Mathematical Methodologies for the Detection and Segmentation of Surface Defects on Fiber Optic End Faces: A Computational Approach

Abstract:
This report provides an exhaustive examination of sophisticated, non-machine learning mathematical methodologies for the detection and segmentation of surface imperfections, specifically scratches and digs, on fiber optic end faces. The analysis is based on processing a raw pixel intensity matrix derived from a CSV file. The methodologies explored encompass fundamental image representation and pre-processing techniques, including Gaussian convolution, anisotropic diffusion, Fast Fourier Transform (FFT) filtering, and Singular Value Decomposition (SVD) for noise reduction. For defect detection, gradient-based operators (Sobel, Prewitt), transform-based methods (Hough, Radon), Laplacian-based techniques (Laplacian of Gaussian), and Hessian-based approaches (Determinant of Hessian, Hessian eigenvalue analysis) are detailed. Segmentation strategies, including mathematical morphology and Partial Differential Equation (PDE)-based models like active contours and level sets, are also presented. Numerical implementation considerations, such as PDE discretization and iterative solvers (Jacobi, Gauss-Seidel), are discussed. The report emphasizes the underlying mathematical principles, analytical formulations, and numerical procedures suitable for manual calculation, simulation, and in-depth understanding, while also considering robustness against noise, defect variability, and discretization artifacts.

1. Introduction

1.1. Problem Statement:

The reliable performance of fiber optic communication systems is critically dependent on the pristine condition of the optical surfaces, particularly the end faces of fibers and connectors. Microscopic surface imperfections, such as "scratches" (elongated, linear, or jagged features) and "digs" (localized, conglomerated, often circular or blob-like defects), can lead to significant signal loss, increased back-reflection, and overall degradation of system performance. The accurate identification and characterization of these defects are paramount during manufacturing, quality control, and maintenance operations.

This report addresses the challenge of detecting and segmenting such defects from raw pixel intensity data representing the fiber optic end face. The input data is provided as a CSV file containing x, y coordinates and corresponding intensity values.[1] An initial analysis of this data reveals the image dimensions to be 1152 pixels in width (x-coordinates from 0 to 1151) and 30 pixels in height (y-coordinates from 0 to 29).[1] This extreme aspect ratio (approximately 38:1) is a notable characteristic of the dataset and may influence the choice of algorithms or their parameterization, particularly for methods that assume more isotropic data structures or are sensitive to boundary conditions. For instance, defects oriented along the wider x-axis could be very long and thin, while those along the y-axis would be very short, potentially requiring anisotropic kernels or adaptive processing strategies.

1.2. Scope and Objectives:
The primary scope of this report is an exhaustive and meticulously detailed exploration of sophisticated, highly computational mathematical methods for the detection and segmentation of scratches and digs, explicitly excluding machine learning or deep learning approaches. The focus is strictly on foundational mathematical calculation procedures, encompassing concepts from calculus (differential operators, partial differential equations), linear algebra (matrix operations, eigenvalue decomposition, singular value decomposition), physics-based models (diffusion processes), and other PhD-level mathematical formalisms.

The main objective is to provide a comprehensive compendium of these calculation procedures, detailed in a manner suitable for manual analysis, step-by-step simulation, and fundamental theoretical understanding. This emphasis on the underlying computational steps is guided by the requirement for methods that can be conceptually simulated or manually traced, rather than relying on black-box automated systems. The report will address the variability inherent in defect characteristics, such as their intensity relative to the background (which can be darker, brighter, or variable) and their morphology (linear, jagged, circular, or blob-like). Furthermore, the techniques discussed will aim for robustness against common image noise and artifacts arising from the discretization of continuous mathematical models onto a pixel grid.

