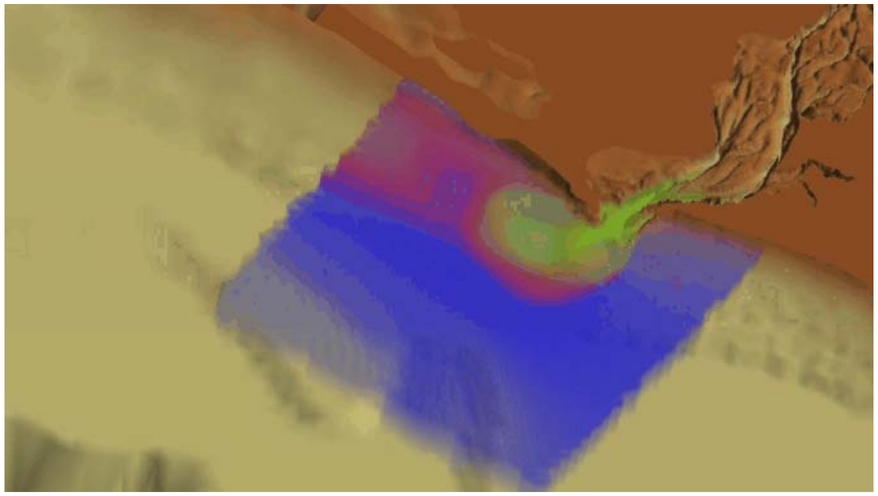
Managing Rapidly-Evolving Scientific Workflows

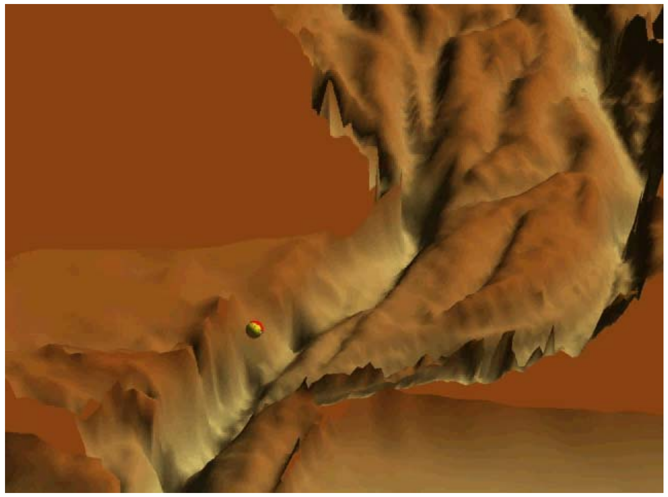
Juliana Freire
Claudio T. Silva
http://www.sci.utah.edu/~vgc/vistrails/
University of Utah

Joint work with:
Steven P. Callahan, Emanuele Santos,
Carlos E. Scheidegger and Huy T. Vo

Our Motivation: CORIE

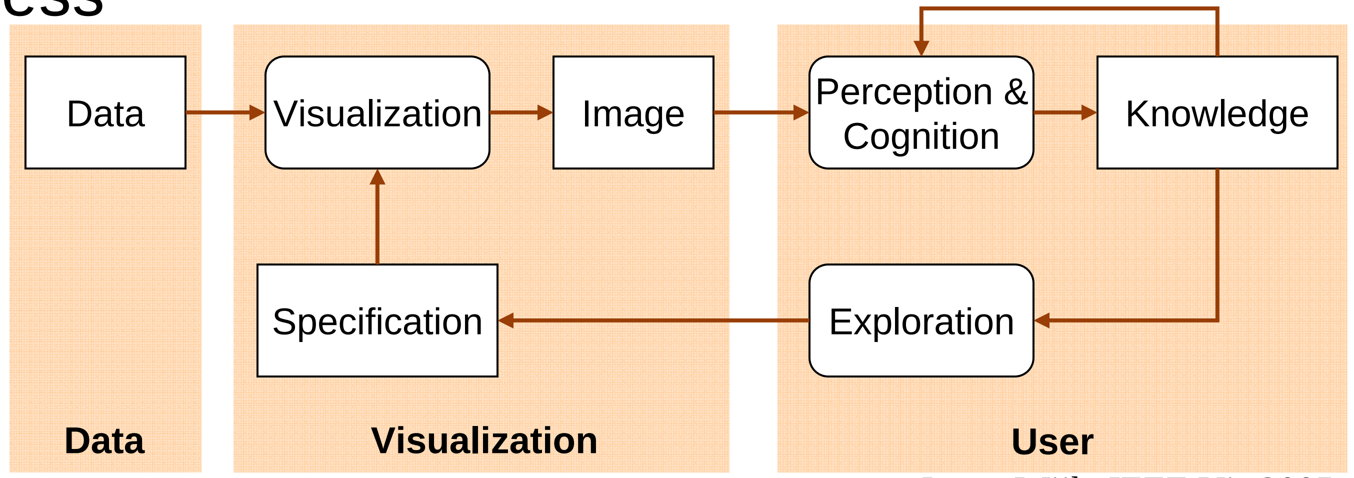
- Environmental observation and forecasting system (EOFS)
  - –Combine real-time sensor measurements with advanced computer models to describe complex, and dynamic environmental systems – focus on the Columbia River
- Initially: goal was to develop 3D visualizations
- *Look at visualization from an information management perspective*





Data Exploration through Visualization

- Hard to make sense out of large volumes of raw data, e.g., sensor feeds, simulations, MRI scans
- Insightful visualizations help analyze and validate various hypothesis
- But creating a visualization is a complex, iterative process



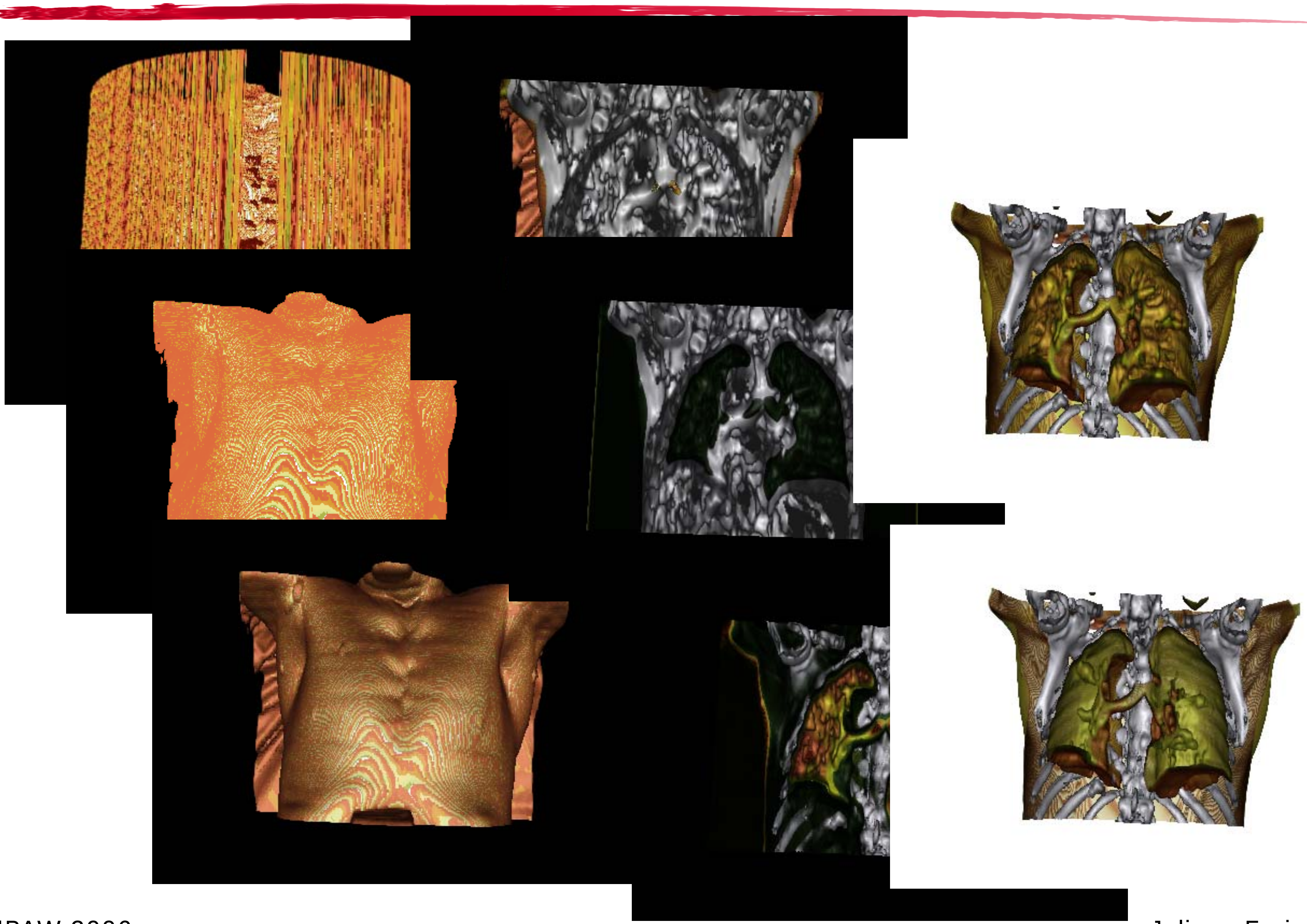
J. van Wijk, IEEE Vis 2005

Visualization Systems: State of the Art

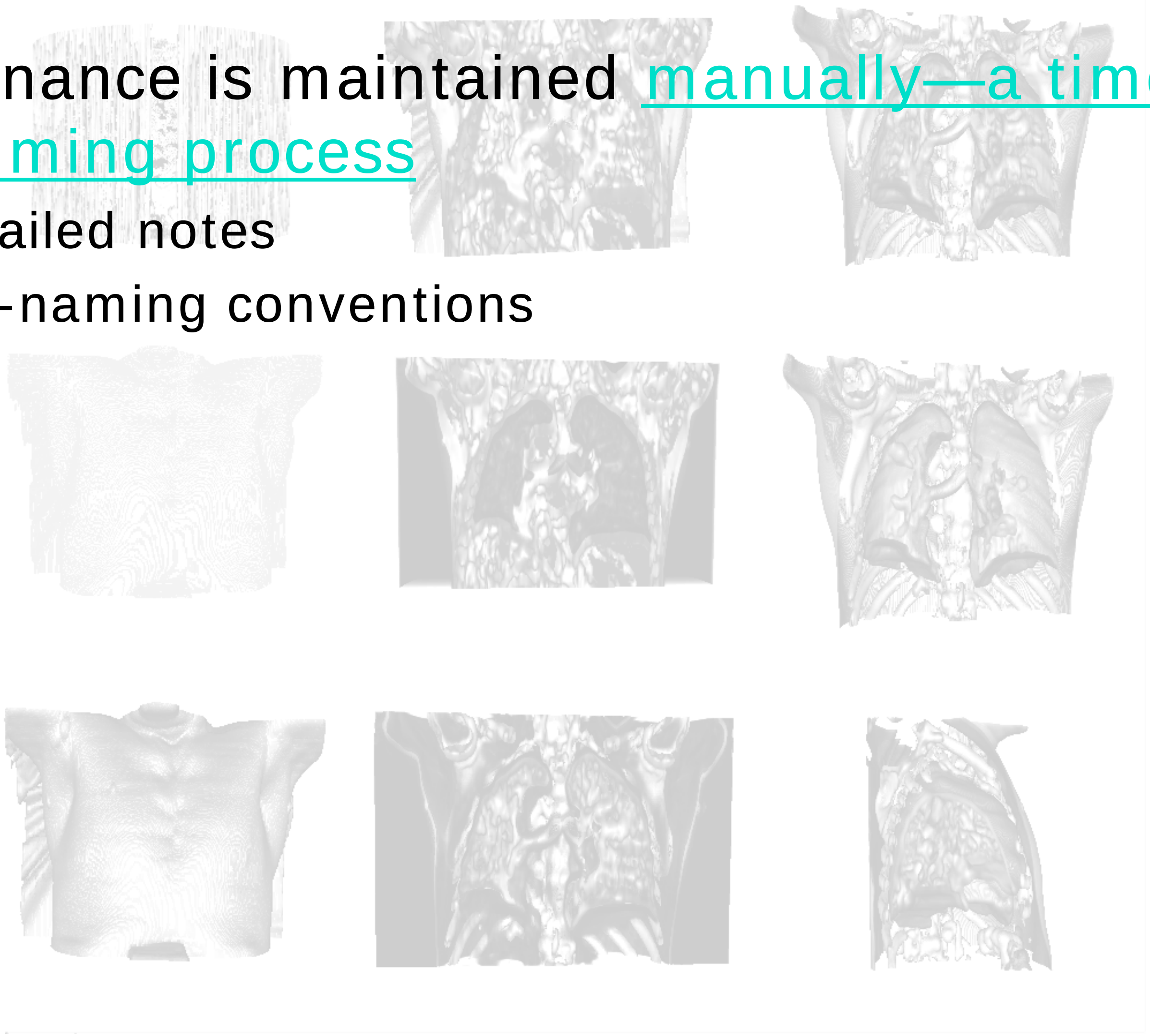
- Interactive creation and manipulation of visualizations
- Systems: SCIRun, ParaView/VTK
- Visual programming for creating *visualization pipelines—dataflows of visualization operations*
- Hard to create and compare a *large number* of visualizations
- Limitations:
  - No separation between the specification of a dataflow and its instances
  - Destructive updates—no provenance tracking mechanism
  - Users need to manage data and metadata

