L12, 7 Mar, Segmentation: MAS Rotor (part 3)

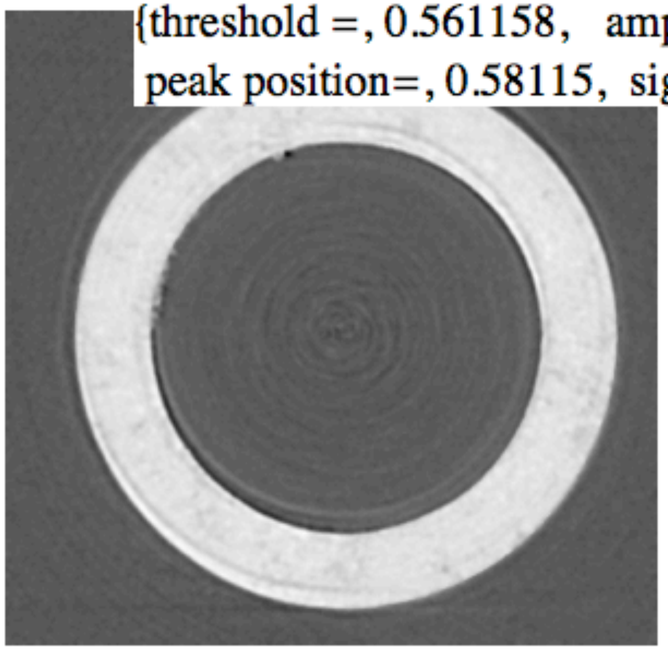
1) Download Moodle/Week 8/Pgm10_Segmentation_MAS_rotor_v2.nb
2) Download Moodle/Week 8/MAS_rotor_cropped.h5 (177MB)

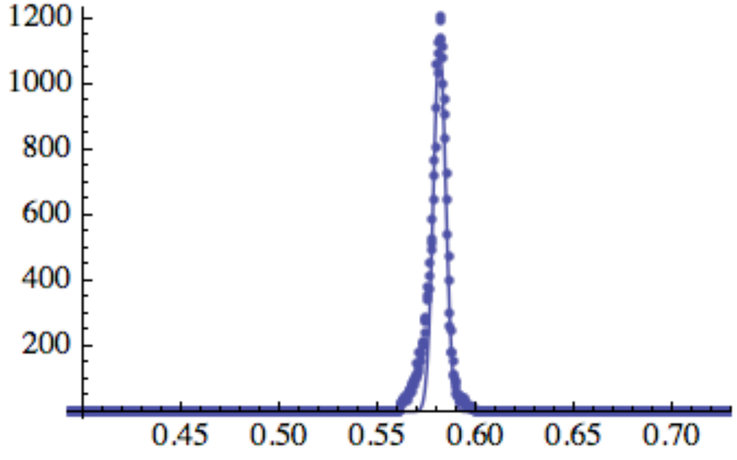
indexRow

threshold

saveParameters

{threshold =, 0.561158, amp=, 1107.07, [redacted]
peak position=, 0.58115, sigma=, 0.0000261375, variance=, 3736.6}





