L14, 19 March: Affine transformations, LLNL VisIT

1) Download Moodle/Week 10/Pgm11_Affine_Transformation.nb

How can we align two volumes? For example, in an MRI, the patient is breathing. In a stress test, the sample is distorting. During heating, the sample is swelling. All of these are examples of data sets that can be aligned with a combination of translation, rotation, shearing, and scaling transformations. The general term is affine transformations (straight lines are preserved).

TranslationTransform

`TranslationTransform[v]`
gives a `TransformationFunction` that represents translation of points by a vector *v*.

MORE INFORMATION

EXAMPLES

Basic Examples (1)

Generate a function representing a translation by the vector $\{a, b\}$:

```
In[1]:= TranslationTransform[{a, b}]
```

Out[1]= TransformationFunction$\left[\left(\begin{array}{cc|c} 1 & 0 & a \\ 0 & 1 & b \\ \hline 0 & 0 & 1 \end{array}\right)\right]$

Apply the transformation function to a vector:

```
In[2]:= %[{x, y}]
Out[2]= {a + x, b + y}
```

```
In[131]:= funcTrans = TranslationTransform[{a, b, c, d}]
```

Out[131]= TransformationFunction$\left[\left(\begin{array}{cccc|c} 1 & 0 & 0 & 0 & a \\ 0 & 1 & 0 & 0 & b \\ 0 & 0 & 1 & 0 & c \\ 0 & 0 & 0 & 1 & d \\ \hline 0 & 0 & 0 & 0 & 1 \end{array}\right)\right]$

```
In[132]:= funcTrans[{x, y, z, t}]
Out[132]= {a + x, b + y, c + z, d + t}
```

RotationTransform

`RotationTransform[θ]`
gives a `TransformationFunction` that represents a rotation in 2D by θ radians about the origin.

`RotationTransform[θ, p]`
gives a 2D rotation about the 2D point p.

`RotationTransform[θ, w]`
gives a 3D rotation around the direction of the 3D vector w.

`RotationTransform[θ, w, p]`
gives a 3D rotation around the axis w anchored at the point p.

`RotationTransform[{u, v}]`
gives a rotation about the origin that transforms the vector u to the direction of the vector v.

`RotationTransform[{u, v}, p]`
gives a rotation about the point p that transforms u to the direction of v.

`RotationTransform[θ, {u, v}, …]`
gives a rotation by θ radians in the hyperplane spanned by u and v.

MORE INFORMATION

EXAMPLES

Basic Examples (4)

A 2D rotation transform by θ radians:

```
In[1]:= r = RotationTransform[θ]
```

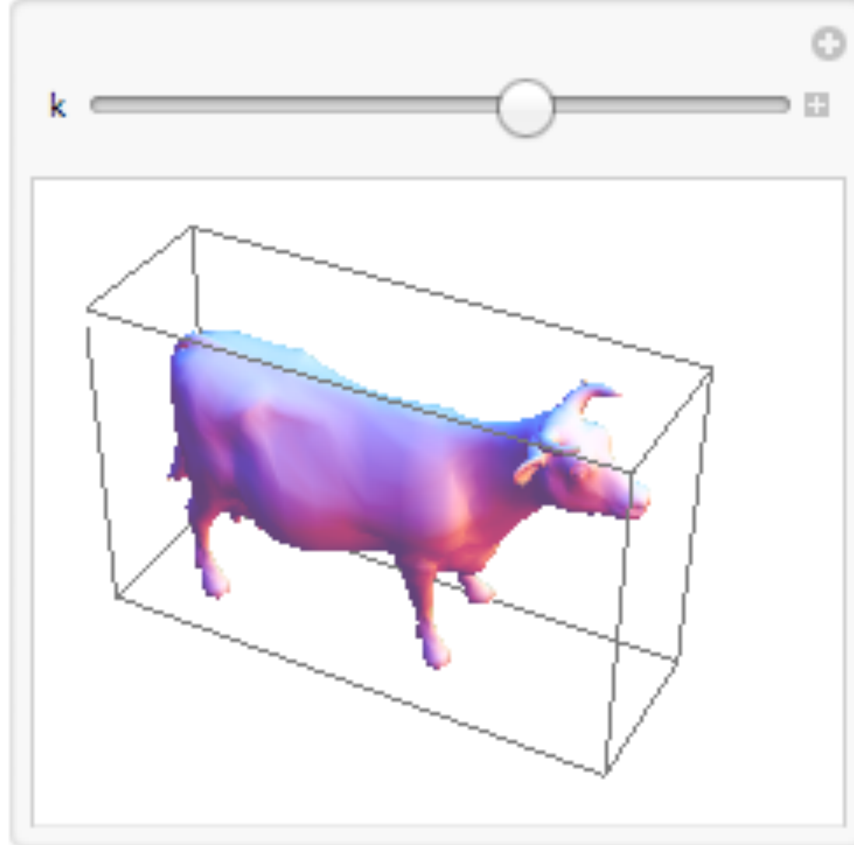
Out[1]= TransformationFunction$\left[\left(\begin{array}{cc|c} \text{Cos}[\theta] & -\text{Sin}[\theta] & 0 \\ \text{Sin}[\theta] & \text{Cos}[\theta] & 0 \\ \hline 0 & 0 & 1 \end{array}\right)\right]$