The generation and maintenance of visualizations is a major bottleneck in the scientific process

Example: Visualizing Medical Data



Issues in Visualizing Data

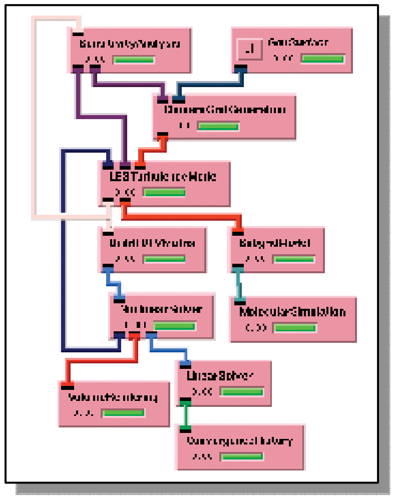


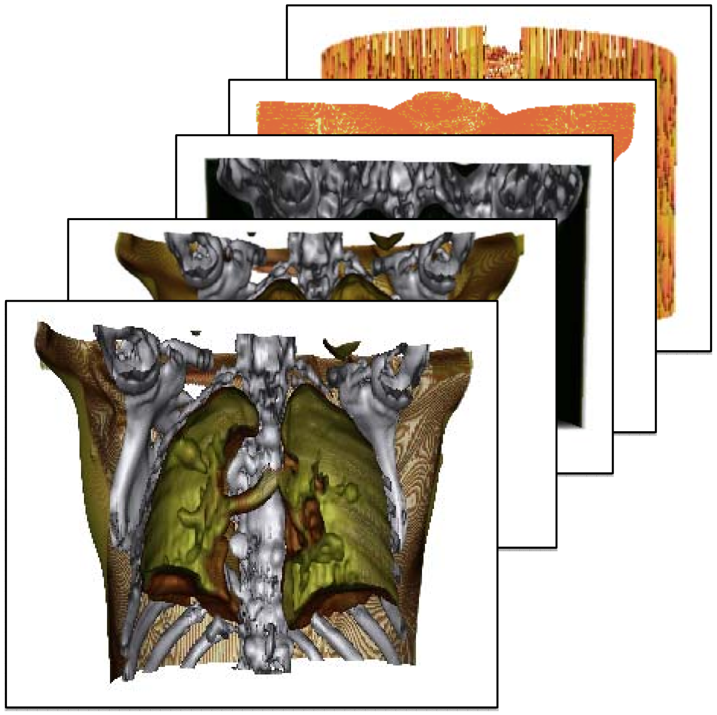
- Provenance is maintained manually—a time-consuming process
  - Detailed notes
  - File-naming conventions

Provenance Captured Manually

raw data

dataflow





anon4877_voxel_scale_1_zspace_20060331.srn

anon4877_textureshading_20060331.srn

anon4877_textureshading_plane0_20060331.srn

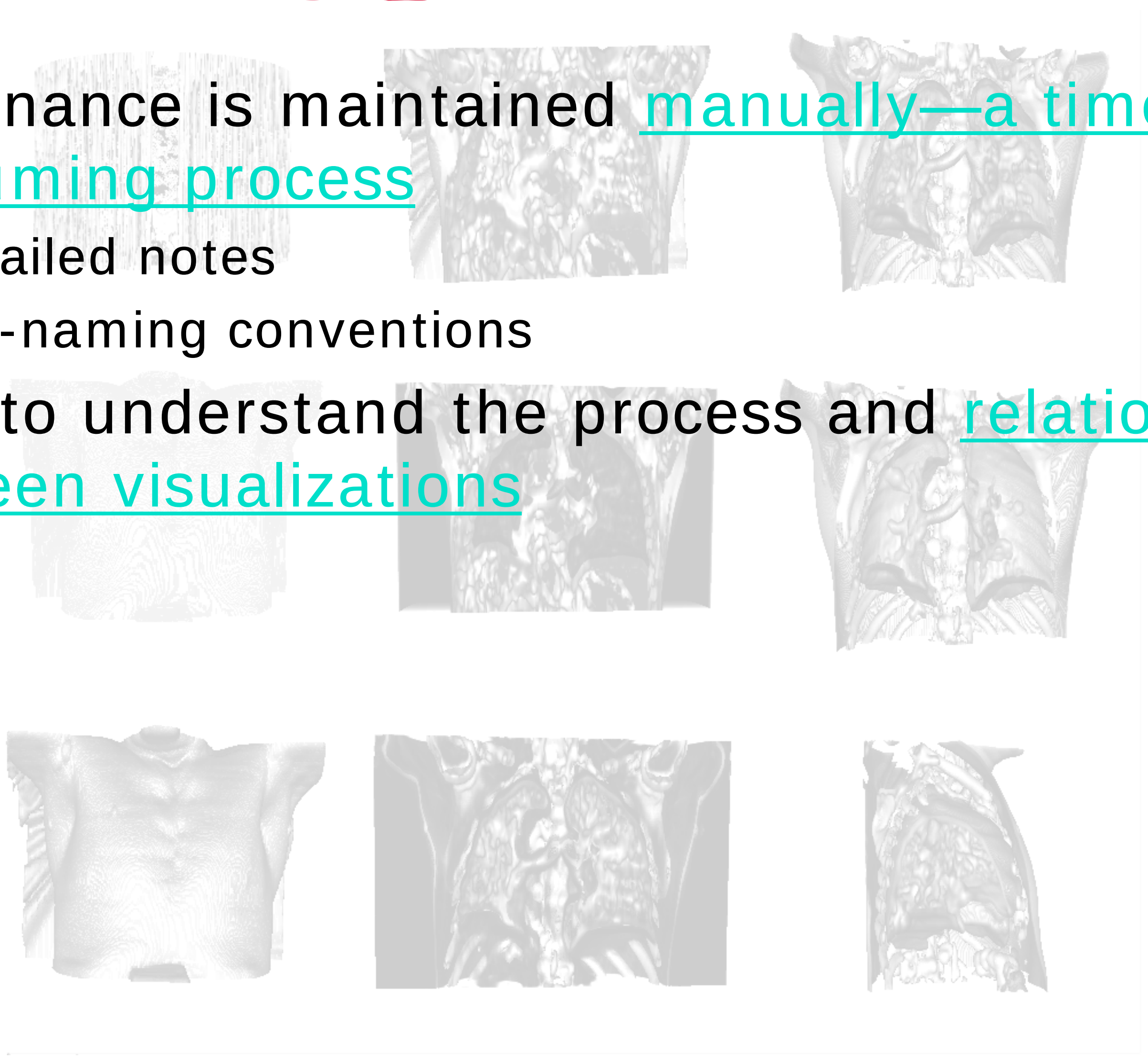
anon4877_goodxferfunction_20060331.srn

anon4877_lesion_20060331.srn

Files

Notes

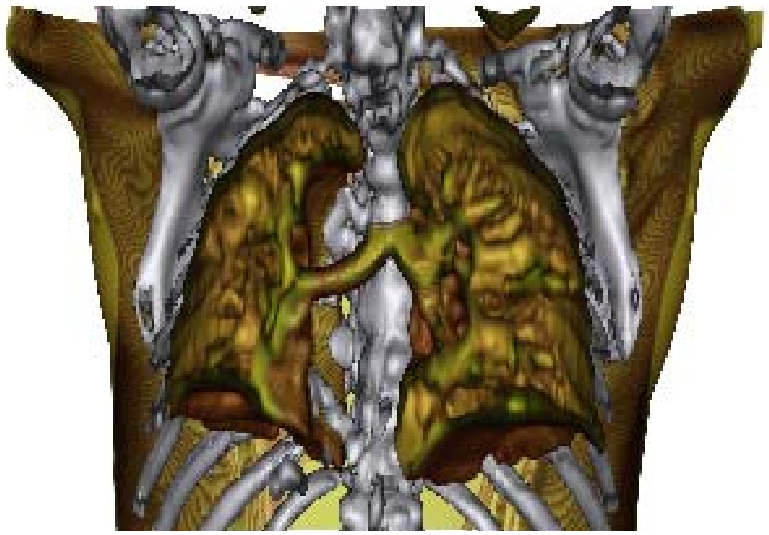
Issues in Visualizing Data



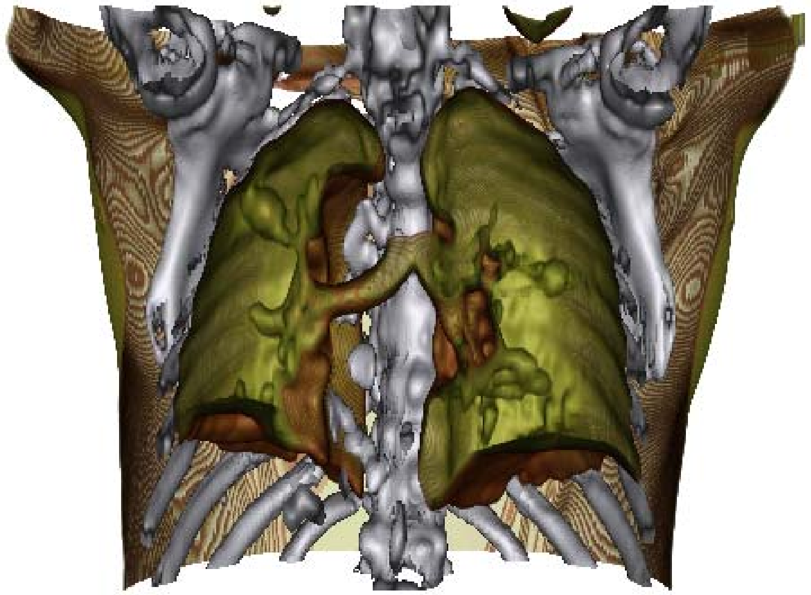
- Provenance is maintained manually—a time-consuming process
  - Detailed notes
  - File-naming conventions
- Hard to understand the process and relationships between visualizations

What’s the difference?

anon4877_base_20060331.srn



anon4877_lesion_20060401.srn

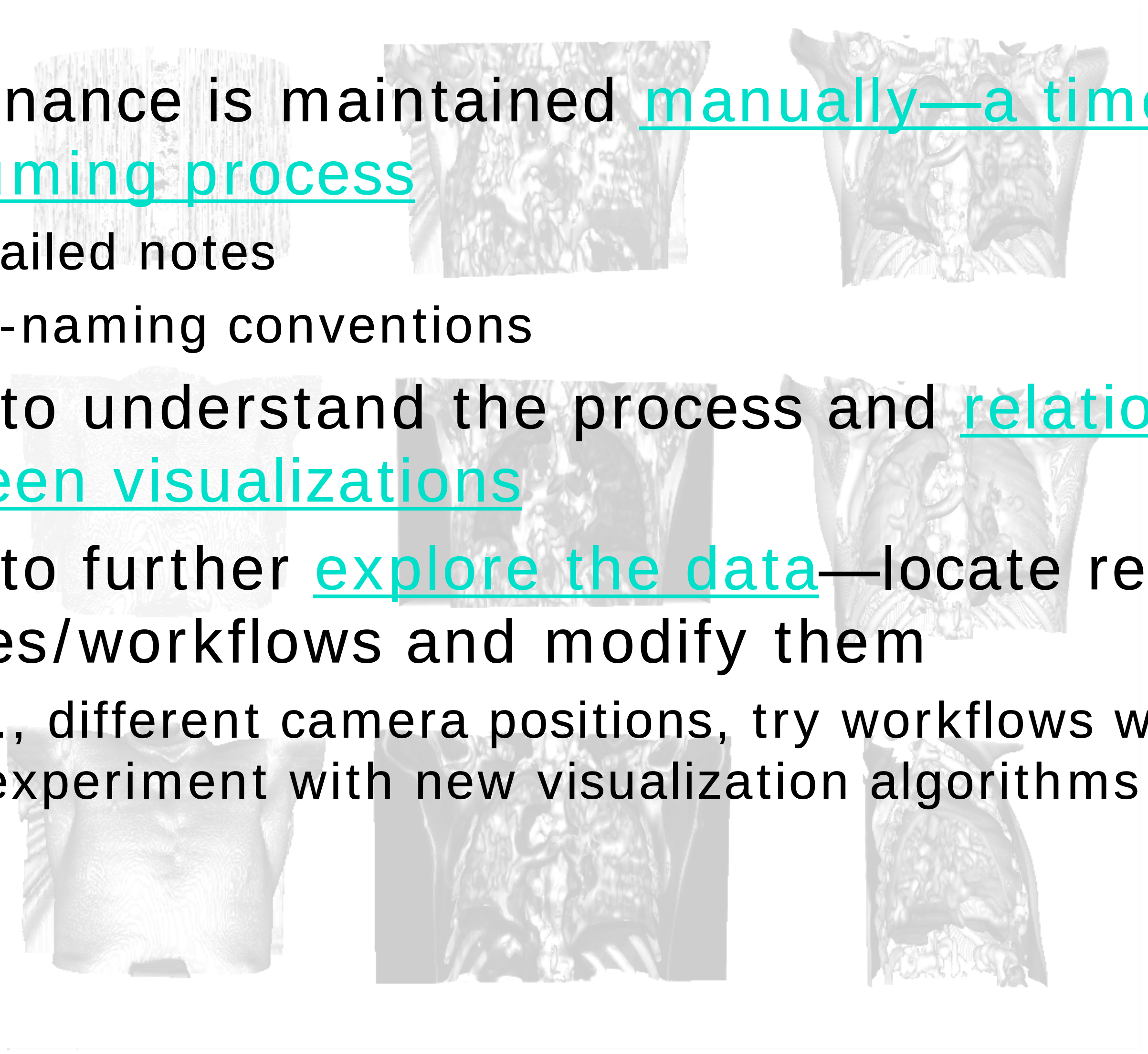


How were these images created?

