Advanced

Kitware Course Week, Clifton Park, NY

24 July, 2019

Marcus D. Hanwell

Outline

- Introduction
- Custom Operators
- User Input for Operators
- External Pipelines
- More

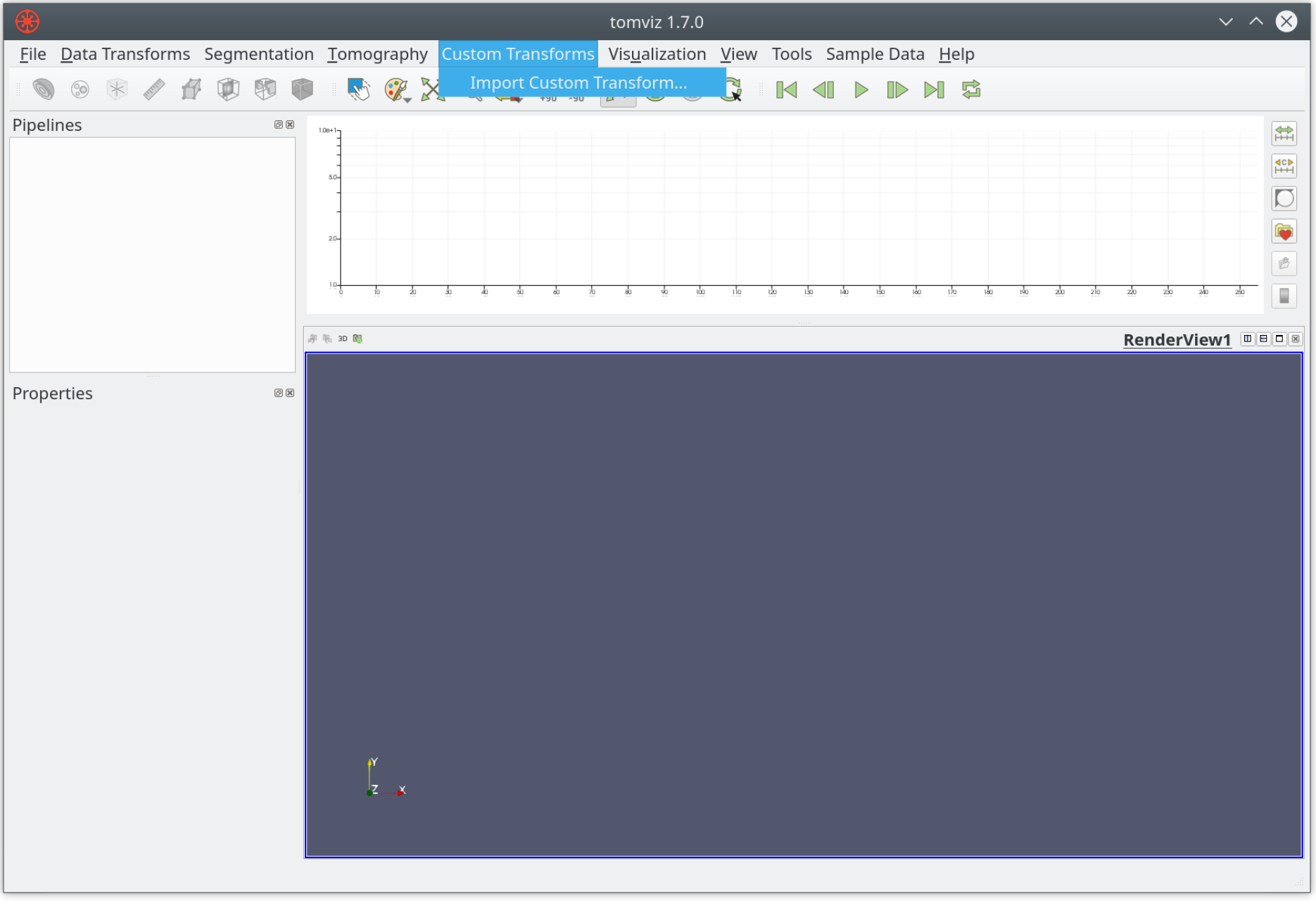
Introduction

- Tomviz can be extended at runtime
  - Custom operators and file formats
  - Primarily using Python scripts/JSON
- The processing pipeline is self-contained
  - Run in Docker for reproducibility
  - Soon external Python for CUDA, etc

