Visualization Knowledge (VisKo): Leveraging the Semantic Web to Support Visualization

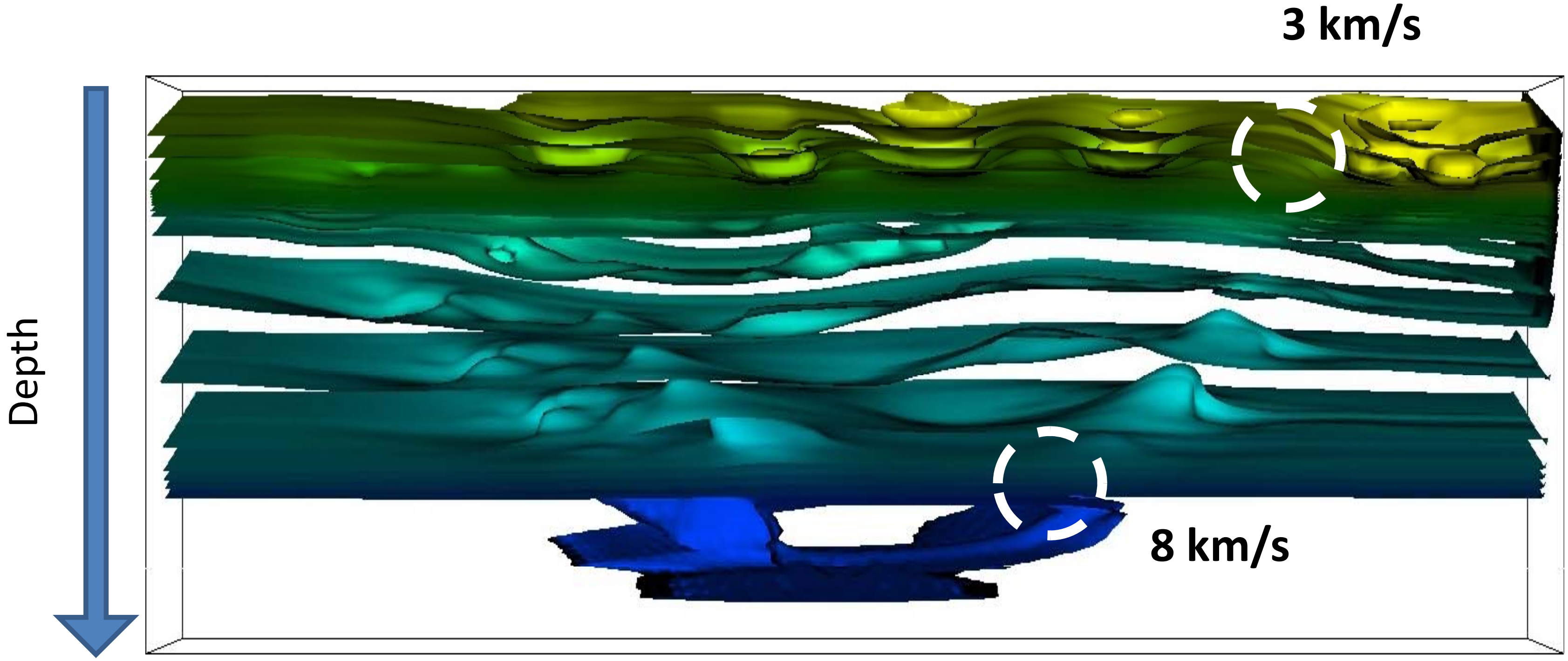
Paulo Pinheiro da Silva and Nicholas Del Rio
CyberShARE Center
University of Texas at El Paso
http://trust.utep.edu/visko

Overview

1. **A Visualization Example**
2. Toolkits and Visualization Pipelines
3. Visualization Query
4. Automated Generation of Visualization pipelines
5. Ontological Description of Visualization Pipelines
6. Conclusions

Velocity Model Visualization

- A set of isosurfaces
  - Visualization derived from a seismic velocity model
  - Covers a region in southern New Mexico



Visualizing the Velocity Model

- Visualization generated by custom Java application
  - relied on Visualization Toolkit (VTK) for rendering
  - VTK was developed for rendering 3D visualizations
  - VTK is supported by Sandia, Los Alamos, ARL, and others

- Writing a custom visualization application:
  - may rely on third party package to support rendering
  - may need to perform some transformations on input dataset before rendering

Program For Velocity Visualization

```
vtkImageReader rdr = new vtkImageReader();
rdr.SetFileName(inputDatasetFilePath);
rdr.SetDataScalarTypeToUnsignedShort();
rdr.SetDataByteOrderToLittleEndian();
rdr.SetFileDimensionality(3);
rdr.SetDataOrigin(0,0,0);
rdr.SetDataSpacing(1,1,1);
rdr.SetDataExtent(0,230,0,25,0,68);
rdr.SetNumberOfScalarComponents(1);
rdr.FileLowerLeftOn();
rdr.Update();


vtkContourFilter contours = new
vtkContourFilter();
contours.SetInput(rdr.GetOutput());
contours.GenerateValues(35,0.0,9000.0);


vtkPolyDataMapper contMapper = new
vtkPolyDataMapper();
contMapper.SetInput(contours.GetOutput());
contMapper.SetScalarRange(0.0,9000.0);

vtkActor contActor = new vtkActor();
contActor.SetMapper(contMapper);
contActor.RotateX(105);


vtkRenderer ren1 = new vtkRenderer();
ren1.AddActor(contActor);
ren1.AddActor2D(outlineActor);
ren1.SetBackground(1,1,1);


vtkRenderWindow renWin = new vtkRenderWindow();
renWin.SetOffScreenRendering(1);
renWin.AddRenderer(ren1);
renWin.SetSize(300,300);
renWin.Render();


vtkJPEGWriter img = new vtkJPEGWriter();
img.SetInputConnection(renWin.GetOutputPort());
img.SetFileName(outputDatasetFilePath);
img.SetQuality(100);
```

What Information is Needed?

- Regarding visualization, users need to know:
  - What view suits their data (e.g., contours, map, surfaces)
  - What properties should the view exhibit (e.g., orientation, projection, color, size)

- Regarding datasets, users may need to know:
  - The format the data is encoded in (e.g., netCDF, ESRI) to read the datasets
  - The semantic type of the data (e.g., gravity, velocity) to help with setting parameter arguments

What Information is Needed?

- Regarding third party rendering software:
  - Can it generate the required renderings?
  - Can it ingest my data in the format it resides in?
  - Does it satisfy my performance requirements?
  - What language can I use to interface with it?
  - What dependent packages do I need to install?

Writing the Program

- Once you have answers for the previous questions, you can begin coding
- Some portion of the code will transform input datasets into formats that can be rendered
  - Most renderers are format specific (*exception Protovis*)
- The rest of the code will:
  - Gather suitable arguments for the rendering
  - Invoking the renderer with the arguments

Overview of the Visualization Program

- The program that generated the velocity model visualization:
  - Relies on VTK for rendering
  - Renders the data as *isosurfaces*
  - Ingests data in format *binaryFloatArray*
  - transforms *binaryFloatArray* to *VTKImageData*
  - Ingests data of type *3DVelocityModel*
  - Is not ready for parallel execution
  - Is written in Java

Overview

1. **A Visualization Example**
2. **Toolkits and Visualization Pipelines**
3. Visualization Query
4. Automated Generation of Visualization pipelines
5. Ontological Description of Visualization Pipelines
6. Conclusions

Visualization Toolkits

- VTK is a ***toolkit*** that provides functions such as:
  - Filtering
  - Gridding/interpolating
  - Mapping (i.e., transform data into views like isosurfaces)
  - Rendering the views
- Functions are referred to as ***operators***
  - Generic mapping tools (GMT): 60 operators
  - VTK: hundreds of operators