Are they really from the same patient?

Do they use the same colormaps?

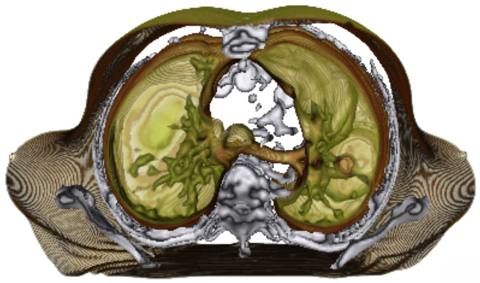
Issues in Visualizing Data

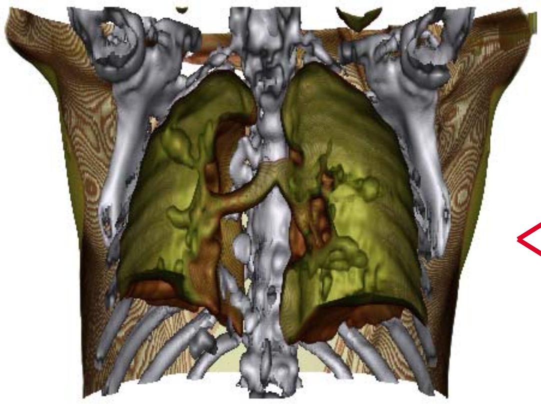


- Provenance is maintained manually—a time-consuming process
  - Detailed notes
  - File-naming conventions
- Hard to understand the process and relationships between visualizations
- Hard to further explore the data—locate relevant images/workflows and modify them
  - E.g., different camera positions, try workflows with new data, or experiment with new visualization algorithms

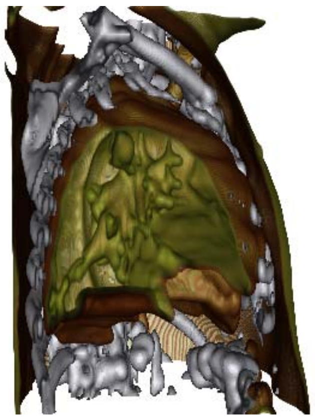
Exploring the Data

axial





sagital



coronal

Breathing cycle

VisTrails: Managing Visualizations

- Streamlines the creation, execution and sharing of complex visualizations
  - VisTrails manages the data and the exploration process, scientists can focus on *science!*
  - *“Reduce the time to insight” (Bill Gates, 2006)*
- Key differentiators:
  - Infrastructure for collaborative data exploration through visualization
  - Systematic maintenance of visualization *provenance*: akin to an electronic lab notebook
  - Interactive comparative visualization
- Not a replacement for visualization (or scientific workflow systems): provides infrastructure that can be combined with and enhance these systems
- Many important applications—some ongoing collaborations:
  - OHSU (environmental observation and forecasting systems); Harvard Medical School (radiation oncology); UCSD (biomedical informatics)

Outline

demonstration

- Vistrail = Evolving Dataflow
- Action-Based Provenance
- Streamlining Data Exploration
- Interacting with Provenance Information
- System: Architecture and Implementation
- Ongoing and Future Work

VisTrails

Evolving dataflow

Link to video:
http://www.cs.utah.edu/~juliana/talks/videos/vistrails_evolvingdataflow_spx.avi

Action-Based Provenance

- Records user interactions with workflows
- Workflow evolution is captured in a *vistrail*—a rooted tree where
  - *nodes* correspond to workflow versions
  - *edges* correspond to actions that transform the parent into the child workflow
- Action algebra:
  - addModule, deleteModule, addConnection, deleteConnection, setParameter, ...
  - Can be easily extended, e.g., addDirector for Ptolemy-based systems

Action-Based Provenance

- Records user interactions with workflows
- Workflow evolution is captured in a *vistrail*—a rooted tree where
  - *nodes* correspond to workflow versions
  - *edges* correspond to actions that transform the parent into the child workflow
- Action algebra:
  - add ction, setF
  - Can ased syst

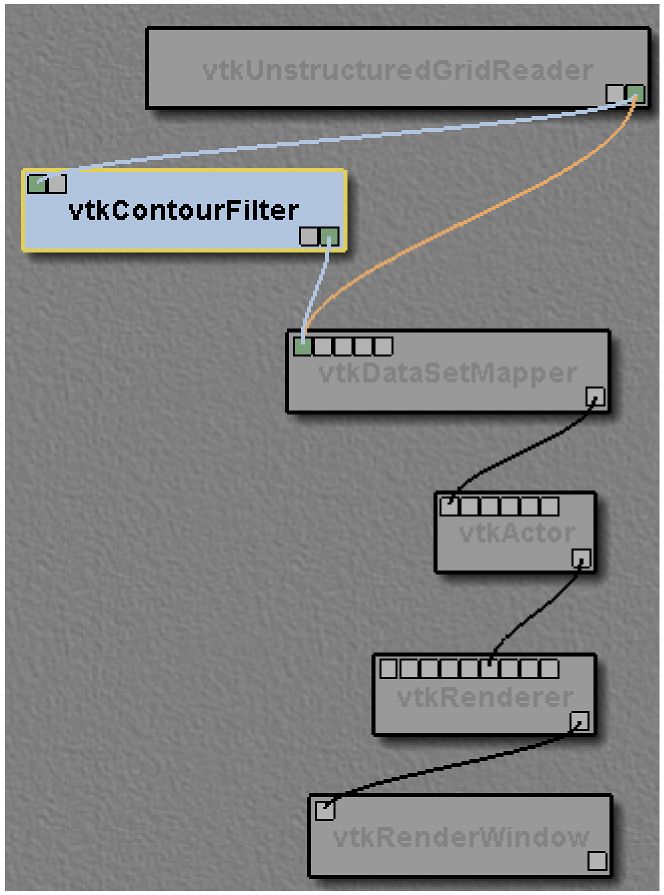
```
type Vistrail = vistrail [ @id, @name, Action*, annotation? ]

type Action = action [ @parent, @time, tag?, annotation?, @userId,

  (AddModule|DeleteModule|ReplaceModule|

 AddConnection|DeleteConnection|SetParameter|…)]
```

Action-Based Provenance: Example



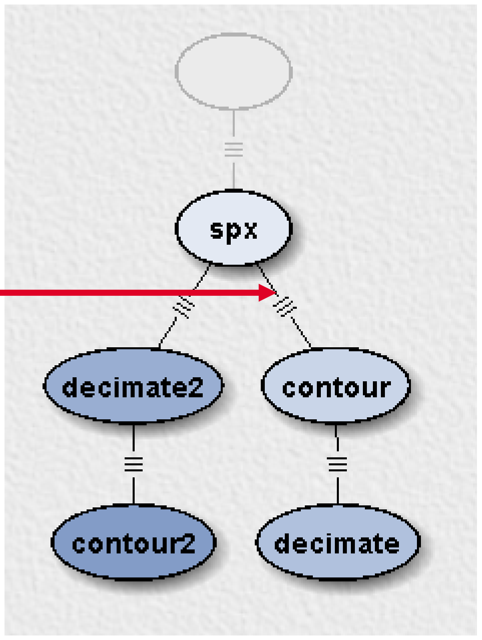
addModule

deleteConnection

addConnection

addConnection

setParameter



Action-Based Provenance: Example

```
<action date="" parent="25" time="26“ user=“juliana">
<addModule>
   <object cache="1" id="5" name="vtkContou
</addModule> </action>
<action date="" parent="26" time="27"
   <deleteConnection connectionId="0"/>
</action>
<action date="" parent="27" time="28" use
   <addConnection connect id="0">
        <filterInput destId="5" destPort="0" sourceId="0"
sourcePort="0"/>
   </addConnection> </action>
<action date="" parent="28" time="29" user=“juliana“>
   <addConnection connect id="4">
        <filterInput destId="1" destPort="0" s
sourcePort="0"/>
      </addConnection> </action>
<action date="" parent=“29" time="30" user="" >
   < changeParameter>
      <set function="SetValue" functionId="0"
moduleId="5" parameter="(unnamed)" param
type="int" value="0"/>
      <set function="SetValue" functionId="0"
moduleId="5" parameter="(unnamed)" parameterId="1"
type="float" value="0.5"/>
 </ changeParameter> </action>
```

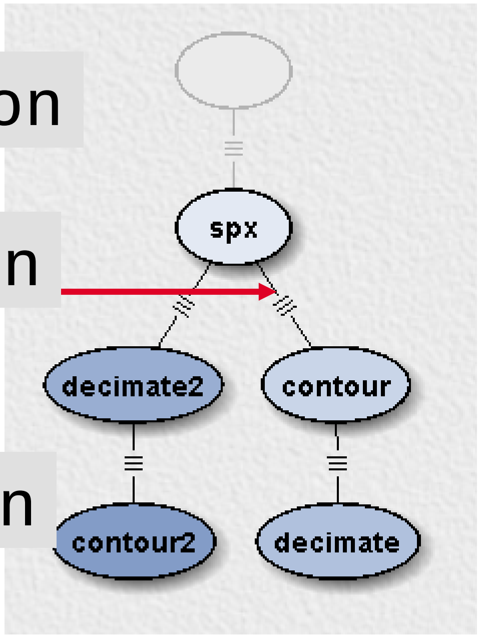
addModule

deleteConnection

addConnection

addConnection

setParameter



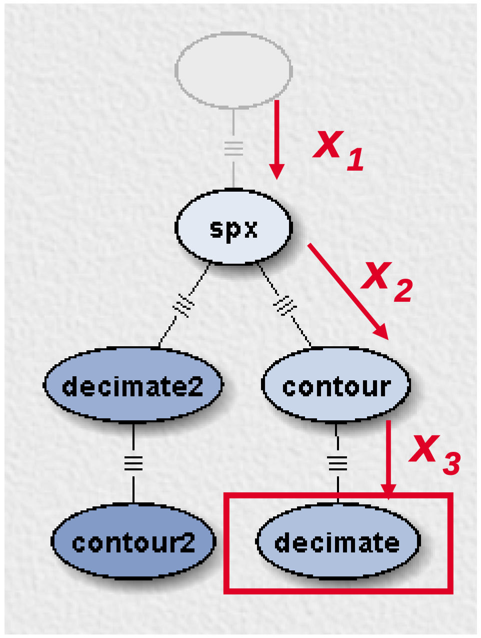
Action-Based Provenance: Formalism

- Let
  - DF be the set of all possible dataflow instances, s.t. $\varnothing \in$ DF
  - x_i: DF→DF be a function that transforms a dataflow $x_i(D_a) = D_b$
- A vistrail node v_t corresponds to the dataflow that is constructed by the sequence of actions from the root to v_t

$$v_t = x_n \circ x_{n-1} \circ \ldots \circ x_1 \circ \varnothing$$

- Vistrail nodes are partially ordered
  - Given v_i and v_j, if v_j is created by applying a sequence of actions to v_i, $v_i < v_j$

Dataflow = sequence of actions



$$\text{decimate} = x_3 \circ x_2 \circ x_1 \circ \emptyset$$

Action-Based Provenance: Summary

- Uniformly captures both data and process provenance
- Records user actions—compact representation
- Detailed information about the exploration process
  - Results can be reproduced
  - Scientists can return to any point in the exploration space
- *Version tree structure enables scalable exploration of the dataflow parameter space*

Provenance and Data Exploration

Useful operations through direct manipulation of version tree:

- Macros: re-use actions for repetitive tasks
- Bulk updates: quickly explore slices of parameter space
- Workflow diffs: visually compare different workflow versions
- Distributed collaboration: groups can collaborate to create visualizations