Rotate around the x axis:

```
In[1]:= cow = ExampleData[{"Geometry3D", "Cow"}, "GraphicsComplex"];

In[2]:= Manipulate[Graphics3D[{EdgeForm[None], GeometricTransformation[
            cow, RotationTransform[k Pi / 8, {1, 0, 0}]]}],
         {k, -1, 1}, SaveDefinitions → True]
```

Out[2]=



Note: For a rotation, the determinant of the rotation matrix is +1.

Later, we'll see determinants near +1 due to rescaling.

A determinant of -1 indicates a change in handedness (inversion).

Finding related points in two data sets. This can be done by eye or "ImageCorrespoindingPoints"

■ Step 2. Find related points

```
133]:= listRelatedPoints = ImageCorrespondingPoints[imageBefore, imageAfter];
       Dimensions[listRelatedPoints]
       TableForm[listRelatedPoints[[All, 1 ;; 10, All]]]
       pointsBefore = listRelatedPoints[[1, All, All]];
       pointsAfter = listRelatedPoints[[2, All, All]];
```

```
[134]= {2, 748, 2}
```

35]//TableForm=

721.349	147.2	519.804	536.477	560.88	659.294	708.308	23.5232	105.075	557.335
513.032	509.188	508.321	508.412	508.304	508.256	508.505	506.31	506.228	502.478
719.863	148.422	519.548	536.461	560.64	658.604	707.933	25.1461	105.541	557.551
573.724	520.12	551.174	552.974	554.99	563.313	568.534	505.918	511.732	548.633

FindGeometricTransform

New in 8

`FindGeometricTransform[`pts_1`, `pts_2`]`
finds a geometric transformation between two geometries pts_1 and pts_2, returning the alignment error together with the transformation function.

MORE INFORMATION

- `FindGeometricTransform` returns an expression of the form `{`*err*`, `*trfun*`}`, where *err* is an estimate of the average alignment error, and *trfun* is a transformation function. The function *trfun* can be applied to the positions pts_2 to align them with the positions pts_1.
- The geometries pts_1 and pts_2 can be given as lists of position coordinates or *Mathematica* graphics objects.
- `FindGeometricTransform[`$image_1$`, `$image_2$`]` uses corresponding points between the two images to find the geometric transformation.
- `FindGeometricTransform` works with points in any dimensions as well as with built-in 2D and 3D graphics primitives.
- `FindGeometricTransform` takes a `"Transformation"` option. By default it automatically finds the most suitable geometric transformation for the given sets of positions.
- Possible settings for the `"Transformation"` option include:

`"Translation"`	translation only
`"Rigid"`	translation and rotation
`"Similarity"`	translation, rotation, and scaling
`"Affine"`	linear transformation and translation
`"Perspective"`	linear fractional transformation

```
{error, funcTranRotScale} = FindGeometricTransform[pointsAfter, pointsBefore,
  "Transformation" → "Translation"]
```

$$\left\{0.527775,\ \text{TransformationFunction}\left[\left(\begin{array}{cc|c} 1 & 0 & 35.4573 \\ 0 & 1 & 23.051 \\ \hline 0 & 0 & 1 \end{array}\right)\right]\right\}$$