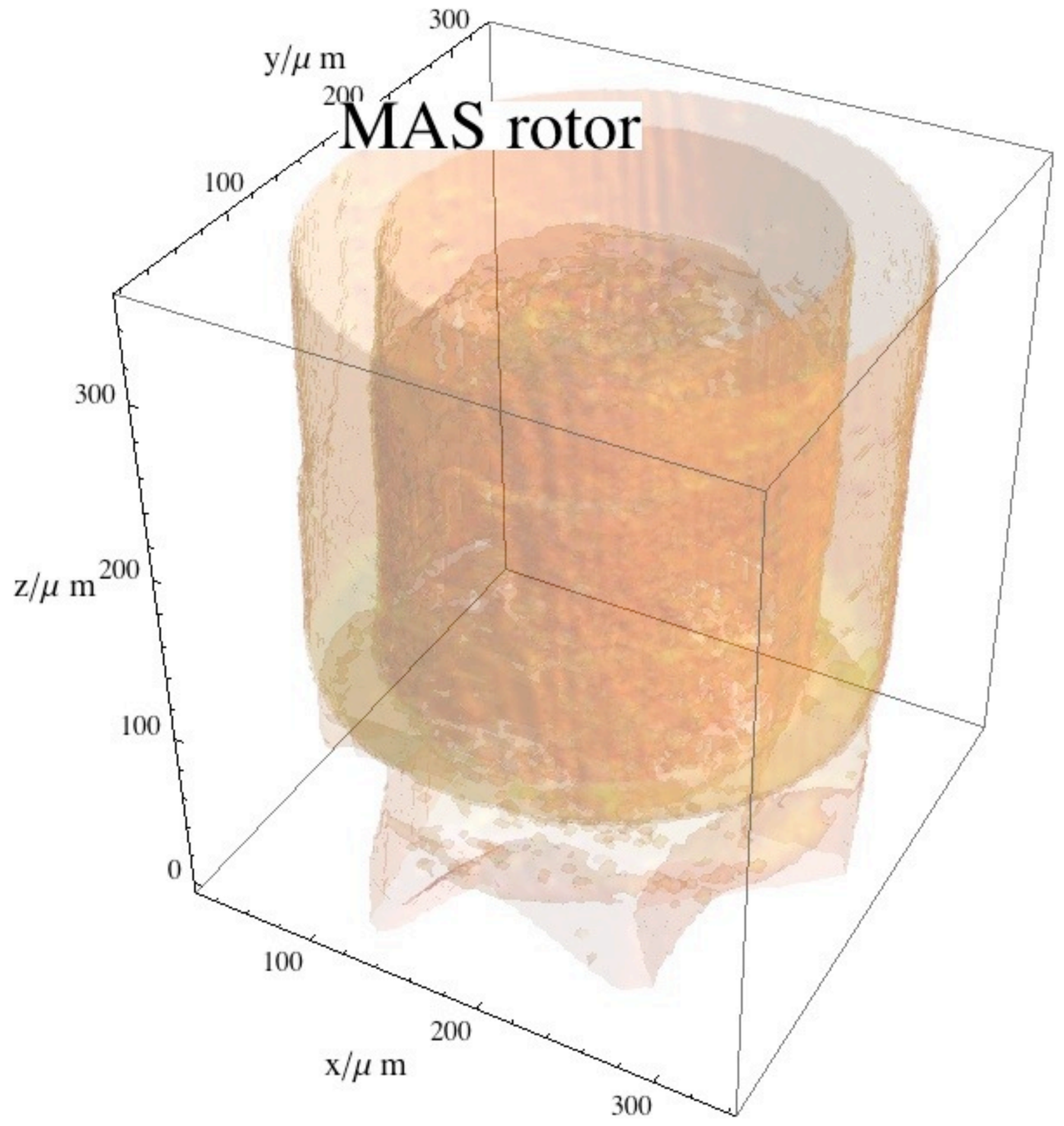
Philosophy

Our *a priori* knowledge about a sample can be used in segmentation.

Two If statements for defining boundaries of only Kel-F and only zirconia.

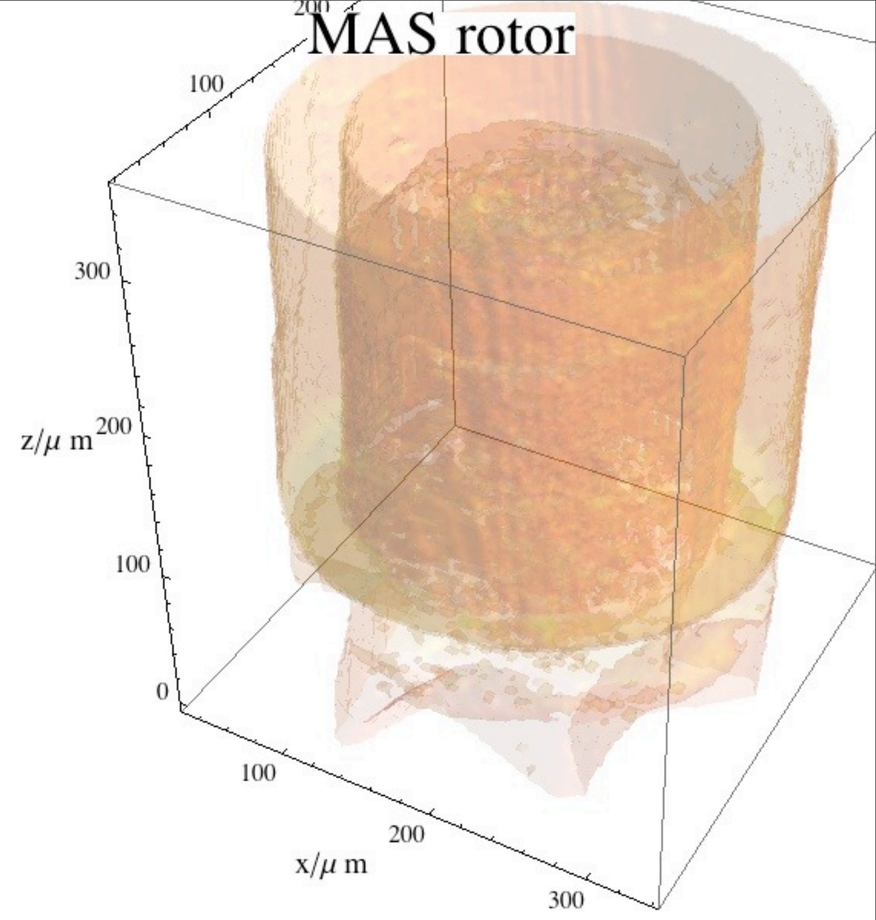
Human eye used to find best values of binarization threshold.

Subtract zirconia binary image from original image to get a Kel-F only image.



Mathematica commands - image processing

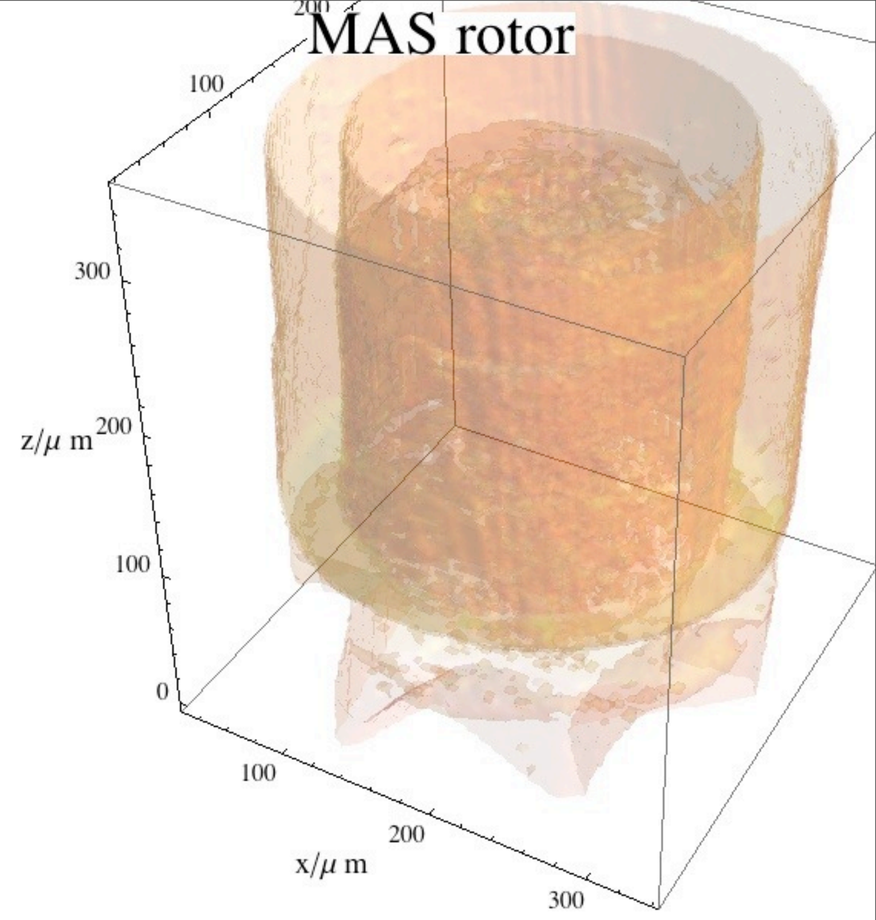
MorphologicalBinarization - a binarization
GeometricMeanFilter - a median filter
TotalVariationFilter - noise reducing filter
ImageMultiply -
ImageSubtract -
ImageData - convert image format to array of numbers
MorphologicalComponents - makes a label field
ComponentMeasurements - inspects a label field
Erosion
Closing
ImageAdjust



NOTE: Know these commands. See Mathematica help files.

