Managing Rapidly-Evolving Scientific Workflows

Juliana Freire, Cláudio T. Silva, Steven P. Callahan,
Emanuele Santos, Carlos E. Scheidegger, and Huy T. Vo

University of Utah

Abstract. We give an overview of VisTrails, a system that provides an infrastructure for systematically capturing detailed provenance and streamlining the data exploration process. A key feature that sets VisTrails apart from previous visualization and scientific workflow systems is a novel action-based mechanism that uniformly captures provenance for data products and workflows used to generate these products. This mechanism not only ensures reproducibility of results, but it also simplifies data exploration by allowing scientists to easily navigate through the space of workflows and parameter settings for an exploration task.

1 Introduction

Workflow systems have been traditionally used to automate repetitive tasks and to ensure reproducibility of results [1,6,9,10]. However, for applications that are exploratory in nature, and in which large parameter spaces need to be investigated, a large number of related workflows must be created. Data exploration and visualization, for example, require scientists to assemble complex workflows that consist of dataset selection, and specification of series of algorithms and visualization techniques to transform, analyze and visualize the data. The workflow specification is then adjusted in an iterative process, as the scientist generates, explores and evaluate hypotheses about the data under study. Often, insight comes from comparing multiple data products. For example, by applying a given visualization process to multiple datasets; by varying the values of simulation parameters; or by applying different variations of a process (e.g., which use different visualization algorithms) to a dataset. This places the burden on the scientist to first generate a data product and then to remember the input data sets, parameter values, and the exact workflow configuration that led to that data product. As a result, much time is spent manually managing these rapidly-evolving workflows, their relationships and associated data.

Consider the problem of radiation treatment planning. Whereas a scanner can create a new dataset in minutes, using advanced dataflow-based visualization tools such as SCIRun [10], it takes from several hours to days to create appropriate visualizations. Fig. 1 shows a series of visualizations generated from a CT scan of a torso—each visualization is created by a different dataflow. During the exploratory process, a visualization expert needs to manually record information about how the dataflows evolve. Often, this is achieved through a combination of written notes and file-naming conventions. For planning the treatment of a

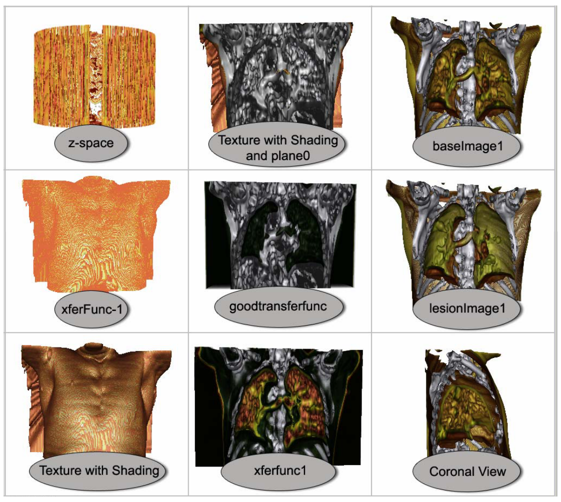
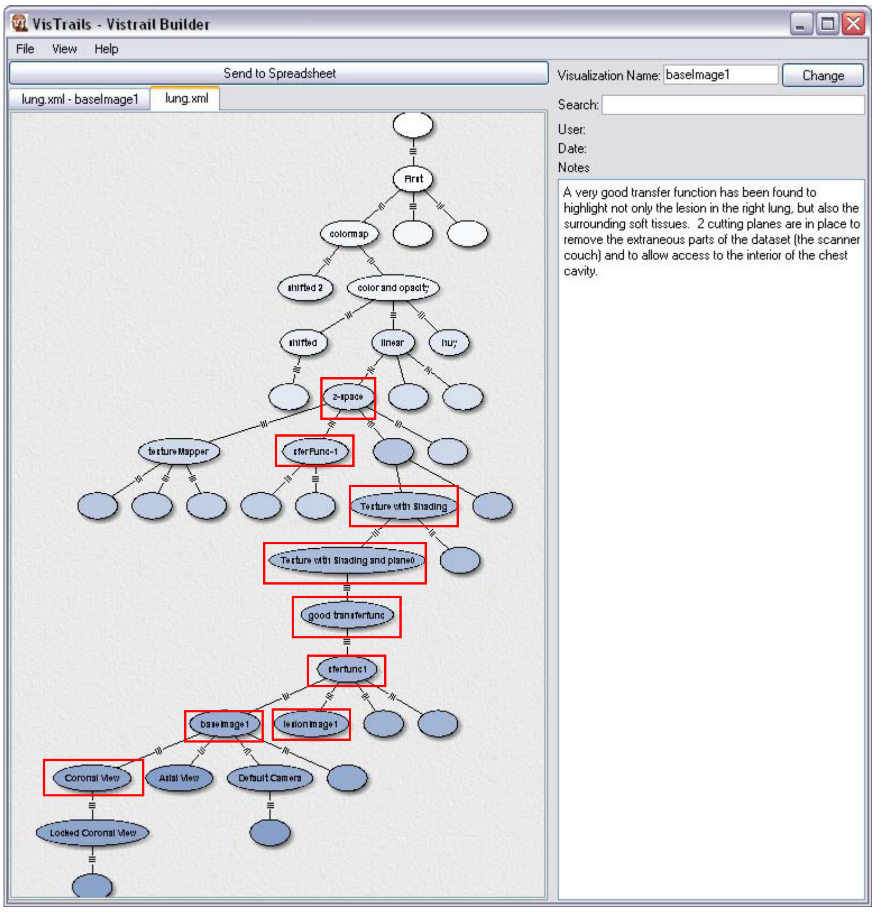