Visualization Pipeline Model

- VTK requires that users write ***pipelines***
  - the output of an operator feeds into the operator next in the pipeline sequence
  - first operator in pipeline is usually data reader

- Thus the Java program that visualizes the velocity model can be seen as a pipeline of VTK operators

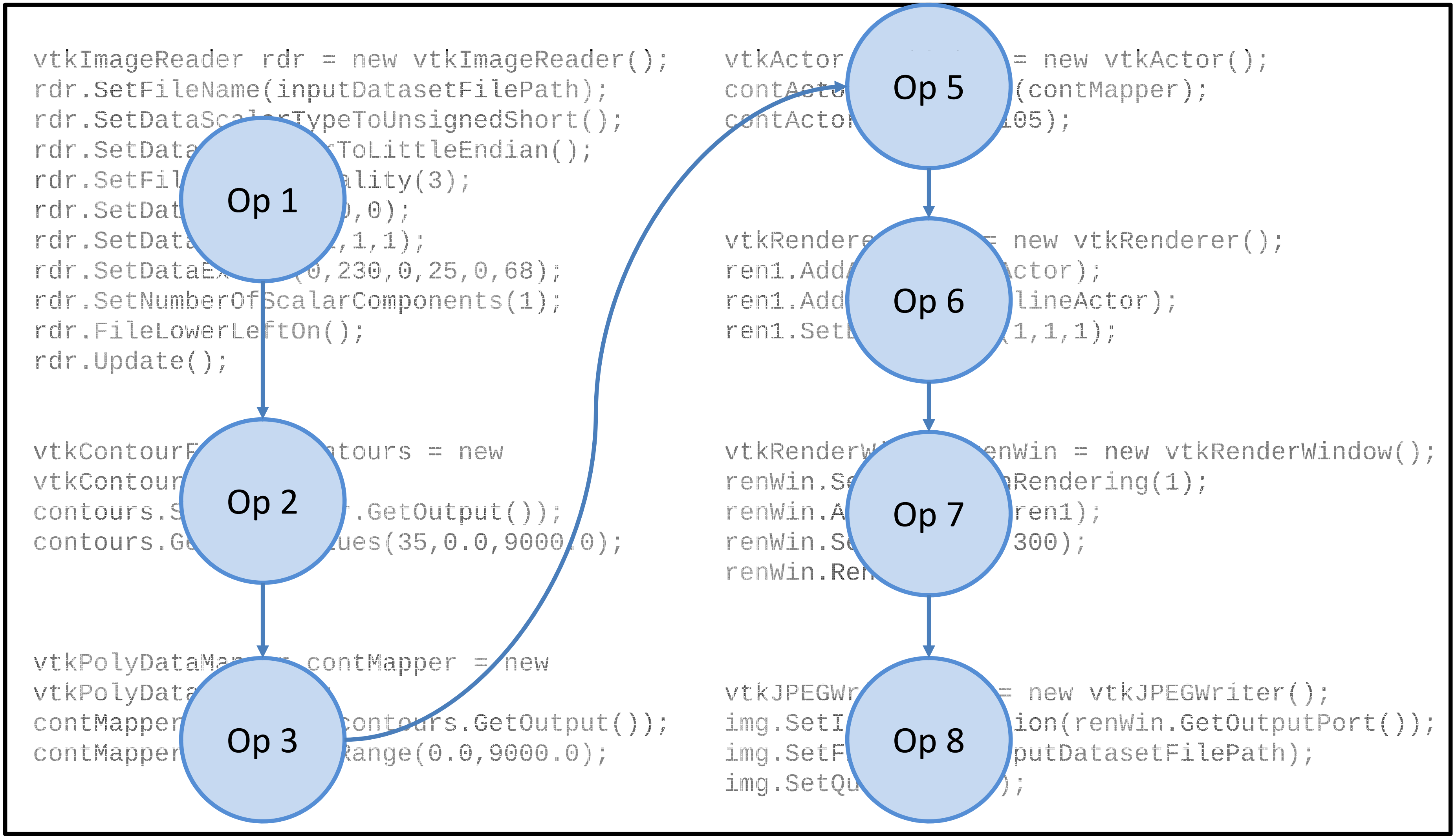
- ***It is up to the users to write these pipelines…***

VTK Java Pipeline For Velocity Model

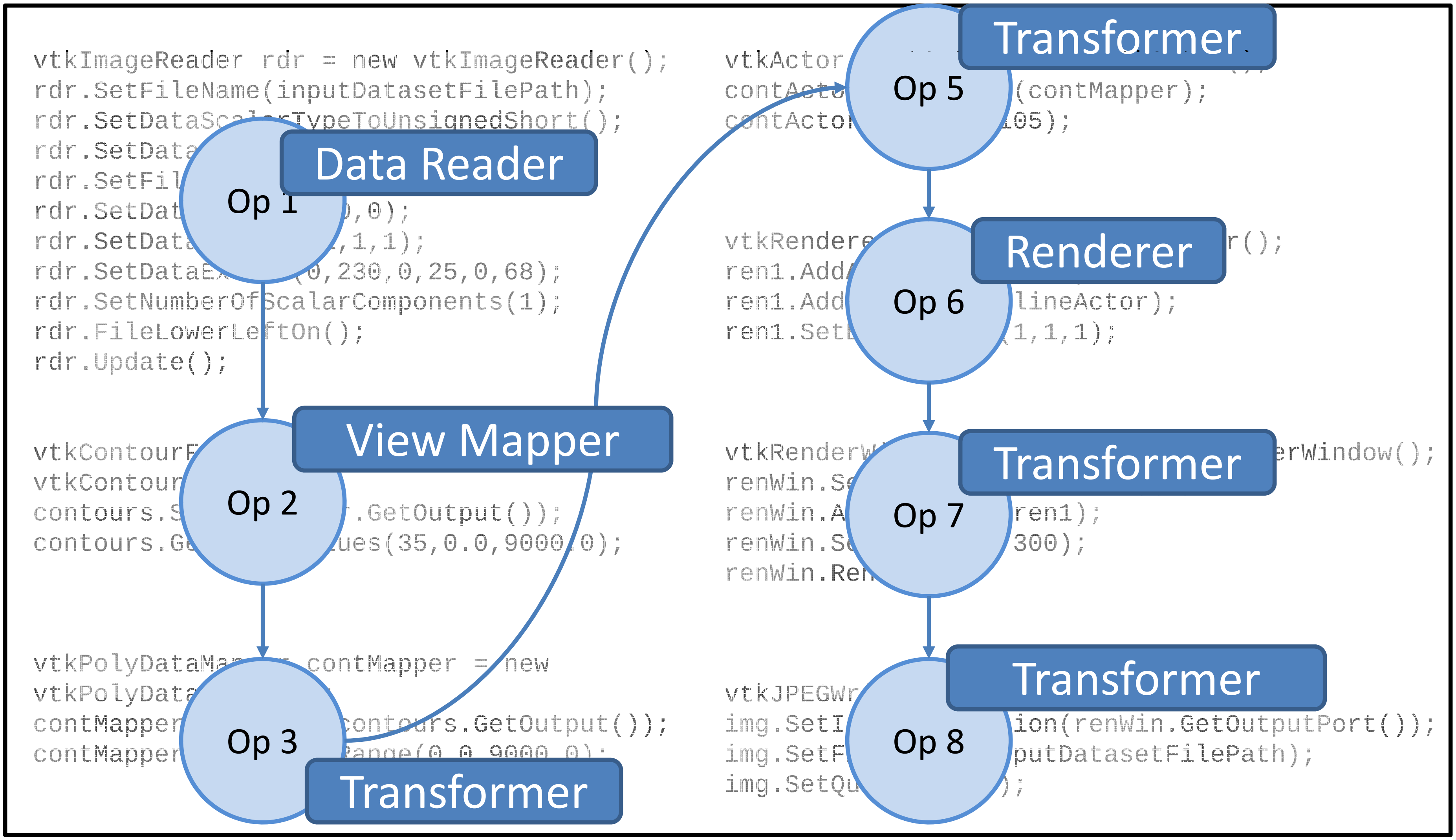
```
vtkImageReader rdr = new vtkImageReader();
rdr.SetFileName(inputDatasetFilePath);
rdr.SetDataScalarTypeToUnsignedShort();
rdr.SetDataByteOrderToLittleEndian();
rdr.SetFileDimensionality(3);
rdr.SetDataOrigin(0,0,0);
rdr.SetDataSpacing(1,1,1);
rdr.SetDataExtent(0,230,0,25,0,68);
rdr.SetNumberOfScalarComponents(1);
rdr.FileLowerLeftOn();
rdr.Update();


vtkContourFilter contours = new
vtkContourFilter();
contours.SetInput(rdr.GetOutput());
contours.GenerateValues(35,0.0,9000.0);


vtkPolyDataMapper contMapper = new
vtkPolyDataMapper();
contMapper.SetInput(contours.GetOutput());
contMapper.SetScalarRange(0.0,9000.0);
```

```
vtkActor contActor = new vtkActor();
contActor.SetMapper(contMapper);
contActor.RotateX(105);


vtkRenderer ren1 = new vtkRenderer();
ren1.AddActor(contActor);
ren1.AddActor2D(outlineActor);
ren1.SetBackground(1,1,1);


vtkRenderWindow renWin = new vtkRenderWindow();
renWin.SetOffScreenRendering(1);
renWin.AddRenderer(ren1);
renWin.SetSize(300,300);
renWin.Render();


vtkJPEGWriter img = new vtkJPEGWriter();
img.SetInputConnection(renWin.GetOutputPort());
img.SetFileName(outputDatasetFilePath);
img.SetQuality(100);
```