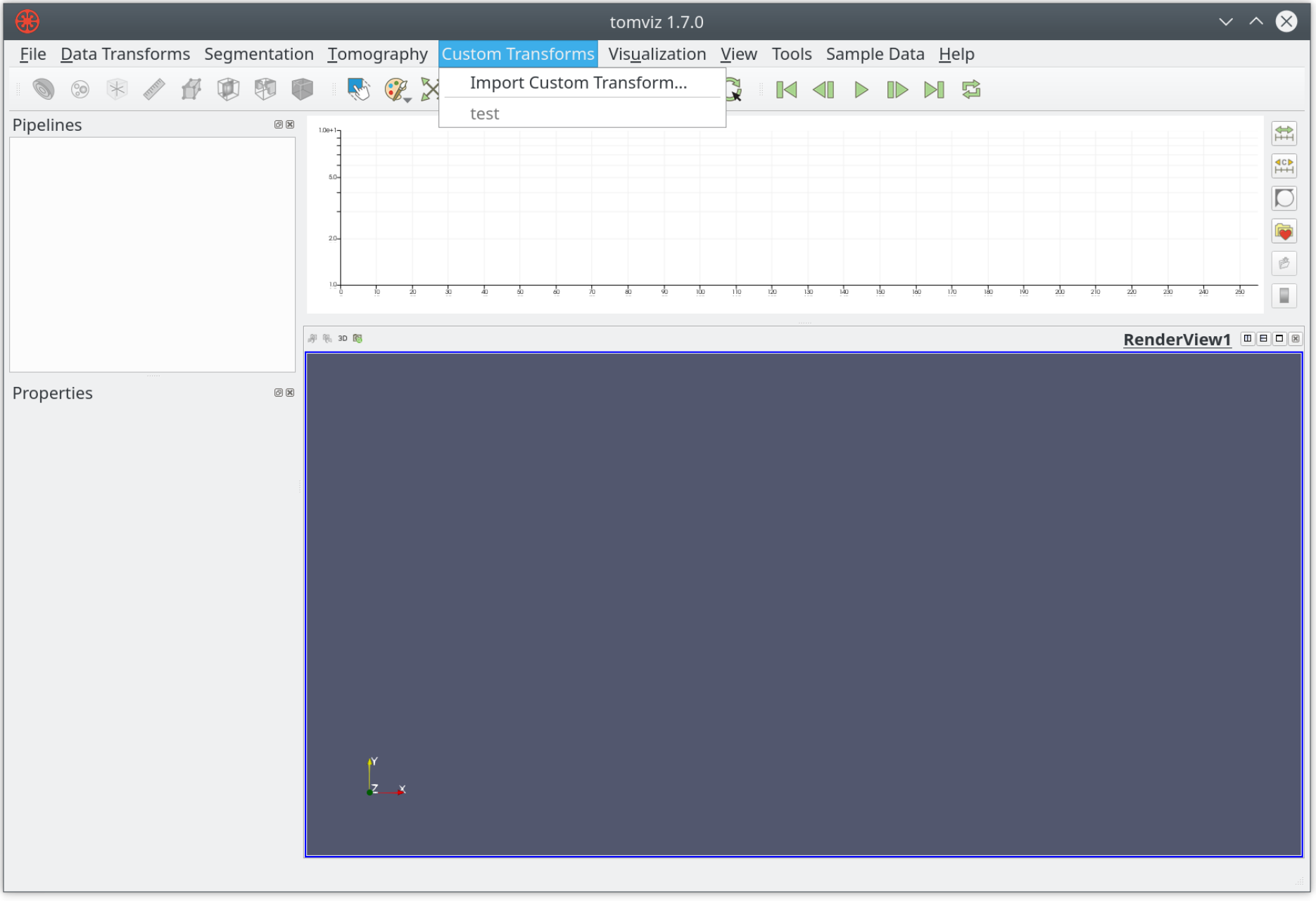
Custom Operators

- Initially just a single entry
  - Import custom transform
- Use import, or copy files over
  - Looks in ~/tomviz and ~/.tomviz
  - Adds suitable Python/JSON files