Macros: Reusing Provenance

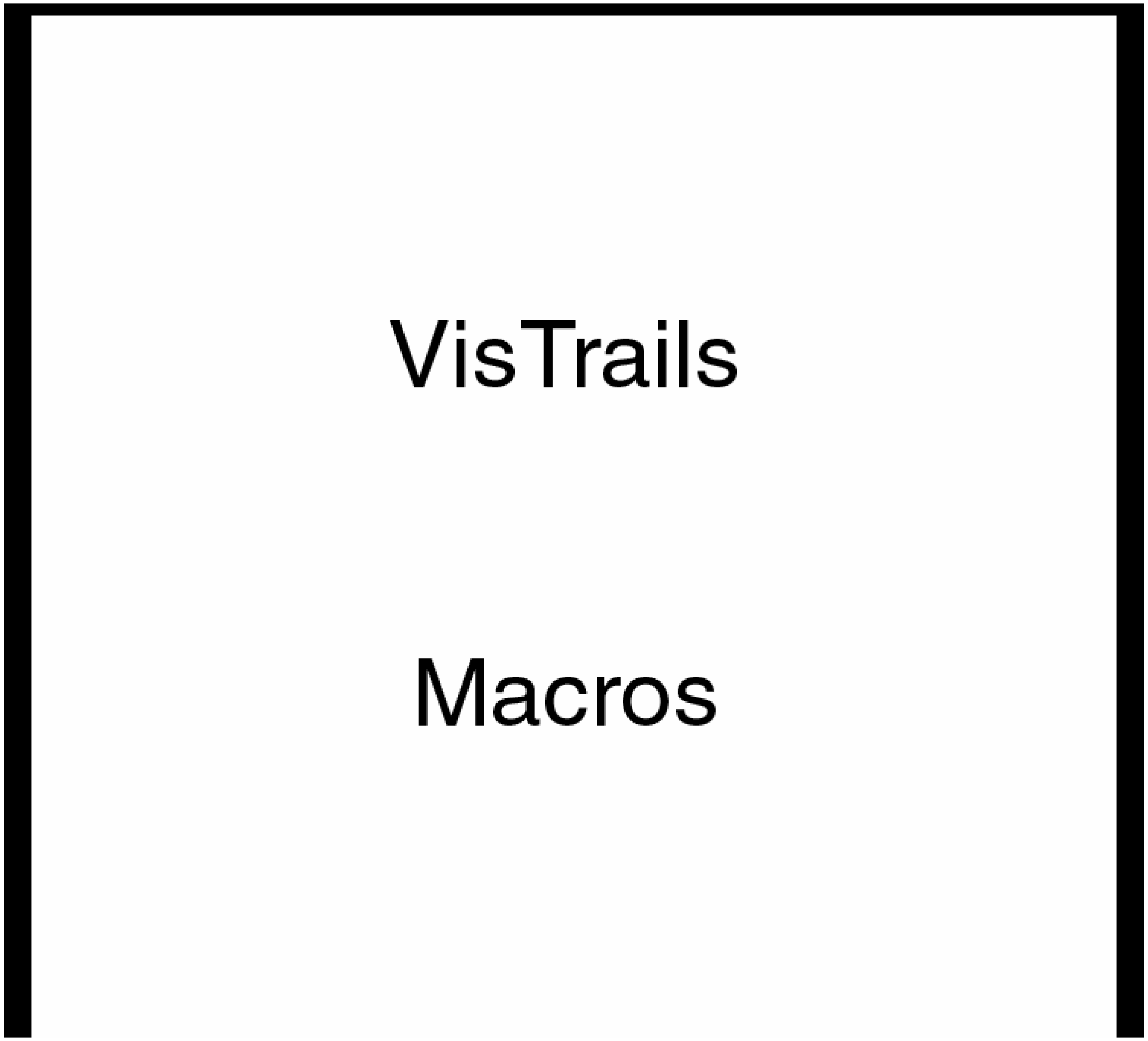
- A macro corresponds to modules and connections—a dataflow fragment
- Represented as a sequence of actions

$$x_j \circ x_{j-1} \circ \ldots \circ x_i$$

- Creating a macro
  - Record a sequence of actions ⟵ implemented
  - Nodes selected from version tree
  - Select dataflow fragment
- Applying a macro to a vistrail node $\boldsymbol{v_t}$

$$x_j \circ x_{j-1} \circ \ldots \circ x_i \circ \boldsymbol{v_t}$$

- Users set parameters and connect the inputs and outputs
  - May be automated in some cases



Link to video:
http://www.cs.utah.edu/~juliana/talks/videos/vistrails_macros.avi

Scalable Derivation of Visualizations

- Scripting dataflows: Bulk updates are simple to specify and apply
- Exploration of parameter space for a workflow $\boldsymbol{v_t}$

$(setParameter(id_n,value_n) \circ \ldots \circ (setParameter(id_1,value_1) \circ \boldsymbol{v_t})$

- Exploration of multiple workflow specifications

$(addModule(id_i,\ldots) \circ (deleteModule(id_i) \circ \boldsymbol{v_1})$

…

$(addModule(id_i,\ldots) \circ (deleteModule(id_i) \circ \boldsymbol{v_n})$

- Results can be conveniently compared in the VisTrails spreadsheet
- Can create animations too!

VisTrails

Bulk updates

Link to video: **http://www.cs.utah.edu/~juliana/talks/videos/vistrails_bulkupdates.avi**

VisTrails

Generating animations

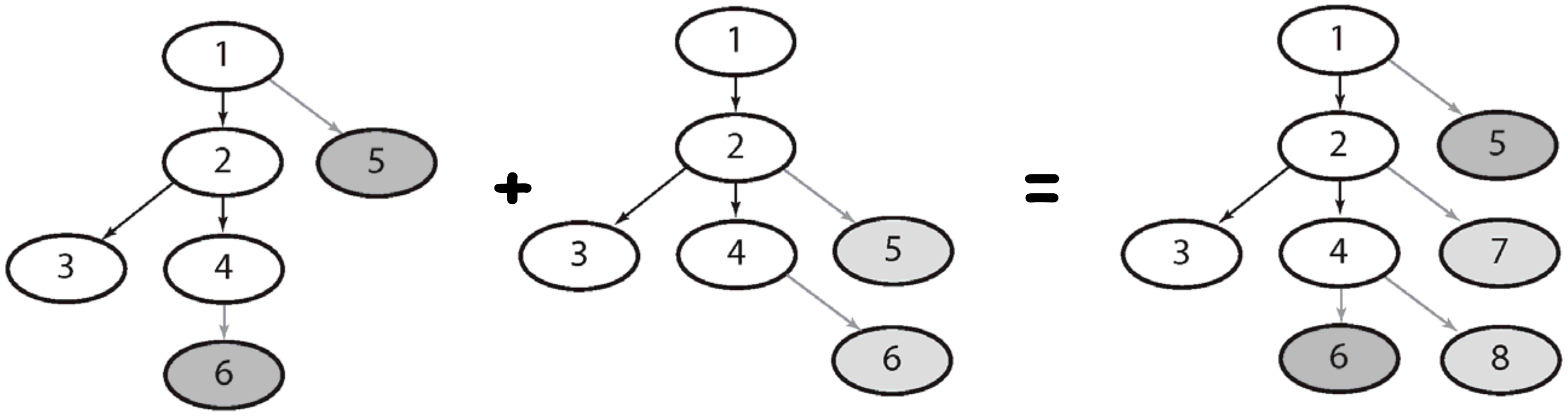
Link to video: http://www.cs.utah.edu/~juliana/talks/videos/vistrails_animation.avi

Collaborative Visualization

- Collaboration is key to data exploration
  - Translational, integrative approaches to science
- Central repository: store information in a database
- Synchronize concurrent updates through locking
- Asynchronous access: similar to version control systems
  - Check out, work offline, synchronize
  - Users exchange patches

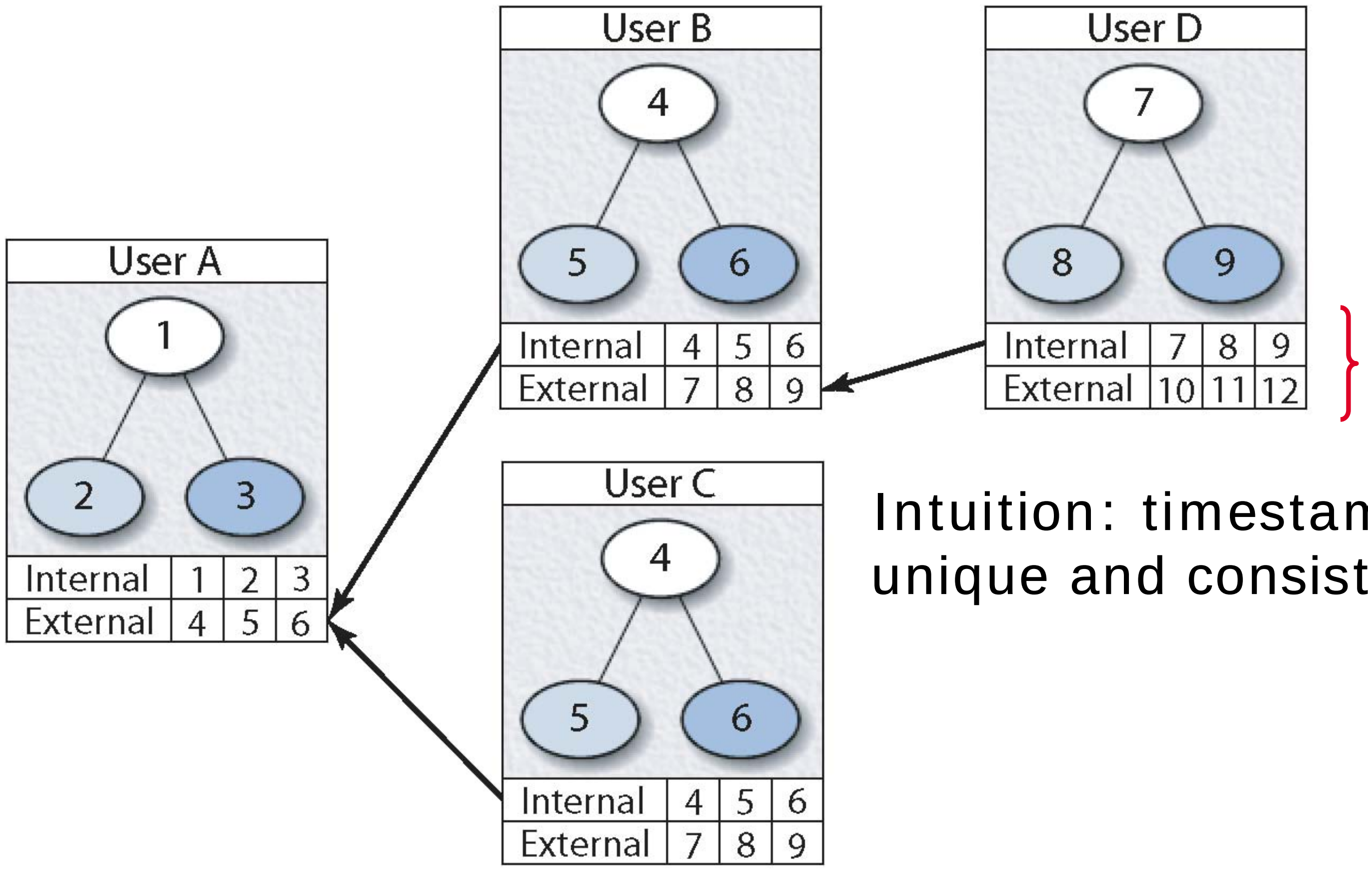
Vistrail Synchronization

- Version tree is *monotonic*
  - *Actions are always added, never deleted*
- Merging two vistrails is simple



Hierarchical Synchronization

- No need for a central repository—can do distributed collaboration



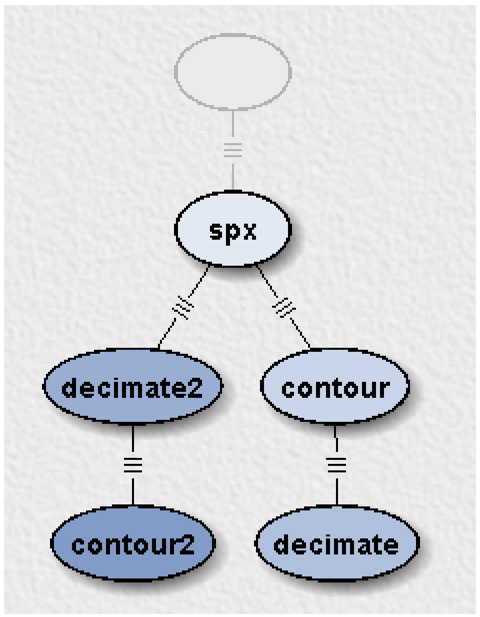
Relabelling map

Intuition: timestamps need to be unique and consistent, but only locally

See Callahan et al, SCI Institute Technical Report, No. UUSCI-2006-016 2006

Interacting with Provenance Information

- Storing detailed information is important
- Need appropriate user interface to
  - leverage information, and
  - deal with the information overload
- Understanding the history
  - Different colors for different users
  - Node age represented by saturation level