Very close to the difference in image size (before to after rotation).

```
{error, funcTranRotScale} = FindGeometricTransform[pointsAfter, pointsBefore,
  "Transformation" → "Rigid"]
```

$$\left\{0.39384,\ \text{TransformationFunction}\left[\left(\begin{array}{cc|c} 0.996497 & -0.0870749 & 45.7699 \\ 0.0869174 & 0.995543 & 0.141122 \\ \hline 0 & 0 & 1 \end{array}\right)\right]\right\}$$

```
{error, funcTranRotScale} = FindGeometricTransform[pointsAfter, pointsBefore,
  "Transformation" → "Similarity"]
```

$$\left\{0.40718,\ \text{TransformationFunction}\left[\left(\begin{array}{cc|c} 0.9959 & -0.0869419 & 45.9519 \\ 0.0869419 & 0.9959 & 0.0310733 \\ \hline 0 & 0 & 1 \end{array}\right)\right]\right\}$$

If we rotated by -5 degrees, the negative sign would move to the lower left corner element.

```
{error, funcTranRotScale} = FindGeometricTransform[pointsAfter, pointsBefore,
  "Transformation" → "Affine"]
```

$$\left\{0.394,\ \text{TransformationFunction}\left[\left(\begin{array}{cc|c} 0.996581 & -0.0872686 & 45.8063 \\ 0.0870661 & 0.995319 & 0.127272 \\ \hline 0 & 0 & 1 \end{array}\right)\right]\right\}$$

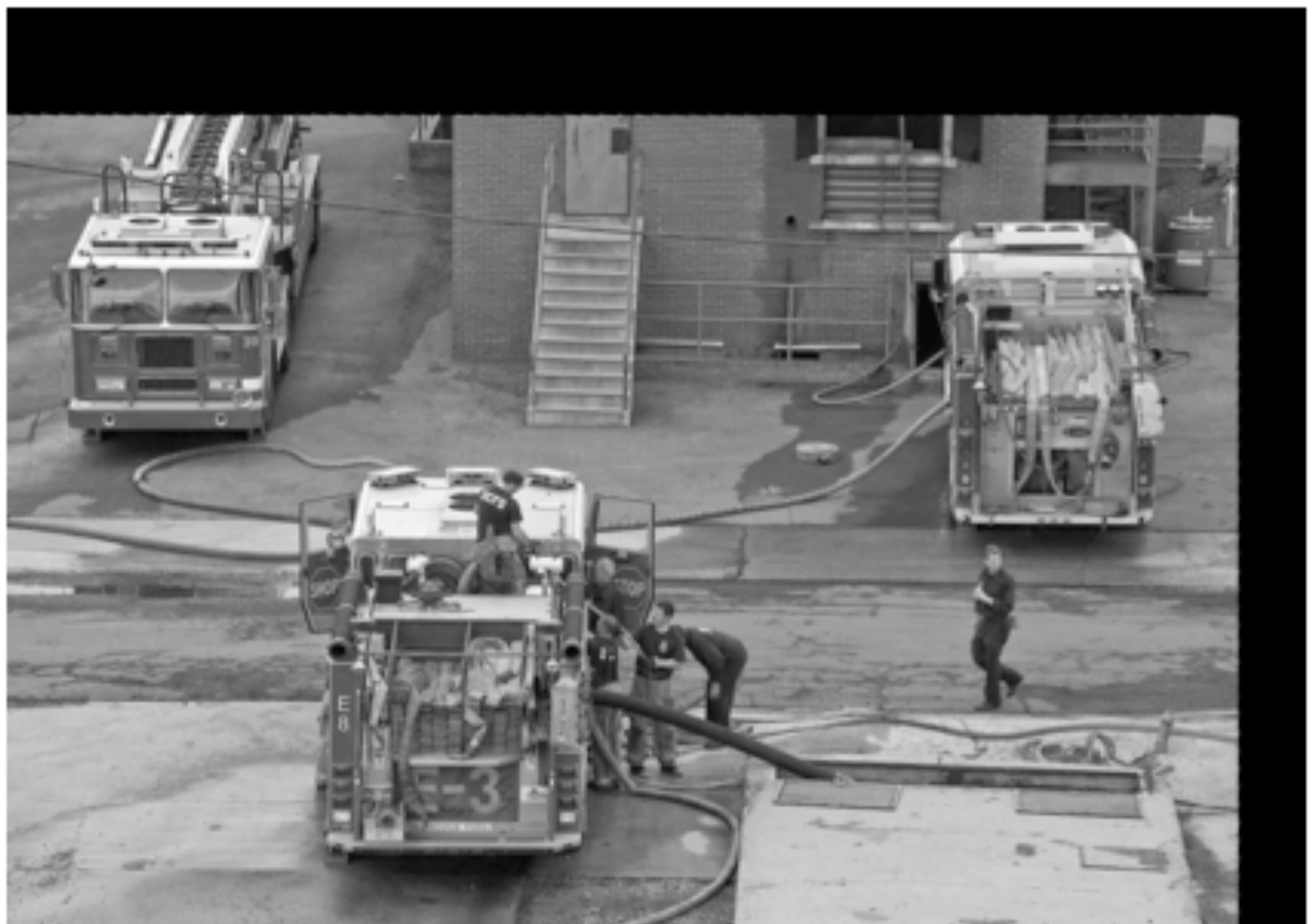
```
{error, funcTranRotScale} = FindGeometricTransform[pointsAfter, pointsBefore,
  "Transformation" → "Perspective"]
```