1.3. Report Structure:
This report is organized into several key sections. Section I details fundamental image representation techniques and pre-processing steps crucial for enhancing image quality and preparing data for defect analysis. Sections II and III delve into specific mathematical methods tailored for the detection of scratches (line-like defects) and digs (blob-like defects), respectively. Section IV discusses advanced segmentation techniques for accurately delineating the boundaries of detected defects. Section V addresses numerical implementation considerations pertinent to solving the mathematical models presented, particularly focusing on the discretization

of partial differential equations and iterative solvers for the resulting linear systems. Section VI explores advanced considerations regarding robustness to defect variability and image artifacts, and the potential for combining different mathematical approaches. Finally, Section VII provides concluding remarks, summarizing the findings and discussing the inherent challenges and trade-offs.

I. Fundamental Image Representation and Pre-processing

Effective defect detection necessitates a robust understanding of image representation and the application of appropriate pre-processing techniques to mitigate noise and enhance features of interest. The choice of pre-processing methods is critical, as it directly impacts the efficacy of subsequent detection and segmentation algorithms. There exists a fundamental interplay: while smoothing techniques reduce noise, they can also blur the very defects targeted for detection. Therefore, methods that selectively smooth or preserve critical features like edges are often preferred.

A. Representing the Pixel Grid as a Matrix:
    The input data, provided as a CSV file with columns `x,y,intensity` [1], is transformed into a two-dimensional matrix, denoted as <span class="math-inline">I\(x,y\)</span>. Given the maximum coordinates <span class="math-inline">X_\{max\}\=1151</span> and <span class="math-inline">Y_\{max\}\=29</span> [1], the matrix <span class="math-inline">I</span> will have dimensions <span class="math-inline">\(Y_\{max\}\+1\) \\times \(X_\{max\}\+1\)</span>, resulting in a <span class="math-inline">30 \\times 1152</span> matrix, assuming 0-indexed coordinates. Each element <span class="math-inline">I\(y,x\)</span> of this matrix represents the intensity value at the spatial location <span class="math-inline">\(x,y\)</span>. The intensity values in the provided data are floating-point numbers (e.g., 181.0), suggesting either a high dynamic range or pre-scaled data, rather than typical 8-bit integer values (0-255). This variability in intensity is a key characteristic to consider. Basic matrix operations, such as element-wise addition, subtraction, multiplication, and scalar multiplication, form the foundation for many image processing algorithms.

B. Mathematical Noise Models and Their Implications:
    Real-world image acquisition is often subject to noise, which can obscure defects or lead to false detections. Common noise models include:
    *   **Additive Gaussian Noise:** Characterized by adding a random value from a Gaussian distribution to each pixel intensity. Its probability density function is <span

class="math-inline">p\(z\) \= \\frac\{1\}\{\\sigma\\sqrt\{2\\pi\}\}e^\{\-\(z\-\\mu\)^2/\(2\\sigma^2\)\}</span>, where <span class="math-inline">\\mu</span> is the mean (often zero) and <span class="math-inline">\\sigma</span> is the standard deviation.
* **Salt-and-Pepper (Impulse) Noise:** Manifests as random occurrences of black and white pixels. Pixels are replaced by <span class="math-inline">p_\{min\}</span> or <span class="math-inline">p_\{max\}</span> with a certain probability.
The presence of such noise necessitates robust detection techniques or effective pre-processing for noise suppression.

C. Image Smoothing and Noise Reduction via Convolution:
Convolution with a smoothing kernel is a fundamental technique for noise reduction.
1. Gaussian Kernel Convolution:
Convolution with a Gaussian kernel is a widely used linear filtering technique for smoothing images and reducing noise, particularly noise that is Gaussian in nature.
* **Mathematical Formulation:** The 2D Gaussian kernel is defined as:
<span class="math-block">G\(x,y;\\sigma\) \= \\frac\{1\}\{2\\pi\\sigma^2\} e^\{\-\\frac\{x^2\+y^2\}\{2\\sigma^2\}\}</span>
where <span class="math-inline">\\sigma</span> is the standard deviation, controlling the scale or extent of smoothing.[2] The convolution of an image <span class="math-inline">I</span> with the Gaussian kernel <span class="math-inline">G</span> is given by:
<span class="math-block">\(I * G\)\(x,y\) \= \\sum_\{u\}\\sum_\{v\} I\(u,v\)G\(x\-u, y\-v\)</span>
The summation is performed over the spatial domain of the kernel.
* **Properties:**
* **Separability:** The 2D Gaussian kernel is separable, meaning <span class="math-inline">G\(x,y;\\sigma\) \= G\(x;\\sigma\)G\(y;\\sigma\)</span>, where <span class="math-inline">G\(x;\\sigma\)</span> is a 1D Gaussian function. This allows a 2D convolution to be efficiently performed as two sequential 1D convolutions (first along rows, then along columns), significantly reducing computational complexity.[2] This efficiency is particularly relevant for simulations or manual step-throughs.
* **Semi-Group Property:** Convolution of two Gaussian kernels <span class="math-inline">G_\{\\sigma_1\}</span> and <span class="math-inline">G_\{\\sigma_2\}</span> results in another Gaussian kernel <span class="math-inline">G_\{\\sqrt\{\\sigma_1^2\+\\sigma_2^2\}\}</span>.[2] This property is useful for building scale-space representations incrementally.
* **Differentiability:** Convolving an image with a Gaussian kernel renders the