Interacting with Provenance Information

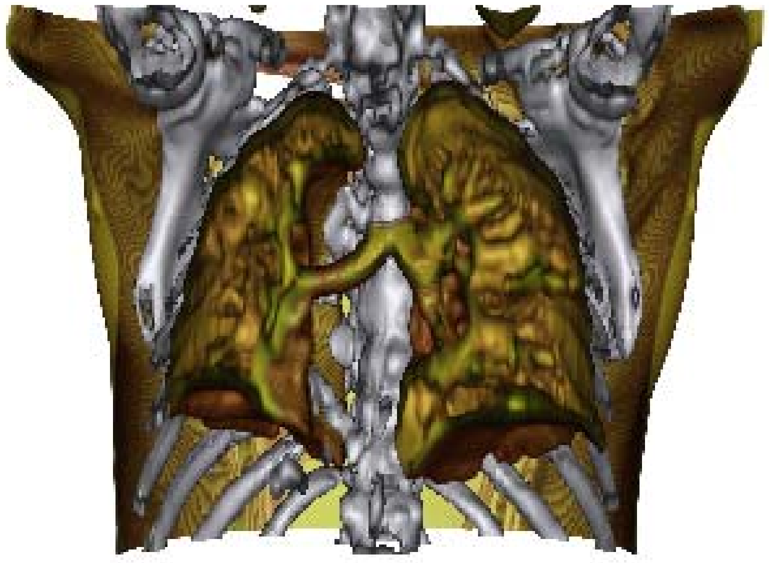
- Storing detailed information is important
- Need appropriate user interface to
  - leverage information, and
  - deal with the information overload
- Understanding the history
  - Different colors for different users
  - Node age represented by saturation level
- Create *views* over the version tree
  - Tagged nodes
  - Search and query

⬅ **Demo**

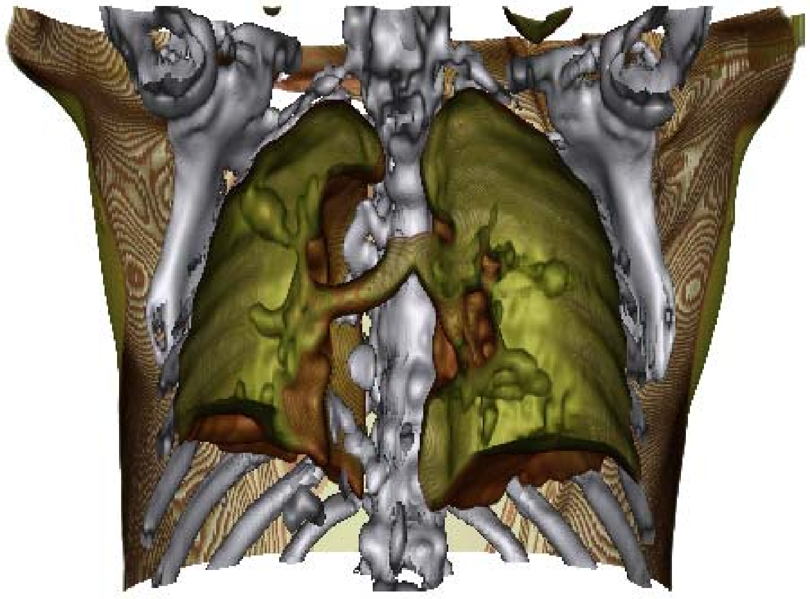
Interacting with Provenance Information

- Storing detailed information is important
- Need appropriate user interface to
  - leverage information, and
  - deal with the information overload
- Understanding the history
  - Different colors for different users
  - Node age represented by saturation level
- Create *views* over the version tree
  - Tagged nodes
  - Search and query
- Understanding the exploratory process
  - Visual workflow diff

What’s the difference?

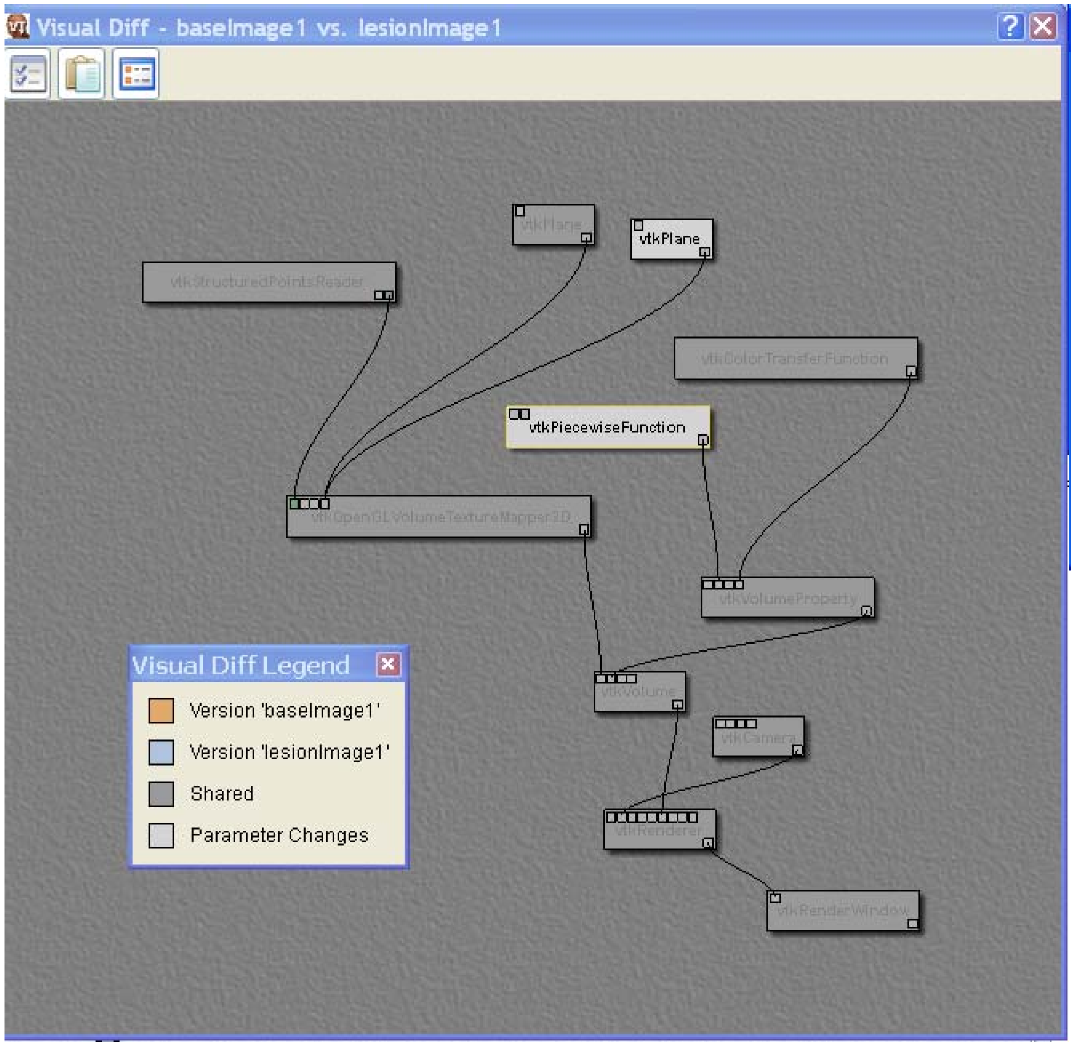


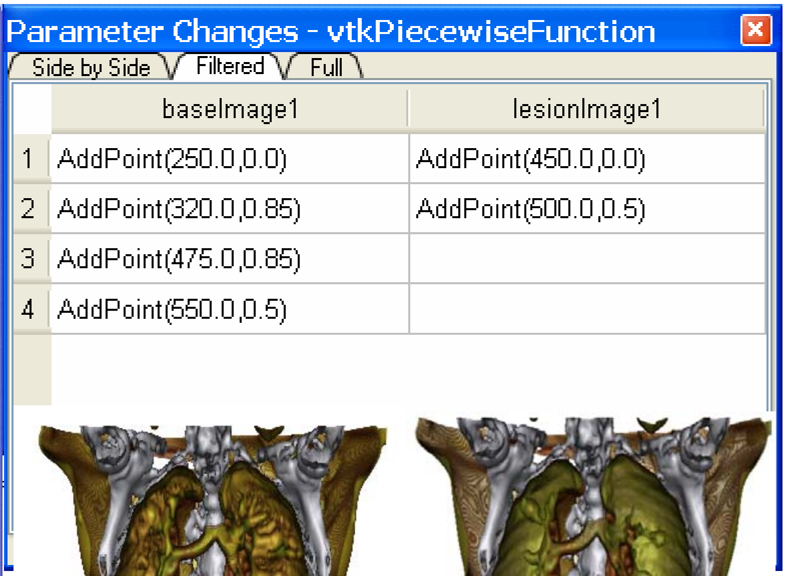
baseImage1



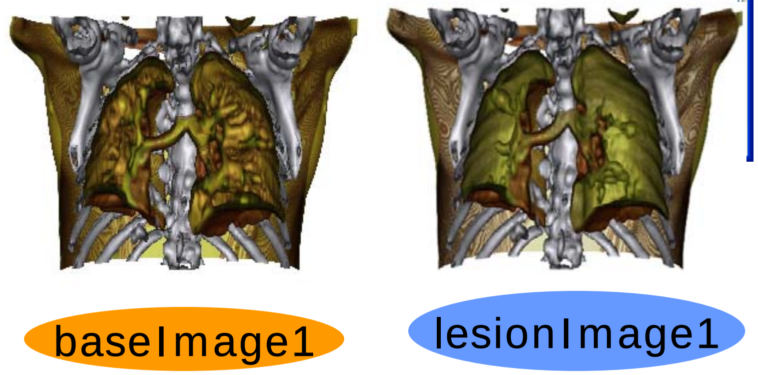
lesionImage1

What’s the difference?

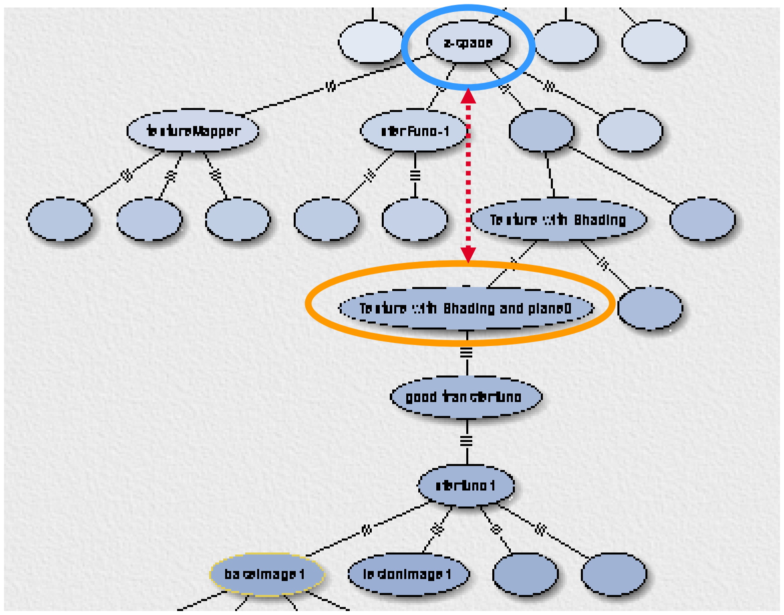


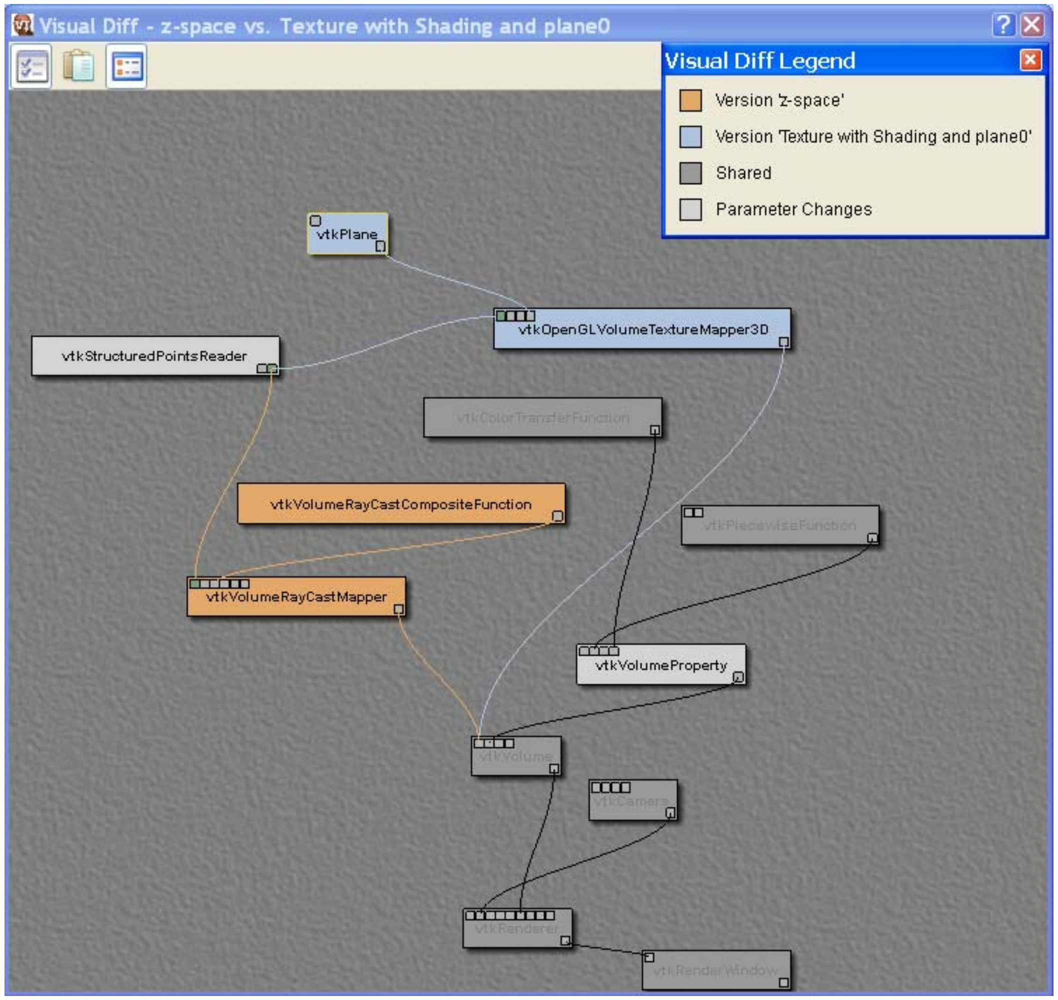


	baseImage1	lesionImage1
1	AddPoint(250.0,0.0)	AddPoint(450.0,0.0)
2	AddPoint(320.0,0.85)	AddPoint(500.0,0.5)
3	AddPoint(475.0,0.85)	
4	AddPoint(550.0,0.5)	