$$\left\{0.390452,\ \text{TransformationFunction}\left[\left(\begin{array}{cc|c} 0.995768 & -0.0873998 & 45.9133 \\ 0.0870113 & 0.995107 & 0.134926 \\ \hline -4.99798\times10^{-7} & -5.80351\times10^{-7} & 1. \end{array}\right)\right]\right\}$$

For this transform, the error (RMS?) over the 748 corresponding points is about 0.39 pixels.

▪ Step 5. Apply the geometric transformation:

```
imageAfterAffine = ImageTransformation[imageAfter, funcTranRotScale, DataRange → All];
imageAfterAffineCropping = ImageCrop[imageAfterAffine, {widthBefore, heightBefore}, {Right, Top}];
GraphicsRow[{imageAfterAffine, imageAfterAffineCropping}, ImageSize → 700]
```



- **Step 6. And then subtraction, binarize, dilation, edge detection, morphological component analysis and component properties, and image addition**

49]:=

```
imageDifference = Dilation[Binarize[ImageDifference[imageAfterAffineCropping, imageBefore], 0.3],
   4];
imagePerimeter = Dilation[EdgeDetect[imageDifference], 1];
imageComponents = MorphologicalComponents[imageDifference];
(* imageComponents// Colorize *)
ComponentMeasurements[imageComponents, "Count"]
GraphicsRow[{ImageAdd[imageBefore, imagePerimeter],
  ImageAdd[imageAfterAffineCropping, imagePerimeter]}, ImageSize → 700]
```

