Automated segmentation of microstructures using the SIDE algorithm

School of Electrical and Computer Engineering

Purdue University

04/08/2011

SIDE for image segmentation

- SIDE (Stabilized Inverse Diffusion Equations) algorithm
  - Evolve feature vectors that are vectorial representation of some specific region over a scale space
  - This evolution forms a region merging process for image segmentation
  - The direction of the evolution is driven by minimizing an energy functional along its gradient

Energy functional

- The energy functional is the objective function for SIDE evolution, trying to minimize:

$$\mathcal{E} = \sum_{\{R_i, R_j\} \in NBR-PRS} b(R_i, R_j) E(\|\mu_i - \mu_j\|) \qquad (1)$$

where

R_i is a region (set of connected pixels) of the image

μ_i is the vector intensity for region R_i

$NBR - PRS$ denotes the set of all pairs of neighbor regions

$b(R_i, R_j)$ is a design parameter

$E(x) = \sqrt{x}$ which denotes a SIDE energy function

PURDUE
ENGINEERING

Velocity of the gradient descent

- The following PDE describes the evolution of the intensity for Region *i*

$$\dot{\mu}_i(t) = \frac{1}{a(R_i)} \sum_{R_j \in NBRS(R_i)} b(R_i, R_j) \cdot \frac{\mu_j - \mu_i}{\left\| \mu_j - \mu_i \right\|} \cdot E'(\left\| \mu_j - \mu_i \right\|) \qquad (2)$$

where

$NBRS(R_i)$ denotes the set of regions that are neighbors of R_i

$a(R_i)$ is a design parameter

PURDUE ENGINEERING

Original SIDE algorithm

1. Initialize all pixels as individual regions, set $t=0$
2. Evolve all feature vectors, until $t = t_{merge}$, according to (2)
3. Merge two neighboring regions if

$$\left\|\mu_i(t) - \mu_j(t)\right\| < \varepsilon \quad \text{for some small } \varepsilon$$

4. If a predetermined number of regions is reached, then stop; otherwise update functions $a(\cdot)$ and $b(\cdot,\cdot)$, increase t_{merge}, and repeat Steps 2-3

Design parameters

- In the original algorithm, design parameters are defined as follows for segmentation of natural images:
  - $a(R_i)$ is some function of the region size $|R_i|$
  - $b(R_i, R_j)$ is the length of boundary between R_i and R_j

PURDUE ENGINEERING

Modifications of the SIDE algorithm

- Intensity penalty
- Boundary length penalty
- Boundary curvature penalty
- Stopping rule

Intensity penalty

- Modify the original energy functional by including an additional term to penalize the evolution of a vector intensity too far from some *desired state:*

$$\mathcal{E} = (1-\lambda) \sum_{\{R_i, R_j\} \in NBR-PRS} b(R_i, R_j) E(\|\mu_i - \mu_j\|) + \lambda \sum_{R_i} G(\|\mu_i^{DES} - \mu_i\|) \qquad (3)$$

where

λ is the weighting factor for the energy function $G(x)$

$G(x) = \frac{1}{2} x^2$ defines the cost of intensity difference x

Intensity penalty (cont'd)

- To find the desired states, we use a scalar feature μ_i and initialize it with the pixel intensity y_i. Then a *clustering* is performed to approximate the distribution of pixel intensity to a Gaussian mixture model:

$$f(y) \approx \sum_{k=1}^{K} ?_k N(m_k, ?_k; y)$$

Intensity penalty (cont'd)

- The desired state is determined by the Mahalanobis distance between the scalar feature and its closest mixture mean:

$$\mu_{\mathrm{i}}^{\mathrm{DES}} \equiv \underset{m_k}{\operatorname{argmin}} \frac{\left|\mu_i - m_k\right|}{\sigma_k}$$

PURDUE
ENGINEERING

Boundary length penalty

- To remedy a segmentation issue with excessively long boundaries from original SIDE algorithm
- Modified function $b(\cdot,\cdot)$ defined as:

$$b(R_i, R_j) = (\text{boundary length})^{\eta}$$

PURDUE
ENGINEERING

Boundary curvature penalty

- Given a continuous curve $c(t)=(x(t),y(t))$, the curvature $\kappa(t)$ can be defined as:

$$\kappa(t) \equiv \frac{\left|x'(t)y''(t) - y'(t)x''(t)\right|}{(x'^2(t) + y'^2(t))^{3/2}} \qquad (8)$$

- The average curvature over a curve of length T can then be defined as:

$$\bar{\kappa} = \frac{1}{T}\int_0^T \kappa(t)dt \qquad (9)$$

PURDUE ENGINEERING

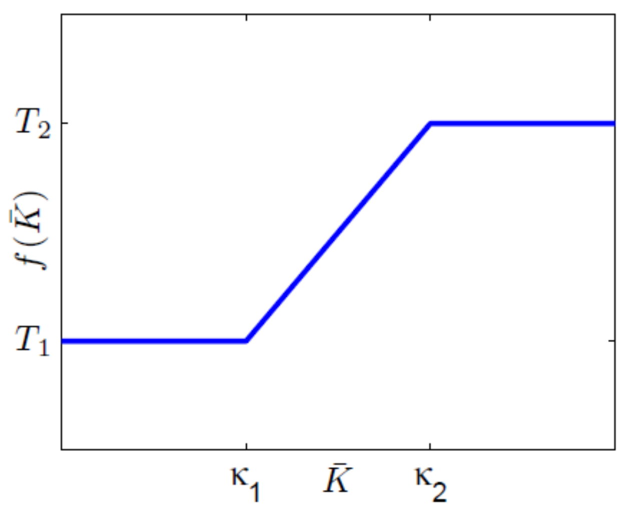
Boundary curvature penalty (cont'd)

- To include the boundary curvature penalty, a realization is to modify the function $b(\cdot,\cdot)$:

$$b(R_i, R_j) = (\text{boundary length})^{\eta} f(\overline{\kappa}) \qquad (10)$$

where

$f(\overline{\kappa})$ is the curvature penalty function, which is a piecewise linear function



Boundary curvature penalty (cont'd)

- Unfortunately, the computation of curvature in discrete domain is ill-posed
  - For example, computed curvature for a 45° straight line may produce multiple nonzero values
- For a discrete pixel on a curve $c[n]$, an efficient and robust way to estimate curvature is to fit a smooth parabola through $c[n-n_0]$, $c[n]$, and $c[n+n_0]$, and compute the curvature of the parabola using (8) *[Hermann06]*
  - $n_0=5$ is empirically found useful for IN-100 dataset

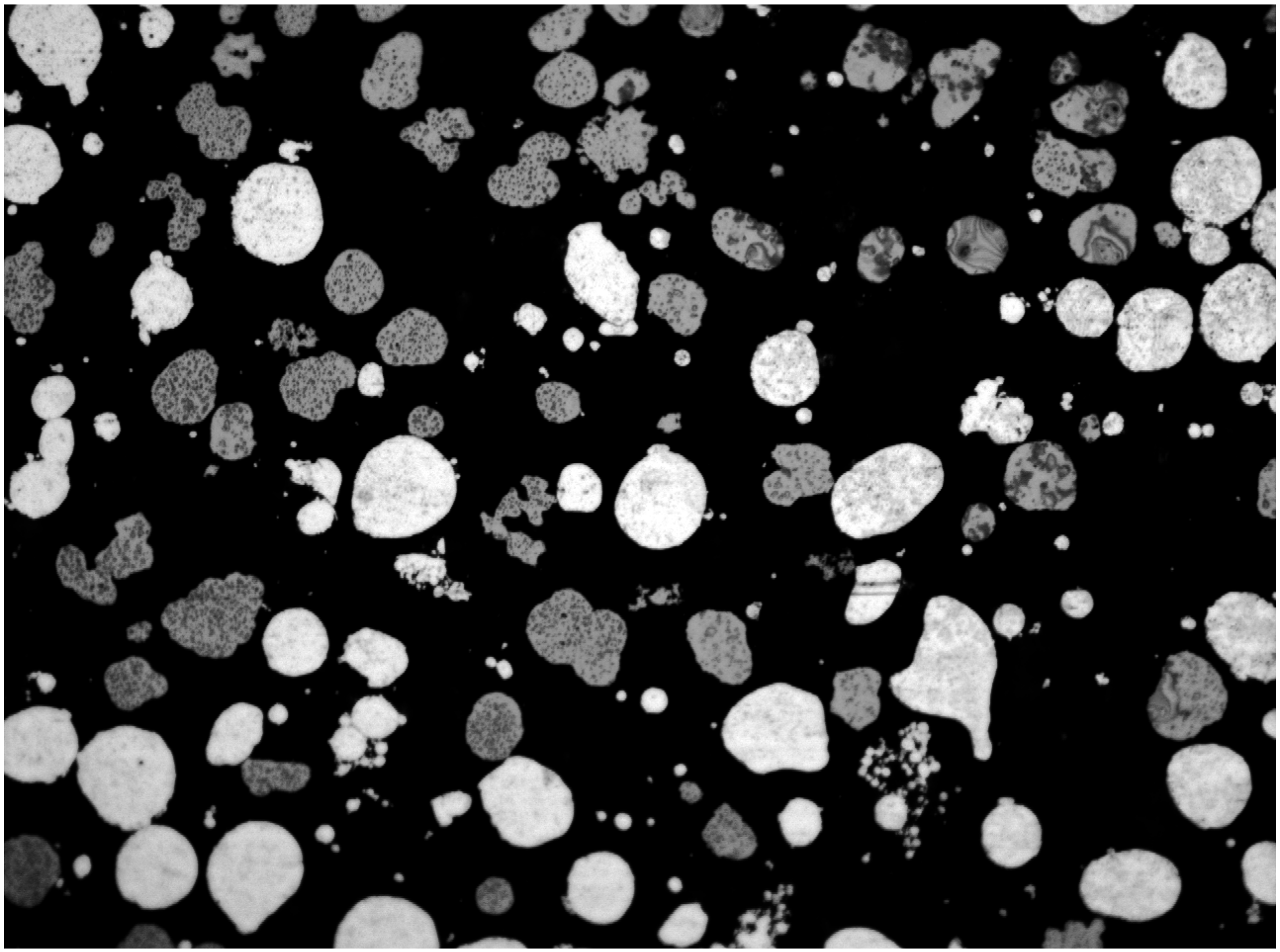
PURDUE
ENGINEERING

Experimental results

- Segmentation of MNML alloy
- Segmentation of IN100 alloy
- Segmentation of Rene88 alloy

