In-Class Lab Activity: Image Segmentation using ITK

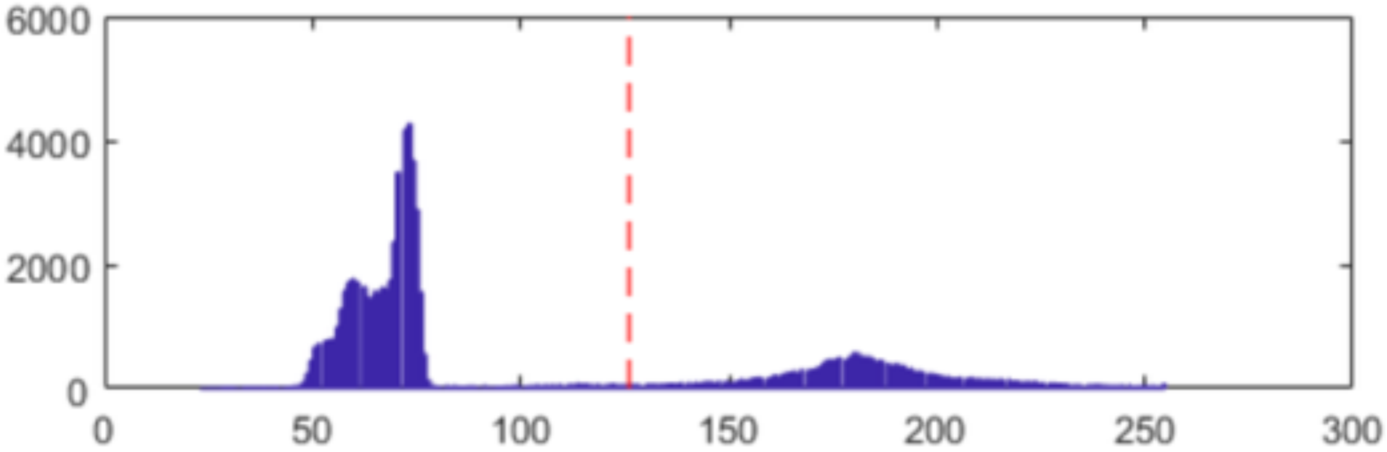
Andrew Hahn (adhahn@wisc.edu)

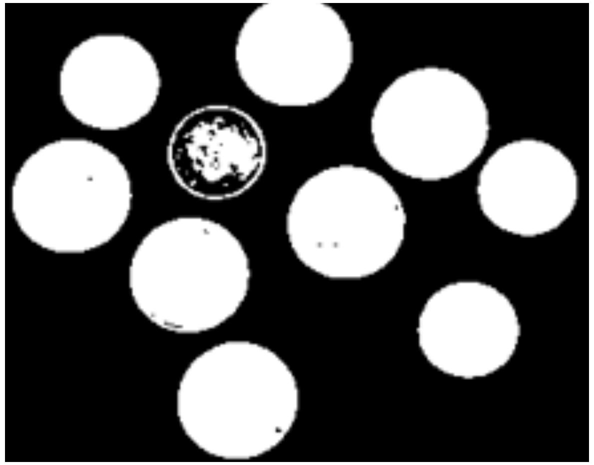
Otsu Thresholding (Reminder)

- Otsu's threshold determines threshold by
  - Minimizing intra-class variance
  - Maximizing inter-class variance

  equivalent

  - $\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$
  - ω_0, ω_1 probabilities of class 1 and 2 respectively





ITK Implementation

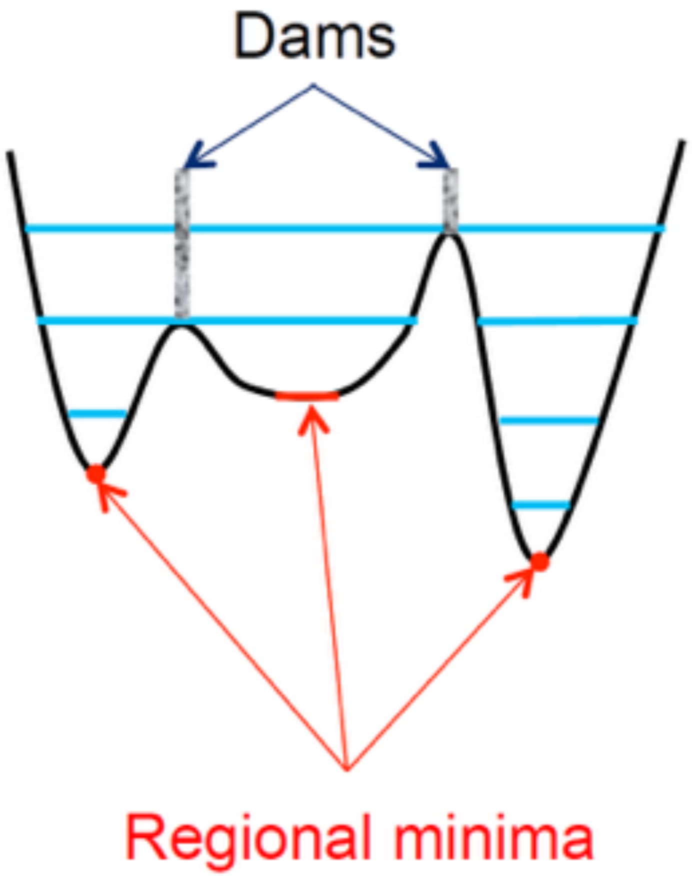
- *Itk::OtsuThresholdImageFilter<InputImageType, OutputImageType>*
  - *SetInput(ImageType img)*
  - *SetInsideValue(T value)*
    - *This value is used in the output image for all pixels lower or equal to the calculated threshold*
  - *SetOutsideValue(T value)*
    - *This value is used in the output image for all pixels larger than the calculated threshold*