Mathematica commands - list commands

Position
Ordering
Reverse
First, Rest
ConstantArray
Length
Dimensions
Flatten
Append



Mathematica commands - set commands

Intersection

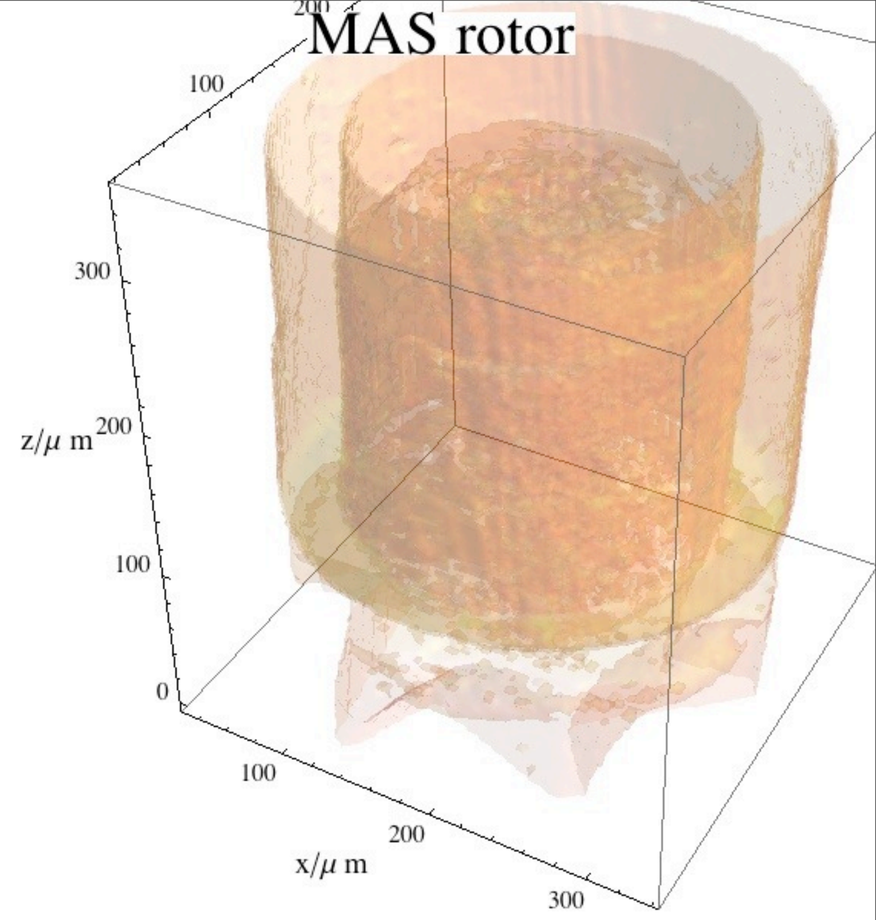
NOTE: Know these commands. See Mathematica help files.

Mathematica commands - procedures

Manipulate
Module
Do
If
Map

Mathematica commands - Input/Output

NotebookDirectory
Import
Export
FileNames



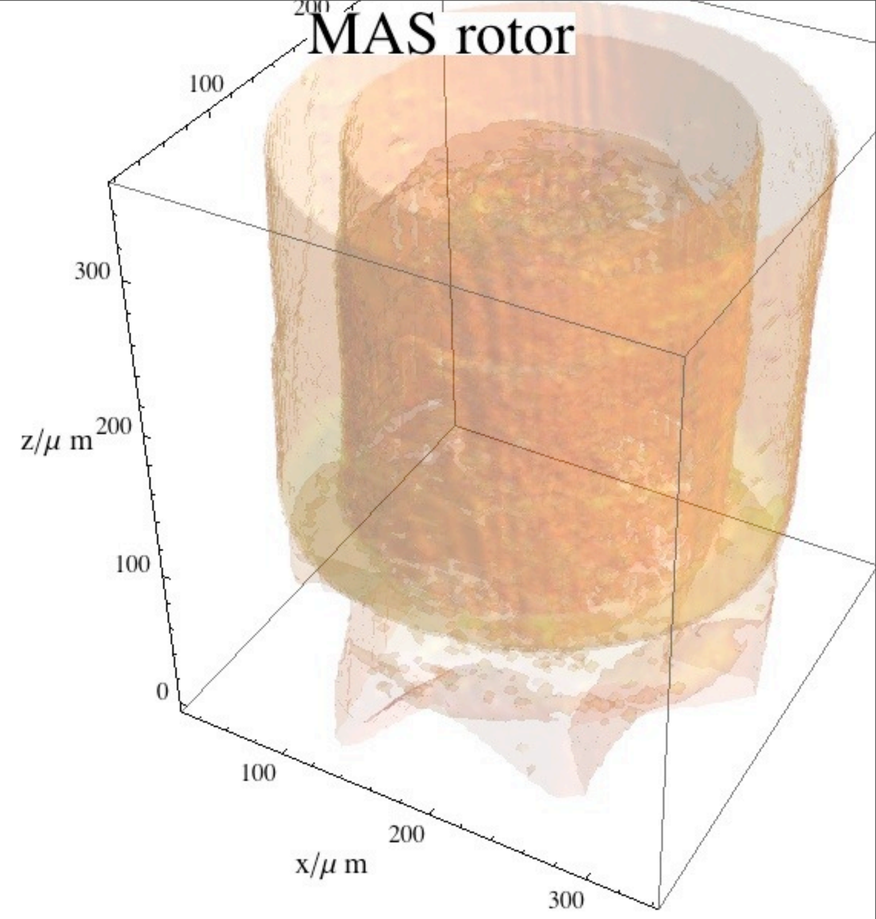
NOTE: Will not be explicitly tested on the exam.