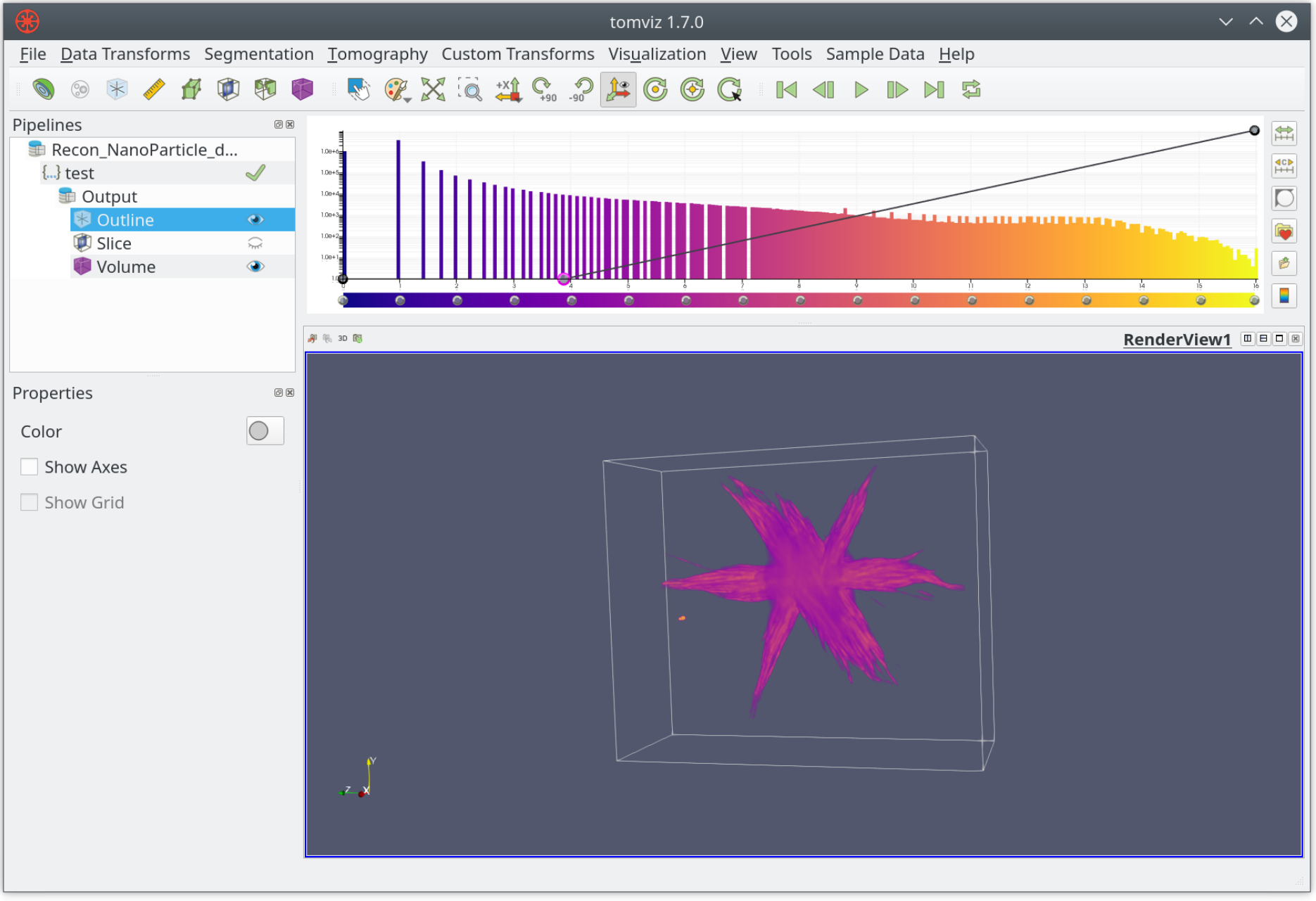
Custom Transforms


tomviz 1.7.0
File
Data Transforms
Segmentation
Tomography
Custom Transforms
Visualization
View
Tools
Sample Data
Help
Import Custom Transform...
Pipelines
Properties
RenderView1
3D
Y
Z
X

Custom Transforms—Test


tomviz 1.7.0
File
Data Transforms
Segmentation
Tomography
Custom Transforms
Visualization
View
Tools
Sample Data
Help
Import Custom Transform...
test
Pipelines
Properties
RenderView1
3D
Y
Z
X

Custom Transforms—Test


tomviz 1.7.0
File
Data Transforms
Segmentation
Tomography
Custom Transforms
Visualization
View
Tools
Sample Data
Help
Pipelines
Recon_NanoParticle_d...
test
Output
Outline
Slice
Volume
Properties
Color
Show Axes
Show Grid
RenderView1

Anatomy of an Operator

```
def transform_scalars(dataset):

  from tomviz import utils
  import numpy as np

  # Get the current volume as a NumPy array.
  array = utils.get_array(dataset)

  # Operate on your data, here we square root it.
  result = np.sqrt(array)

  # Set the transformed data, displayed in Tomviz.
  utils.set_array(dataset, result)
```

Classier Operators—Outline

```
import tomviz.operators

class SquareRootOperator(tomviz.operators.CancelableOperator):

  def transform_scalars(self, dataset):
    from tomviz import utils
    import numpy as np
    self.progress.maximum = 10
    scalars = utils.get_scalars(dataset)
    # Process dataset in chunks, user can cancel.
    result = np.float32(scalars)
    step = 0
    # Processing loop goes here...
    utils.set_scalars(dataset, result)
```

Classier Operators—Loop

```
def transform_scalars(self, dataset):
  # ...other stuff from previous slide
  scalars = utils.get_scalars(dataset)
  # Process dataset in chunks, user can cancel.
  result = np.float32(scalars)
  for chunk in np.array_split(result, 10):
    if self.canceled:
      return
    np.sqrt(chunk, chunk)
    step += 1
    self.progress.value = step

  utils.set_scalars(dataset, result)
```

Classier Operators—Complete

```
import tomviz.operators
CHUNKS = 10

class SquareRootOperator(tomviz.operators.CancelableOperator):

  def transform_scalars(self, dataset):
    from tomviz import utils
    import numpy as np
    self.progress.maximum = CHUNKS
    scalars = utils.get_scalars(dataset)
    # Process dataset in chunks, user can cancel.
    result = np.float32(scalars)
    step = 0
    for chunk in np.array_split(result, CHUNKS):
      if self.canceled:
        return
```

User Input for Operators

- You have a custom operator now
  - How can we change the chunk size?
  - Edit the code?
- Generating user interface
  - Inject input into operators