Fig. 1. Series of images generated from an CT scan for planning the radiation treatment of a lung-cancer patient

single patient, it is not uncommon that a few hundred files are created to store dataflow instances and their associated images [2]. To help the radiation oncologists understand the resulting images and ascertain their accuracy, a detailed log of the exact process used to create the images is necessary—this often requires many pages of notes detailing the process.

At the University of Utah, we have been developing VisTrails, a system whose goal is to simplify and streamline the process of scientific data exploration. VisTrails provides an infrastructure which can be combined with and enhance existing visualization and workflow systems. A novel feature of VisTrails is an action-based mechanism which uniformly captures provenance information for both data products and workflows used to generate these products. As shown in Fig. 2, the action-based provenance is stored as a rooted tree, where each node corresponds to a *version* of a workflow, and edges between nodes correspond to the action applied to create one from the other. This tree reflects the process followed by the visualization expert to construct the necessary images, and concisely represents all the workflow versions explored. Although the issue of provenance in the context of scientific workflows has received substantial attention recently, most works focus on data provenance, i.e., maintaining information of how a given data product is generated [6,7,11]. To the best of our knowledge, VisTrails is the first system to provide support for the systematic tracking of workflow evolution.

By maintaining detailed provenance of the exploration process, VisTrails not only ensures reproducibility, but it also allows scientists to easily navigate through the space of workflows and parameter settings used in a given exploration task. In particular, this gives them the ability to return to previous versions



Visualization Name: baseImage1

Search:

User:

Date:

Notes

A very good transfer function has been found to highlight not only the lesion in the right lung, but also the surrounding soft tissues. 2 cutting planes are in place to remove the extraneous parts of the dataset (the scanner couch) and to allow access to the interior of the chest cavity.

Fig. 2. A snapshot of the VisTrails provenance management interface. Each node in this vistrail version tree represents a workflow version. The nodes highlighted in the tree correspond to the images shown in Fig. 1. This tree captures all the steps followed by a visualization expert to derive the images needed for the radiation treatment planning of a patient.

of a workflow and compare their results. Powerful operations are also possible through direct manipulation of the version tree. These operations, combined with an intuitive interface for comparing the results of different workflows, greatly simplify the scientific discovery process. These include the ability to re-use workflows and workflow fragments through a macro feature; to explore a multi-dimensional slice of the parameter space of a workflow and generate a large number of data products through bulk-updates; to analyze (and visualize) the differences between two workflows; and to support collaborative data exploration in a distributed and disconnected fashion.