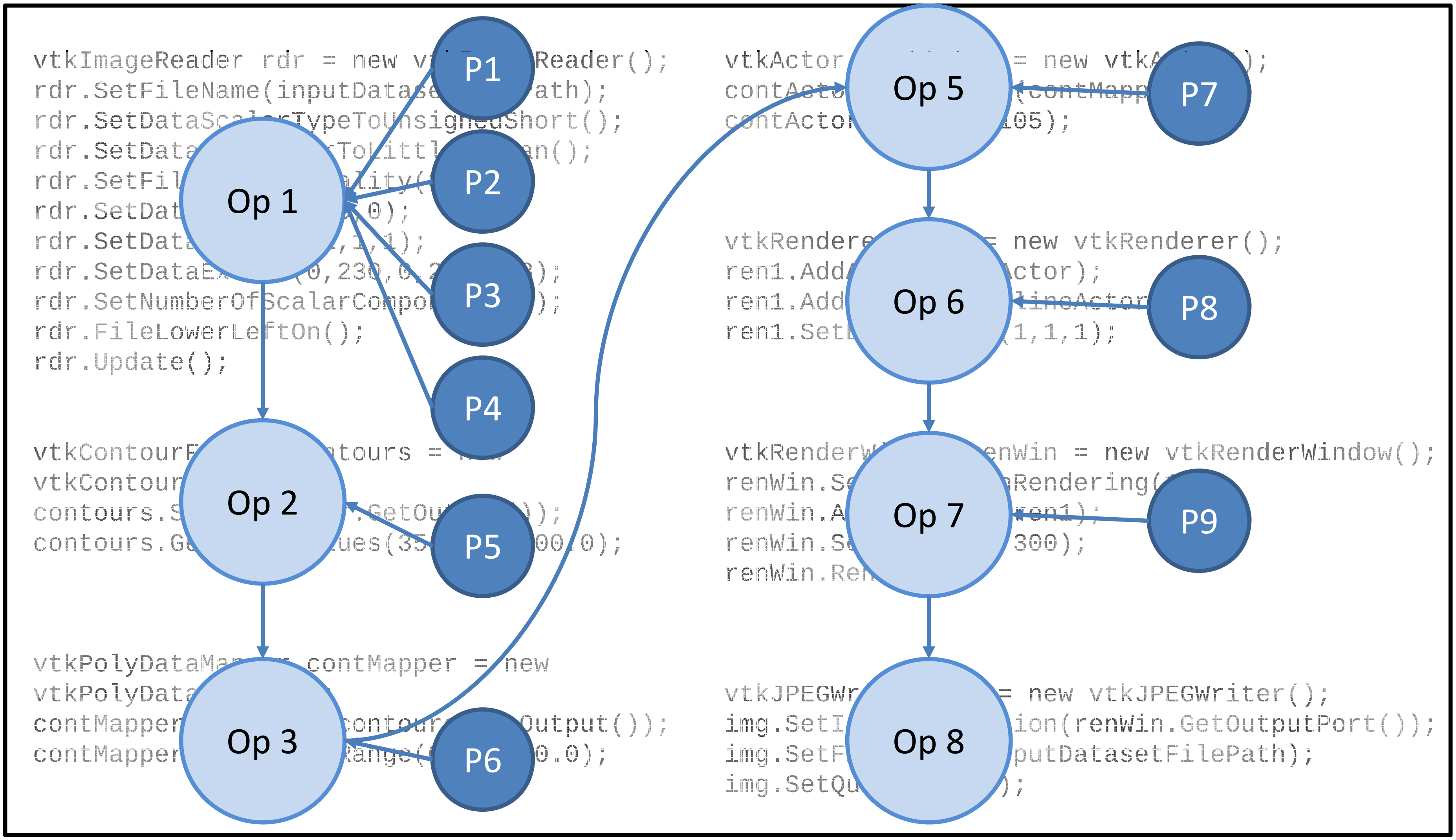
Pipeline of Visualization Operators



Different Types of Operators



Operators are Parameterized



What Kind of Skills are Currently required by a user?

- Knows about different views
- Knows what toolkits support a particular view
- Knows what toolkits operate on a particular data
- Knows how to install a visualization toolkit
- Knows what language the toolkit is built on
- Knows what operators need to compose a pipeline
- Knows suitable arguments for the operators
- Knows how to develop software

What Kind of Skills are Currently required by a user?

scientist

- Knows about different views

Visualization Expert

- Knows what toolkits support a particular view
- Knows what toolkits operate on a particular data

Engineer

- Knows how to install a visualization toolkit
- Knows what language the toolkit is built on
- Knows what operators need to compose a pipeline
- Knows suitable arguments for the operators
- Knows how to develop software

Overview

1. **A Visualization Example**
2. **Toolkits and Visualization Pipelines**
3. **Visualization Query**
4. Automated Generation of Visualization pipelines
5. Ontological Description of Visualization Pipelines
6. Conclusions