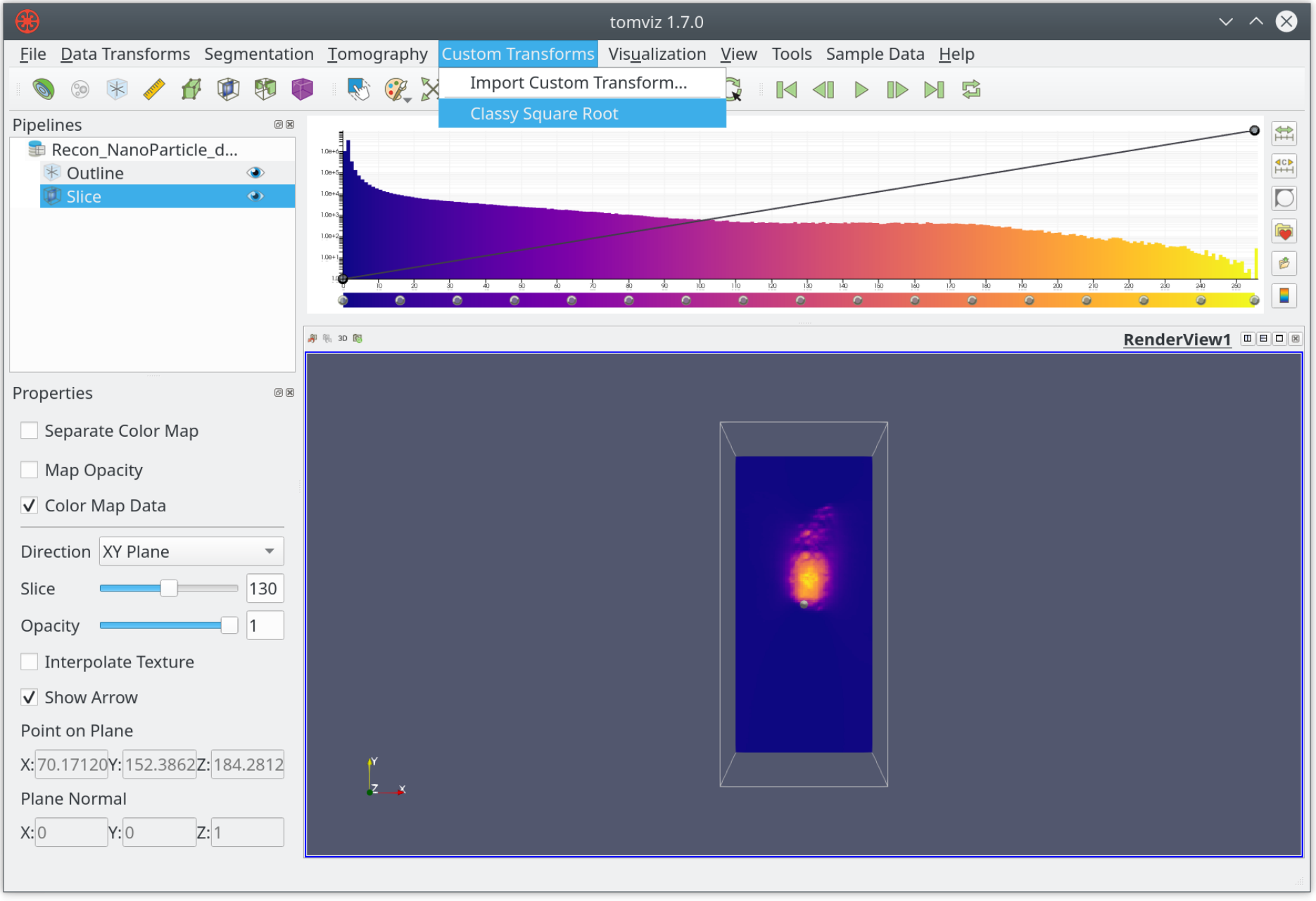
JSON for Increased Control

```
{
  "name": "Fancy Square Root",
  "label": "Classy Square Root",
  "description": "This is the fanciest square root operator, it does
  "parameters": [
    {
      "name": "number_of_chunks",
      "label": "Number of Chunks",
      "type": "int",
      "default": 10,
      "minimum": 1,
      "maximum": 1000
    }
  ]
}
```

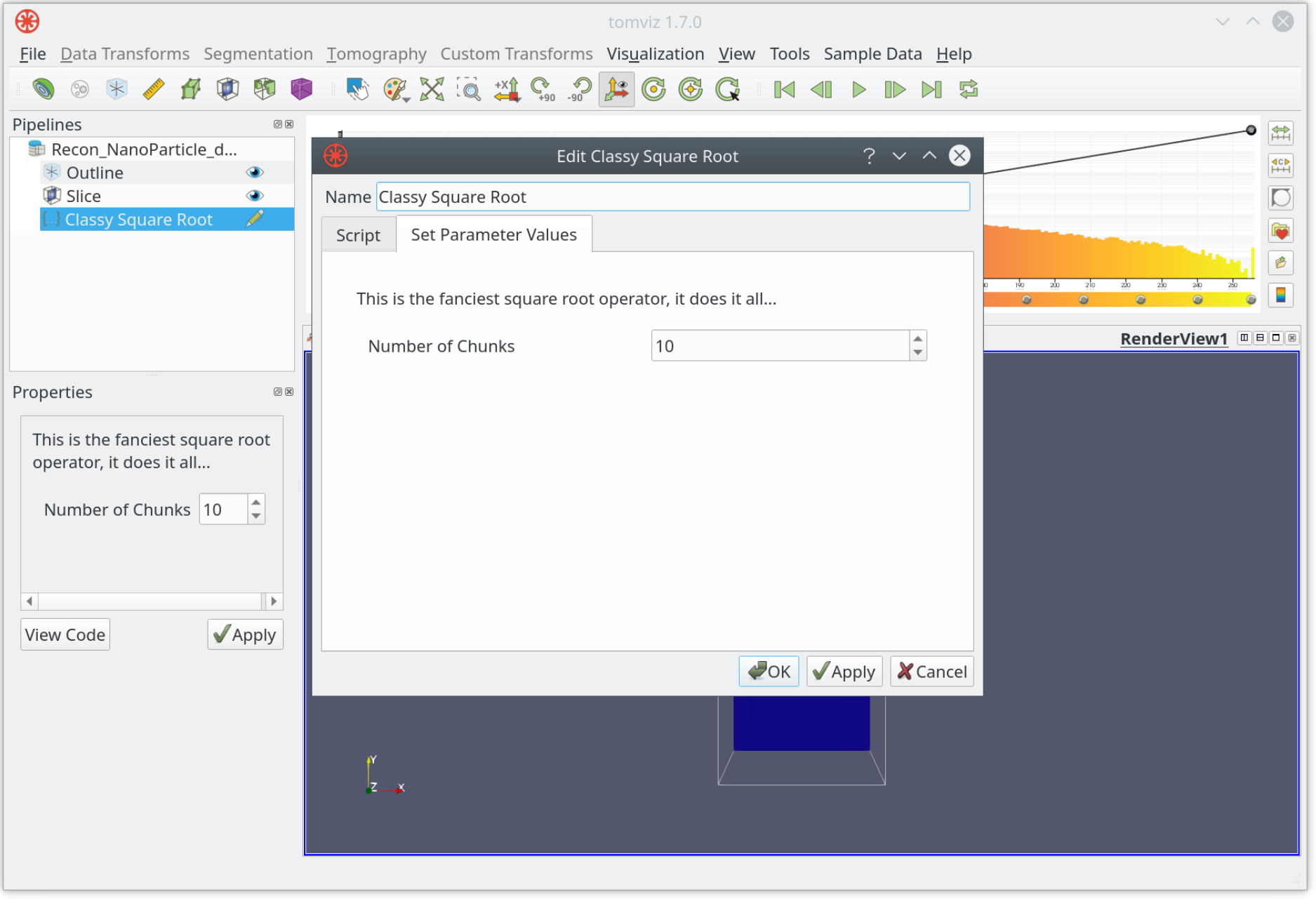
Classier Operators—Complete

```
import tomviz.operators

class SquareRootOperator(tomviz.operators.CancelableOperator):

  def transform_scalars(self, dataset, number_of_chunks = 10):
    from tomviz import utils
    import numpy as np
    self.progress.maximum = number_of_chunks
    scalars = utils.get_scalars(dataset)
    # Process dataset in chunks, user can cancel.
    result = np.float32(scalars)
    step = 0
    for chunk in np.array_split(result, number_of_chunks):
      if self.canceled:
        return
      np.sqrt(chunk, chunk)
```