Mathematica commands - computation

FindFit

ReplaceAll (/.)

Interpolation

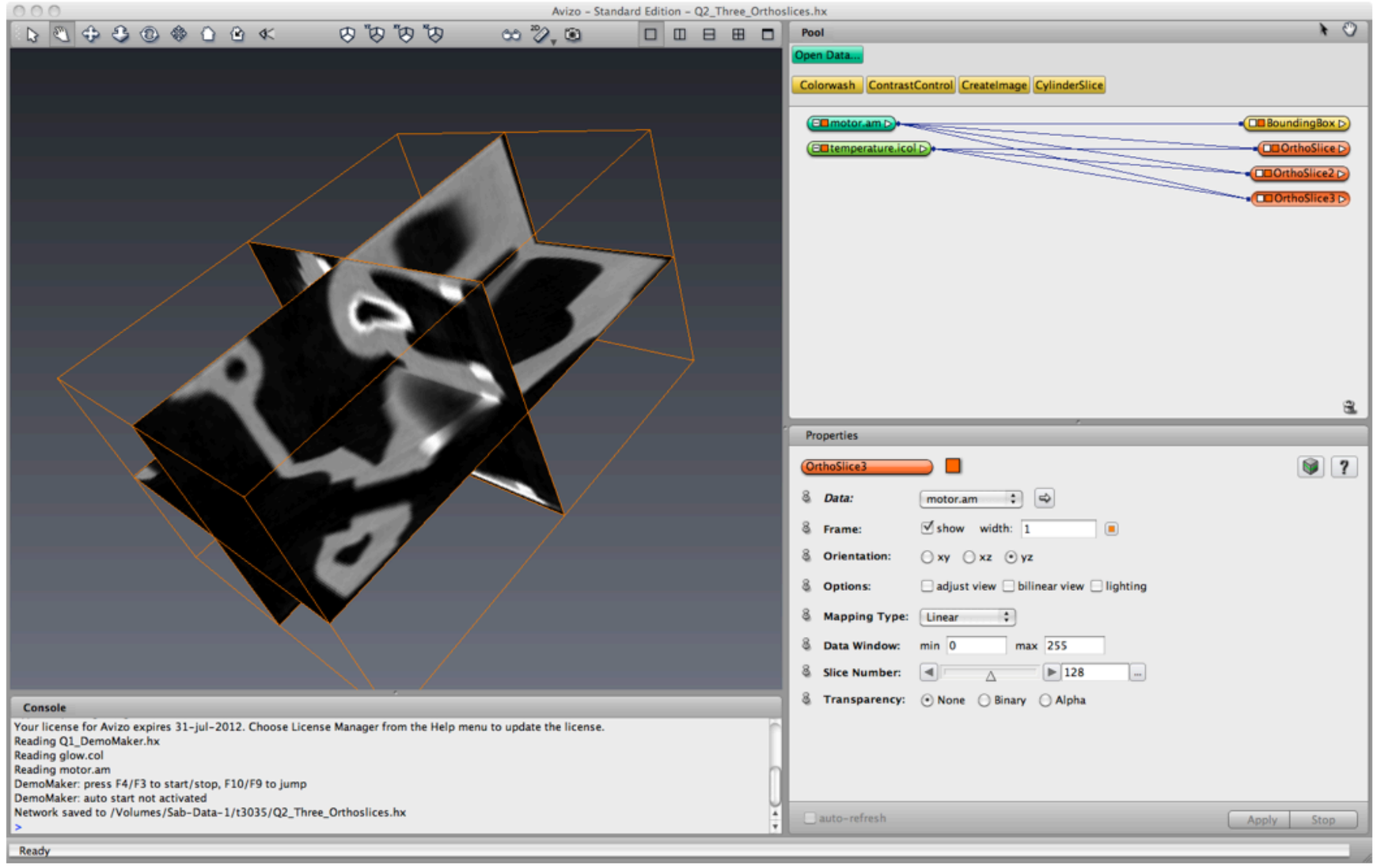


NOTE: Will not be explicitly tested on the exam.

Q2. Please explain how three orthoslices can be shown at one time.

Key: (page 41-42) By now, most Avizo users can probably do this in their sleep. The programming elements are medium quality: (1) Right mouse on the data element and "pool window" shows common options like an OrthoSlice. (2)_ The OrthoSlice defaults to "xy" plane. (3) A bounding box nicely frames the slices.

Curiously, the user gets no visible clues about axes orientation; it's a complete surprise as to which plane will show up. I don' like the itsy-bitsy on/off button on OrthoSlice; it's used way to often to be left so small.



It would be nice to be see the connections between the main/console/pool windows. What if I moused over a slice in the main window and the icon in Pool flashed, and the console showed the data structure essentials (dimensions, min/max). Maybe the line between data source and OrthosSlice should flash. Conversely, if we highlight an OrthoSlice icon, we might like to see the slice highlighted (perimeter?) in the main window. Your thoughts?