image infinitely differentiable. This is because the Gaussian function itself is infinitely differentiable, allowing for stable computation of image derivatives by convolving the image with derivatives of the Gaussian.[2]
    *   **Connection to the Heat Equation:** Gaussian smoothing is mathematically equivalent to solving the heat equation (a parabolic partial differential equation) <span class="math-inline">\\frac\{\\partial u\}\{\\partial t\} \= D \\nabla^2 u</span>, where <span class="math-inline">u</span> is the image intensity, <span class="math-inline">t \= \\sigma^2/2</span> represents time, and <span class="math-inline">D</span> is the diffusion coefficient.[3] This physical analogy provides insight into the smoothing process as a diffusion of intensity values, leading to a more uniform distribution. The `itk-DiscreteGaussianImage` filter is a practical implementation of this concept.[3]
    *   **Relevance:** Gaussian smoothing is a fundamental pre-processing step. The choice of <span class="math-inline">\\sigma</span> is critical: a small <span class="math-inline">\\sigma</span> may leave too much noise, while a large <span class="math-inline">\\sigma</span> can blur out the very defects (especially fine scratches or small digs) that need to be detected. The extreme aspect ratio of the input image (1152x30) might warrant consideration of anisotropic <span class="math-inline">\\sigma</span> values if defects are expected to have preferential orientations, though standard Gaussian smoothing is isotropic.

D. Anisotropic Diffusion for Edge-Preserving Smoothing:
  While Gaussian smoothing is effective for noise reduction, its isotropic nature causes blurring of edges and fine details, which can be detrimental for detecting subtle scratches or precisely delineating the boundaries of digs. Anisotropic diffusion methods address this by varying the smoothing behavior spatially, smoothing more in homogeneous regions and less near strong edges.[3, 4]
  1. The Perona-Malik Equation:
    This is a well-known model for anisotropic diffusion, introduced by Perona and Malik.[4, 5]
    *   **Formulation:** The evolution of the image intensity <span class="math-inline">I</span> over time <span class="math-inline">t</span> is described by the PDE:
        <span class="math-block">\\frac\{\\partial I\}\{\\partial t\} \= \\nabla \\cdot \(c\(\\\|\\nabla I\\\|\) \\nabla I\) \= \\operatorname\{div\}\(c\(\\\|\\nabla I\\\|\)\\nabla I\)</span>
        where <span class="math-inline">\\nabla \\cdot</span> is the divergence operator, <span class="math-inline">\\nabla I</span> is the image gradient, and <span class="math-inline">c\(\\\|\\nabla I\\\|\)</span> is the diffusion coefficient (or