Outline. In this paper, we give an overview of VisTrails. The architecture of the system is described in Section 2. In Section 3, we present the action-based provenance mechanism and discuss some of the data exploration operations it enables. We review the related work and conclude in Section 4, where we also outline directions for future work.

2 VisTrails: System Overview

With VisTrails, we aim to give scientists a dramatically improved and simplified process to analyze and visualize large ensembles of simulations and observed phenomena. Although the initial motivation for developing VisTrails was to provide support for data exploration through visualization, the system is extensible and provides infrastructure for managing metadata and processes involved in the creation of data products in general, not just visualizations. The high-level architecture of the system is shown in Fig. 2. Below we briefly describe its key components. For more details, the reader is referred to [3,5].

Users create and edit workflows using the *Vistrail Builder*, which provides a visual programming interface similar to those of visualization and workflow systems [9,10]. The workflow specifications are saved in the *Vistrail Repository*. Users may interact with saved workflows 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 workflow instance; users can modify the parameters of a workflow as well as synchronize parameters across different cells. The spreadsheet layout makes efficient use of screen space, and the row/column groupings can conceptually help the user explore the workflow parameter space.

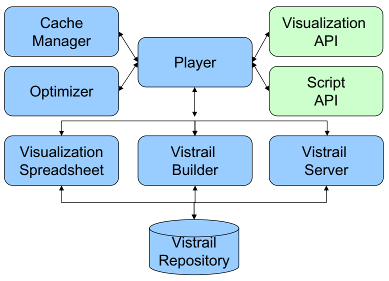
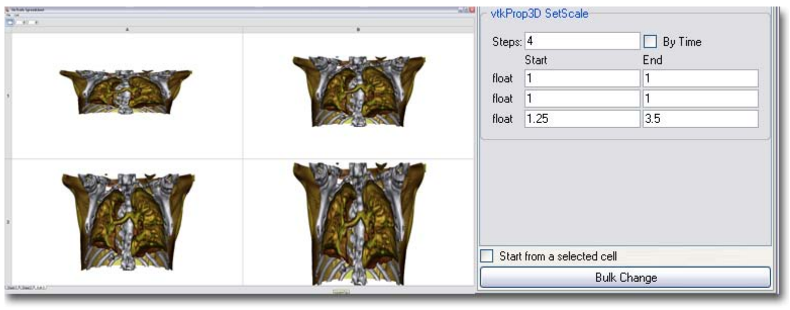


Fig. 3. VisTrails Architecture

Workflow execution is controlled by the *Vistrail Cache Manager*, which keeps track of invoked operations 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 workflow specifications.

3 Action-Based Provenance and Data Exploration

Vistrail: An Evolving Workflow. To provide full provenance of the exploration process, we introduce the notion of a vistrail. *A vistrail captures the evolution of a workflow—all the trial-and-error steps followed to construct a set of data products.* A vistrail consists of a collection of workflows—several versions


vtkProp3D SetScale

Steps: 4 ☐ By Time

	Start	End
float	1	1
float	1	1
float	1.25	3.5

☐ Start from a selected cell

Bulk Change

Fig. 4. Results of a bulk update exploration voxel scaling in a single dimension shown in the VisTrails Spreadsheet