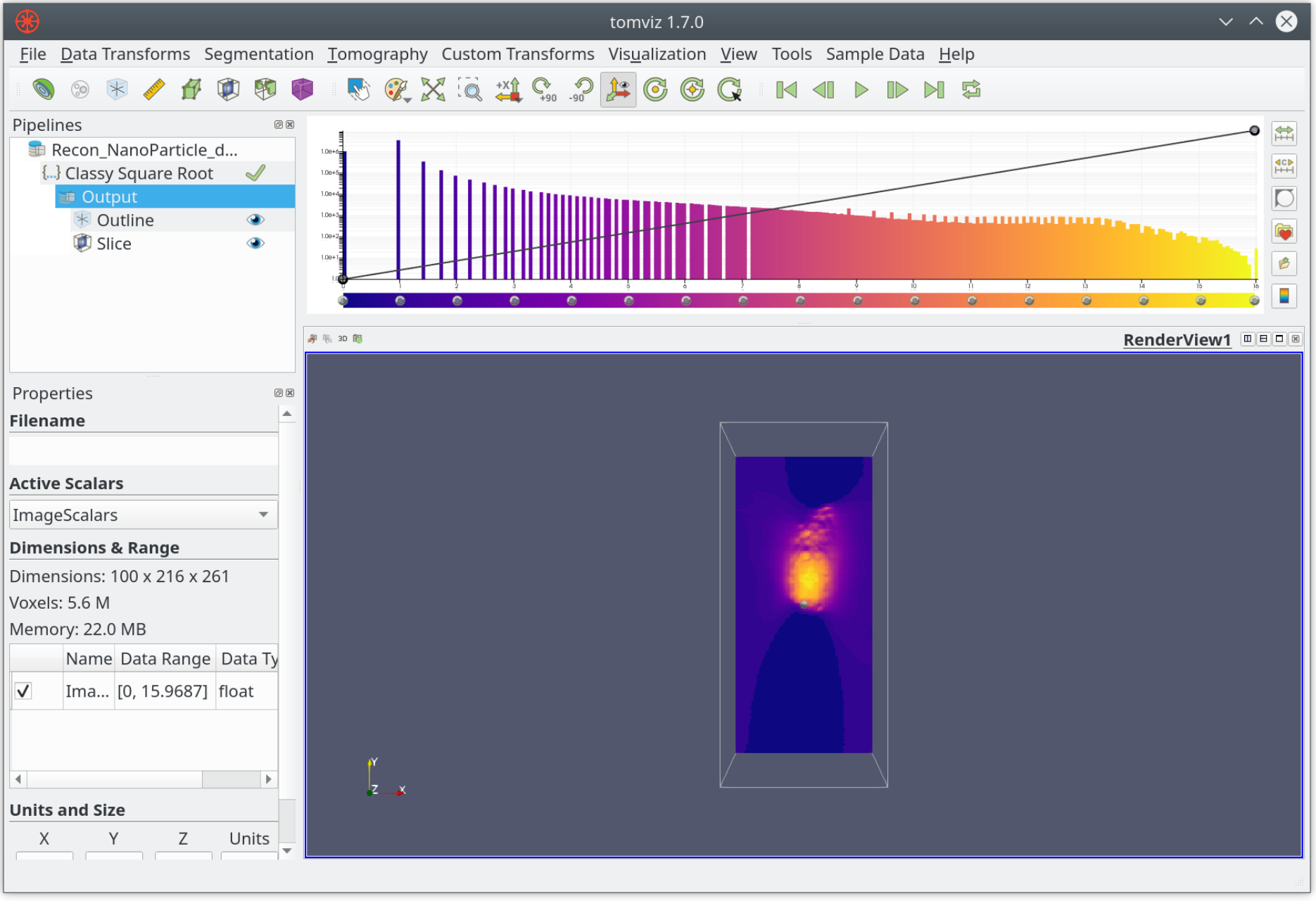
Custom Transforms—JSON Menu


tomviz 1.7.0
File
Data Transforms
Segmentation
Tomography
Custom Transforms
Visualization
View
Tools
Sample Data
Help
Import Custom Transform...
Classy Square Root
Pipelines
Recon_NanoParticle_d...
Outline
Slice
Properties
Separate Color Map
Map Opacity
Color Map Data
Direction
XY Plane
Slice
130
Opacity
1
Interpolate Texture
Show Arrow
Point on Plane
X: 70.17120 Y: 152.3862 Z: 184.2812
Plane Normal
X: 0 Y: 0 Z: 1
RenderView1