Exercise 5
[02_OtsuThreshold]

1. Read a DICOM image
   - You can use the lungImage2.dcm from Exercise 1
2. Perform Otsu’s method to binarize the image
3. Display the binarized image using VTK

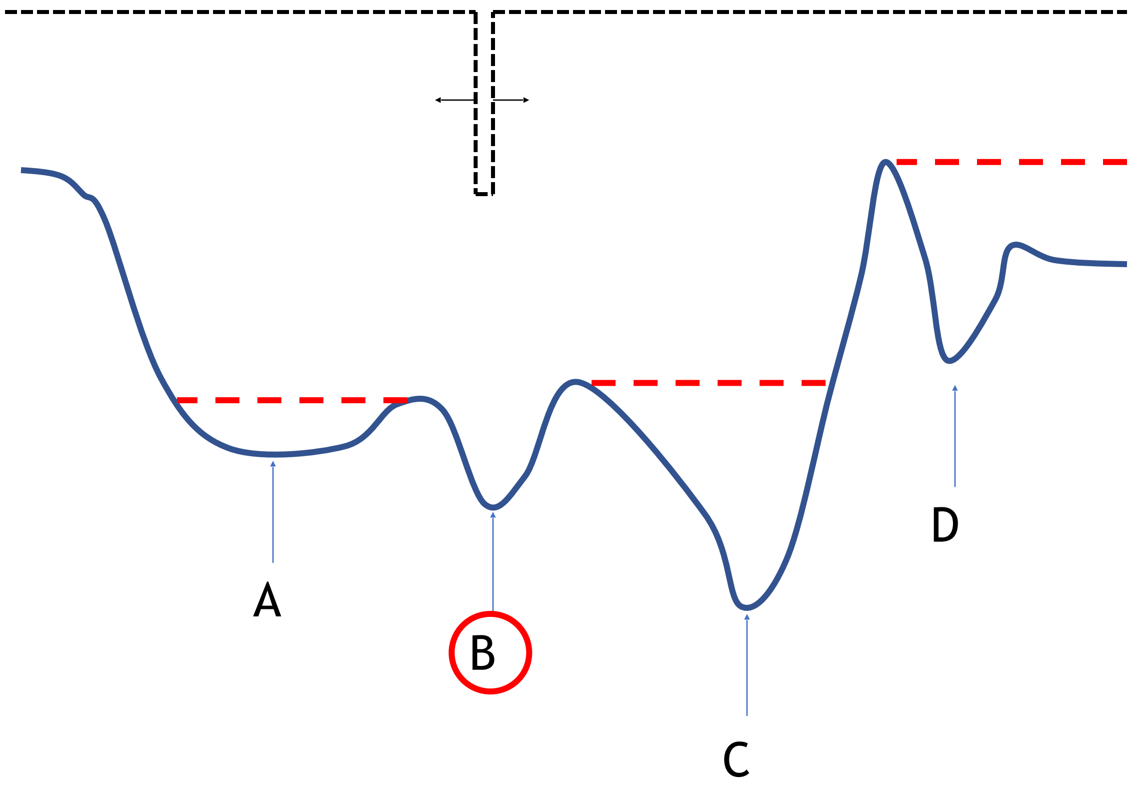
Watershed Segmentation

- Suppose that a hole is punched in each **regional minimum** and that the entire topography is flooded from below by letting **water rise** through the holes at a uniform rate
- When rising water in distinct catchment basins is about to **merge**, a **dam** is built to prevent merging. These dam boundaries correspond to the watershed lines.
- This method is often applied on **gradient** images or **distance transforms** of the original image.