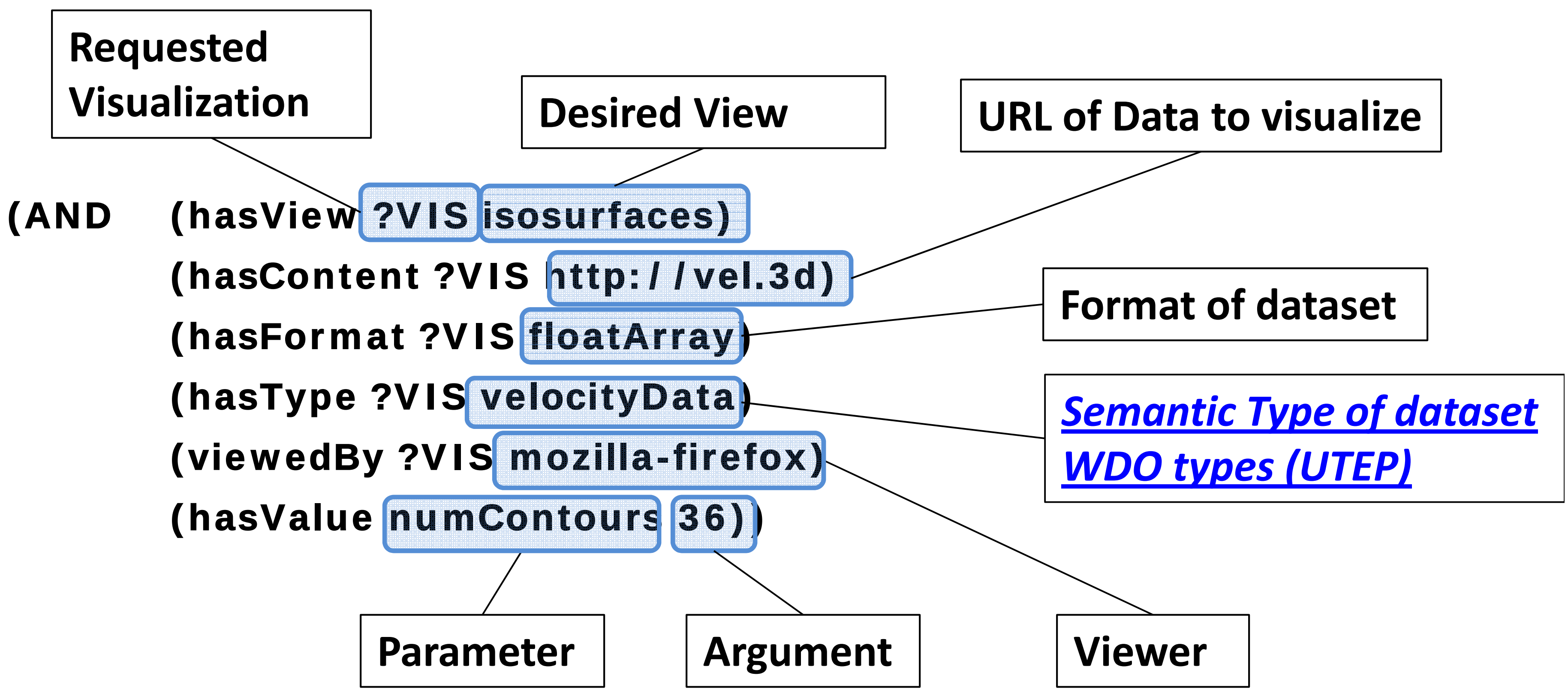
Declarative Requests

- Many user skills needed to visualize data
- Aside from cognitive aspects of visualization, a large part of the problem is engineering
- Stems from fact that we generate visualizations imperatively (i.e., write code)

Can we provide a means for users to generate visualizations declaratively (i.e., specify what visualization they want without having to code)?

Visualization Query

- The velocity model visualization was actually a result of a ***visualization query***



Visualization Queries and SQL

- Visualization queries mirror SQL queries
  - query request is specified declaratively
  - request is then translated into a query plan
  - query plan computes the result requested by the query
- Information specified in visualization queries is used to derive pipelines rather than query plans
- The pipeline in turn generates the visualization requested in the query

Visualization Query Challenges

What kind of knowledge is needed to generate pipelines that answer visualization queries?

What infrastructure can leverage the knowledge to support the generation and execution of the pipelines?

VisKo Project Claims

- VisKo supplements user skills with ***visualization knowledge*** to compose pipelines

- VisKo is a framework for:
  - Encoding user skills into visualization knowledge
  - Managing visualization knowledge
  - Automatically generate pipelines from visualization knowledge
  - Automatically generate visualizations by executing pipelines

Old Way vs VisKo

```
vtkImageReader rdr = new vtkImageReader();
rdr.SetFileName(inputDatasetFilePath);
rdr.SetDataScalarTypeToUnsignedShort();
rdr.SetDataByteOrderToLittleEndian();
rdr.SetFileDimensionality(3);
rdr.SetDataOrigin(0,0,0
rdr.SetDataSpacing(1,1,
rdr.SetDataExtent(0,230
rdr.SetNumberOfScalarCo
rdr.FileLowerLeftOn();
rdr.Update();


vtkContourFilter contou
vtkContourFilter();
contours.SetInput(rdr.GetOutput());
contours.GenerateValues(35,0.0,9000.0);



vtkPolyDataMapper contMapper = new
vtkPolyDataMapper();
contMapper.SetInput(contours.GetOutput());
contMapper.SetScalarRange(0.0,9000.0);
```

```
(AND    (hasView ?VIS isosurfaces)
        (hasContent ?VIS http://vel.3d)
        (hasFormat ?VIS floatArray)
        (hasType ?VIS velocityData)
        (viewedBy ?VIS mozilla-firefox)
        (hasValue numContours 36))
```

Overview

1. **A Visualization Example**
2. **Toolkits and Visualization Pipelines**
3. **Visualization Query**
4. **Automated Generation of Visualization pipelines**
5. Ontological Description of Visualization Pipelines
6. Conclusions

A Pipeline Synthesized by VisKo

operators

data flow

1. **Service Implementation Name:** float2shortThr.c
   - **No Parameters**
   - **Supporting Toolit:** Visualization Toolkit
   - **Supportng OWLS Service:** http://rio.cs.utep.edu/ciserver/ciprojects/viskoOWLS/float2ShortThrTransformerFixedParams.owl

---

2. **Service Implementation Name:** vtkImageDataReader1
   - **No Parameters**
   - **Supporting Toolit:** Visualization Toolkit
   - **Supportng OWLS Service:** http://rio.cs.utep.edu/ciserver/ciprojects/viskoOWLS/velocityAndTimeShortIntToImageData3D.owl

---

3. **Service Implementation Name:** vtkContourFilter1
   - **No Parameters**
   - **Supporting Toolit:** Visualization Toolkit
   - **Supportng OWLS Service:** http://rio.cs.utep.edu/ciserver/ciprojects/viskoOWLS/velocityImageData3DToContoursPolyData.owl