Custom Transforms—User Input


tomviz 1.7.0
File Data Transforms Segmentation Tomography Custom Transforms Visualization View Tools Sample Data Help
Pipelines
Recon_NanoParticle_d...
Outline
Slice
Classy Square Root
Properties
This is the fanciest square root operator, it does it all...
Number of Chunks 10
View Code
Apply
Edit Classy Square Root
Name Classy Square Root
Script
Set Parameter Values
This is the fanciest square root operator, it does it all...
Number of Chunks 10
OK
Apply
Cancel
RenderView1

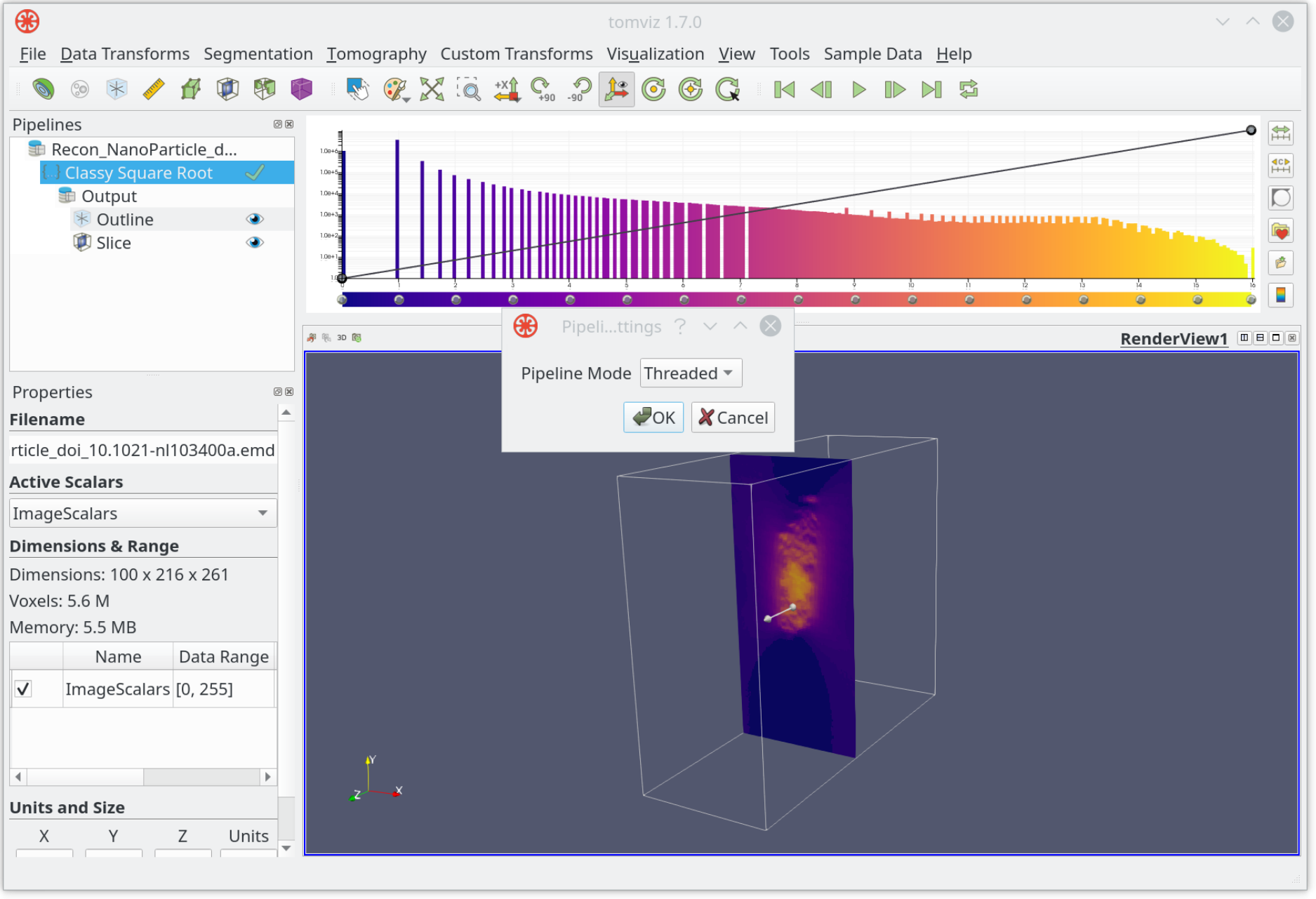
Custom Transforms—Result


tomviz 1.7.0
File Data Transforms Segmentation Tomography Custom Transforms Visualization View Tools Sample Data Help
Pipelines
Recon_NanoParticle_d...
Classy Square Root
Output
Outline
Slice
Properties
Filename
Active Scalars
ImageScalars
Dimensions & Range
Dimensions: 100 x 216 x 261
Voxels: 5.6 M
Memory: 22.0 MB
Name
Data Range
Data Ty
Ima...
[0, 15.9687]
float
Units and Size
X
Y
Z
Units
RenderView1