Differences in Specification





Dataflow Diff

- Vistrail is a rooted tree: all nodes have a common ancestor—diffs are well-defined

$vt_1 = x_i \circ x_{i-1} \circ \ldots \circ x_1 \circ Ø$

$vt_2 = x_j \circ x_{j-1} \circ \ldots \circ x_1 \circ Ø$

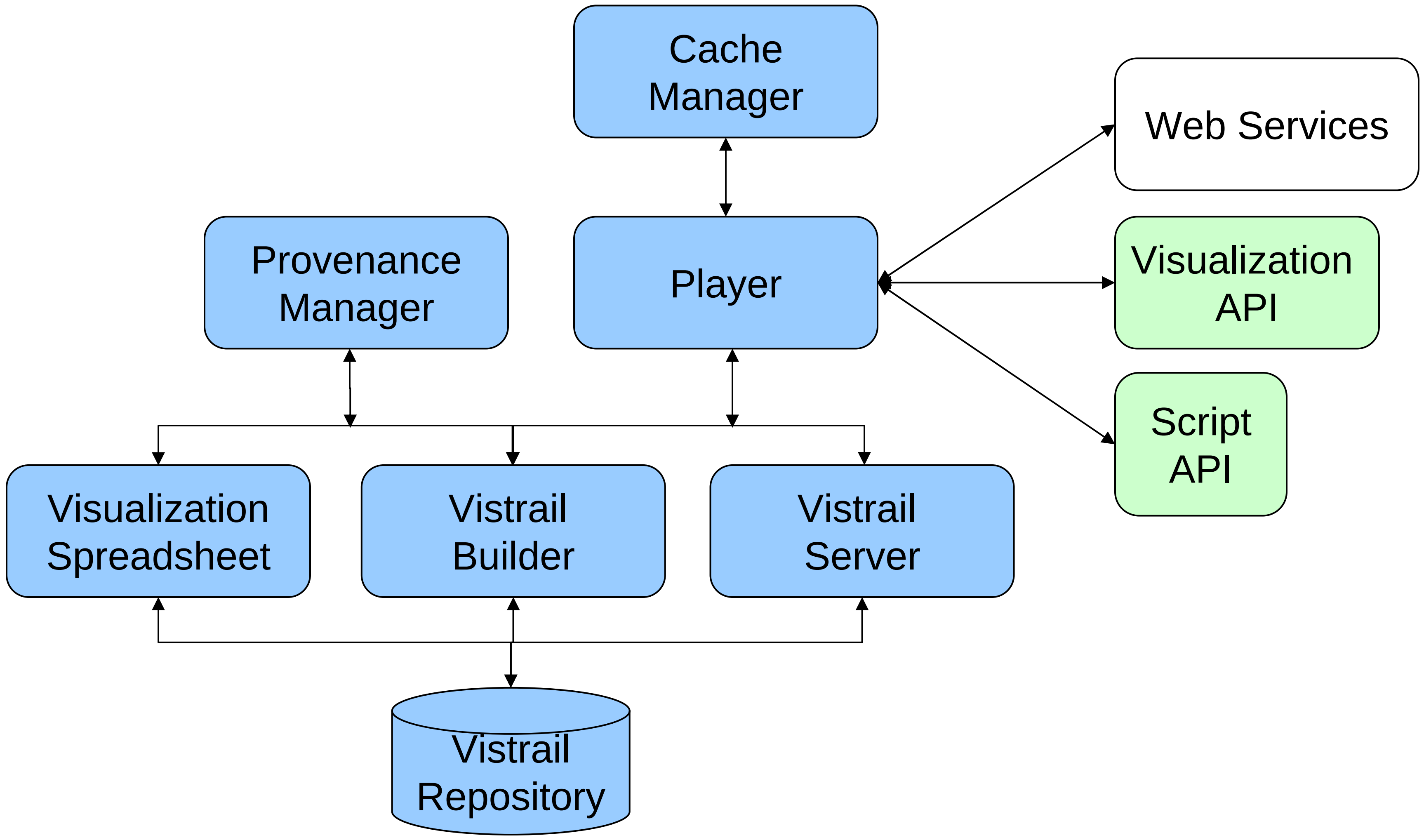
$vt_1\text{-}vt_2 = \{x_i, x_{i-1}, \ldots, x_1, Ø\} - \{x_j, x_{j-1}, \ldots, x_1, Ø\}$

- Different semantics:
  - Exact, based on ids
  - Approximate, based on module/connection signatures

Outline

- Vistrail = Evolving Dataflow
- Action-Based Provenance
- Streamlining Data Exploration
- Interacting with Provenance Information
- System: Architecture and Implementation
- Ongoing and Future Work

VisTrails Architecture

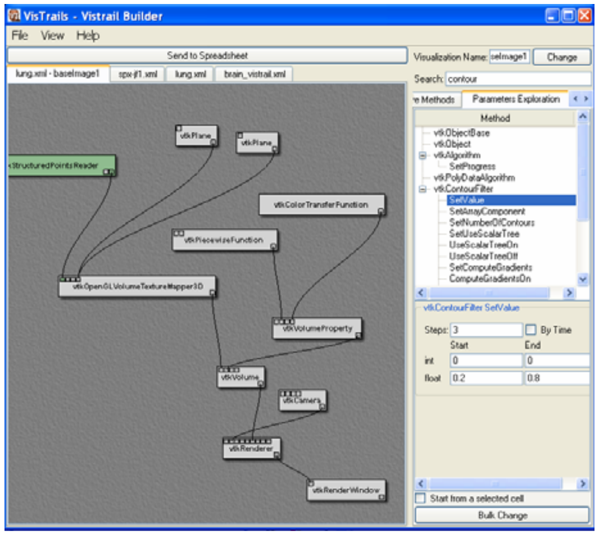


VisTrails Implementation

- Code written in Python (~20k lines)
  - Extensibility—easy to include new modules
  - Cool feature: Workflows can be exported as Python scripts!
- GUI for module interactions automatically generated
  - No additional code needed for Python or swigged apps
- Re-use open-source components: QT/PyQT, OpenGL, VTK
- Portability: Mac, Linux, Windows (even 64 bit!)
  - Also some bugs
- Repository: MySQL vs. eXist
- Simple workflow execution model—not our focus