PURDUE
ENGINEERING

Segmentation of MNML alloy

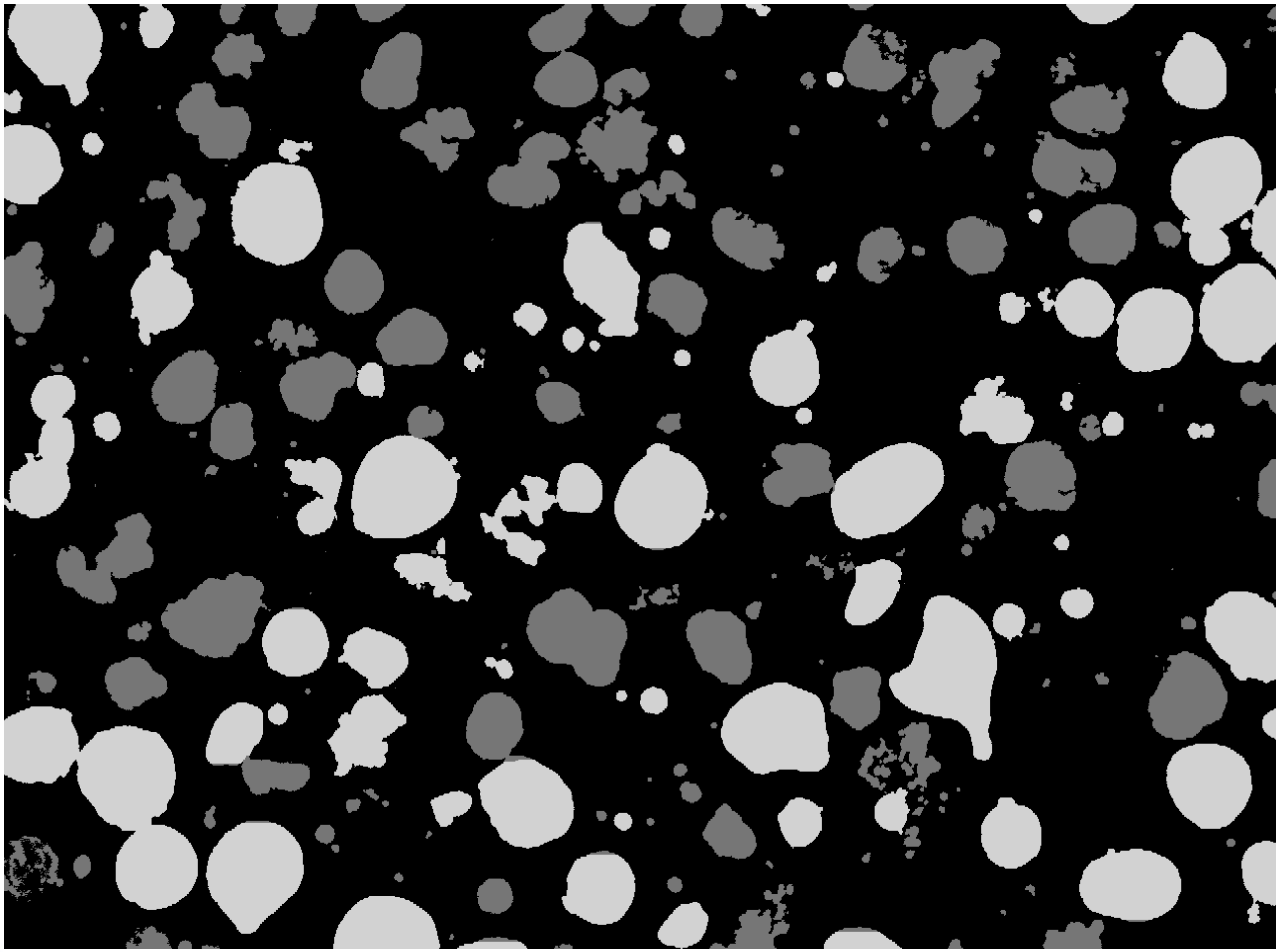


Segmentation procedure

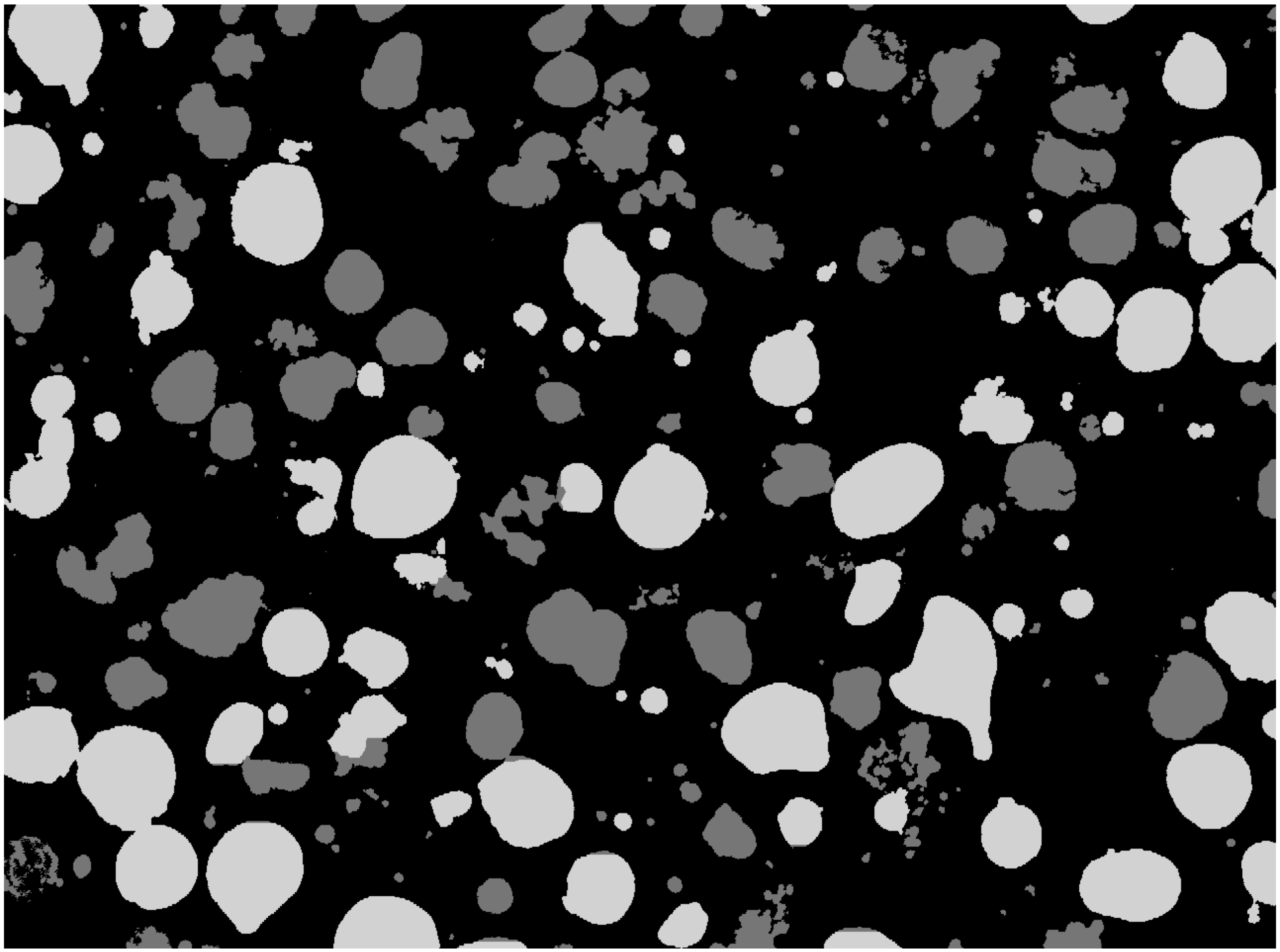
1. Background isolation by fitting a bimodal Gaussian to the intensity histogram; scale the background standard deviation with $\sigma_1 \leftarrow 100\sigma_1$
2. Segment the foreground (two classes) with $\lambda=0.99$ (intensity penalty weight) using a bimodal Gaussian and $\eta=2$ (boundary length penalty exponent)
3. Stopping rule: stop when 500 gradient descent iterations evolve without any regions begin merged

PURDUE
ENGINEERING

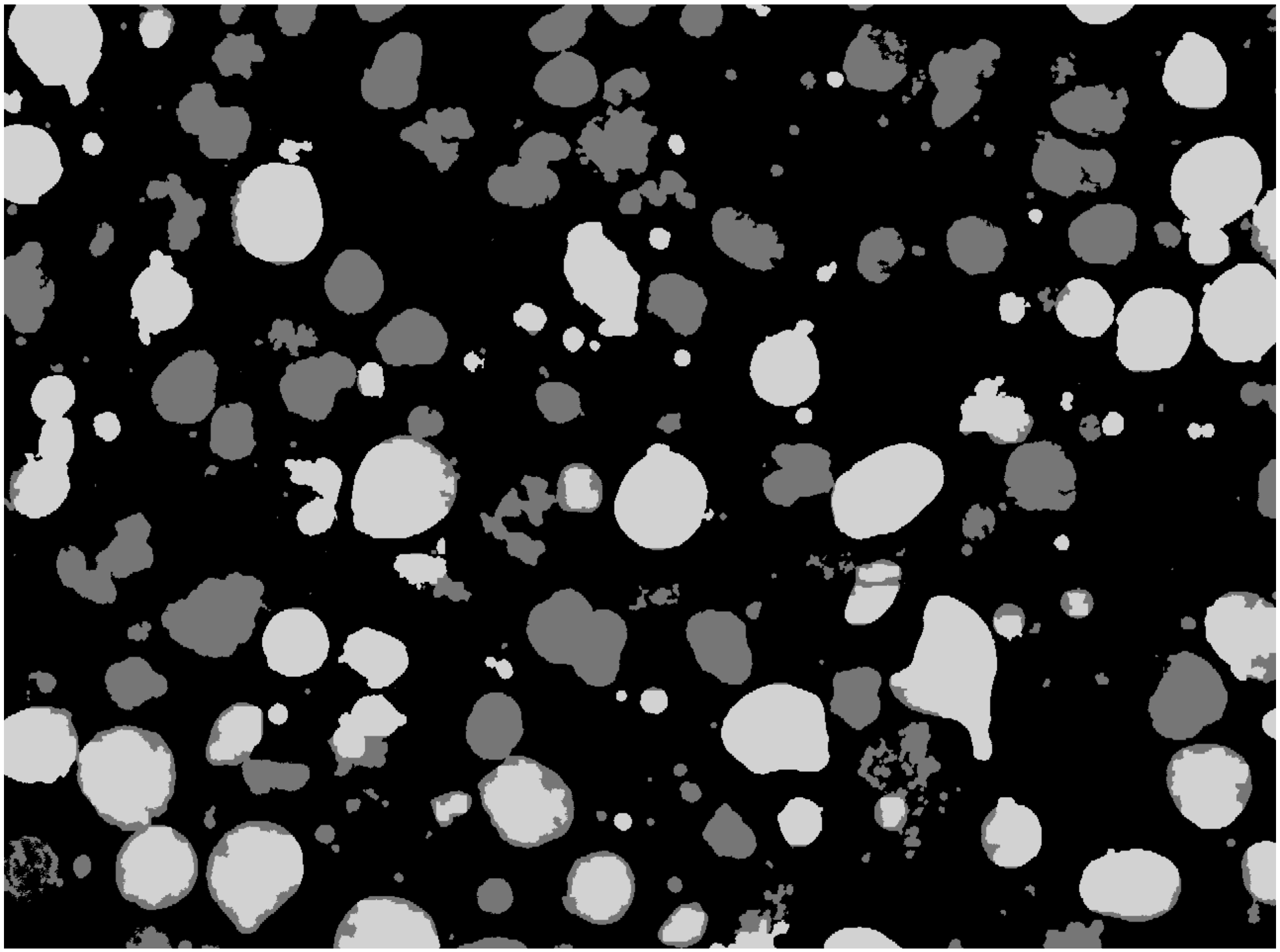
Results—λ=0.5(intensity penalty)