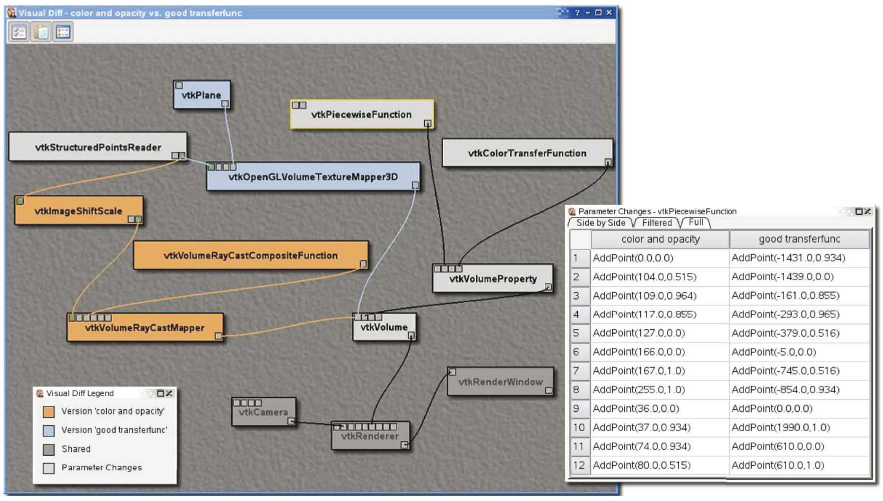
of a workflow and its instances. It allows scientists to explore data products by returning to and modifying previous versions of a workflow.

A vistrail is depicted in Fig. 2. Instead of storing a set of related workflows, we store the operations (actions) that are applied to the workflows. A vistrail is essentially a tree in which each node corresponds to a *version* of a workflow, and the edge between nodes P and C, where P is the parent of C, corresponds to one or more actions applied to P to obtain C. More formally, let WF be the domain of all possible workflow instances, where $\emptyset \in WF$ is a special empty workflow. Also, let $x : WF \rightarrow WF$ be a function that transforms a workflow instance into another, and $\mathcal{W}$ be the set of all such functions. A vistrail node corresponds to a workflow f constructed by a sequence of actions x_i, where each $x_i \in \mathcal{W}$:

$$f = x_n \circ x_{n-1} \circ \ldots \circ x_1 \circ \emptyset$$

Workflow Change Actions. In the current VisTrails prototype, we implemented a set of operators that correspond to common actions applied to workflows in the exploratory process, including: adding or replacing a module, deleting a module, adding a connection between modules, and setting parameter values. We also have an import operator that adds a workflow to an empty vistrail—this is useful for starting a new exploration process. Internally, the vistrail tree is represented in XML. This allows users to query the workflows and the provenance information, as well as share information easily. For a description of the vistrail schema, see [5].

The action-oriented provenance mechanism captures important information about the exploration process through the very simple process of tracking (and recording) the steps followed by a user. Although quite simple and intuitive, this mechanism has important benefits. Notably, it uniformly captures both changes to workflow instances (i.e., parameter value changes) and to workflow specifications (i.e., changes to modules and connections). In addition, it enables several operations that greatly simplify the data exploration process. We outline some of these operations below, for more details, see [5].


Visual Diff - color and opacity vs. good transferfunc
vtkPlane
vtkPiecewiseFunction
vtkStructuredPointsReader
vtkColorTransferFunction
vtkOpenGLVolumeTextureMapper3D
vtkImageShiftScale
vtkVolumeRayCastCompositeFunction
vtkVolumeProperty
vtkVolumeRayCastMapper
vtkVolume
vtkRenderWindow
vtkCamera
vtkRenderer
Visual Diff Legend
Version 'color and opacity'
Version 'good transferfunc'
Shared
Parameter Changes
Parameter Changes - vtkPiecewiseFunction
Side by Side
Filtered
Full

	color and opacity	good transferfunc
1	AddPoint(0.0,0.0)	AddPoint(-1431.0,0.934)
2	AddPoint(104.0,0.515)	AddPoint(-1439.0,0.0)
3	AddPoint(109.0,0.964)	AddPoint(-161.0,0.855)
4	AddPoint(117.0,0.855)	AddPoint(-293.0,0.965)
5	AddPoint(127.0,0.0)	AddPoint(-379.0,0.516)
6	AddPoint(166.0,0.0)	AddPoint(-5.0,0.0)
7	AddPoint(167.0,1.0)	AddPoint(-745.0,0.516)
8	AddPoint(255.0,1.0)	AddPoint(-854.0,0.934)
9	AddPoint(36.0,0.0)	AddPoint(0.0,0.0)
10	AddPoint(37.0,0.934)	AddPoint(1990.0,1.0)
11	AddPoint(74.0,0.934)	AddPoint(610.0,0.0)
12	AddPoint(80.0,0.515)	AddPoint(610.0,1.0)

Fig. 5. Visual diff interface. This figure shows the differences between the nodes (workflows) labeled `color and opacity` and `good transferfunc`.