VisTrails User Interface

VisTrails Builder



Visualization Name: selmage1

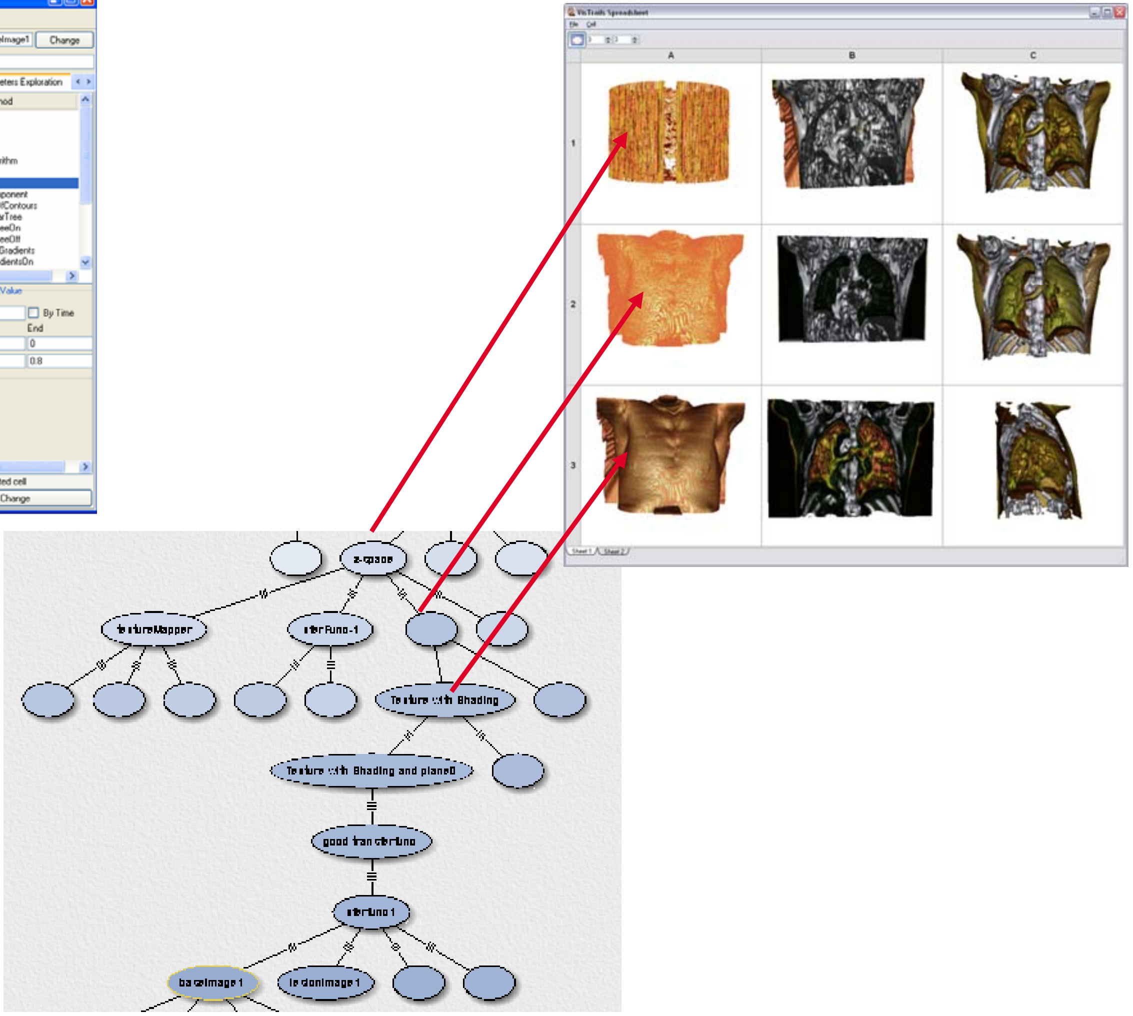
Search: contour

Steps: 3 ☐ By Time

	Start	End
int	0	0
float	0.2	0.8

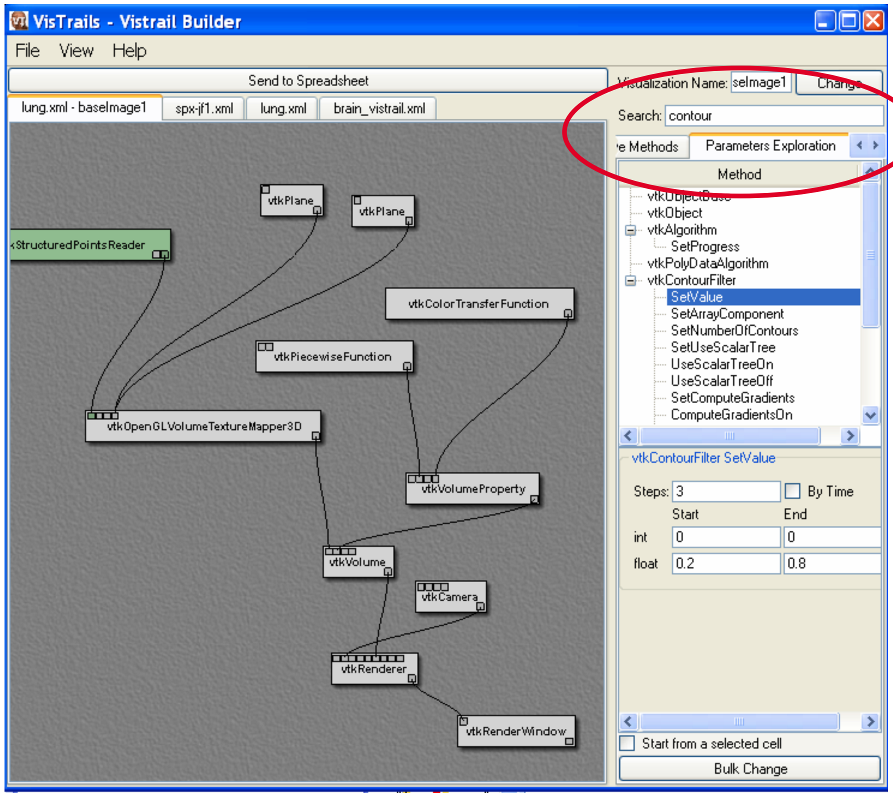
☐ Start from a selected cell

VisTrails Spreadsheet



VisTrails Version Tree

VisTrails User Interface: Search



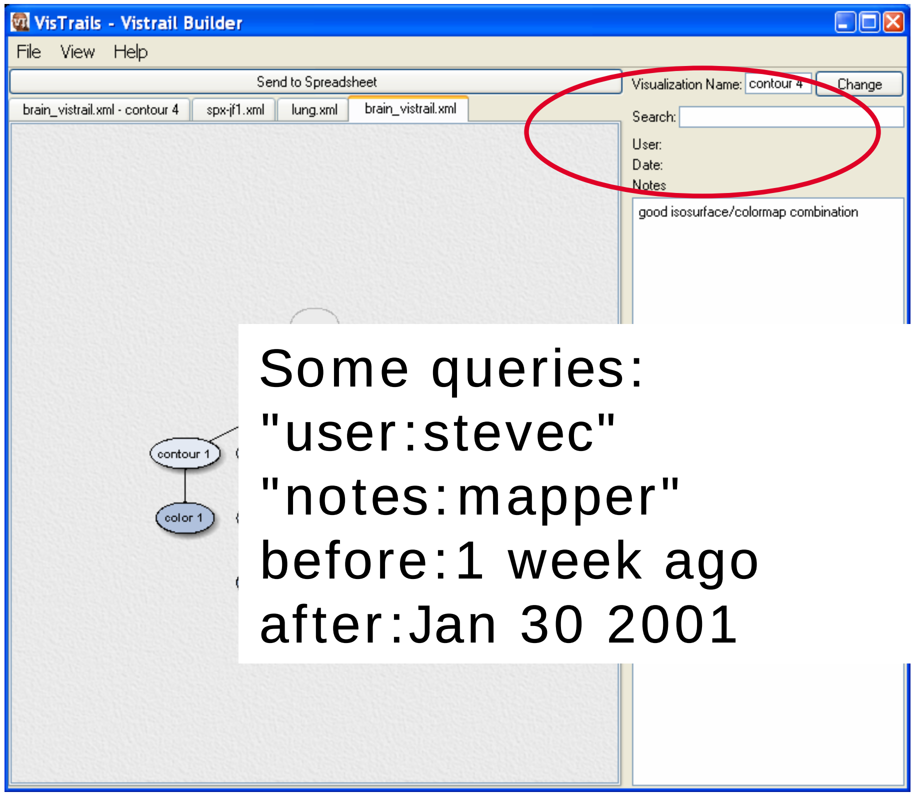
Visualization Name: selmage1

Search: contour

Steps: 3

	Start	End
int	0	0
float	0.2	0.8

Searching for modules



Visualization Name: contour 4

Search:

User:

Date:

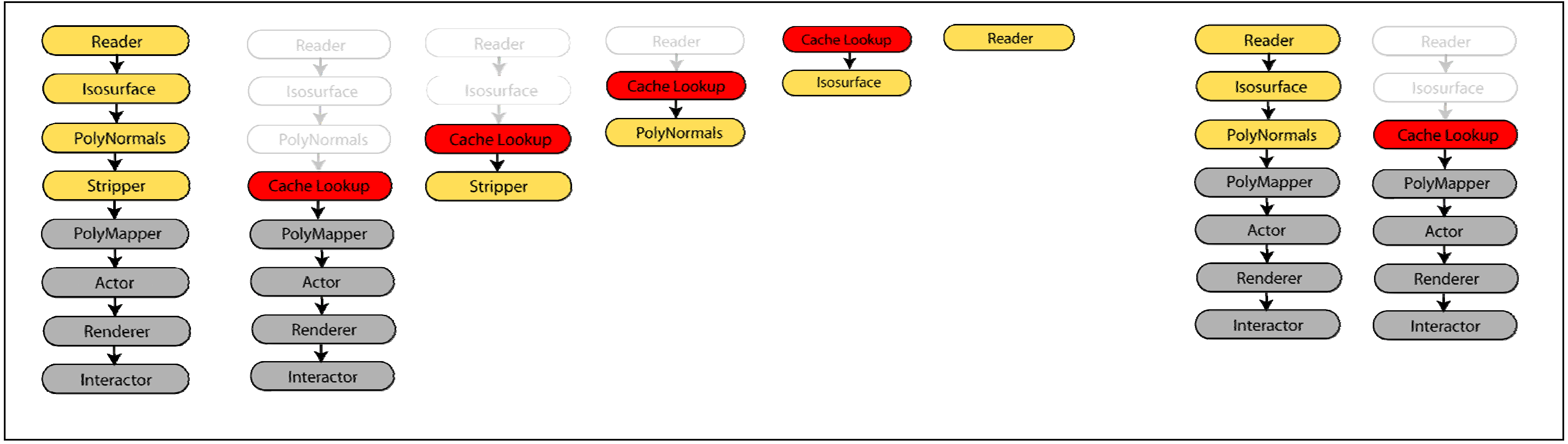
Notes:

good isosurface/colormap combination

Some queries:
"user:stevec"
"notes:mapper"
before:1 week ago
after:Jan 30 2001

Searching for dataflows

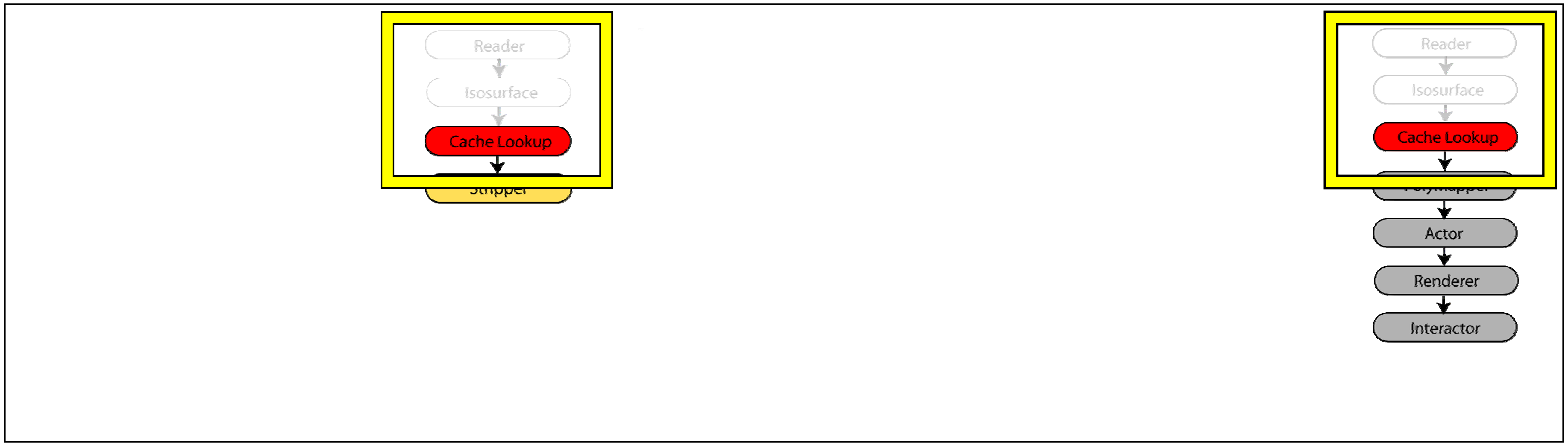
The Cache Manager



- Important for scalability
- The Cache Manager determines pipeline sharing
- Each module is broken into a series of subnetworks
- Each subnetwork receives a unique ID, comprising its modules, connectivity and parameters
- Results are linked to the ID, and only computed if missing in the cache

See Bavoil et al, IEEE Visualization, 2005

The Cache Manager



- Important for scalability
- The Cache Manager determines pipeline sharing
- Each module is broken into a series of subnetworks
- Each subnetwork receives a unique ID, comprising its modules, connectivity and parameters
- Results are linked to the ID, and only computed if missing in the cache

VisTrails: Summary

- A new system that enables interactive, multiple-view visualizations
- Simplifies the creation and maintenance of a large number of visualizations
- Detailed provenance of visualization results and *process*
- Streamlines execution through caching

Conclusions

- Identified the problem and proposed a solution for managing rapidly-evolving workflows
- Detailed data and process provenance automatically captured
- The VisTrails system

Streamlines the data exploration process

Enables collaborative and distributed exploration through visualization

And scientists can do (a lot of) it!

- Focus on visualization, but ideas are applicable to general workflows

Beyond Scientific Workflows

- Ideas useful in other domains
- Adobe Lightroom[1]
  - multiple-view visualization, non-destructive editing, synchronization (=bulk changes)
- Recent comment about WikiCalc in news.com[2]

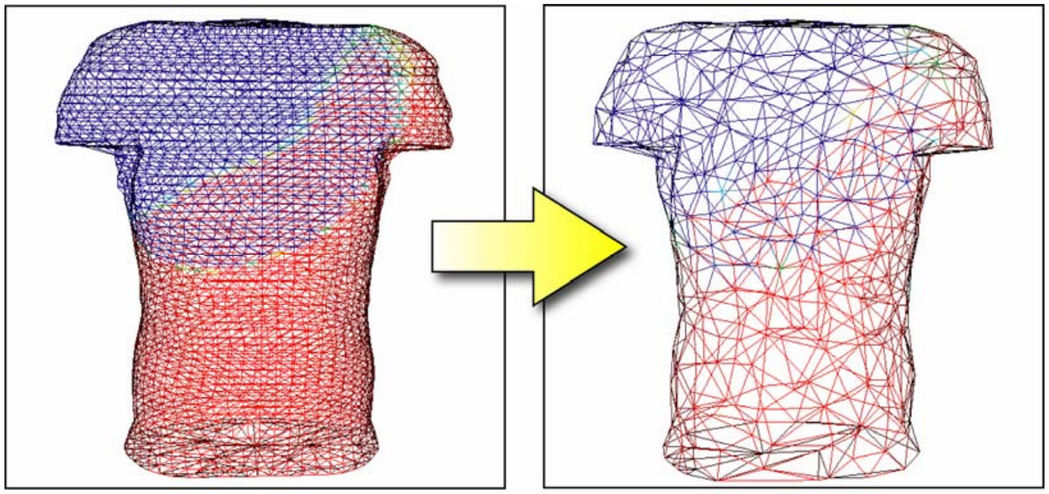
“spreadsheets have traditionally been a single-user application screaming for functionality that could let multiple people edit data quickly and easily”

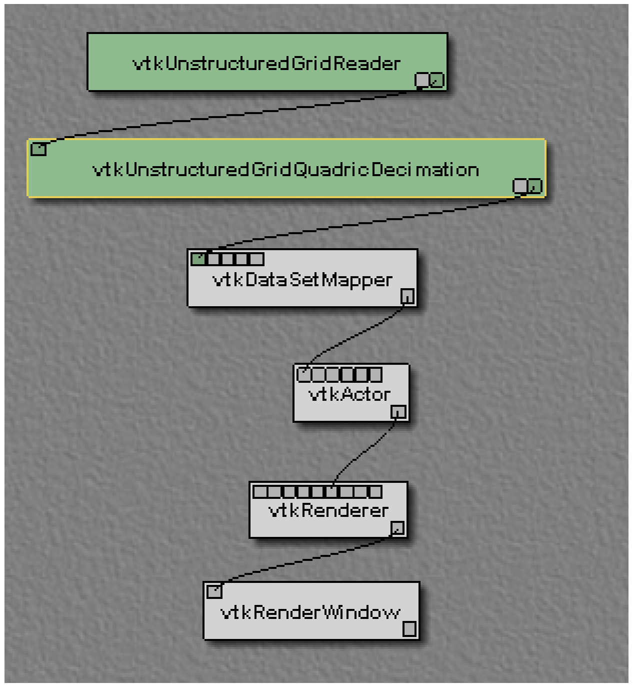
1. http://labs.macromedia.com/technologies/lightroom/video/overview/
2. http://news.com.com/Software+pioneer+Bricklin+tackles+wikis/2100-1032_3-6040867.html?tag=nefd.lede