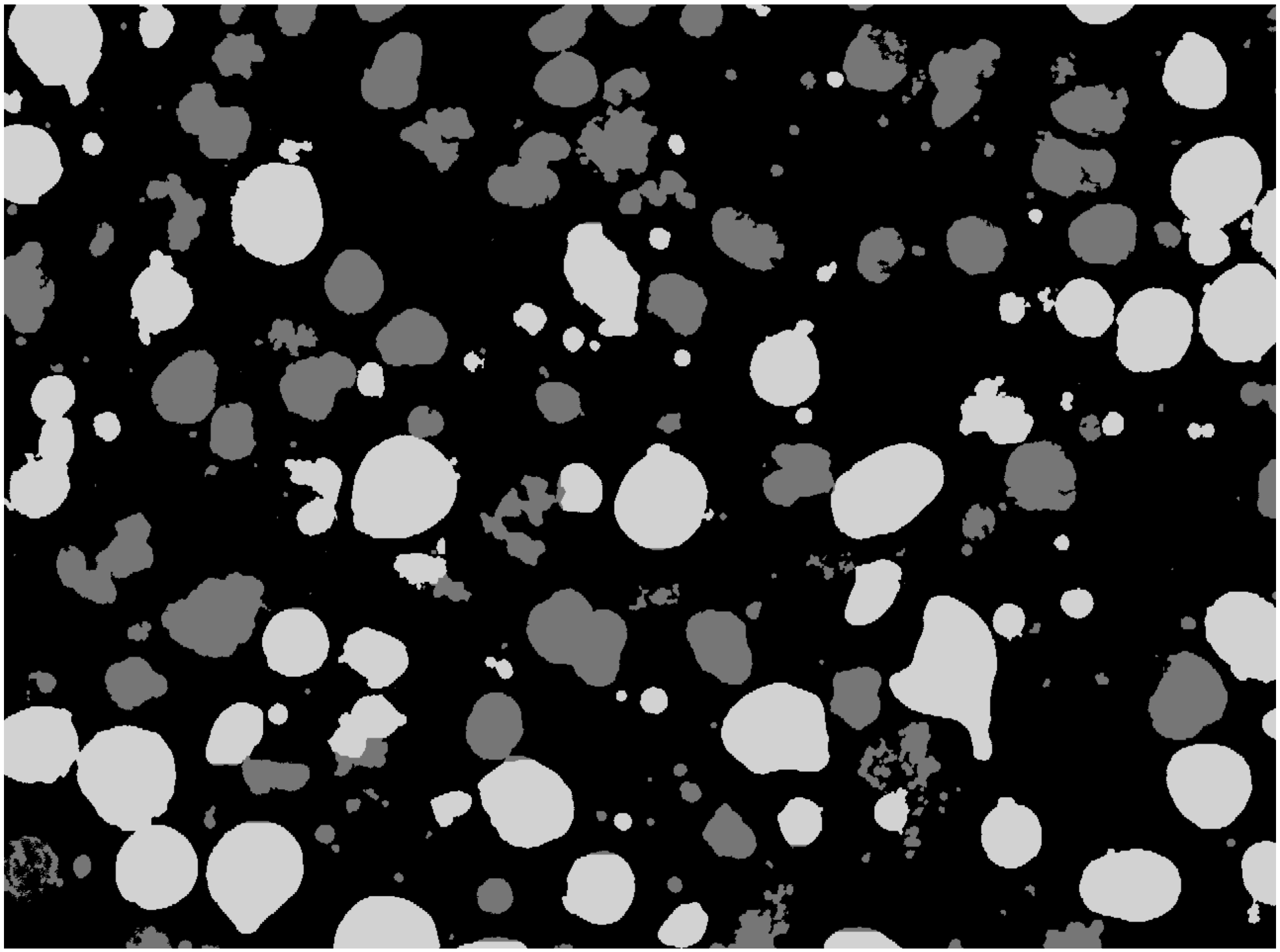
Results—λ=0.99(intensity penalty)



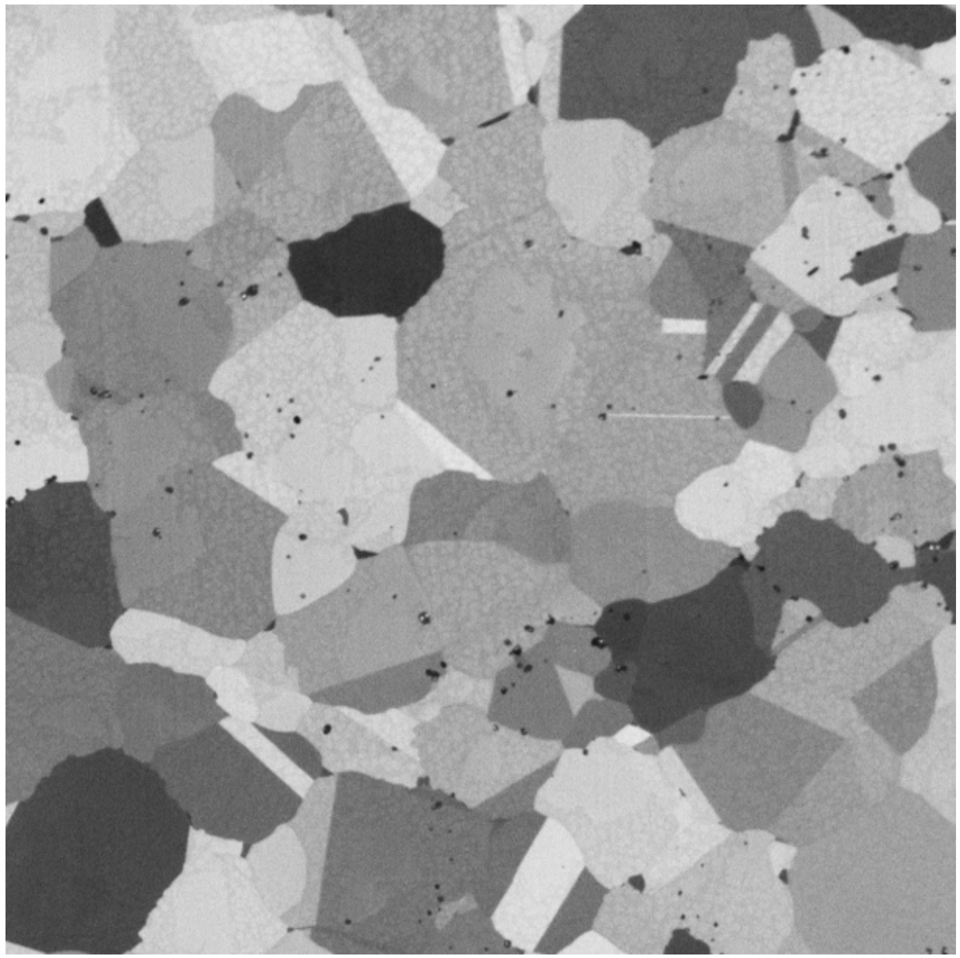
Results—η=1(boundary length penalty)



Results—η=2(boundary length penalty)