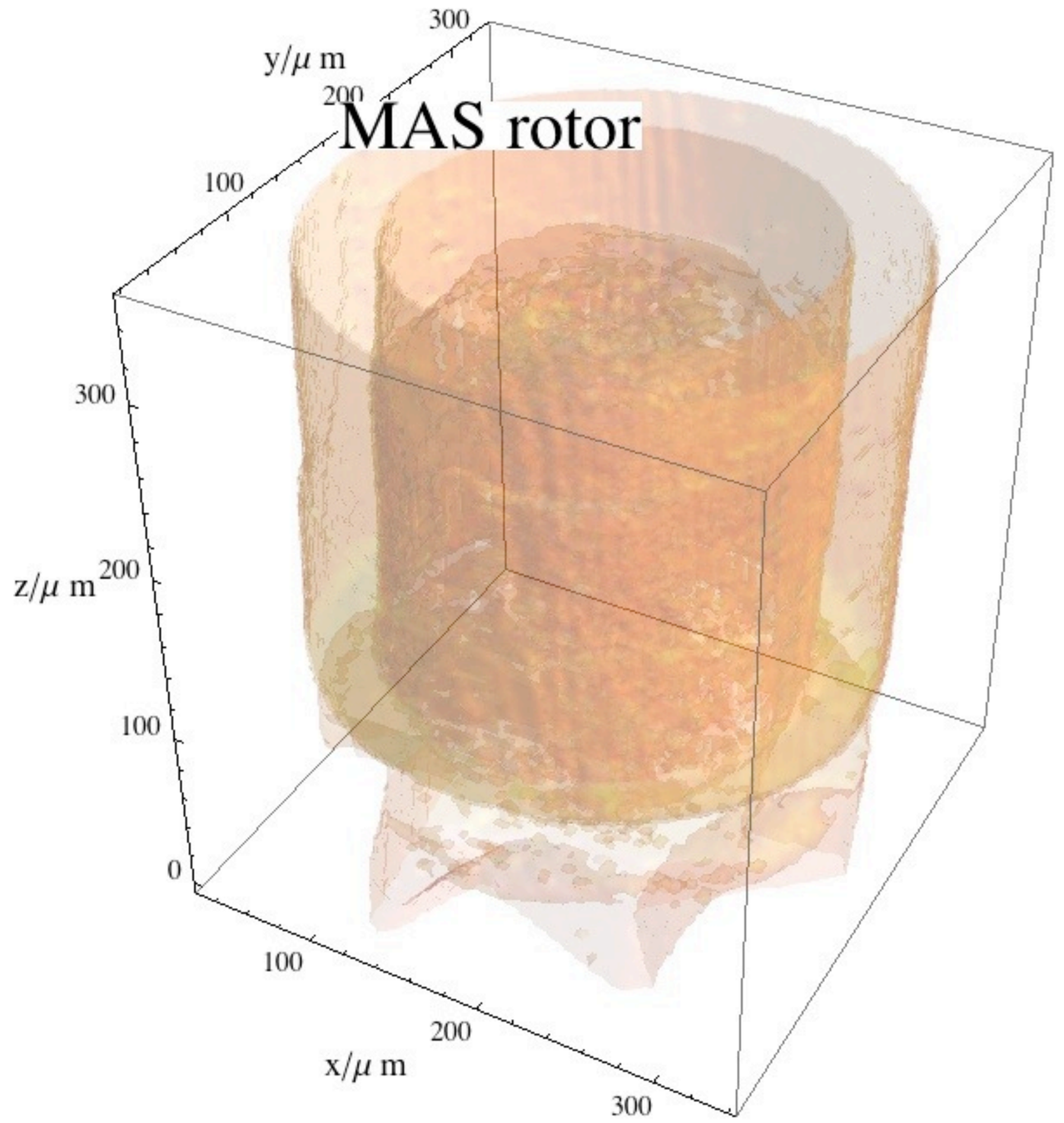
Philosophy

Our *a priori* knowledge about a sample can be used in segmentation.

Two If statements for defining boundaries of only Kel-F and only zirconia.

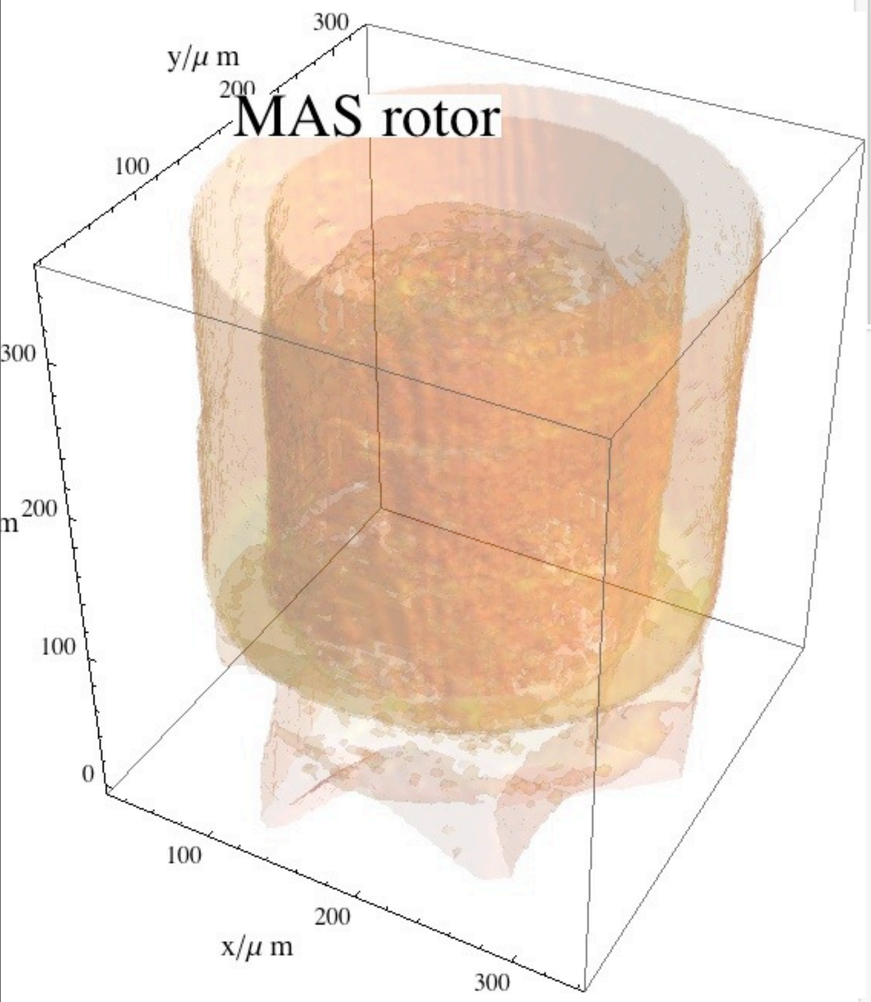
Human eye used to find best values of binarization threshold.