---

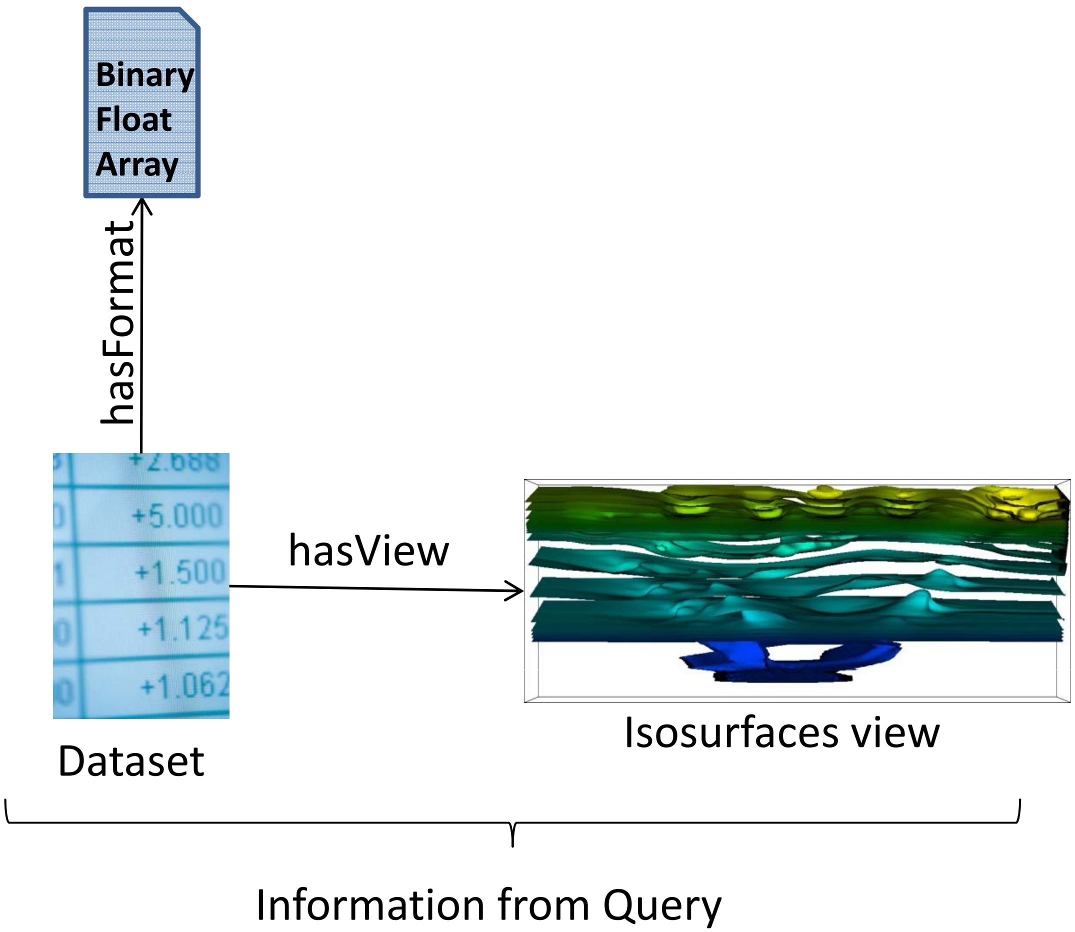
4. **Service Implementation Name:** vtkPolyDataMapper1
   - **No Parameters**
   - **Supporting Toolit:** Visualization Toolkit
   - **Supportng OWLS Service:** http://rio.cs.utep.edu/ciserver/ciprojects/viskoOWLS/velocityContoursPolyDataToJPEG.owl

VisKo Pipeline Composition

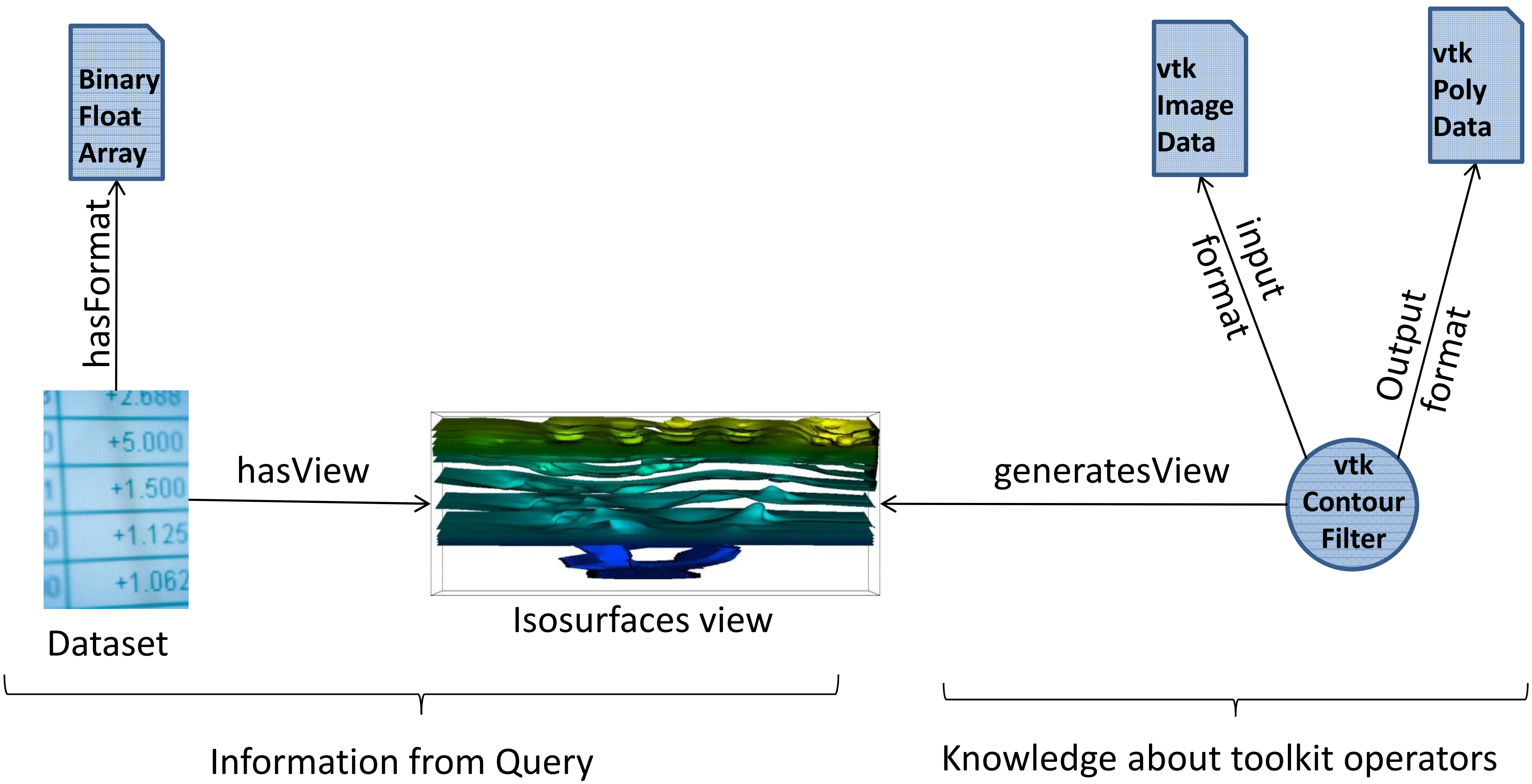
- The query tells the system:
  - the input format
  - the target view
- From target view:
  - Identify operator that generates view (i.e. view mapper)
- From operator:
  - identify format it operates on (i.e., target format)

- Find sequence of operators that transforms the input format (and type) to target format (and type)