Segmentation of IN100 alloy



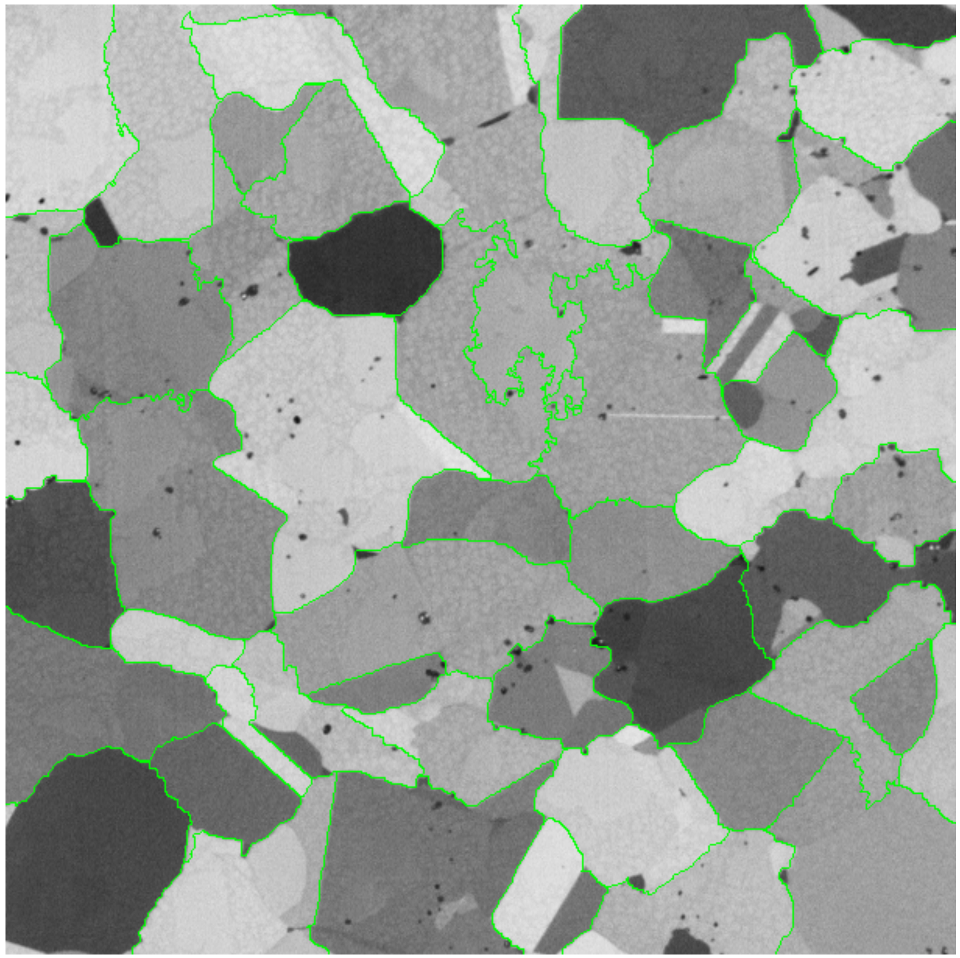
PURDUE ENGINEERING

Segmentation procedure

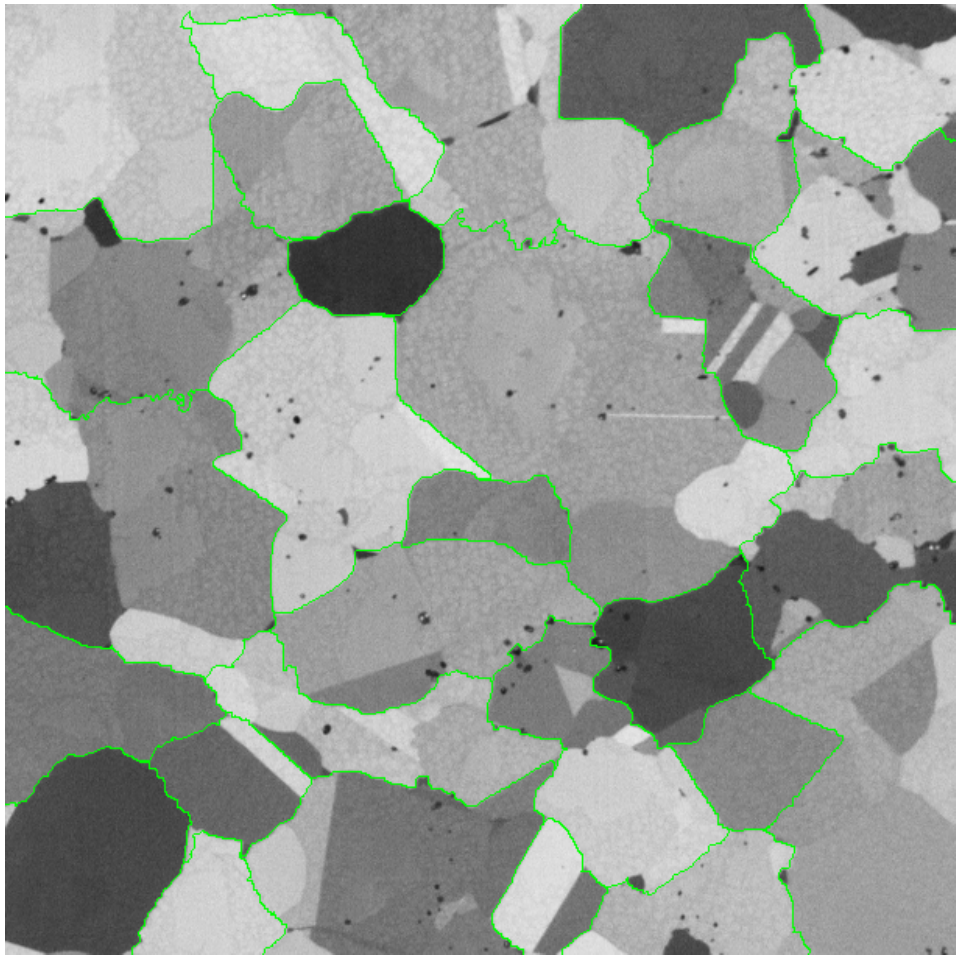
1. Due to the large range of pixel intensity values, λ is set to zero in this segmentation
2. Set η=2, κ_1=κ_2=0.2, and T_1=0.1, T_2=10 for boundary curvature penalty
3. Stopping rule: by observing that the energy functional of this segmentation decreases exponentially, we locate the termination as the point of maximum derivative of energy evolution
   - Not particularly robust and new methods could be investigated

PURDUE
ENGINEERING

Results—η=1(boundary length penalty)

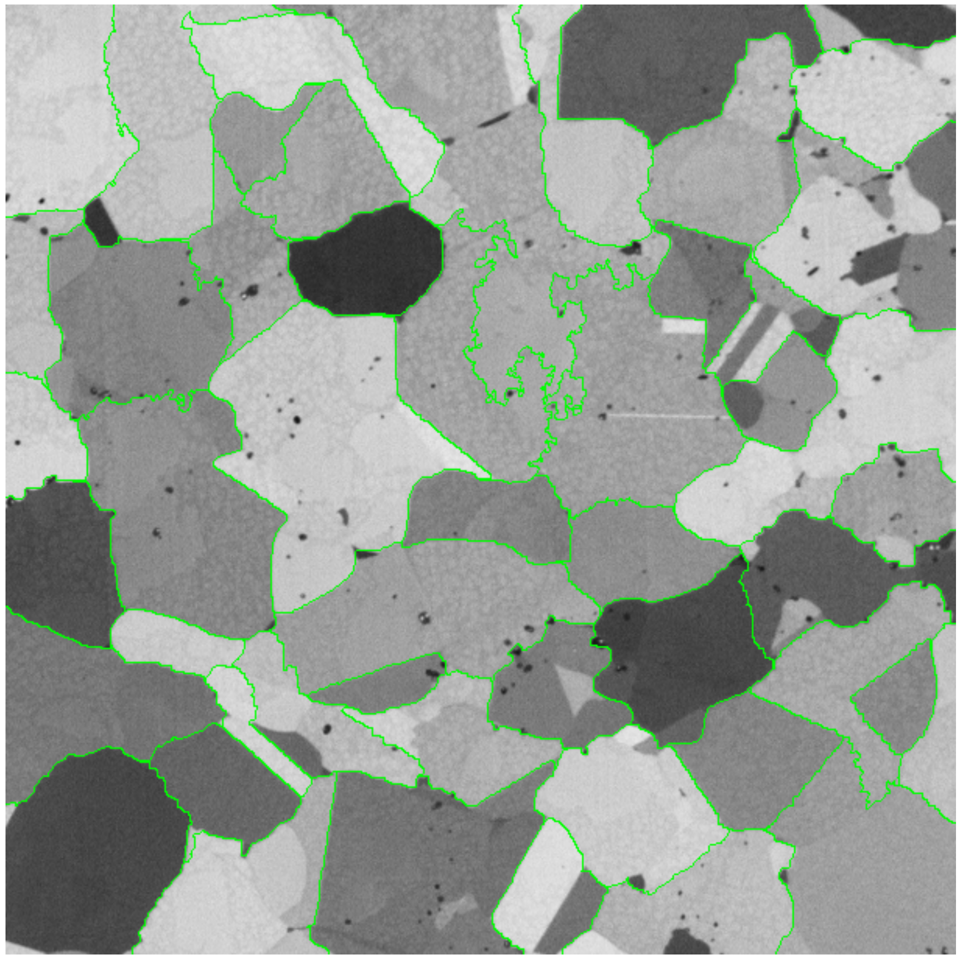


Results—η=2(boundary length penalty)

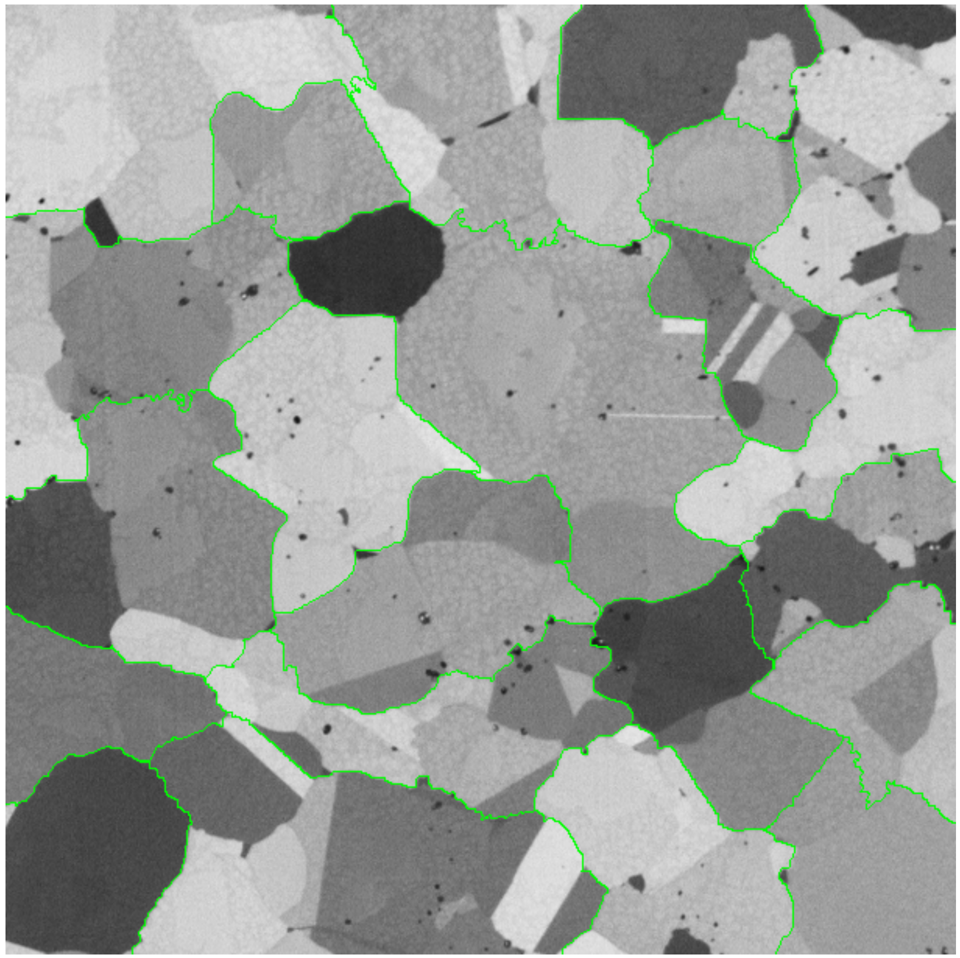


PURDUE
ENGINEERING

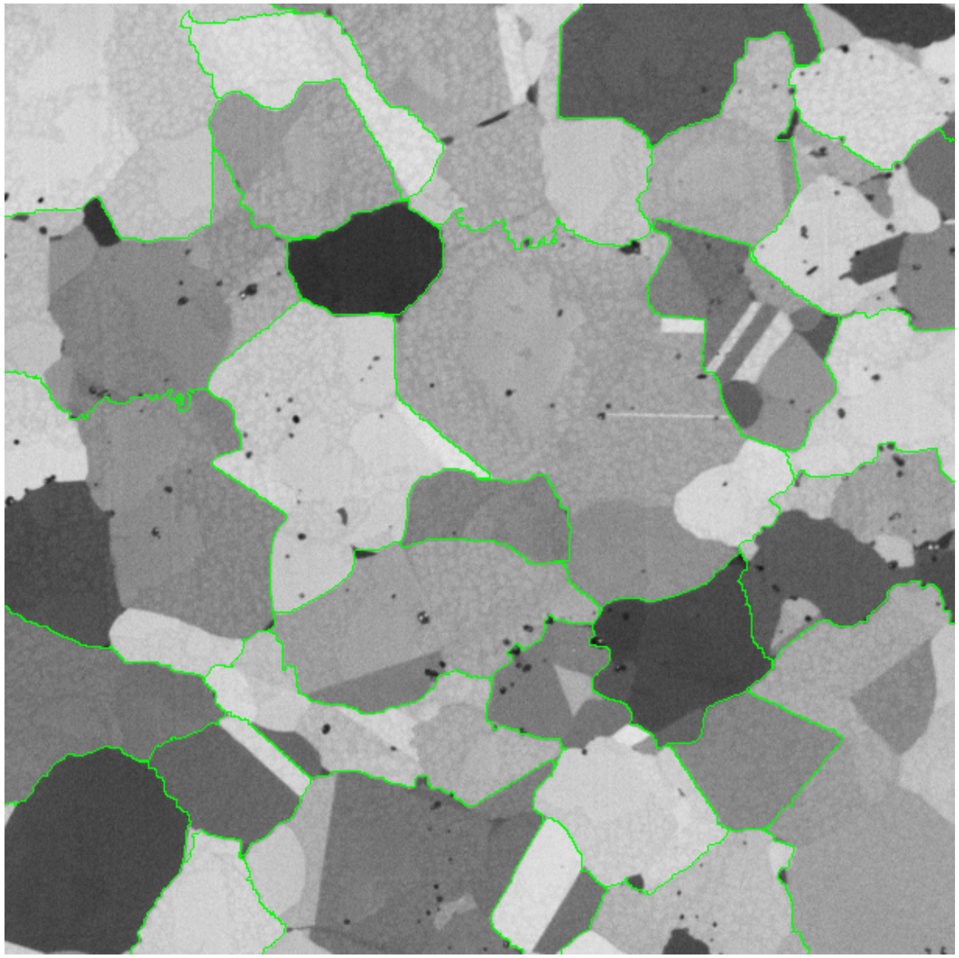
Results—without boundary curvature penalty