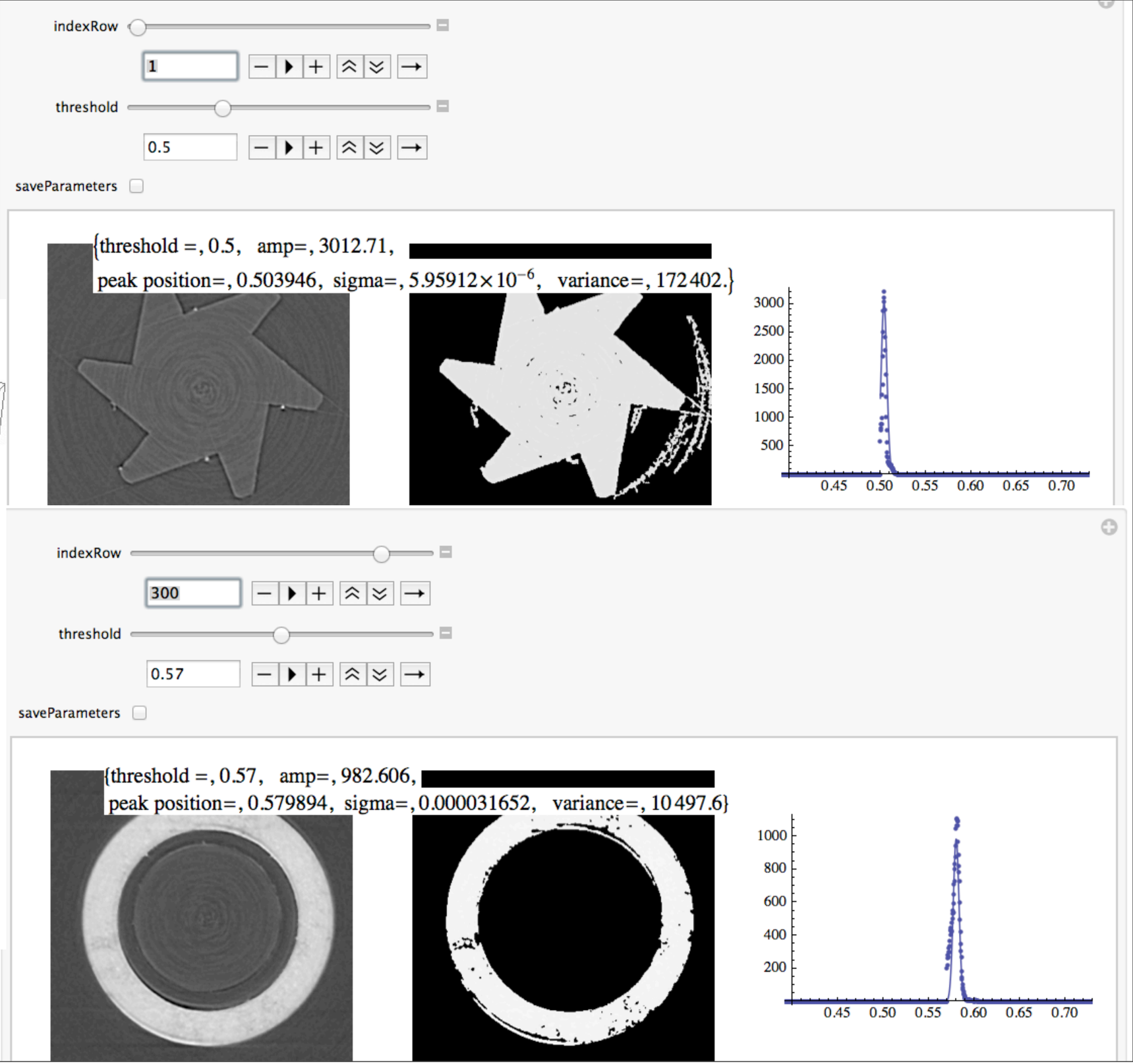
Subtract zirconia binary image from original image to get a Kel-F only image.



Philosophy

Our *a priori* knowledge about a sample can be used in segmentation.





Philosophy

Our *a priori* knowledge about a sample can be used in segmentation.

1) Run Part 4, Step 2 and use the human eye to get really good values for threshold, and the gaussian peak parameters for a few slices in the volume.

2) Manually expand the list of good values. In this case, there are 351 rows. Copy closest parameters into rows 1 and 351 for both components. This helps the Interpolation command.

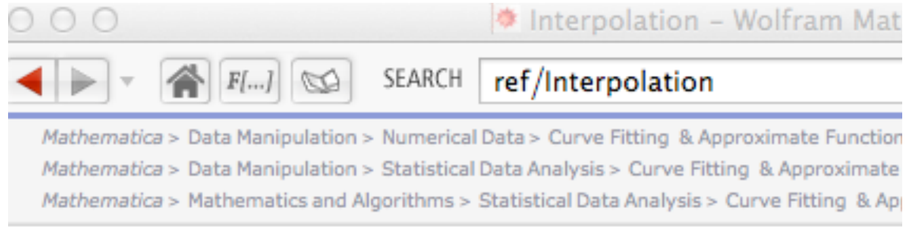
Kel-F

index	threshold	amplitude	position	sigma	variance
1	0.5	3012.71	0.503946	5.95912×10^{-6}	172 402.
25	0.5	3393.23	0.503751	6.56236×10^{-6}	278 923.
50	0.5	3890.22	0.503602	8.4914×10^{-6}	244 004.
75	0.5	5387.17	0.503891	6.60352×10^{-6}	601 834.
100	0.5	5387.17	0.503891	6.60352×10^{-6}	601 834.
351	0.5	5387.17	0.503891	6.60352×10^{-6}	601 834.