conductivity function).
    *   **Diffusion Coefficient:** The diffusion coefficient <span class="math-inline">c\(\\cdot\)</span> is a non-negative, monotonically decreasing function of the gradient magnitude <span class="math-inline">\\\|\\nabla I\\\|</span>. Common choices proposed by Perona and Malik are [4]:
        1. <span class="math-inline">c\(s\) \= e^\{\-\(s/K\)^2\}</span>
        2. <span class="math-inline">c\(s\) \= \\frac\{1\}\{1\+\(s/K\)^2\}</span>
        where <span class="math-inline">s \= \\\|\\nabla I\\\|</span> is the gradient magnitude, and <span class="math-inline">K</span> is a contrast parameter that acts as an edge threshold.
    *   **Mechanism:** In regions where the gradient magnitude is small (i.e., <span class="math-inline">\\\|\\nabla I\\\| \\ll K</span>), <span class="math-inline">c\(\\\|\\nabla I\\\|\)</span> is close to 1, and the equation behaves like the isotropic heat equation, leading to strong smoothing. In regions where the gradient magnitude is large (i.e., <span class="math-inline">\\\|\\nabla I\\\| \\gg K</span>, such as at edges), <span class="math-inline">c\(\\\|\\nabla I\\\|\)</span> approaches 0, inhibiting diffusion across the edge and thus preserving it. The parameter <span class="math-inline">K</span> is crucial and typically needs to be tuned based on image noise levels and edge strengths.
    *   **Numerical Solution:** The Perona-Malik equation is a non-linear PDE that is solved numerically using an iterative approach. Discretization via finite differences transforms the PDE into a system of equations that are updated at each iteration (time step).
    *   **Relevance:** This method is highly relevant for the current task due to its ability to reduce noise while preserving the sharpness of linear scratches and the distinct boundaries of digs, aligning with the user's requirement for robust techniques. The `itk-GradientAnisotropicDiffusionImage` filter is an example of such an implementation.[3] The physical interpretation of this as a diffusion process with spatially varying conductivity aligns with the user's interest in physics-based methods.

E. Frequency Domain Filtering for Noise Reduction:
  Filtering in the frequency domain provides an alternative approach to noise reduction by operating on the image's Fourier spectrum.
  1. Fast Fourier Transform (FFT):
    The FFT is an efficient algorithm to compute the Discrete Fourier Transform (DFT), which decomposes an image into its constituent frequencies.[6, 7]
    *   **Principle:** The image <span class="math-inline">I\(x,y\)</span> is transformed into its frequency domain representation <span class="math-inline">F\(u,v\)</span>. Noise often manifests as high-frequency

components, while the main structural information of the image is typically concentrated in lower frequencies.[7]
*   **Forward FFT:** The 2D DFT is given by:
<span class="math-block">F\(u,v\) \= \\sum_\{x\=0\}^\{M\-1\}\\sum_\{y\=0\}^\{N\-1\} I\(x,y\) e^\{\-j2\\pi\(\\frac\{ux\}\{N\}\+\\frac\{vy\}\{M\}\)\}</span>
where <span class="math-inline">M, N</span> are image dimensions in y and x respectively.
*   **Filtering in Frequency Domain:** A filter function <span class="math-inline">H\(u,v\)</span> is multiplied element-wise with <span class="math-inline">F\(u,v\)</span>. For noise reduction, a low-pass filter is commonly used, which attenuates or removes high-frequency components. Examples include ideal low-pass filters, Butterworth filters, or Gaussian low-pass filters. The process often involves [6]:
1. Computing the power spectrum <span class="math-inline">P\(u,v\) \= \|F\(u,v\)\|^2</span>.
2. Determining a threshold or designing a mask <span class="math-inline">H\(u,v\)</span> based on the power spectrum to isolate noise frequencies.
3. Applying the mask: <span class="math-inline">F_\{filtered\}\(u,v\) \= F\(u,v\)H\(u,v\)</span>.
*   **Inverse FFT:** The filtered image <span class="math-inline">I'\(x,y\)</span> is obtained by applying the inverse DFT:
<span class="math-block">I'\(x,y\) \= \\frac\{1\}\{MN\}\\sum_\{u\=0\}^\{N\-1\}\\sum_\{v\=0\}^\{M\-1\} F_\{filtered\}\(u,v\) e^\{j2\\pi\(\\frac\{ux\}\{N\}\+\\frac\{vy\}\{M\}\)\}</span>
*   **Relevance:** FFT-based filtering can be effective for removing periodic noise or attenuating general high-frequency noise. However, sharp cutoffs in the frequency domain can lead to ringing artifacts (Gibbs phenomenon) in the spatial domain. Careful filter design is essential.