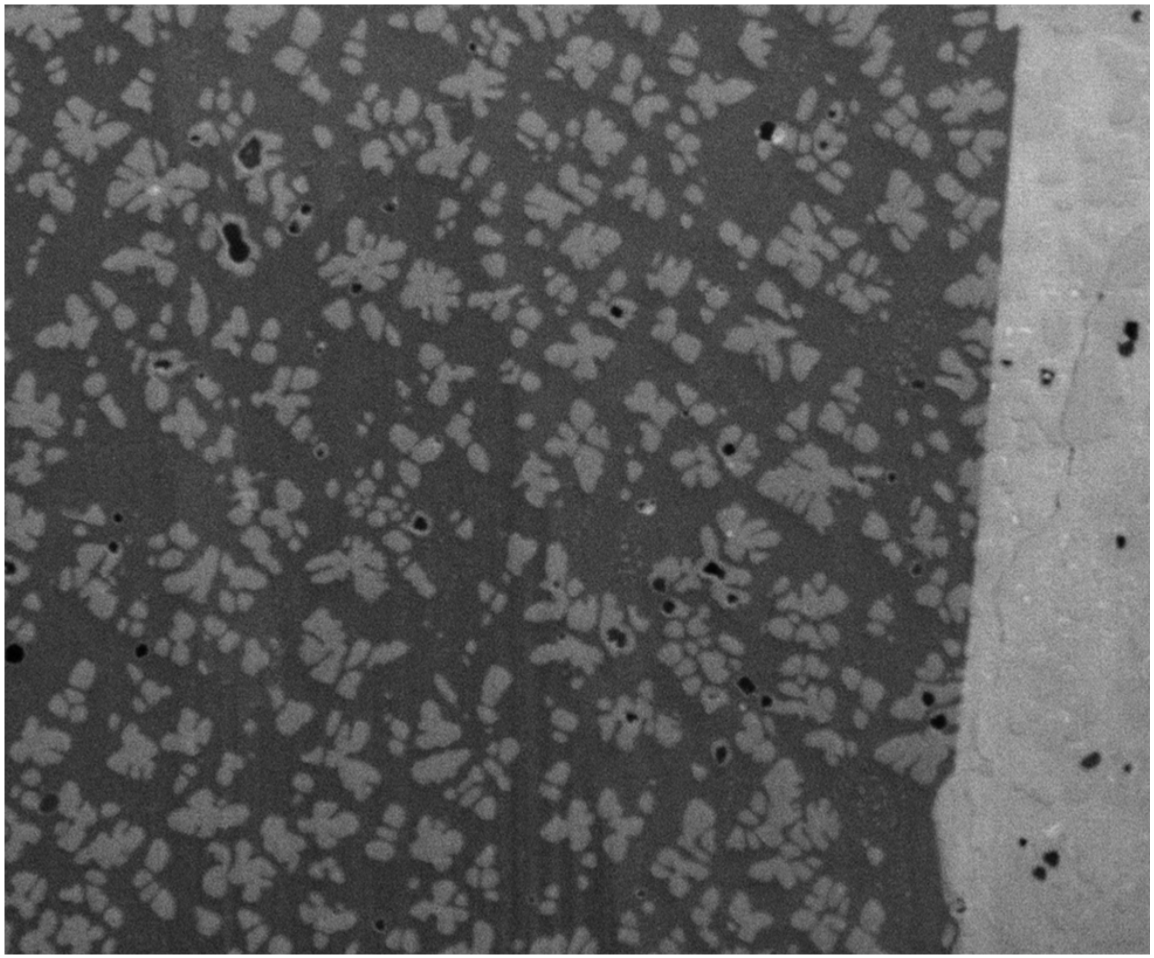
Results—without boundary curvature penalty



Results—with boundary curvature penalty



Segmentation of Rene88 alloy



PURDUE
ENGINEERING

Segmentation procedure

- Two stage segmentation each with specific parameter setting for different features
  - Stage 1: segmentation of large blocks (or large regions)
  - Stage 2: segmentation of particles (or small regions)