Zirconia

index	threshold	amplitude	position	sigma	variance
1	0.57	211.127	0.575044	0.0000539752	385.75
110	0.57	211.127	0.575044	0.0000539752	385.75
150	0.57	1025.38	0.577505	0.0000283423	12 251.6
200	0.57	1024.92	0.57856	0.0000266244	13 466.5
250	0.57	1171.98	0.57971	0.0000188064	7759.85
300	0.57	982.606	0.579894	0.000031652	10 497.6
351	0.57	1129.66	0.581064	0.0000251528	4898.23

3) A cubic-spline interpolation is used to get threshold values and gaussian parameters for any row in between (inclusive) of rows 1 to 351.



index	threshold	amplitude	position	sigma	variance
1	0.57	211.127	0.575044	0.0000539752	385.75
110	0.57	211.127	0.575044	0.0000539752	385.75
150	0.57	1025.38	0.577505	0.0000283423	12 251.6
200	0.57	1024.92	0.57856	0.0000266244	13 466.5
250	0.57	1171.98	0.57971	0.0000188064	7759.85
300	0.57	982.606	0.579894	0.000031652	10 497.6
351	0.57	1129.66	0.581064	0.0000251528	4898.23

```
thresholdCenter = Interpolation[allParametersZirconia[[All, {1, 2}]]][indexRow];
guessA = Interpolation[allParametersZirconia[[All, {1, 3}]]][indexRow];
guessB = Interpolation[allParametersZirconia[[All, {1, 4}]]][indexRow];
guessC = Interpolation[allParametersZirconia[[All, {1, 5}]]][indexRow];
estVariance = Interpolation[allParametersZirconia[[All, {1, 6}]]][indexRow];
```

4) Below is a function to calculate even better threshold values for any given row. It uses the interpolated values of threshold and the gaussian peak parameters as initial guesses.

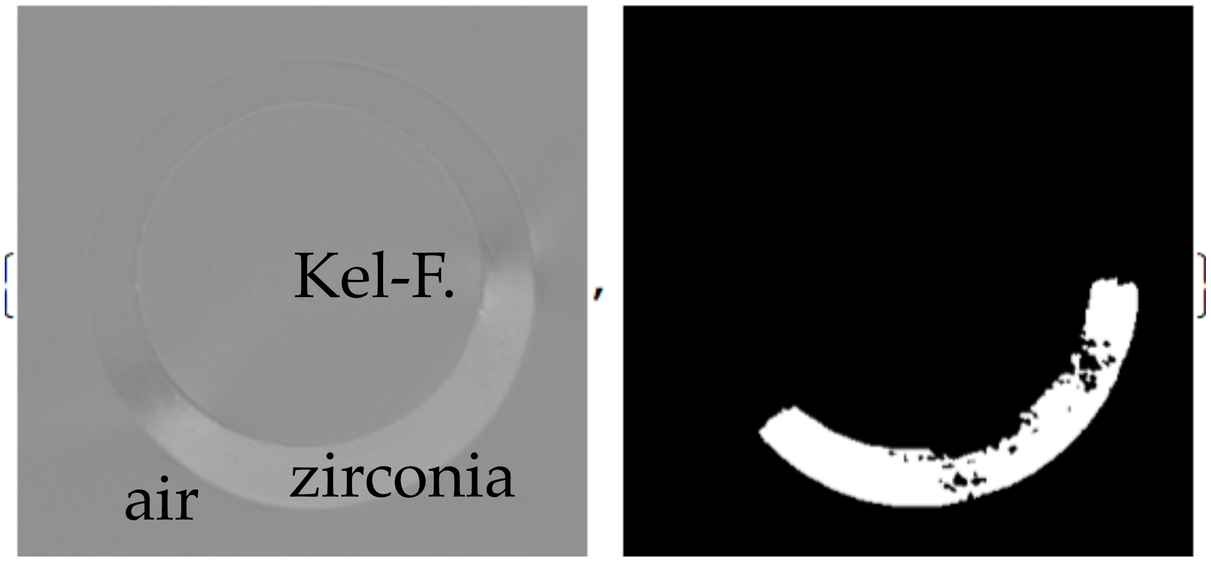
As it turns out, this function is not used.

■ **Step 3: Function to fit zirconia using interpolated starting values for the coefficients.**

```
funcBestThresholdZirconia[indexRow_] := Module[{},
  thresholdCenter = Interpolation[allParametersZirconia[[All, {1, 2}]]][indexRow];
  guessA = Interpolation[allParametersZirconia[[All, {1, 3}]]][indexRow];
  guessB = Interpolation[allParametersZirconia[[All, {1, 4}]]][indexRow];
  guessC = Interpolation[allParametersZirconia[[All, {1, 5}]]][indexRow];
  estVariance = Interpolation[allParametersZirconia[[All, {1, 6}]]][indexRow];
  allParameters = {};
  slice = volMAS[[indexRow, All, All]];
  image = Image[slice, "Bit16"];
  imageTV = TotalVariationFilter[GeometricMeanFilter[image, 1], 0.35, Method → "Laplacian"];
  Do[Module[{},
    imageHighBinary = MorphologicalBinarize[imageTV, {0.0, 0.02} + threshold];
    imageHigh = ImageMultiply[imageTV, imageHighBinary];
    If[Max[ImageData[imageHighBinary, "Bit"]] == 1, Module[{},
      temp = Rest[ImageLevels[imageHigh, 100, {-0.1, 0.1} + threshold]];
      (* g1=ListPlot[temp,PlotRange→{{minVolMAS,maxVolMAS},All}]; *)
      index = Flatten[Position[temp[[All, 2]], p_? (# > 0 &)]];
      tempFit = temp[[index, All]];
      If[Length[tempFit] > 5, Module[{},
        indexTempFit = First[Reverse[Ordering[tempFit[[All, 2]]]]];
        indexTempFit = Reverse[Ordering[tempFit[[All, 2]]]];
```

5) The MAS rotor has some slices which have both zirconia and Kel-F. The bright zirconia is easy to segment with interpolated threshold values.