F. Noise Reduction using Matrix Factorization:
Matrix factorization techniques can also be employed for noise reduction by separating the image into principal components and noise.
1. Singular Value Decomposition (SVD):
SVD decomposes the image matrix <span class="math-inline">I</span> (of size <span class="math-inline">M \\times N</span>) into the product of three matrices: <span class="math-inline">I \= U\\Sigma V^T</span>.[8, 9]
*   **Principle:** <span class="math-inline">U</span> is an <span

class="math-inline">M \\times M</span> orthogonal matrix, <span class="math-inline">\\Sigma</span> is an <span class="math-inline">M \\times N</span> diagonal matrix containing the singular values <span class="math-inline">\\sigma_i</span> in decreasing order (<span class="math-inline">\\sigma_1 \\ge \\sigma_2 \\ge \\dots \\ge 0</span>), and <span class="math-inline">V^T</span> is the transpose of an <span class="math-inline">N \\times N</span> orthogonal matrix.
    *   **Noise Separation:** The larger singular values in <span class="math-inline">\\Sigma</span> correspond to the dominant energy and structural components of the image. Smaller singular values are often associated with noise and finer details.
    *   **Procedure [8]:**
        1. Compute the SVD of the image matrix <span class="math-inline">I</span> to obtain <span class="math-inline">U, \\Sigma, V^T</span>.
        2. Create a truncated singular value matrix <span class="math-inline">\\Sigma_k</span> by keeping only the <span class="math-inline">k</span> largest singular values and setting the rest to zero. The choice of <span class="math-inline">k</span> is critical.
        3. Reconstruct the denoised image as <span class="math-inline">I' \= U\\Sigma_k V^T</span>.
    *   **Relevance:** SVD can effectively reduce noise by discarding components associated with small singular values. However, determining the optimal rank <span class="math-inline">k</span> is crucial; a <span class="math-inline">k</span> that is too small can lead to significant blurring and loss of important image details, including the defects themselves.[8] This method provides a global approach to noise reduction.

The selection of an appropriate pre-processing strategy, or a combination thereof, must consider the specific noise characteristics, the nature of the defects, and the computational constraints. For instance, the physical basis of diffusion models (Gaussian smoothing and anisotropic diffusion) aligns well with the request for physics-based approaches and offers intuitive parameters for adjustment during simulation. The parameter sensitivity of all these methods underscores the importance of understanding their mathematical behavior for effective manual simulation and analysis, as exhaustive automated parameter sweeps are outside the defined scope.

Table I.A: Comparison of Noise Reduction Techniques

Method	Mathematical Principle	Key Parameters	Strengths	Weaknesses/Artifacts	Robustness to Noise Types	Suitability for Scratch/Dig Preservation
Gaussian Convolution	Isotropic diffusion (Heat Equation)	Kernel size (σ)	Simple, effective for Gaussian noise, computationally efficient (separable)	Blurs edges and fine details	Good for Gaussian noise	Poor for fine scratches/ digs if σ is large
Anisotropic Diffusion	Edge-dependent diffusion (Perona-Malik PDE)	Conductivity function parameters (K), iterations	Preserves edges while smoothing regions, good for various noise types	More complex, iterative, sensitive to parameter K, potential for artifacts if unstable	Good for general noise, preserves structure	Good, designed to preserve edges of defects
FFT Filtering	Frequency domain suppression	Filter type (e.g., low-pass), cutoff frequency	Effective for periodic noise, can target specific frequency bands	Can introduce ringing artifacts (Gibbs phenomenon), global effect	Good for periodic noise, some high-frequency noise	Moderate, depends on defect frequency signature
SVD	Low-rank	Number of	Global	Can cause	Good for	Moderate,

	approximation	singular values to keep (k)	noise reduction, can separate structured noise	blurring/loss of detail if k is too small, computationally intensive for large images	random noise, some structured noise	depends on how defect energy is distributed in singular values

II. Mathematical Methods for Scratch (Line-like Defect) Detection
Scratches are characterized as elongated, linear, or jagged features. Their detection often relies on identifying their linear structure or the strong intensity gradients they produce perpendicular to their orientation.