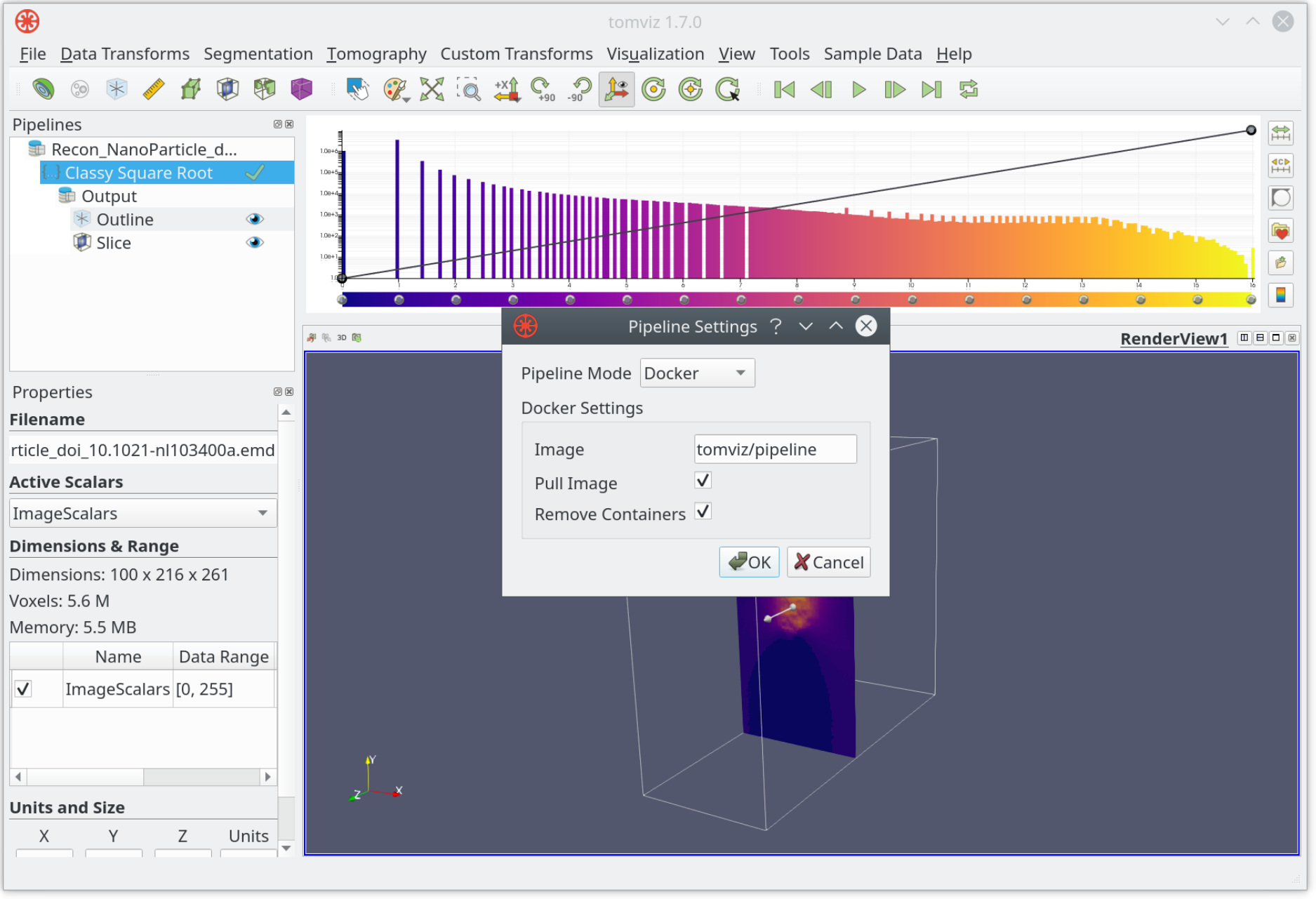
External Pipelines

- Tomviz usually runs pipelines in a background thread
  - This can be changed in Pipeline Settings
  - Threaded (default) or Docker
- Working on making this more flexible in the future