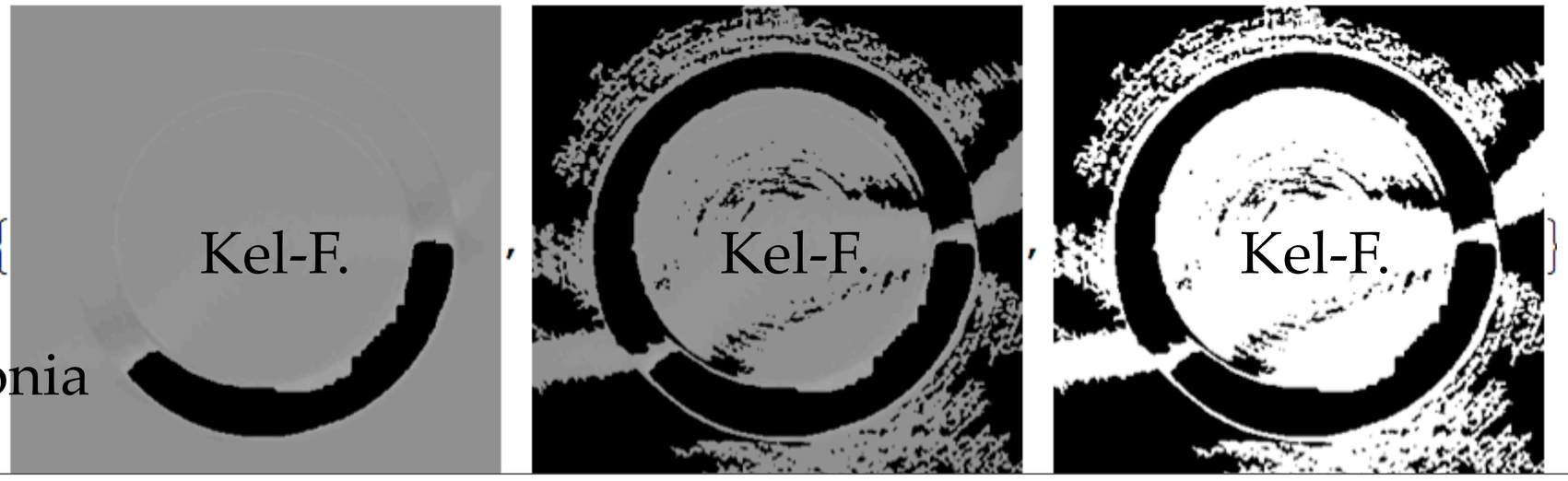
row=110



zirconia binary

6) Then, subtract the binary version of the zirconia from the original image. Does the Kel-F show up now?

no zirconia



7) The MAS rotor has some slices which have both zirconia and Kel-F. The bright zirconia is easy to segment with interpolated threshold values. row=110

Kel-F binary | after some erosion | after closing | label field

8) It takes a little work to find the right label field. It is usually the largest and it is centered in the middle of the image.

```
kelFAreas = ComponentMeasurements[componentsKelF, "Area"];
kelFCentroids = ComponentMeasurements[componentsKelF, "Centroid",
  (Abs[#[[1]] - columns / 2] < 50  && Abs[#[[2]] - slices / 2] < 50  &)];
indexKelFAreas = Reverse[Ordering[kelFAreas[[All, 2]]]];
indexKelFCentroids = kelFCentroids[[All, 1]];
indexBestKelFComponent = First[Intersection[indexKelFAreas, indexKelFCentroids]];
labelBestKelFComponent  =  kelFAreas[[indexBestKelFComponent, 1]];
```

Philosophy

Our *a priori* knowledge about a sample can be used in segmentation.

ImageJ and Manipulate are used to find the rows of Kel-F and zirconia. That row information is used in the If statements.

```
Timing[
 Do[Module[{},
   funcMakeZirconiaAndKelFImages[indexRow, columns, slices];

   If[indexRow ≥ 106, Module[{},
     componentsZirconia = MorphologicalComponents[Closing[imageZirconiaBinary, 5]];
     zirconiaAreas = ComponentMeasurements[componentsZirconia, "Area"];
     indexBestZirconia = First[Reverse[Ordering[zirconiaAreas[[All, 2]]]]];
     labelBestZirconiaComponent = zirconiaAreas[[indexBestZirconia, 1]];
     coordinatesZirconiaPixels = Position[componentsZirconia, p_?(# == labelBestZirconiaComponent &)]
     Do[Module[{x, y},
       {x, y} = coordinatesZirconiaPixels[[i, All]];
       segmentedSlice[[x, y]] = 2; ], {i, Length[coordinatesZirconiaPixels]}];
    ]];

   If[indexRow ≤ 318, Module[{},
     componentsKelF = MorphologicalComponents[imageKelFBinary];
     kelFAreas = ComponentMeasurements[componentsKelF, "Area"];
     kelFCentroids = ComponentMeasurements[componentsKelF, "Centroid",
       (Abs[#[[1]] - columns / 2] < 50 && Abs[#[[2]] - slices / 2] < 50 &)];
     indexKelFAreas = Reverse[Ordering[kelFAreas[[All, 2]]]];
     indexKelFCentroids = kelFCentroids[[All, 1]];
     indexBestKelFComponent = First[Intersection[indexKelFAreas, indexKelFCentroids]];
     labelBestKelFComponent = kelFAreas[[indexBestKelFComponent, 1]];
     coordinatesKelFPixels = Position[componentsKelF, p_?(# == labelBestKelFComponent &)];
     Do[Module[{x, y},
       {x, y} = coordinatesKelFPixels[[i, All]];
       segmentedSlice[[x, y]] = 1; ], {i, Length[coordinatesKelFPixels]}];
    ]];

   segmentedVolume[[indexRow, All, All]] = segmentedSlice;]
  , {indexRow, 1, rows}]
]
```