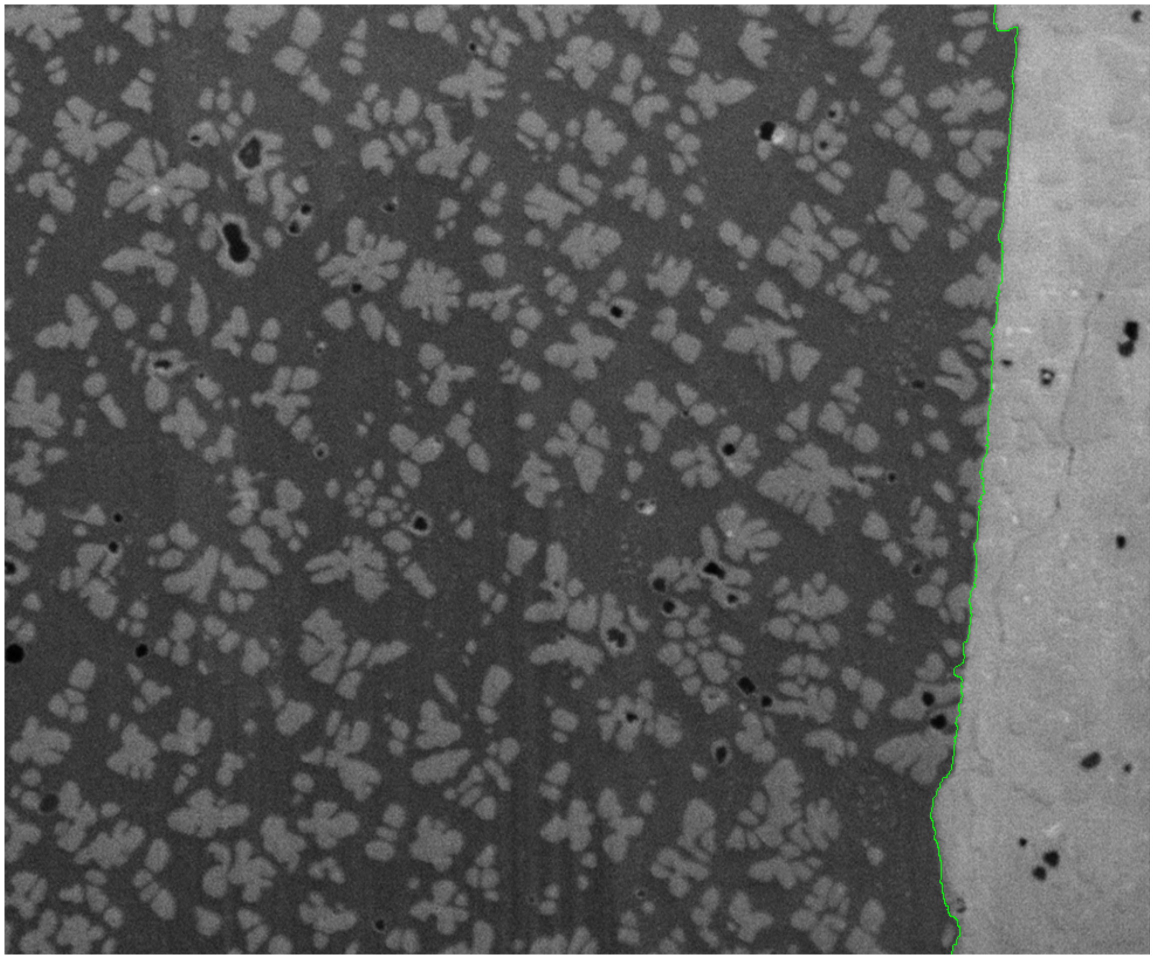
PURDUE
ENGINEERING

Segmentation of large regions

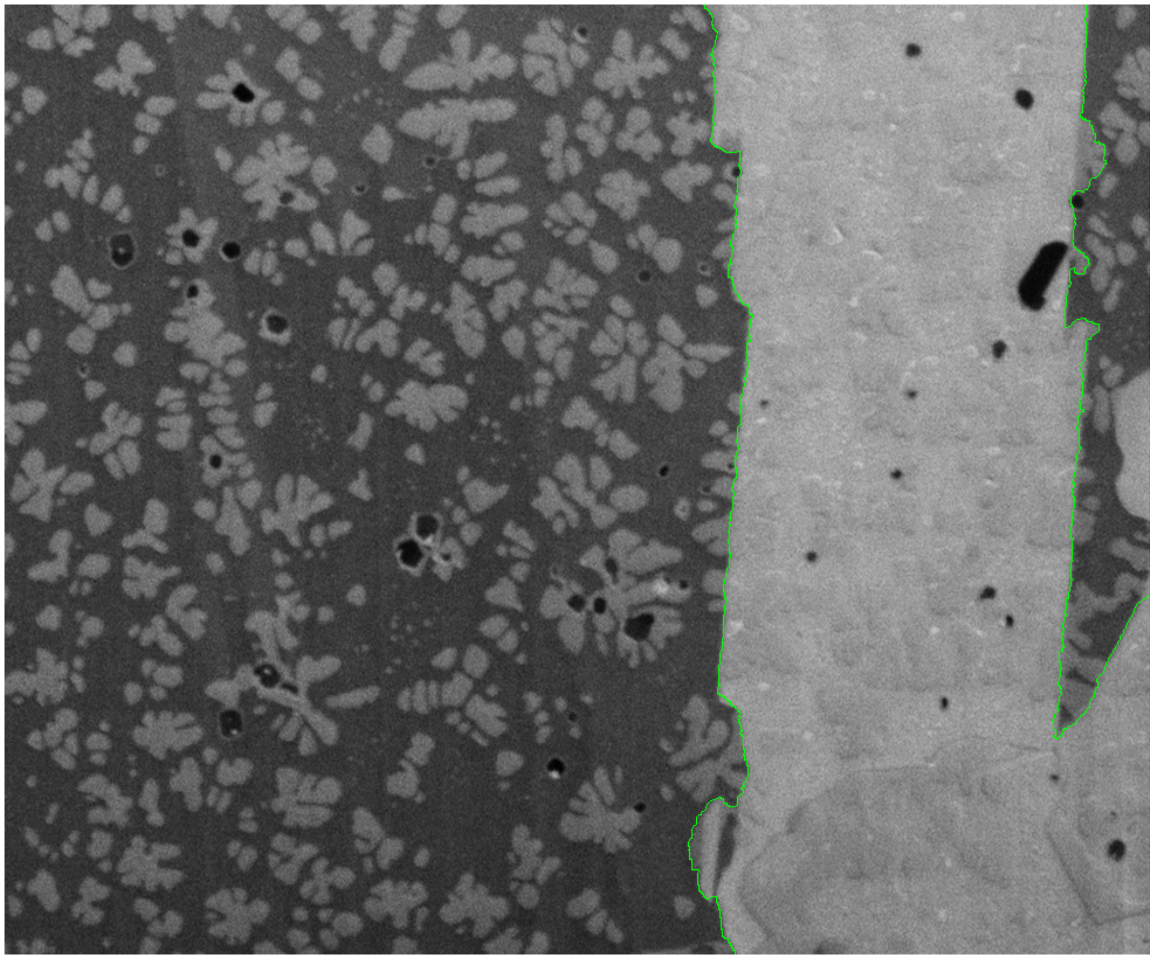
1. Set λ=0.85 and η=1 for intensity and boundary length penalties, respectively
2. Set $\kappa_1=\kappa_2=0.15$, $T_1=0.1$ and $T_2=10$ for boundary curvature penalty
3. Set $a(R_i)=|R_i|$ when region mass $|R_i|>5000$ and $a(R_i) = 0.05$ otherwise
4. Stopping rule: find the number of regions smaller than 40 in which SIDE spent the largest number of iterations without a merge