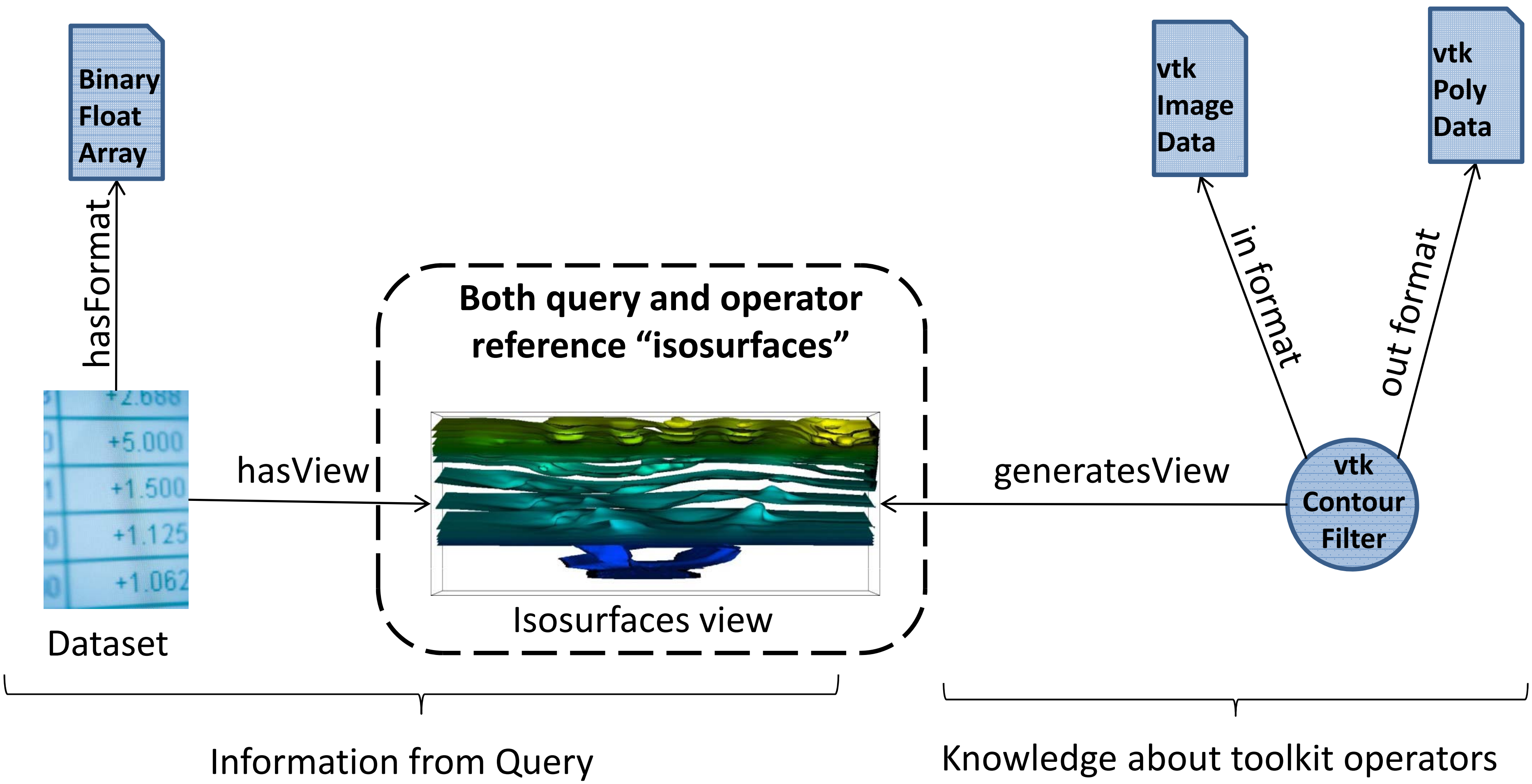
Information From Query



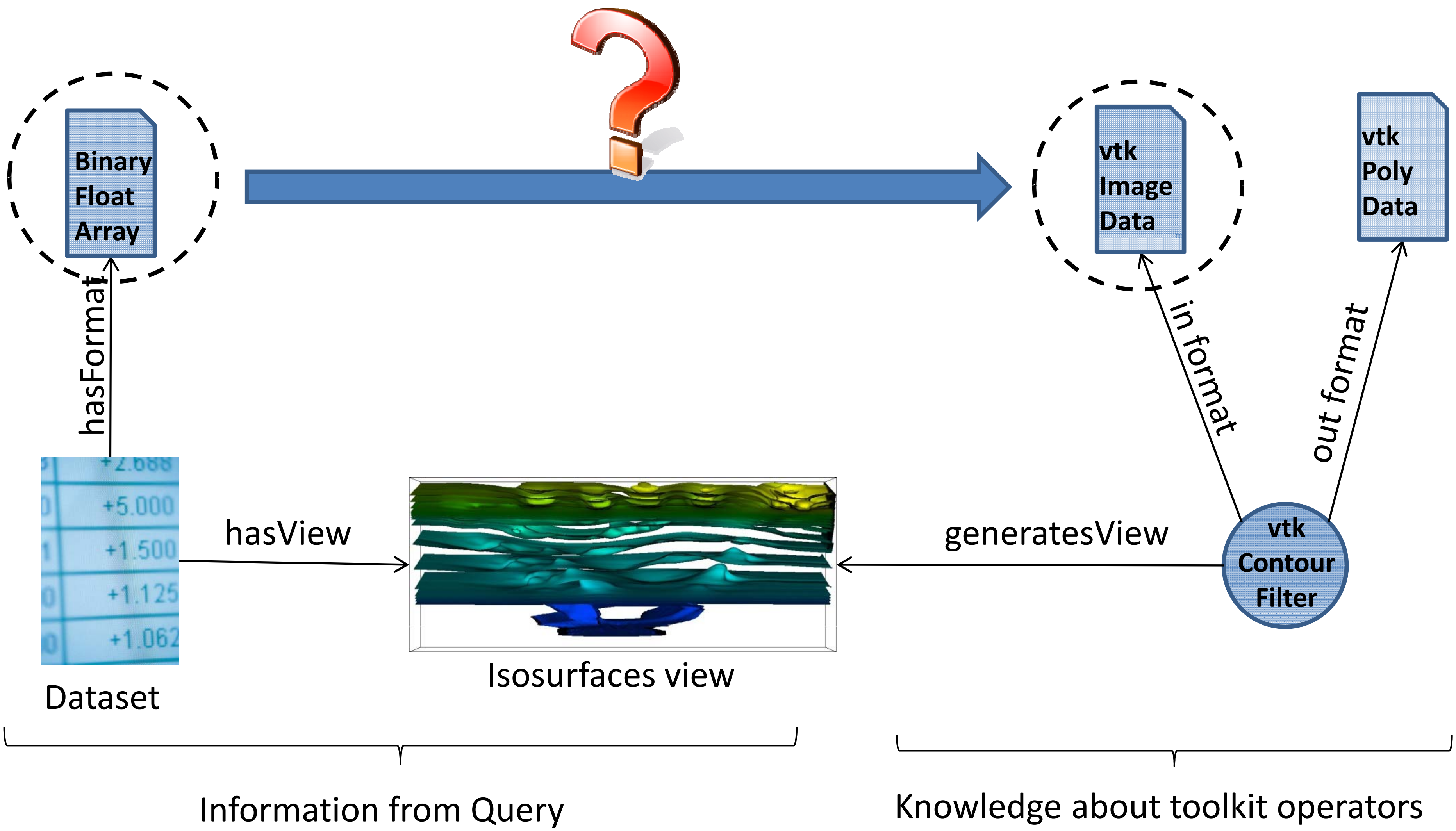
Knowledge about Toolkit Operators



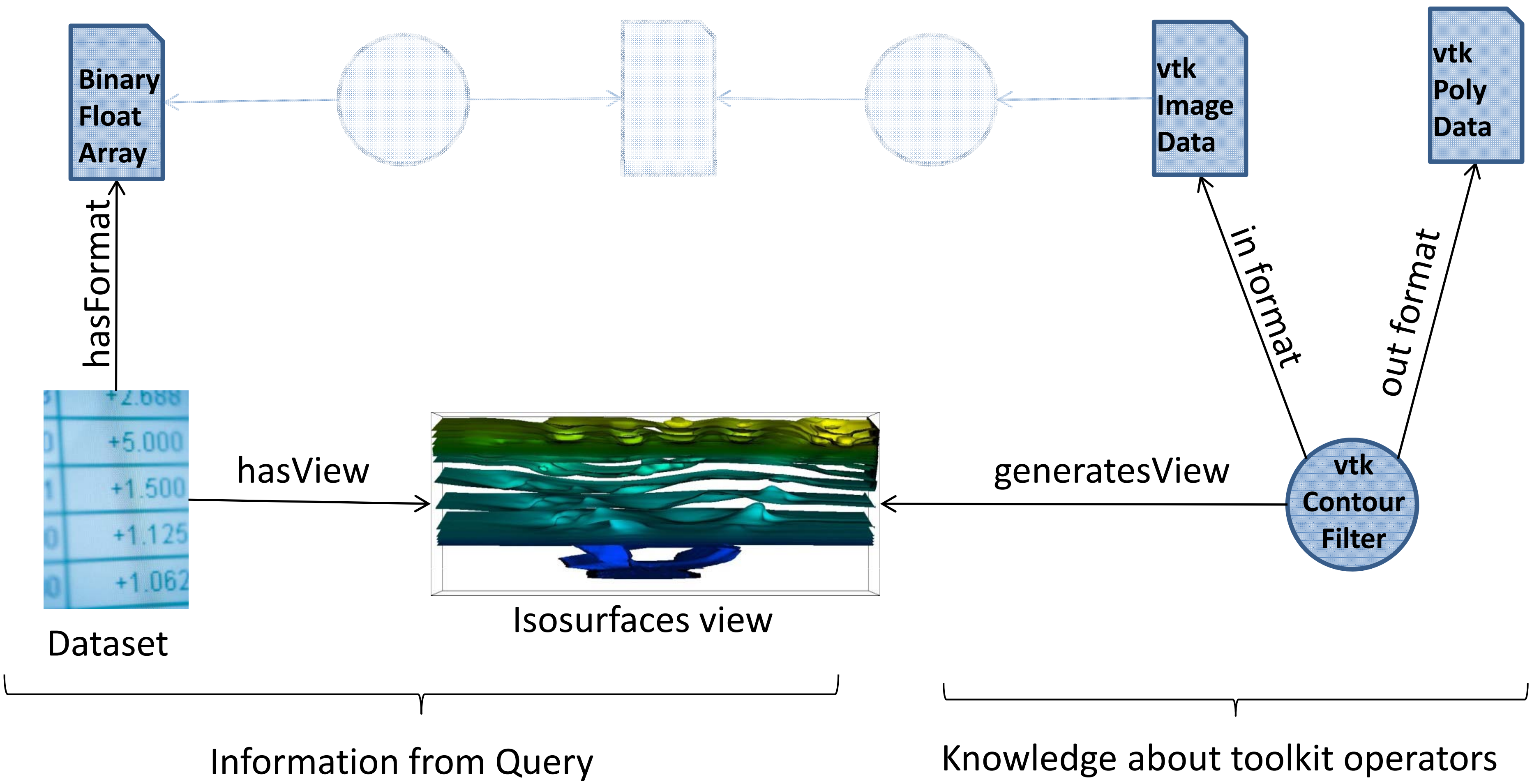
The Knowledge is Linked



Format Transformation?



Required Pipeline



Multiple Results Example

Visualization Query

(AND (hasView ?DATA ?VIEW)

(hasContent ?DATA http://rio.cs.utep.edu/ciserver/ciprojects/GravityMapProvenance/esriGrid.txt)

(hasFormat ?DATA http://rio.cs.utep.edu/ciserver/ciprojects/formats/ESRIGRID.owl#ESRIGRID)