Variant: Morphological Watershed

- Performs flooding operation from selected minima only
  - Mask image may be manually selected or created by preprocessing steps
- Minima are propagated through the image using morphological operations (erosion)
- Steps over unwanted minima



itk::MorphologicalWatershedImageFilter

- SetInput(InputImageType)
  - Sets the image to segment (usually a preprocessed version of the original image)
- SetMarkWatershedLine(bool)
  - Selects between two different algorithms to perform the watershed segmentation [false]
- SetLevel(double)
  - Initial flood level: avoids small regions when in noisy images with many local minima [1.0]

Preprocessing

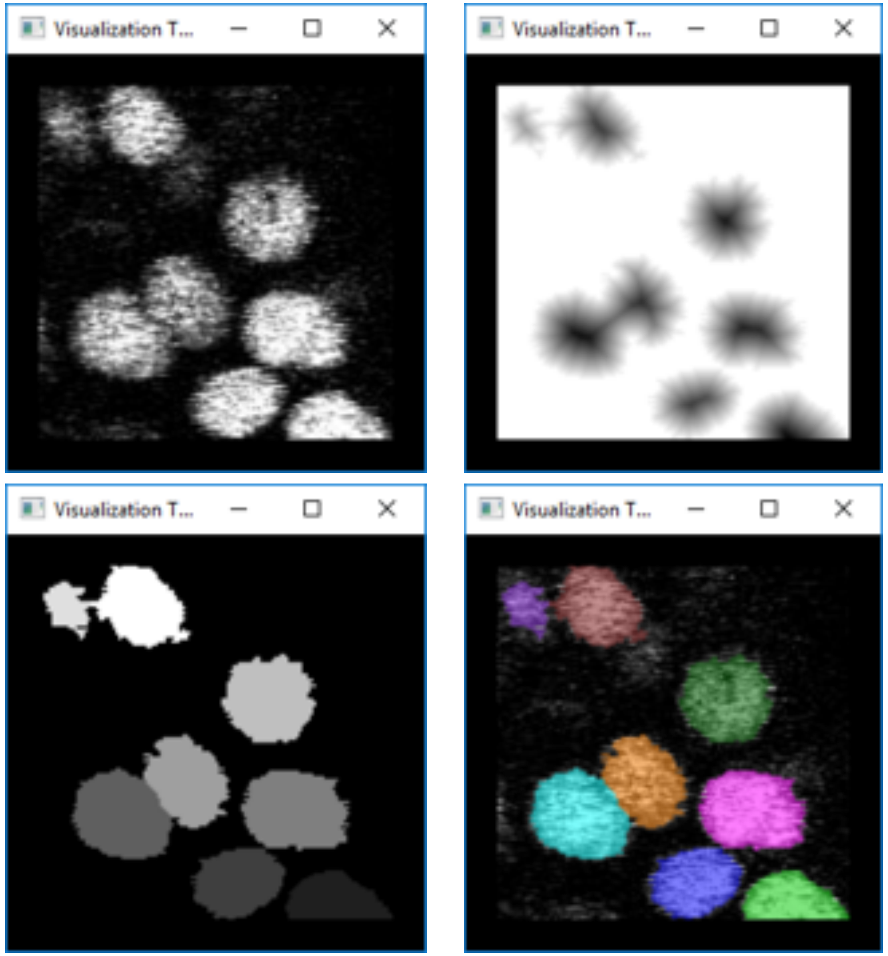
1. Noise reduction using Median filter: itk::MedianImageFilter
2. Binarize the image using: itk::BinaryThresholdImageFilter [lower threshold = 55]
3. Remove small regions: itk::OpeningByReconstructionImageFilter with itk::BinaryBallStructuringElement< bool, 2 > [radius: 5]
4. Fill holes: itk::GrayscaleFillholeImageFilter
5. Calculate inverse distance map
    1. Invert Image: itk::InvertIntensityImageFilter
    2. Distance map: itk::DanielssonDistanceMapImageFilter
    3. Invert distance map: itk::InvertIntensityImageFilter [Maximum: 255]

Postprocessing

1. Multiply segmented image with binary image (after fill holes preprocessing step): itk::MaskImageFilter
   - SetInput(0, SegmentedImage)
   - SetMaskImage(BinaryMaskImage)
2. Overlay colored regions: itk::LabelOverlayImageFilter
   - SetInput(OriginalInputImage)
   - SetLabelImage(MaskedSegmentationImage)
3. Display RGB Image

Exercise 6
[04_MorphologicalWatershed]

1. Read a PNG image
   - cells2.png
2. Perform described preprocessing steps to create a distance map
3. Segment the cells using the morphological watershed technique
4. Display a colored overlay of the original images with the labeled regions