152]=

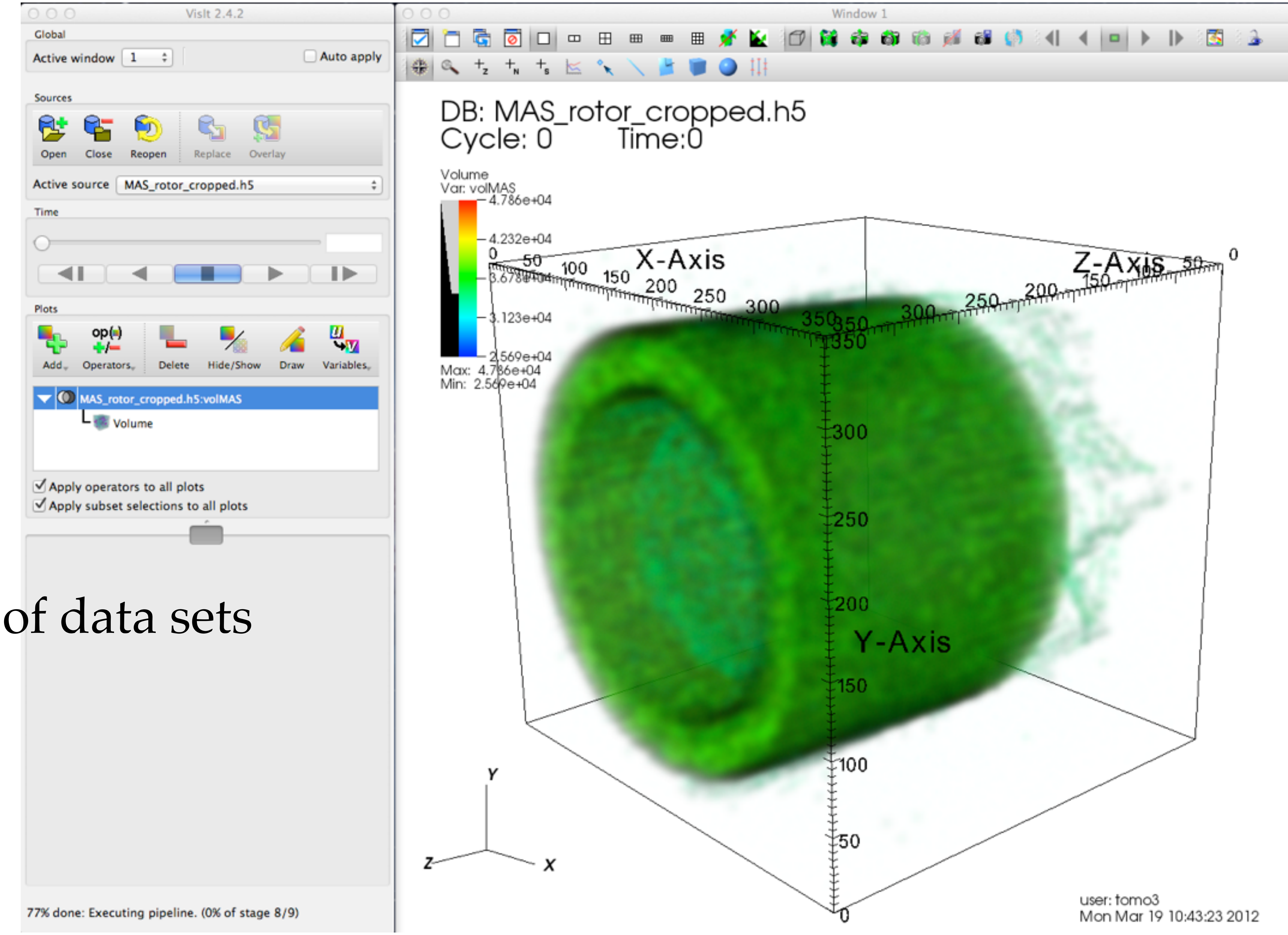
```
{1 → 834, 2 → 1615, 3 → 310, 4 → 810, 5 → 244, 6 → 678,
 7 → 1017, 8 → 159, 9 → 1777, 10 → 2074, 11 → 90, 12 → 81, 13 → 193}
```

Whoops! Missed the “8” to “3”.