A. Gradient-Based Approaches:
These methods operate on the principle that edges and lines, such as scratches, correspond to regions of high intensity gradient in an image.
1. The Sobel Operator:
The Sobel operator computes an approximation of the image gradient by convolving the image with a pair of 3x3 kernels, one for detecting horizontal edges and the other for vertical edges.[10, 11]
*   **Kernels [10]:**
The kernel for approximating the gradient in the x-direction (vertical lines/edges) is:
<span class="math-block">G_x \= \\begin\{bmatrix\} \-1 & 0 & \+1 \\\\ \-2 & 0 & \+2 \\\\ \-1 & 0 & \+1 \\end\{bmatrix\}</span>
The kernel for approximating the gradient in the y-direction (horizontal lines/edges) is:
<span class="math-block">G_y \= \\begin\{bmatrix\} \-1 & \-2 & \-1 \\\\ 0 & 0 & 0 \\\\ \+1 & \+2 & \+1 \\end\{bmatrix\}</span>
*   **Gradient Approximation:** The gradient components <span class="math-inline">S_x</span> and <span class="math-inline">S_y</span> are computed by convolving the input image <span class="math-inline">I</span> with these kernels:
<span class="math-inline">S_x \= G_x * I</span>

        <span class="math-inline">S_y \= G_y * I</span>
    *   **Gradient Magnitude:** The overall gradient magnitude <span class="math-inline">S</span> at each pixel is then calculated as:
        <span class="math-block">S \= \\sqrt\{S_x^2 \+ S_y^2\}</span>
        High values of <span class="math-inline">S</span> indicate the presence of an edge or line.
    *   **Gradient Direction:** The direction of the gradient <span class="math-inline">\\Theta</span> can also be computed, providing information about the orientation of the detected feature:
        <span class="math-block">\\Theta \= \\operatorname\{atan2\}\(S_y, S_x\)</span>
    *   **Relevance:** The Sobel operator is computationally efficient and simple to implement. Its inherent smoothing effect, due to the weighting of neighboring pixels (e.g., the '2's in the kernels), provides a degree of robustness to noise compared to simpler difference operators.[11] It is effective for detecting pronounced linear features like scratches, especially if their contrast with the background is significant.
  2. The Prewitt Operator:
    Similar to the Sobel operator, the Prewitt operator also uses a pair of 3x3 kernels to approximate the image gradient.[12, 13]
    *   **Kernels [13]:**
        The kernel for the x-direction gradient is:
        <span class="math-block">P_x \= \\begin\{bmatrix\} \-1 & 0 & \+1 \\\\ \-1 & 0 & \+1 \\\\ \-1 & 0 & \+1 \\end\{bmatrix\}</span>
        The kernel for the y-direction gradient is:
        <span class="math-block">P_y \= \\begin\{bmatrix\} \-1 & \-1 & \-1 \\\\ 0 & 0 & 0 \\\\ \+1 & \+1 & \+1 \\end\{bmatrix\}</span>
    *   **Gradient Approximation, Magnitude, and Direction:** These are calculated in the same manner as with the Sobel operator, using the Prewitt kernels.
    *   **Relevance:** The Prewitt operator is computationally very similar to the Sobel operator. It provides a simpler averaging scheme (uniform weights in the direction perpendicular to the derivative) compared to Sobel's weighted averaging. While computationally efficient, it is often considered slightly more sensitive to noise than the Sobel operator due to this simpler averaging.[12]

  The local nature of gradient operators makes them fast but potentially sensitive to noise if not preceded by adequate smoothing. They produce an "edge map" or gradient magnitude image, which then typically requires thresholding to identify candidate scratch pixels. The output of these operators can serve as input to more global line detection methods.