ACM: Snake-Model Energy Term

$$E^{*}_{snake} = \int \left(E_{internal}\big(v(s)\big) + E_{image}\big(v(s)\big) + E_{con}\big(v(s)\big) \right) ds$$

$$E_{internal} = \frac{1}{2}\left(\alpha(s)|v_s(s)|^2\right) + \frac{1}{2}\left(\beta(s)|v_{ss}(s)|^2\right)$$

$$E_{image} = w_{line}E_{line} + w_{edge}E_{edge} + w_{term}E_{term}$$

- Internal energy continuity and smoothness of contour
- External / image energy contains
  - Line energy: pulls contour towards bright or dark lines
  - Edge energy: pulls contour towards strong gradients
  - Termination energy: pulls contour towards corners
- Constraint energy: used to include interactive user input

Level Sets

- Different way of representing object boundaries in discrete images
- Curve is represented by zero crossing
- Adds an additional time dimension
- Signed distance map to boundary

7	6	5	4	4	4	3	2	1	1	1	2	3	4	5
6	5	4	3	3	3	2	1	0	0	0	1	2	3	4
5	4	3	2	2	2	1	0	-1	-1	-1	0	1	2	3
4	3	2	1	1	1	0	-1	-2	-2	-2	-1	0	1	2
3	2	1	0	0	0	-1	-2	-3	-3	-2	-1	0	1	2
2	1	0	-1	-1	-1	-2	-3	-3	-2	-1	0	1	2	3
2	1	0	-1	-2	-2	-3	-3	-2	-1	0	1	2	3	4
2	1	0	-1	-2	-2	-2	-2	-1	0	1	2	3	4	5
3	2	1	0	-1	-1	-1	-1	-1	0	1	2	3	4	5
4	3	2	1	0	0	0	0	-1	-1	0	1	2	3	4
5	4	3	2	1	1	1	1	0	0	1	2	3	4	5
6	5	4	3	2	2	2	2	1	1	2	3	4	5	6

Geodesic Active Contours

- Combines Snake-Model (Kass et al.) with Level-Set approach (Osher and Sethian)
- Contours can split and merge
- Contour moves perpendicular to local curve
- Speed of evolution depends on local curvature

ITK Geodesic Active Contour Classes

- itk::GeodesicActiveContourLevelSetImageFilter: Implements the calculation of the energy functional
  - SetPropagationScaling(double): Sets speed of evolution [10.0]
  - SetCurvatureScaling(double): Curvature weight for energy function [1.0]
  - SetAdvectionScaling(double): External energy weight [1.0]
  - SetMaximumRMSError(double): Stopping criterion [0.02]
  - SetNumberOfIterations(int): Stopping criterion [500]
  - SetInput(FastMarchingOutput)
  - SetFeatureImage(ImageType): Preprocessed gradient image
- itk::FastMarchingImageFilter: Implements curve evolution
  - SetTrialPoints(FastMarchingImageFilter::NodeContainer): Sets the seed positions [x=56, y=92]
  - SetSpeedConstant(double): Sets the propagation speed [1.0]
  - SetOutputSize(SizeType): Size of the original image (GetBufferedRegion().GetSize())

Preprocessing

1. Reduce noise using itk::CurvatureAnisotropicDiffusionImageFilter
   - SetTimeStep(double): Speed of diffusion [0.125]
   - SetNumberOfIterations(int): Number of iteratinos [5]
   - SetConductanceParameter(double): Controls gradient preservation [9.0]
   - SetInput(ImageType): Original image
2. Calculate gradient magnitude itk::GradientMagnitudeRecursiveGaussianImageFilter
   - SetSigma(double): Standard deviation of Gaussian [1.0]
   - SetInput(ImageType): Smoothing output
3. Normalize magnitude using itk::SigmoidImageFilter
   - sigmoid->SetOutputMinimum(double): Minimum output value [0.0]
   - sigmoid->SetOutputMaximum(double): Maximum output value [1.0]
   - sigmoid->SetAlpha(double): Slope of sigmoid curve [-0.3]
   - sigmoid->SetBeta(double): Translation of sigmoid curve [2.0]
   - sigmoid->SetInput(ImageType): Gradient magnitude image

Postprocessing

1. Threshold level set using itk::BinaryThresholdImageFilter
   - SetLowerThreshold(double): Lower threshold [-1000.0]
   - SetUpperThreshold(double): Upper threshold [0.0]
   - SetOutsideValue(double): Background value [0]
   - SetInsideValue(double): Object pixel value [255]
   - SetInput(ImageType): Output of Geodesic active contour filter
2. Display segmented binary image using VTK

Exercises 7 & 8
[04_ActiveContour]
[05_ActiveContourWithVTKSeed]

1. Read a PNG image
   - BrainProtonDensitySlice6.png
2. Perform described preprocessing steps to create a normalized gradient image
3. Segment white matter using geodesic active contours
4. Display binary image using VTK

- Exercise 8: Allow user to select seed points using vtkSeedWidget