153]=

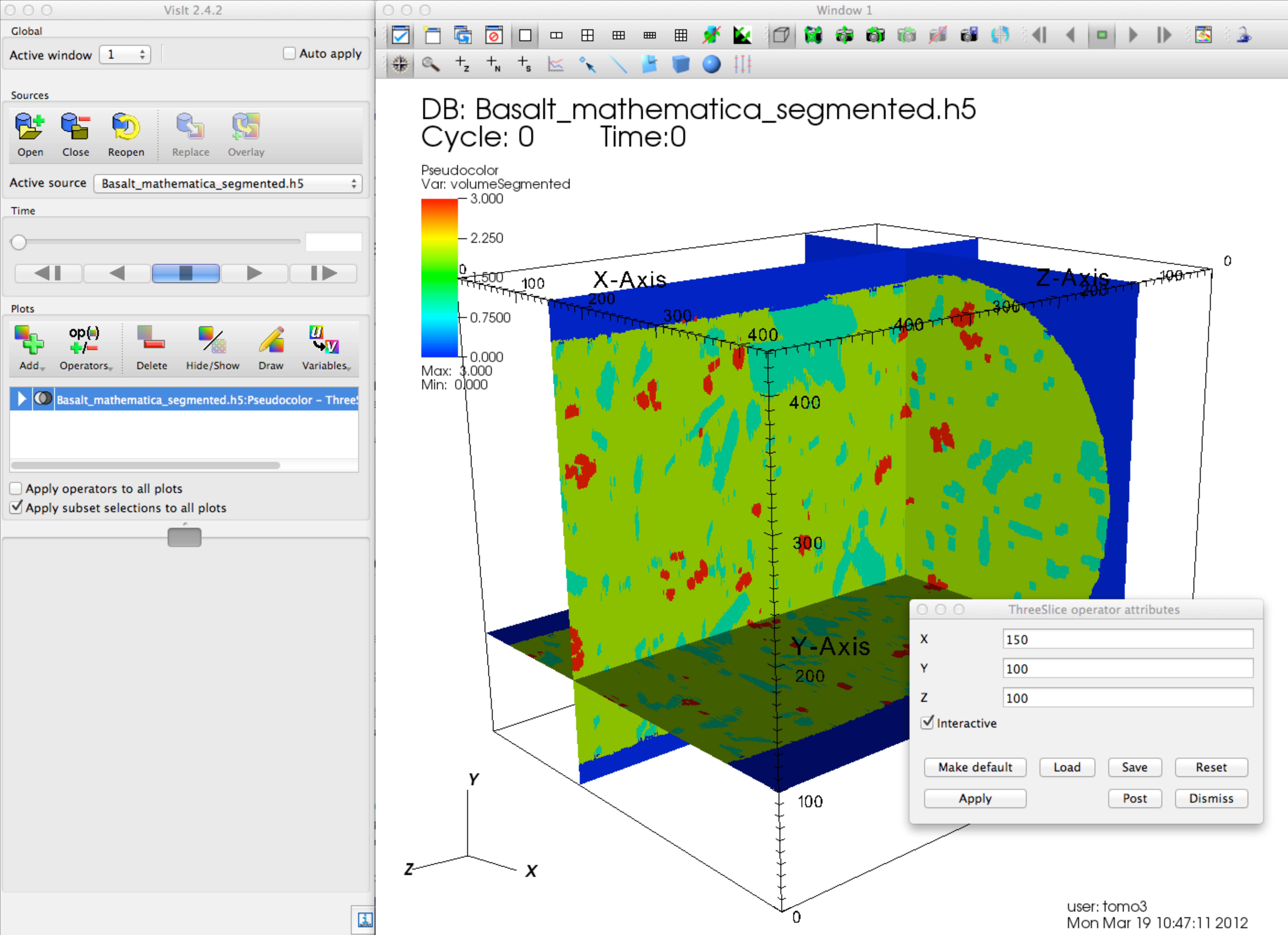
LLNL VisIT

- free
- PC, Mac, cluster
- imports HDF5
- GUI
- Python scripting
- optimized for sequences of data sets



https://wci.llnl.gov/codes/visit/

http://www.visitusers.org/


VisIt 2.4.2
Global
Active window 1
Auto apply
Sources
Open
Close
Reopen
Replace
Overlay
Active source Basalt_mathematica_segmented.h5
Time
Plots
Add
Operators
Delete
Hide/Show
Draw
Variables
Basalt_mathematica_segmented.h5:Pseudocolor - Three
Apply operators to all plots
Apply subset selections to all plots
Window 1
DB: Basalt_mathematica_segmented.h5
Cycle: 0 Time:0
Pseudocolor
Var: volumeSegmented
3.000
2.250
1.500
0.7500
0.000
Max: 3.000
Min: 0.000
X-Axis
Y-Axis
Z-Axis
ThreeSlice operator attributes
X 150
Y 100
Z 100
Interactive
Make default
Load
Save
Reset
Apply
Post
Dismiss
user: tomo3
Mon Mar 19 10:47:11 2012

LLNL VisIT

1) Menu bar / Controls / Command
2) Menu bar / Operator Attributes / Slicing / ThreeSlice
3) In Commands, press Record
4) In ThreeSlice, change one number and press Apply
5) In Commands, press Stop

The result is a short Python program.