B. Transform-Based Line Detection:
These methods transform the image or its features (like edge points) into a different parameter space where lines have a more convenient and detectable representation, often as peaks or intersections.
1. The Hough Transform:
The Hough Transform is a powerful technique for detecting lines (and other parametric shapes) in images, particularly effective when lines are broken or partially occluded.[14, 15]
* **Parameter Space:** To avoid issues with infinite slopes in the Cartesian line equation <span class="math-inline">y \= mx \+ c</span>, the Hough Transform typically uses the polar parameterization of a line [14]:
<span class="math-block">\\rho \= x\\cos\\theta \+ y\\sin\\theta</span>
Here, <span class="math-inline">\\rho</span> is the perpendicular distance from the origin to the line, and <span class="math-inline">\\theta</span> is the angle of this perpendicular with respect to the x-axis. Each point <span class="math-inline">\(x,y\)</span> in the image space maps to a sinusoidal curve in the <span class="math-inline">\(\\rho, \\theta\)</span> parameter space.
* **Accumulator Array:** A 2D array, known as the accumulator <span class="math-inline">H\(\\rho, \\theta\)</span>, is created. The dimensions of this array correspond to quantized ranges of <span class="math-inline">\\rho</span> and <span class="math-inline">\\theta</span>.
* **Voting Process [14, 15]:**
1. An edge detection algorithm (e.g., Sobel, Prewitt, or more commonly Canny [15]) is applied to the image to identify candidate edge pixels.
2. For each detected edge pixel <span class="math-inline">\(x_i, y_i\)</span>:
For a range of discrete <span class="math-inline">\\theta</span> values, the corresponding <span class="math-inline">\\rho</span> value is calculated using the polar equation.
The cell <span class="math-inline">\(\[\\rho\], \[\\theta\]\)</span> in the accumulator array corresponding to the quantized <span class="math-inline">\\rho</span> and <span class="math-inline">\\theta</span> values is incremented. This is akin to casting a "vote".
* **Line Identification:** Collinear points in the image space will produce sinusoidal curves in the <span class="math-inline">\(\\rho, \\theta\)</span> space that intersect at a common point. Thus, cells in the accumulator array with high vote counts (peaks) correspond to lines in the original image. These peaks are typically found by applying a threshold to the accumulator.
* **Relevance:** The Hough Transform is robust to noise and gaps in lines,

making it well-suited for detecting jagged or discontinuous scratches.[14] Its global nature aggregates evidence from all edge pixels. However, it is computationally more intensive than local gradient operators, and its accuracy depends on the resolution of the <span class="math-inline">\(\\rho, \\theta\)</span> accumulator and the quality of the input edge map. The quantization of the parameter space is a critical step that influences both accuracy and computational load.
2. The Radon Transform:
The Radon Transform is closely related to the Hough Transform and can also be used for line detection.[16, 17]
* **Mathematical Principle:** The Radon transform of a 2D function <span class="math-inline">I\(x,y\)</span> is defined as its line integral along lines parameterized by their distance from the origin <span class="math-inline">\\rho</span> and their angle <span class="math-inline">\\theta</span>:
<span class="math-block">R\(\\rho, \\theta\) \= \\int_\{\-\\infty\}^\{\\infty\} \\int_\{\-\\infty\}^\{\\infty\} I\(x,y\) \\delta\(\\rho \- x\\cos\\theta \- y\\sin\\theta\) \\,dx\\,dy</span>
where <span class="math-inline">\\delta\(\\cdot\)</span> is the Dirac delta function. In essence, it projects the image intensity along lines oriented at various angles.
* **Line Detection:** Straight lines in the image <span class="math-inline">I\(x,y\)</span> manifest as peaks in the Radon transform space <span class="math-inline">R\(\\rho, \\theta\)</span>.[17] If a scratch has a consistently different intensity from its surroundings, it will produce a strong signal in the Radon transform at the corresponding <span class="math-inline">\(\\rho, \\theta\)</span>.
* **Relevance:** Similar to the Hough Transform, the Radon Transform provides a global measure for line presence and is robust to noise and minor discontinuities. It is particularly effective if the scratches themselves have a distinct intensity profile that can be integrated.