(hasType ?DATA http://rio.cs.utep.edu/ciserver/ciprojects/CrustalModeling/CrustalModeling.owl#d12)

(viewedBy ?DATA http://rio.cs.utep.edu/ciserver/ciprojects/viskoOperator/mozilla-firefox.owl#mozilla-firefox))

Input format is ESRI Gridded

Data is of type *Gravity Data*

No view specified!

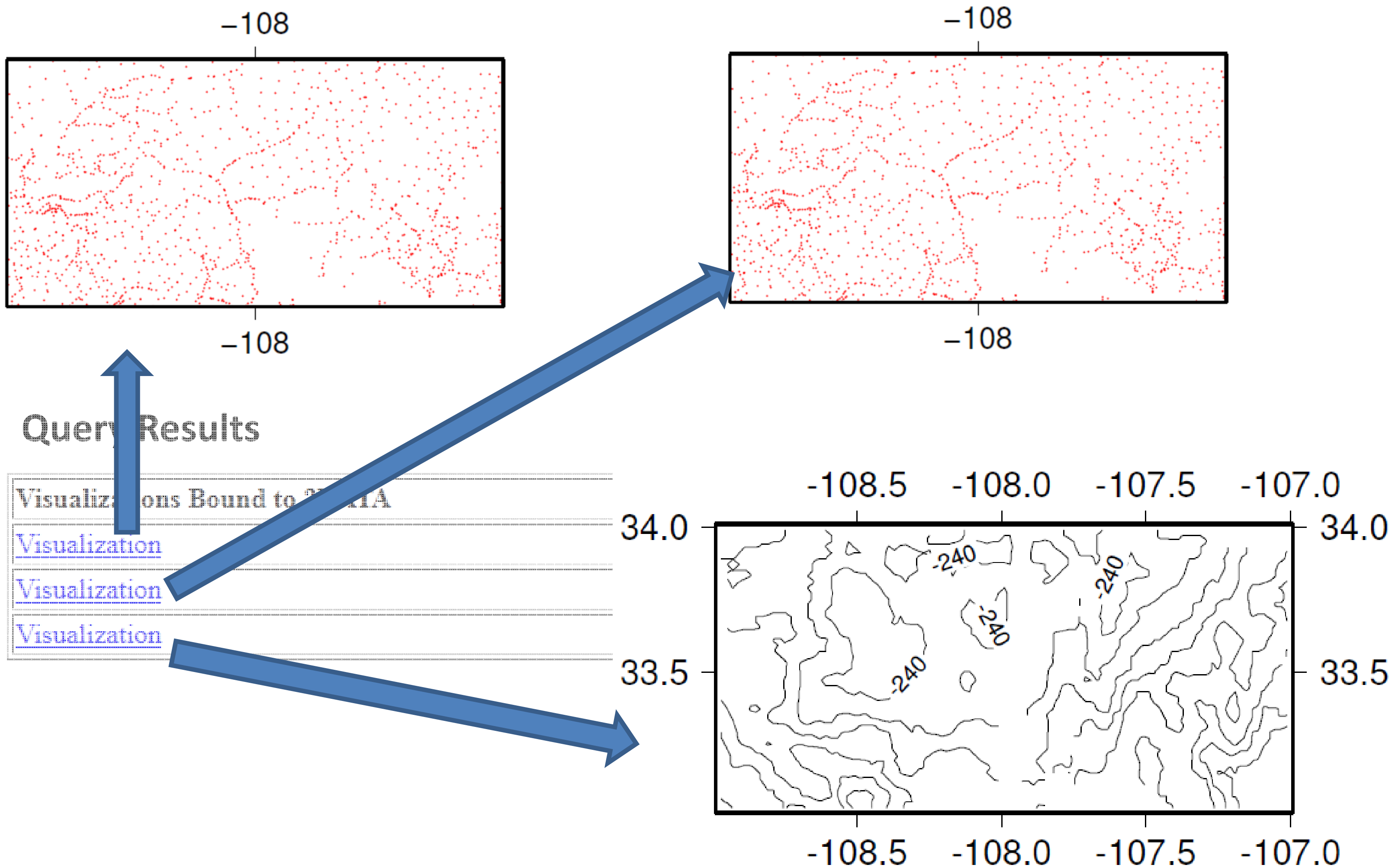
Multiple Results Example

VisKo was able to generate three different pipelines, given the query and visualization knowledge currently loaded