PURDUE ENGINEERING

Segmentation of large regions—results

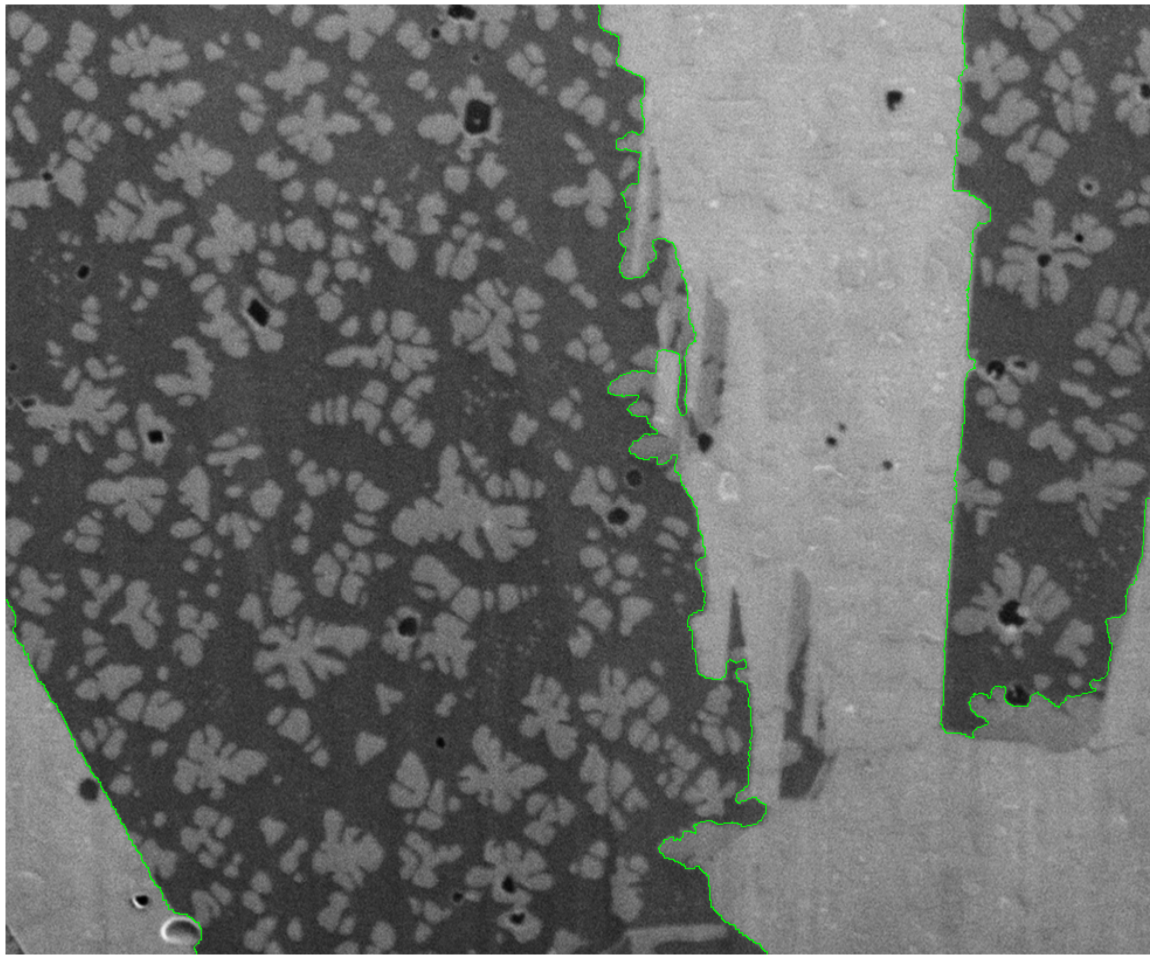


Segmentation of large regions—results



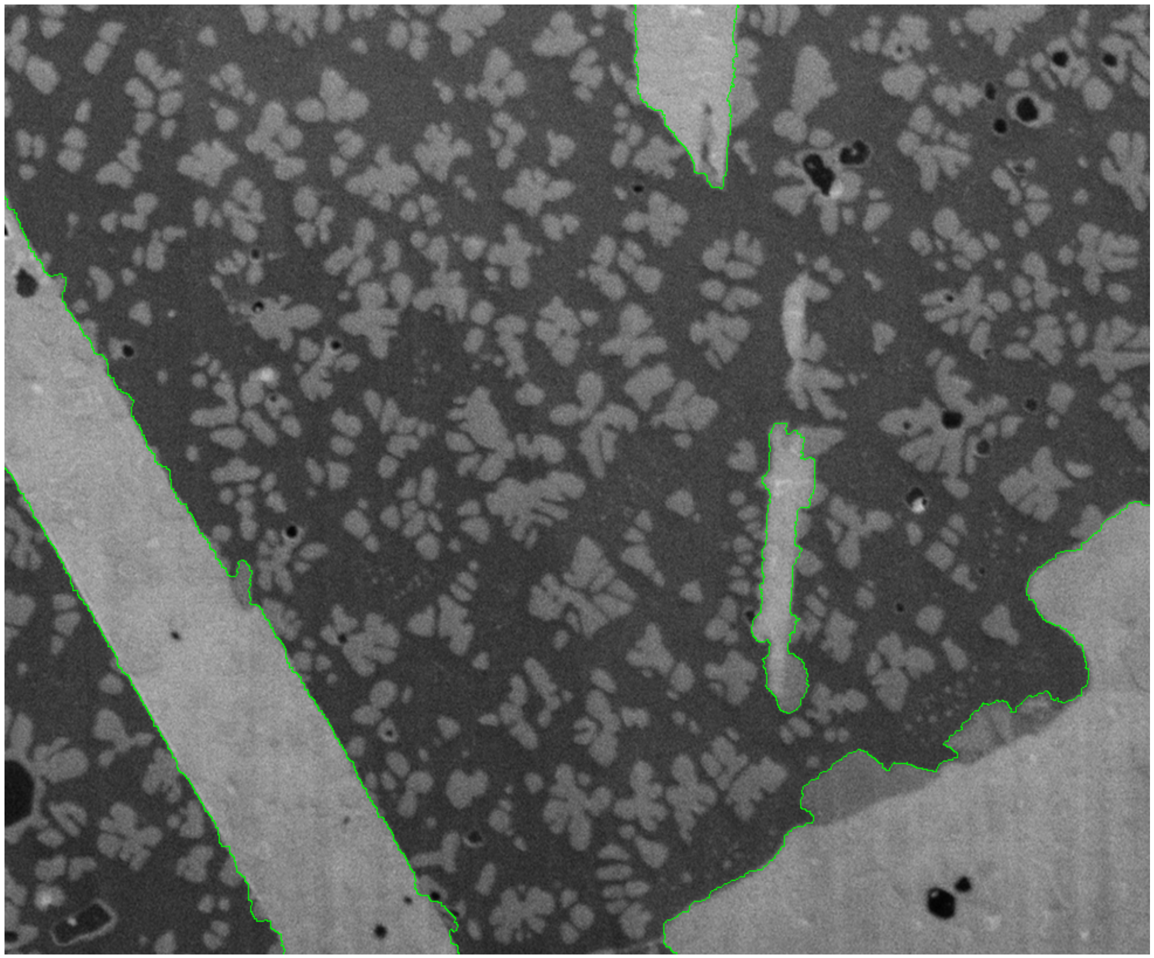
PURDUE
ENGINEERING

Segmentation of large regions—results

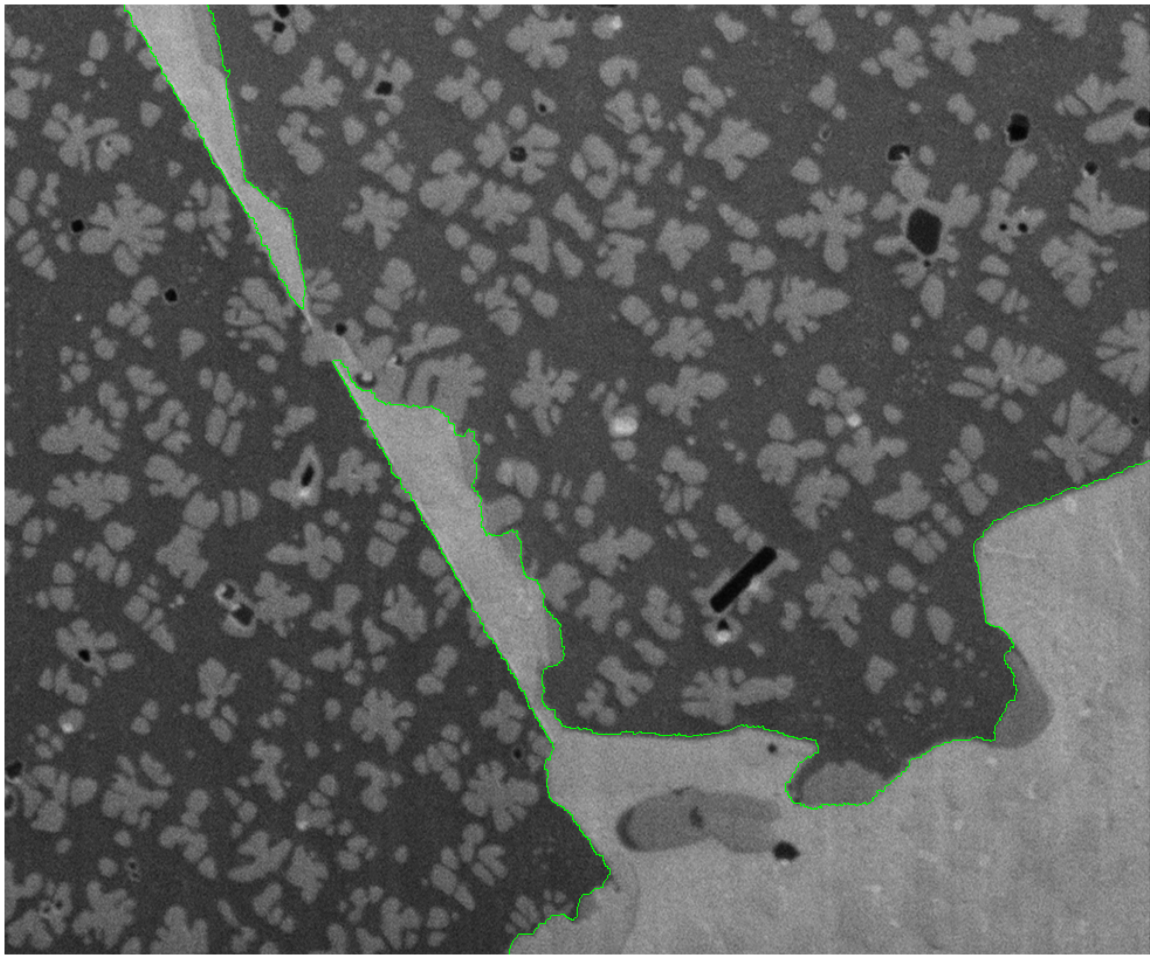


PURDUE ENGINEERING

Segmentation of large regions—results



Segmentation of large regions—results



PURDUE
ENGINEERING

Segmentation of small regions

- Segmenting large regions individually for finding small regions
  - Some large regions contain no small regions, and so we can use a bimodal Gaussian mixture to fit histogram of pixel data within the large region. If one mode is significantly smaller than the other($\pi_1<0.1$ or $\pi_2<0.1$), then stop; otherwise continue

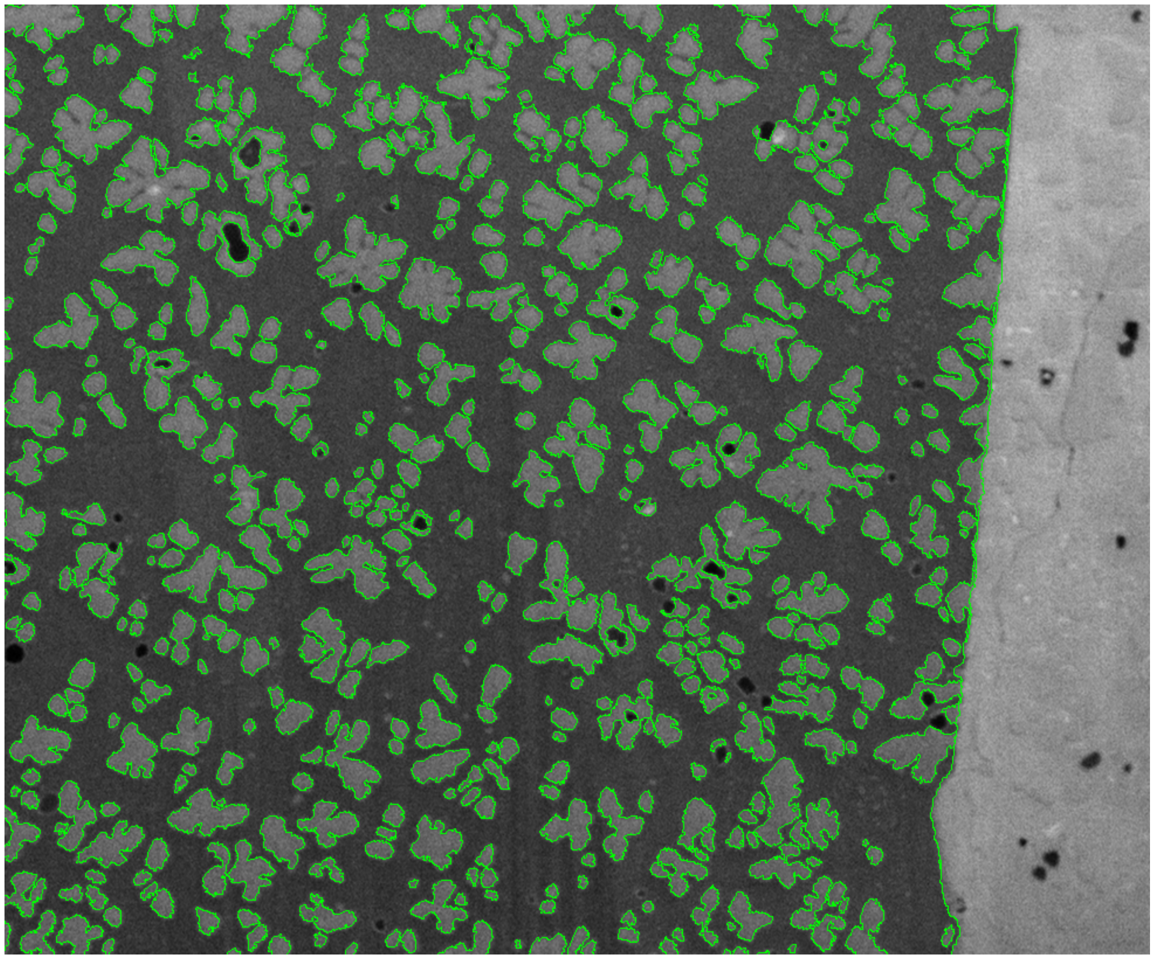
PURDUE
ENGINEERING

Segmentation of small regions (cont'd)

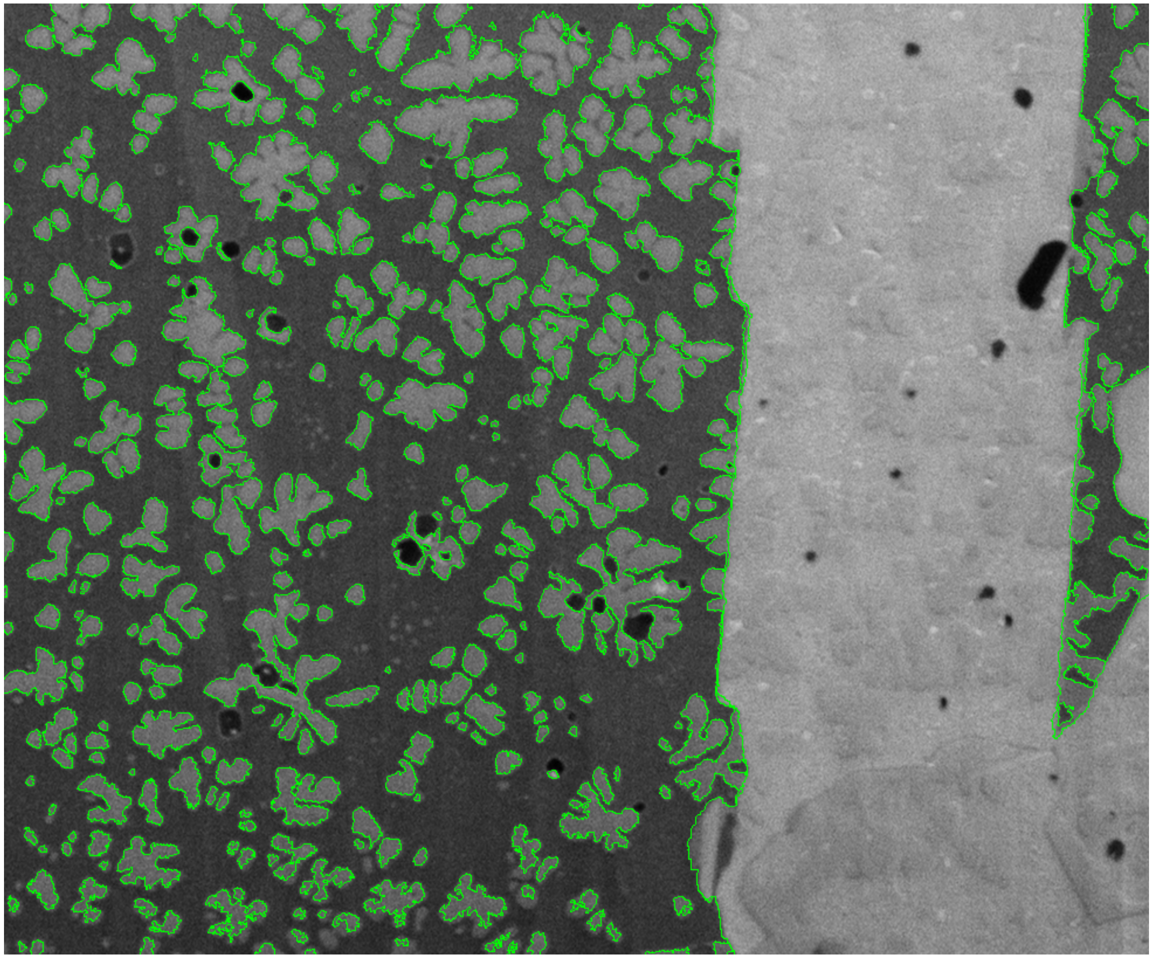
- For each large region containing small regions, we use a region-based initialization
  - Classify each pixel into one of the two classes using an ML estimator with a bimodal Gaussian mixture model
  - Initialize the segmentation regions R_i as contiguous collections of neighboring pixels with the same label, and initialize the scalar feature μ_i as the sample mean of the pixel intensities within R_i
- Set $\lambda=0.95, \eta=1$, and use no boundary curvature penalty
- Stopping rule: stop when 500 gradient descent iterations evolve without any regions being merged