Scalable Derivation of Data Products. The action-oriented model leads to a very natural means to *script* workflows. For example, to execute a given workflow f over a set of n different parameter values, one just needs to apply a sequence of set parameter actions to f:

$$(setParameter(id_n, value_n) \circ \ldots (setParameter(id1, value1) \circ f) \ldots)$$

Or to compare the results of different data transformation algorithms represented by modules R_1 and R_2, a *bulk update* can be applied that replaces all occurrences of R_1 with R_2 modules. Fig. 3 shows the VisTrails bulk-change interface. For the workflow corresponding to the node labeled `baseImage1` in the tree of Fig. 2, the user instructs the system to create visualizations varying the voxel size from 1.25 to 3.5 in four steps. VisTrails executes the workflow using the interpolated values and automatically displays the four images in the spreadsheet, where the specialist can easily select the most accurate one. Since some scanners use different resolution in different axes, correcting non-uniform resolution is a common task while dealing with CT scans. To perform this task using SCIRun [10], the visualization expert must go through the lengthy process of manually setting these parameters, one by one through a GUI and saving the resulting images into files.

Re-Use of Stored Provenance. To construct complex scientific workflows, users must have deep knowledge of underlying tools and libraries. Even for experts, creating these workflows can be time-consuming. Thus, mechanisms that allow the re-use of workflows or workflow fragments are key to streamlining the exploratory process. In VisTrails, users can create macros by selecting a sequence of actions in the version tree, or by selecting a workflow fragment. Internally, a macro m is represented as a sequence of operations $x_j \circ x_{j-1} \circ \ldots \circ x_k$. To apply m to a workflow f in the version tree, VisTrails simply composes m with the actions of f.

Interacting with Provenance Information. At any point in time, the scientist can choose to view the entire history of changes, or only the workflows important enough to be given a name, i.e., the tagged nodes in the version tree. The version tree in Fig. 2 represents the history of the changes applied to a workflow to generate the visualizations shown in Fig. 1. Note that in this figure, only tagged nodes are displayed. Edges that hide untagged nodes are marked with three short perpendicular lines. In addition, since the tree structure only shows the dependencies among the workflow versions, different saturation levels are used to indicate the chronological order in which the versions were created—darker nodes are more recent.

To better understand the exploratory process, users often need to compare different workflows. The difference between two nodes in the vistrail tree can be derived by computing the difference between the sequences of actions associated with the nodes. The visual diff interface of VisTrails is illustrated in Fig. 5.

Collaborative Data Exploration. Data exploration is a complex process that requires close collaboration among domain scientists, computer scientists and visualization experts. The ability to collaboratively explore data is key to the scientific discovery process. A distinctive feature of the VisTrails provenance mechanism is monotonicity: nodes in the vistrail version tree are never deleted or modified—once pipeline versions are created, they never change. Having monotonicity makes it possible to adopt a collaboration infrastructure similar to modern version control systems (e.g., GNU Arch, BitKeeper, DARCS). A user's local copy can act as a repository for other users. This enables scientists to work offline, and only commit changes they perceive as relevant. Scientists can also exchange patches and synchronize their vistrails. The vistrail synchronization algorithm is described in [5].

4 Related Work and Discussion

In this paper, we gave an overview of VisTrails, a system that provides a novel infrastructure for tracking provenance of both data products and workflow evolution. VisTrails is not intended to replace visualization and scientific workflow systems, instead it can be combined with and enhance these systems.

Although provenance in the context of scientific workflows has received substantial attention recently, most works focus on data provenance. To the best of our knowledge, VisTrails is the first system to provide support for tracking workflow evolution. Provenance has also been investigated in other areas. In their pioneering work on the GRASPARC system, Broadlie *et al.* [4] proposed the use of a history mechanism that allowed scientists to steer an ongoing simulation by backtracking a few steps, changing parameters, and resuming execution.