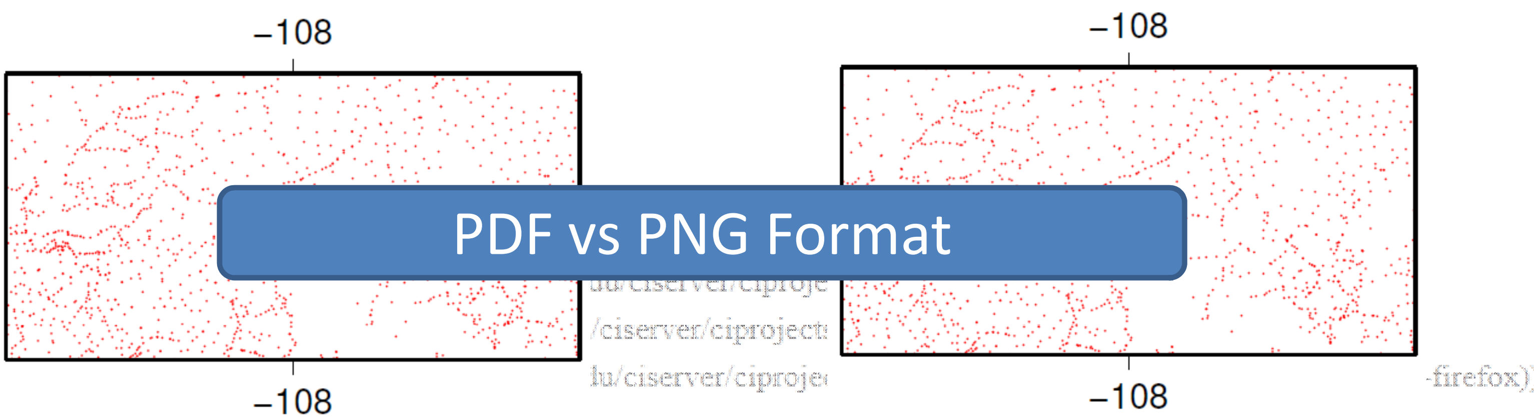
Query Results

Visualizations Bound to ?DATA	How to Generate ?DATA
Visualization	Pipeline
Visualization	Pipeline
Visualization	Pipeline

Multiple Visualizations Example



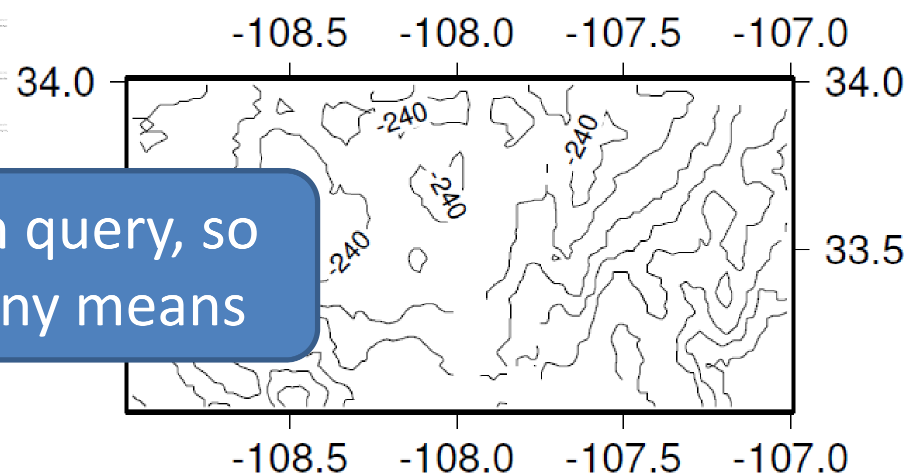
Multiple Visualizations Example



PDF vs PNG Format

Query Results

Visualizations Bound to ?DATA
Visualization
Visualization



View was left unspecified in query, so system visualized data by any means

Composition by Rules

- Pipeline composition is actually derived through application of ***rules***
  - rules are applied to statements comprising our visualization knowledge
  - rules are simple horn clauses

- Rules are not described in this seminar
- Before we can apply rules, we need to know what statements comprise our knowledge base

Overview

1. **A Visualization Example**
2. **Toolkits and Visualization Pipelines**
3. **Visualization Query**
4. **Automated Generation of Visualization pipelines**
5. **Ontological Description of Visualization Pipelines**
6. Conclusions

VisKo Visualization Language

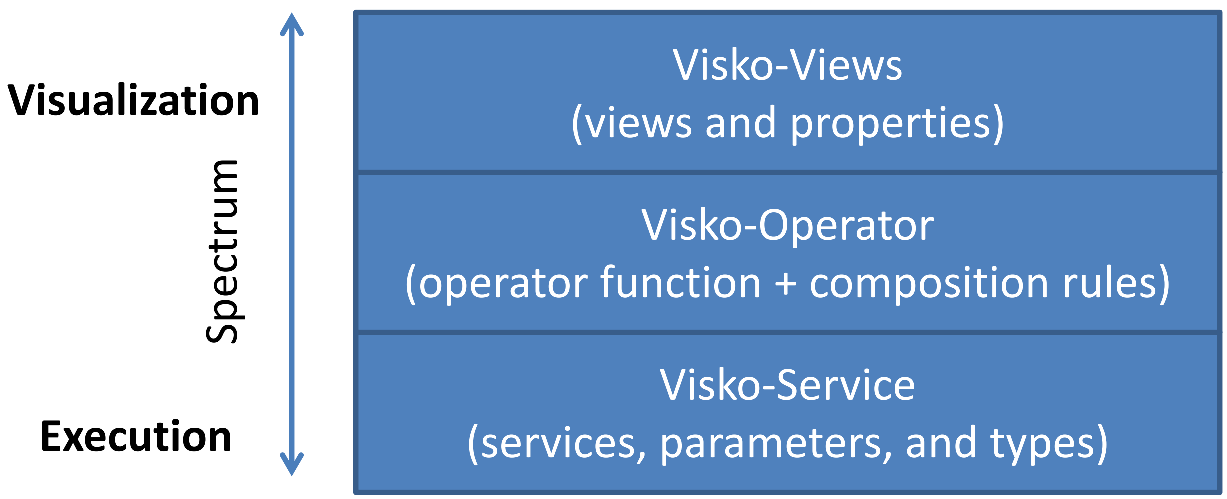
- VisKo provides a language to describe operators and how they can be composed into pipelines

- The language's expressivity is focused on describing:
  - Views and view properties
  - Different operator types
  - Parameters associated with operators

- The language is defined by ontologies encoded in [Ontology Web Language (OWL)]

VisKo Language Layers

- The VisKo language encompasses three different ontologies to describe toolkit operators from different perspectives



Encoding Velocity Model View

- Velocity model is visualized as a set of isosurfaces, so this view is defined as a set of ESIP ***surfaces***

- We need to describe this resource *isosurfaces* in terms of the ontology:

Isosurfaces *isa* Surfaces
Isosurfaces *isa* Geometry
Isosurfaces *isa* AtomicView
Isosurfaces *isa* View

} **Isosurfaces description**

Note: VisKo relies on Resource Document Framework (RDF) for encoding statements

Encoding Velocity Model Operator

- A *contouring* operator generated the isosurfaces
- The contouring operator
  - operated on data in format *3DImageData*
  - generated the view *isosurfaces*
  - output plot in format *PolyData*

contouringOperator *isa* Mapper
contouringOperator *operatesOn* 3DImageData
contouringOperator *transformsTo* PolyData
contouringOperator *mapsTo* isosurfaces

} **vtkContourFilter description**

Note: *contouring* operator is conceptual and cannot be executed

Encoding Velocity Model Service

- The contouring operator is *implemented* by the VTKContourFilter service

VisKo-Service

vtkContourFilter *isa* Service
vtkContourFilter *implements* contouring
vtkContourFilter *supportedBy* VTK

OWL-S

vtkContourFilter *hasInput* contourSpacing
vtkContourFilter *hasInput* numberOfContours
vtkContourFilter *hasGrounding* wsdlGrounding

vtkContourFilter description

Executable VisKo service implements operator ***contouring***

Overview

1. **A Visualization Example**
2. **Toolkits and Visualization Pipelines**
3. **Visualization Query**
4. **Automated Generation of Visualization pipelines**
5. **Ontological Description of Visualization Pipelines**
6. **Conclusions**

Conclusions

- VisKo demonstrates that visualization pipelines can be specified declaratively through the use of visualization queries
- VisKo is a systematic way of reusing knowledge about visualization toolkits
- VisKo has been in use for projects in the area of Earth sciences and environmental sciences
- VisKo has knowledge about the use of GMT, VTK, NCL, ImageJ
- Together we can create an ESIP Visualization Knowledge Base