PURDUE
ENGINEERING

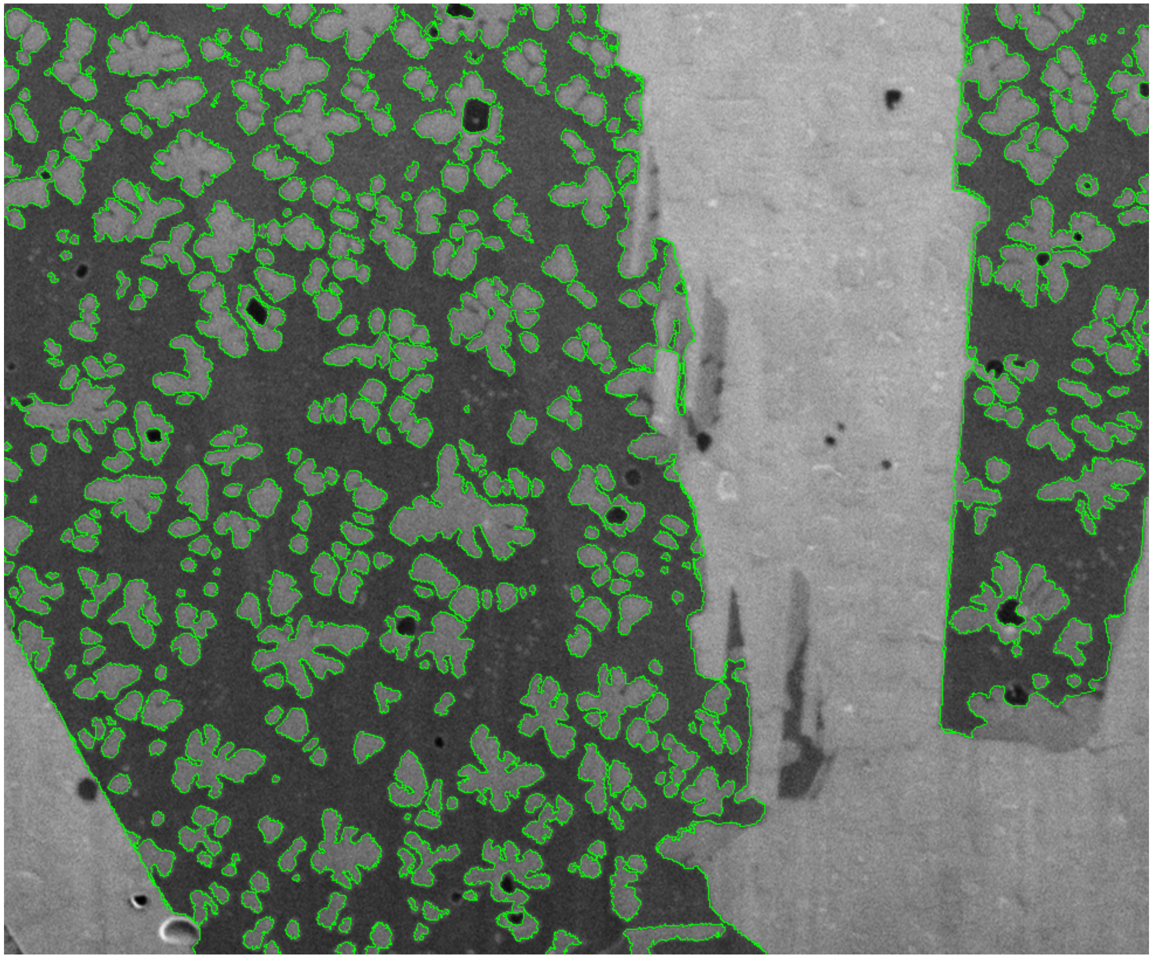
Results—complete segmentation



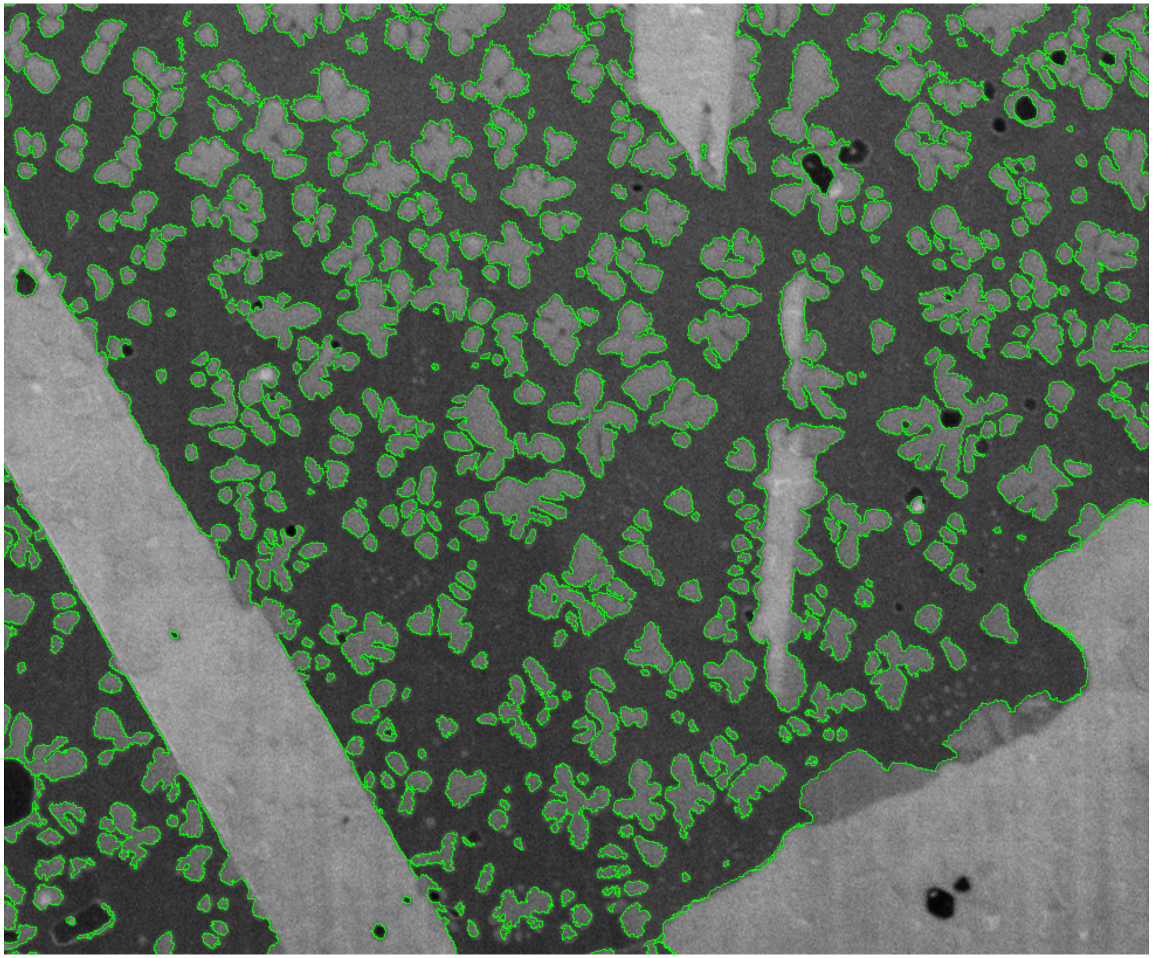
Results—complete segmentation



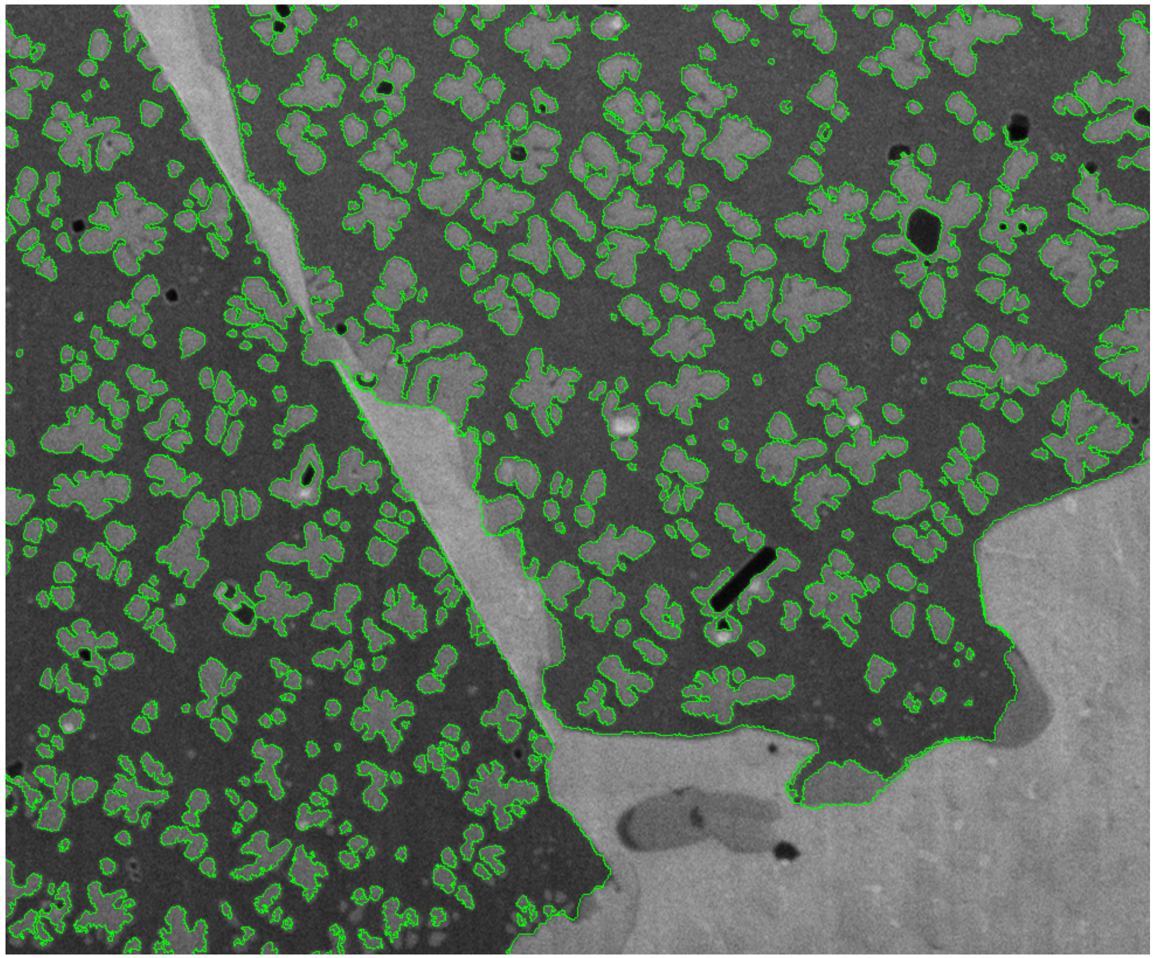
Results—complete segmentation



Results—complete segmentation



Results—complete segmentation



Summary

	MNML-3	IN100	Rene88
Intensity Penalty	Very useful, as there are only three different intensities.	Probably not usable for IN100, as there are too many different intensities.	Very useful, as there are only two different intensities.
Boundary length penalty	Useful.	Useful.	Useful.
Boundary smoothness penalty	Very useful, to ensure the smoothness of boundaries of the circular regions.	Useful, to avoid fractal-like boundaries.	Very useful, especially in the large-region segmentation stage, to identify long straight twin boundaries.
Stopping rule	Useful.	Useful.	Useful.

PURDUE
ENGINEERING