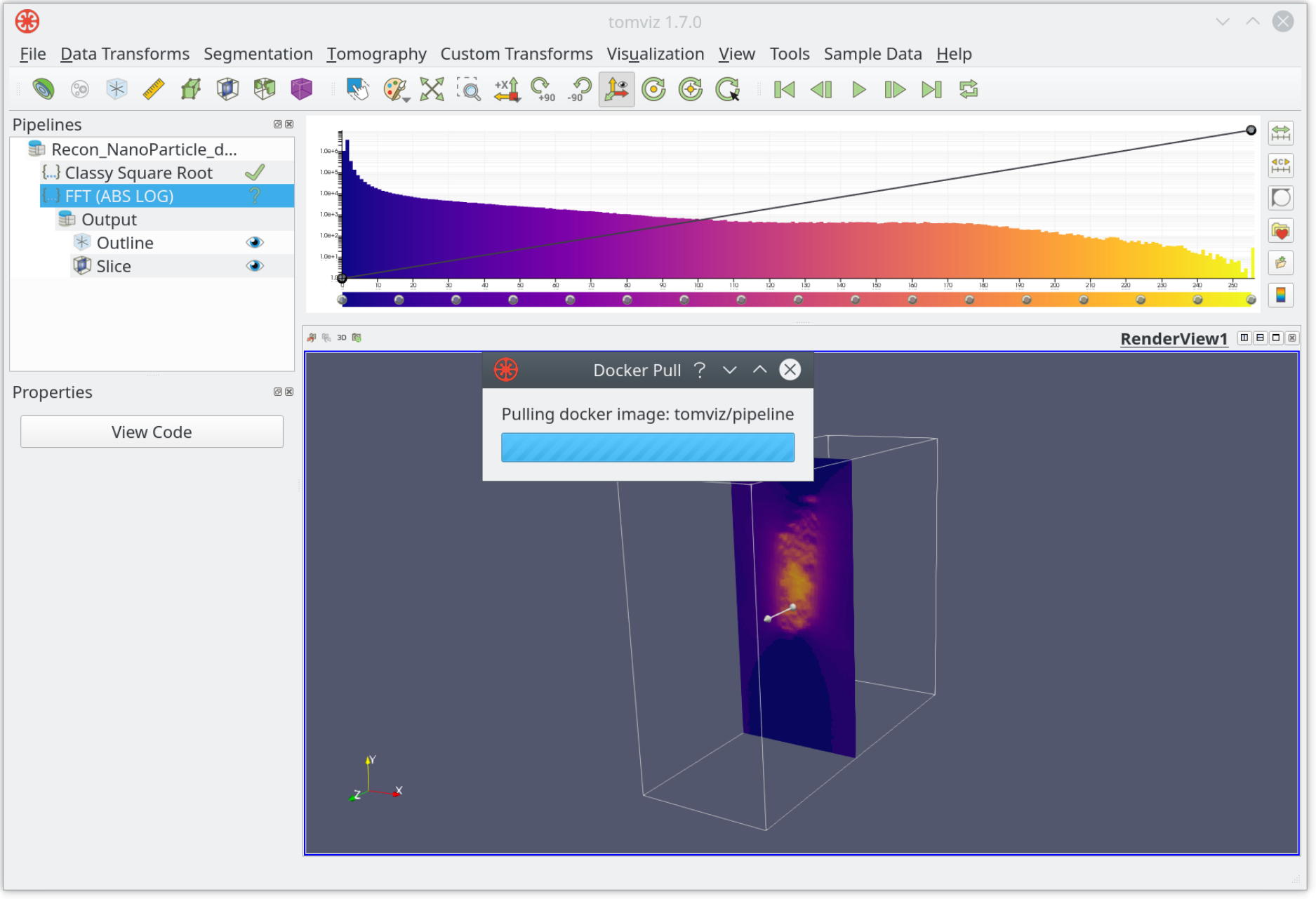
Pipeline Settings


tomviz 1.7.0
File Data Transforms Segmentation Tomography Custom Transforms Visualization View Tools Sample Data Help
Pipelines
Recon_NanoParticle_d...
Classy Square Root
Output
Outline
Slice
Pipeli...ttings
Pipeline Mode
Threaded
OK
Cancel
RenderView1
Properties
Filename
rticle_doi_10.1021-nl103400a.emd
Active Scalars
ImageScalars
Dimensions & Range
Dimensions: 100 x 216 x 261
Voxels: 5.6 M
Memory: 5.5 MB
Name
Data Range
ImageScalars
[0, 255]
Units and Size
X
Y
Z
Units

Pipeline Settings—Docker


tomviz 1.7.0
File Data Transforms Segmentation Tomography Custom Transforms Visualization View Tools Sample Data Help
Pipelines
Recon_NanoParticle_d...
Classy Square Root
Output
Outline
Slice
Properties
Filename
rticle_doi_10.1021-nl103400a.emd
Active Scalars
ImageScalars
Dimensions & Range
Dimensions: 100 x 216 x 261
Voxels: 5.6 M
Memory: 5.5 MB
Name
Data Range
ImageScalars
[0, 255]
Units and Size
X
Y
Z
Units
Pipeline Settings
Pipeline Mode
Docker
Docker Settings
Image
tomviz/pipeline
Pull Image
Remove Containers
OK
Cancel
RenderView1

Pulling Docker Image


tomviz 1.7.0
File Data Transforms Segmentation Tomography Custom Transforms Visualization View Tools Sample Data Help
Pipelines
Recon_NanoParticle_d...
Classy Square Root
FFT (ABS LOG)
Output
Outline
Slice
Properties
View Code
RenderView1
Docker Pull
Pulling docker image: tomviz/pipeline

Running Pipelines

- Create a virtual environment, install

```
$ git clone --recursive git://github.com/openchemistry/
$ cd tomviz/tomviz/python
$ mkvirtualenv tomviz
$ pip install -e .
```

- Use Tomviz to build a pipeline
  - Save the state file (JSON)

Running Pipelines

- Run the pipeline in the virtual environment

```
$ tomviz-pipeline -s state.tvsm -d data.emd -o output.emd
[2019-07-23 14:14:59,647] INFO: Executing 'Invert Data' op
[2019-07-23 14:14:59,963] INFO: Writing transformed data.
```

- Input must be an EMD file right now
- Operators are executed in sequence
- Output is written to an EMD file

Summary

- Adding custom operators is easy
  - NumPy centric approach
  - Runs in a background thread
  - Options to customize input, etc
- The pipeline can be run externally
  - A few bugs we will fix in the next release
  - Docker out of the box, Python batch possible
- File formats and more are in the works