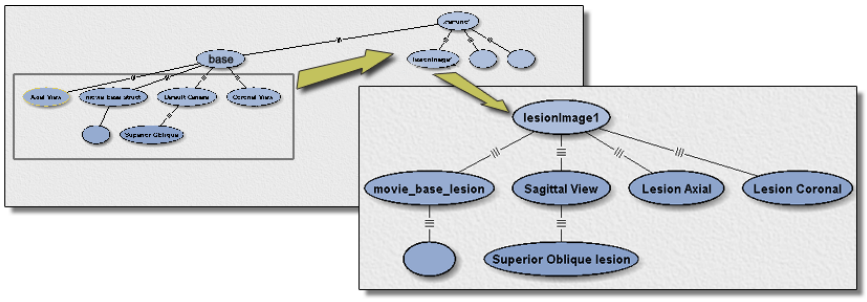

Figure 6: Creation of visualizations by analogy. The set of operations applied to the vistrail node *base* is applied to a different node in the tree (labeled *lesionimage1*).

appropriate visualizations. This can be seen most evidently in the event of a change of transfer function to a visualization that would form a full animation. The analogy would be applied to all datasets in the bulk-change parameterization, and the new animation would be automatically generated for each analogy-based change the user requires. This operation is illustrated in the video included with this submission.

Visualization by Analogy. After the initial visualization process has been completed and the oncologists and clinicians give their feedback, the refinement process can begin as it did in the case discussed in Section 2. With VisTrails, however, the refinement process of constructing high-quality images is completed in a fraction of the time. Besides no longer needing to manually record the process provenance, visualizations can easily created by analogy. As illustrated in Figure 6, the action-based structure of the history lends itself naturally to the incremental refinement of a visualization regardless of the types of parameters being changed or the magnitude of the changes to them to be performed easily. The figure shows the process by which a group of visualizations can be generated by applying a set of actions that had been previously applied to a different pipeline.

Using Provenance to Explore Different Visualization Strategies. In the event that a pipeline requires massive change to overcome some limitation or ambiguity in the visualization, the amount of work originally required just to document such a modification is mind-boggling—the visualization process needed to be re-created from scratch. The history tree in VisTrails system, simply branching the version tree at the appropriate node, annotating the node with a complete description of why the change was necessary, and then reformulating the visualization pipeline properly is all that is required.

5 Scientific Implications

The two processes for creating, manipulating, and finalizing visualizations and their respective pipelines discussed above and in Section 2 described two distinct approaches for solving visualization problems in Radiation Oncology. In systems without a robust data provenance tracking system, detailed handwritten notes and many files must be created to properly record the parameterizations necessary to produce a visualization. Consequently, the level of confusion involved in differentiating between two visualizations and their parameterizations is increased as many sources of information must be analyzed to fully understand the motivations and results of the re-parameterization. However, using a visualization system equipped with a robust provenance management tool, the analyzation of unique visualizations stemming from a common ancestor is immediately apparent. Using VisTrail's history management tree, a user can easily select two visualizations and visually compare the pipelines *and their respective parameterizations* to more quickly and thoroughly analyze the underlying data. Since the entire provenance of the visualizations are kept through out the lifetime of the vistrail, the confusion relating to multiple filenames and even handwritten notes in a laboratory notebook is eliminated. The ability to annotate visualizations at VisTrail's version level means that any important annotations, such as motivation and any other records needed to adequately document a change, can be included in a visually meaningful and easily queriable way. Furthermore, the spreadsheet style of VisTrail's multiple visualization capabilities allow any user to quickly and thoroughly explore an entire parameter space resulting in a much more rapid convergence of the visualization to an accurate representation of the data being displayed. Additionally, through the use of analogy-based transformations, actions resulting in good visualizations in one branch of the version tree can easily and quickly be applied to any other branch with a common ancestral node. This method of refinement of visualizations allows a non-expert to rapidly develop complex and meaningful visualizations of a wide variety of datasets and visualization modalities while still maintaining a fully descriptive provenance of the actions performed on the data.

6 Conclusion

VisTrails is a new visualization management system that provides the necessary infrastructure to streamline the process of data exploration through visualization. In this paper, we discussed how VisTrails can improve the visualization process in Radiation Oncology. By automatically and unobtrusively capturing detailed provenance of visualizations and associated pipelines, users need not manually maintain a laboratory notebook. Besides, since the history tree naturally models the relationships amongst pipelines—it represents their evolution—the differences between two pipelines are explicitly stored by the system. Last, but not least, VisTrails greatly simplifies and speeds up the exploration of large parameter spaces. This is possible due to a unique combination of features VisTrails provides, notably, the ability to display and interact with multiple visualizations through the Visualization Spreadsheet[1, 3]; the bulk-update facility; the ability to generate images by analogy and to quickly produce animations.

The first author (Anderson) used SCIRun extensively in the course of a 7-month internship at the Massachusetts General Hospital performing the visualization tasks described in this paper. Our initial study has shown that the same tasks can be accomplished in a small fraction of the time with the kind of functionality available in VisTrails. In the near future, we would like to perform a user study to more precisely quantify the benefit of VisTrails in Radiation Oncology.

Acknowledgments.

This work was partially supported by the National Science Foundation under grants IIS-0513692, CCF-0401498, EIA-0323604, CNS-0541560, and OISE-0405402, the Department of Energy, an IBM Faculty Award and a University of Utah Seed Grant.

REFERENCES

[1] L. Bavoil, S. Callahan, P. Crossno, J. Freire, C. Scheidegger, C. Silva, and H. Vo. Vistrails: Enabling interactive multiple-view visualizations. In *IEEE Visualization 2005*, pages 135–142, 2005.

[2] S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. Managing the evolution of dataflows with vistrails *(Extended Abstract)*. In *IEEE Workshop on Workflow and Data Flow for Scientific Applications (SciFlow)*, 2006. To appear.

[3] E. H. Chi, P. Barry, J. Riedl, and J. Konstan. A spreadsheet approach to information visualization. In *IEEE Information Visualization Symposium*, pages 17–24, 1997.

[4] Kitware. Paraview. http://www.paraview.org.

[5] Kitware. The Visualization Toolkit. http://www.vtk.org.

[6] Kitware. The Visualization Toolkit (VTK) and Paraview. http://www.kitware.com.

[7] J. Kniss, G. Kindlmann, and C. Hansen. Multi-dimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, July 2002.

[8] E. A. Lee and T. M. Parks. Dataflow Process Networks. *Proceedings of the IEEE*, 83(5):773–801, 1995.

[9] M. Levoy, H. Fuchs, S. Pizer, J. Rosenman, E. L. Chaney, G. W. Sherouse, V. Interrante, and J. Kiel. Volume rendering in radiation treatment planning. In *Proceedings of the First Conference on Visualization in Biomedical Computing*, May 1990.

[10] NEMA. The DICOM Standard. http://medical.nema.org.

[11] S. G. Parker and C. R. Johnson. SCIRun: a scientific programming environment for computational steering. In *Supercomputing*, 1995.

[12] C. A. Pelizzari and G. T. Y. Chen. Volume visualization in radiation treatment planning. *Critical Reviews in Diagnostic Imaging*, 41(6):379–364, 2000.