Future

- Reproducible science
  - *Publish image/results and their associated workflows—deep annotations*
  - Track files, versions of systems (executables)—ensure reproducibility
- Train scientists
- Simplify scientific discovery: automate generation of data products

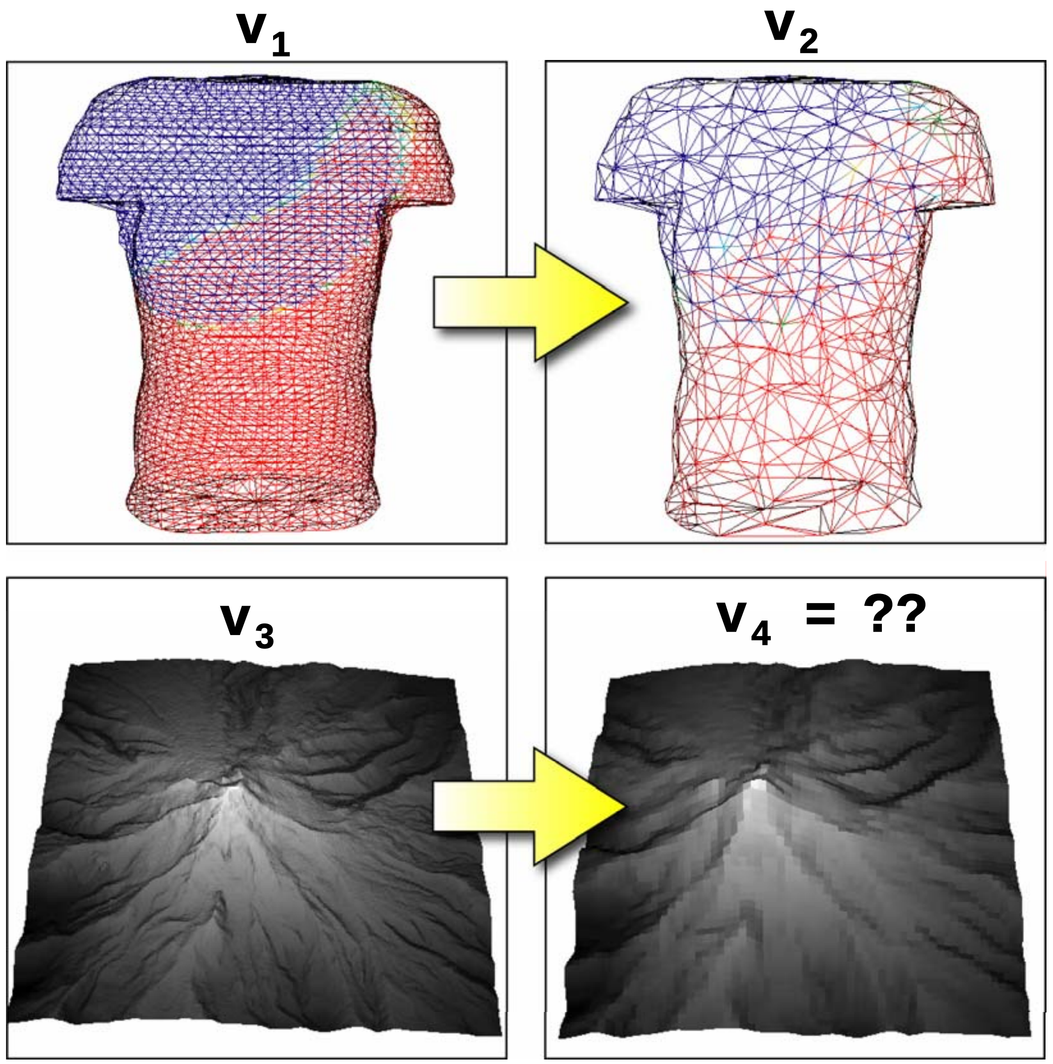
Automating Workflow Creation: Visualization by Analogy

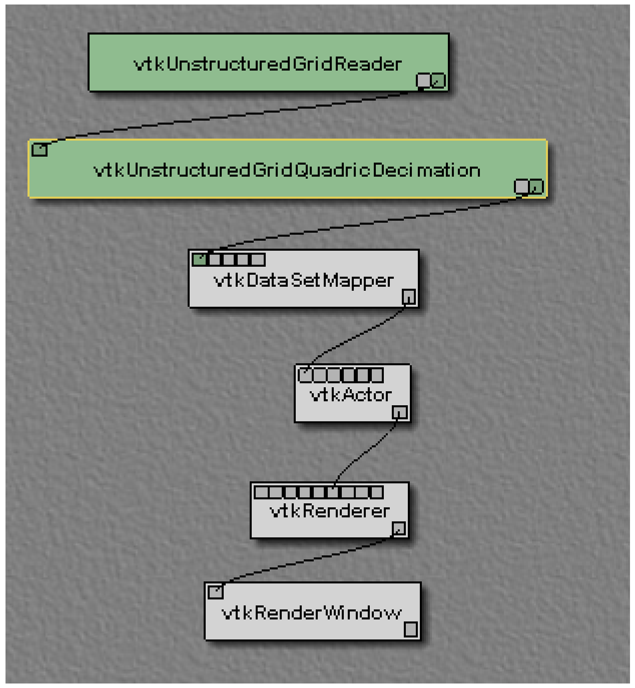



vtkUnstructuredGridReader
vtkUnstructuredGridQuadricDecimation
vtkDataSetMapper
vtkActor
vtkRenderer
vtkRenderWindow

By analogy, specialist can do it!

Automating Workflow Creation: Visualization by Analogy




vtkUnstructuredGridReader
vtkUnstructuredGridQuadricDecimation
vtkDataSetMapper
vtkActor
vtkRenderer
vtkRenderWindow

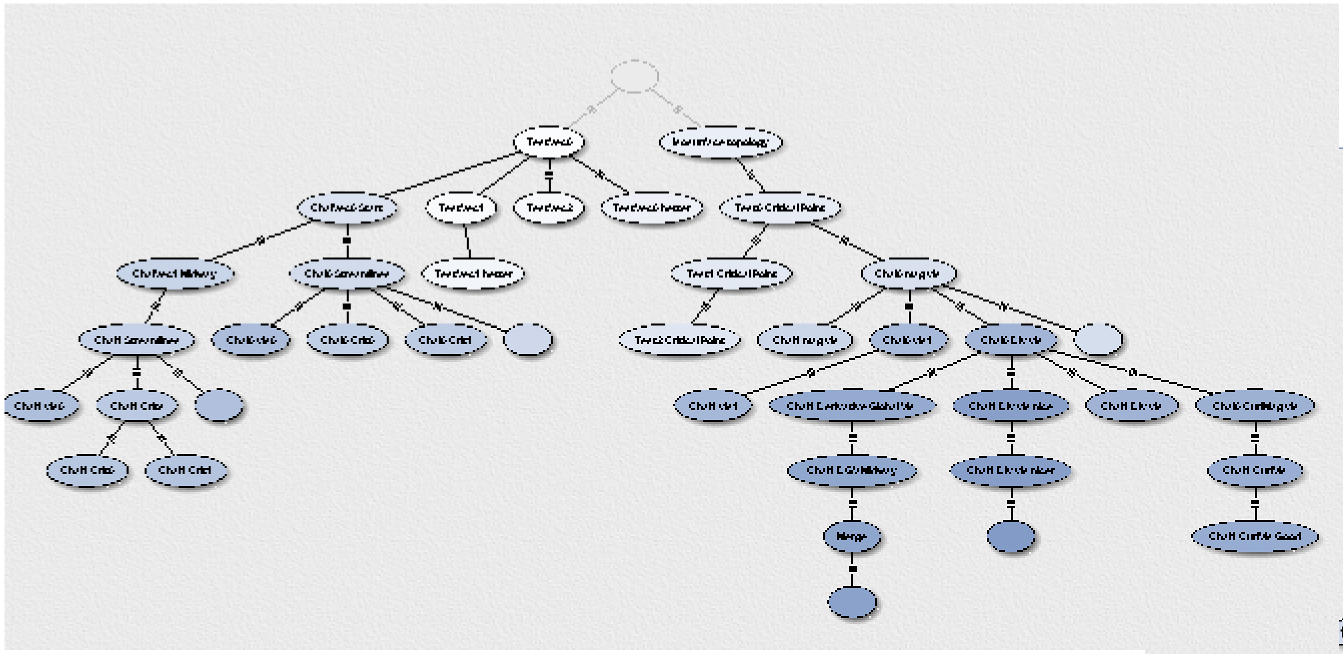
By analogy, specialist can do it!

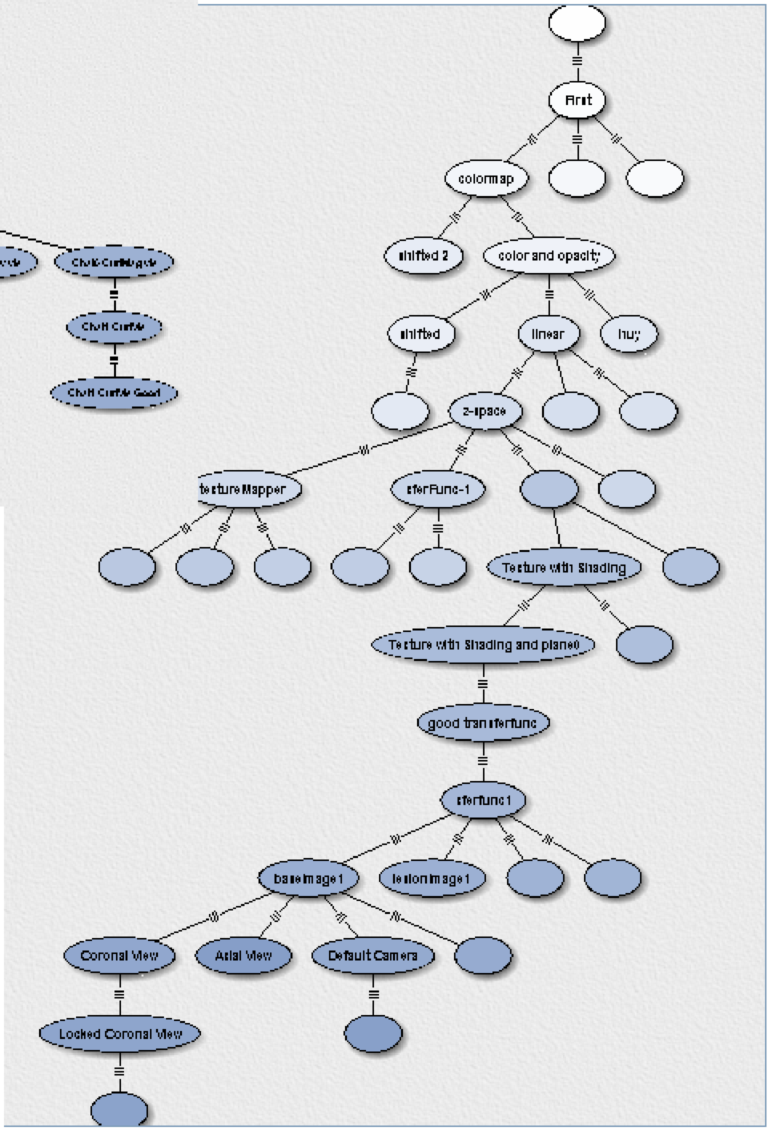
Simple in VisTrails:

$$v_4 = (v_2 - v_1) \circ v_3$$

Future

- Reproducible science
  - Publish image/results and their associated workflows—deep annotations
  - Track files, versions of systems (executable)—ensure reproducibility
- Querying and interacting with provenance
- Automate generation of data products
- Mine history—potentially useful information about good data exploration strategies
  - Automate generation of derived data
  - Simplify exploration, e.g., discover incompatible parameter settings
  - Understand problem-solving strategies





Different exploration strategies

Future

- Reproducible science
  - Publish image/results and their associated workflows—deep annotations
  - Track files, versions of systems (executable)—ensure reproducibility
- Querying and interacting with provenance
- Automate generation of data products
- Mine history—potentially useful information about good data exploration strategies
  - Automate generation of derived data
  - Simplify exploration, e.g., discover incompatible parameter settings
  - Understand problem-solving strategies
- Vision: scientists steering their own explorations

Acknowledgements

- This work is partially supported by the National Science Foundation (under grants IIS-0513692, CCF-0401498, EIA-0323604, CNS-0514485, IIS-0534628, CNS-0528201, OISE-0405402), the Department of Energy, an IBM Faculty Award, and a University of Utah Seed Grant.
- We thank
  - Dr. Antonio Baptista (OHSU) for motivation and input on the system design
  - Dr. George Chen (Harvard Medical School) for the lung datasets, and Erik Andersen for creating the visualizations
  - Gordon Kindlmann (SCI) for the brain data set; and
  - The Visible Human Project for the head

More info about VisTrails

google vistrails

Or

http://www.sci.utah.edu/~vgc/vistrails/