However, their focus was on steering time-dependent simulations, not on data exploration. Kreuseler*et al.* [8] proposed a history mechanism for exploratory data mining. They used a tree-structure, similar to a vistrail, to represent the change history, and described how undo and redo operations could be calculated in this tree structure. Whereas their theoretical framework attempted to capture the complete state of a software system, VisTrails uses a simpler model and only tracks the evolution of workflows. This allows for the much simpler action-based provenance mechanism described above.

Maintaining detailed provenance has many benefits, but it also presents many challenges. A potential problem is information overflow—too much data can actually confuse users. An important challenge we need to address is how to design intuitive interfaces and provide adequate functionality to help the user interact with and use the provenance information productively. We are currently investigating interfaces and languages that facilitate the querying and exploration of the provenance data as well as efficient storage strategies.

A big barrier to a more wide-spread use of scientific workflow systems has been complexity. Although most systems provide visual programming interfaces, assembling workflows requires deep knowledge of the underlying tools and libraries. This often makes it hard for domain scientists to create workflows and steer the data exploration process. An important goal of our research is to eliminate, or at least reduce this barrier. VisTrails already presents a significant step towards this goal. The existing facilities for scalable parameter exploration and workflow re-use give domain scientists a high degree of flexibility to steer their own investigations. Since VisTrails records all user interactions, an interesting direction we intend to pursue is to try to identify exploration patterns in the version tree and use this knowledge to help users create new workflows and/or solve similar problems.

Acknowledgments. We thank Dr. George Chen (MGH/Harvard University) for providing us the lung datasets, and Erik Anderson for creating the lung visualizations. This work is partially supported by the NSF (under grants IIS-0513692, CCF-0401498, EIA-0323604, CNS-0514485, IIS-0534628, CNS-0528201, OISE-0405402), the DOE, and an IBM Faculty Award. E. Santos is partially supported by a CAPES/Fulbright fellowship.

References

1. G. Alonso and C. Mohan. Workflow management: The next generation of distributed processing tools. In S. Jajodia and L. Kerschberg, editors, *Advanced Transaction Models and Architectures*, chapter 2. Kluwer, 1997.
2. E. Anderson, S. Callahan, G. Chen, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. Visualization in radiation oncology: Towards replacing the laboratory notebook. Technical Report UUSCI-2006-017, SCI Institute–Univ. of Utah, 2006.
3. 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.

4. K. Brodlie, A. Poon, H. Wright, L. Brankin, G. Banecki, and A. Gay. GRASPARC: a problem solving environment integrating computation and visualization. In *IEEE Visualization '93*, pages 102–109, 1993.
5. S. Callahan, J. Freire, E. Santos, C. Scheidegger, C. Silva, and H. Vo. Using provenance to streamline data exploration through visualization. Technical Report UUSCI-2006-016, SCI Institute–Univ. of Utah, 2006.
6. I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying and automating data derivation. In *Statistical and Scientific Database Management (SSDBM)*, pages 37–46, 2002.
7. P. Groth, S. Miles, W. Fang, S. C. Wong, K.-P. Zauner, and L. Moreau. Recording and using provenance in a protein compressibility experiment. In *Proceedings of the 14th IEEE International Symposium on High Performance Distributed Computing (HPDC'05)*, July 2005.
8. M. Kreuseler, T. Nocke, and H. Schumann. A history mechanism for visual data mining. In *IEEE Symposium on Information Visualization*, pages 49–56, 2004.
9. B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Tao, and Y. Zhao. Scientific Workflow Management and the Kepler System. *Concurrency and Computation: Practice & Experience*, 2005.
10. S. G. Parker and C. R. Johnson. SCIRun: a scientific programming environment for computational steering. In *Supercomputing*, 1995.
11. Y. L. Simmhan, B. Plale, and D. Gannon. A survey of data provenance in e-science. *SIGMOD Record*, 34(3):31–36, 2005.