Both Hough and Radon transforms offer advantages for detecting scratches that are not perfectly continuous or straight, which is a common characteristic of real-world defects. Their ability to aggregate evidence globally makes them less susceptible to local noise compared to simple gradient operators. However, their computational cost, especially for achieving high resolution in the parameter space, is a consideration for manual simulation or basic implementations.

Table II.A: Comparison of Line Detection Methods

Method	Mathematical Principle	Key Parameters	Strengths for Scratch Detection	Weaknesses	Computational Complexity for Manual Simulation
Sobel Operator	Discrete 1st derivative approximation (weighted average)	Kernel (fixed 3x3)	Simple, fast, some noise smoothing, good for prominent edges/lines	Sensitive to noise (less than basic diff), orientation bias, fixed scale	Low
Prewitt Operator	Discrete 1st derivative approximation (simple average)	Kernel (fixed 3x3)	Very simple, fast, good for prominent edges/lines	More sensitive to noise than Sobel, orientation bias, fixed scale	Low
Hough Transform	Parameter space voting for line parameters (ρ,θ)	Accumulator resolution (Δρ,Δθ), edge threshold, peak threshold	Robust to gaps and noise, detects multiple lines, handles jagged lines	Computationally intensive, sensitive to accumulator resolution and thresholds, detects infinite lines	High
Radon Transform	Projection of image intensity along lines (ρ,θ)	Angular and spatial resolution, peak threshold	Robust to gaps and noise, good for features with consistent intensity	Computationally intensive, can be sensitive to intensity variations along the line	High

III. Mathematical Methods for Dig (Blob-like Defect) Detection

Digs are typically localized, conglomerated defects that appear as roughly circular or blob-like regions with intensities differing from the background. Their detection often involves identifying regions of local intensity extrema or specific curvature patterns. A key aspect of blob detection is handling variations in blob size, which often necessitates scale-space approaches.

A. Laplacian-based Methods:
The Laplacian operator, <span class="math-inline">\\nabla^2 \= \\frac\{\\partial^2\}\{\\partial x^2\} \+ \\frac\{\\partial^2\}\{\\partial y^2\}</span>, is a second-order derivative operator that is sensitive to regions of rapid intensity change and is often used to find edges and blobs. For an idealized blob (a peak or valley in intensity), the Laplacian response will be strong at its center.
1. Laplacian of Gaussian (LoG):
To make the Laplacian operator more robust to noise, the image is typically smoothed with a Gaussian kernel before applying the Laplacian. This combined operation is known as the Laplacian of Gaussian (LoG).[18, 19, 20, 21, 22, 23, 24]
* **Formulation:** Due to the properties of convolution, the LoG can be computed by convolving the image <span class="math-inline">I</span> with an LoG kernel:
<span class="math-block">\\text\{LoG\}\(I\) \= \(\\nabla^2 G_\\sigma\) * I</span>
where <span class="math-inline">G_\\sigma</span> is a Gaussian kernel with scale <span class="math-inline">\\sigma</span>. The LoG kernel itself is given by:
<span class="math-block">\\text\{LoG\}\(x,y;\\sigma\) \= \\nabla^2 G_\\sigma\(x,y\) \= \\frac\{1\}\{\\pi\\sigma^4\} \\left\(\\frac\{x^2\+y^2\}\{2\\sigma^2\} \- 1 \\right\) e^\{\-\\frac\{x^2\+y^2\}\{2\\sigma^2\}\}</span>
The response of this operator will be high at the center of blobs whose size "matches" the scale <span class="math-inline">\\sigma</span> of the Gaussian.
* **Scale-Space Representation:** To detect blobs of various sizes, the LoG operator is applied at multiple scales (i.e., with different values of <span class="math-inline">\\sigma</span>). The responses are then stacked, creating a 3D scale-space volume <span class="math-inline">\(x, y, \\sigma\)</span>. Blobs are identified as local extrema (maxima for bright blobs on a dark background, or minima for dark blobs on a bright background) in this 3D scale-space.[19, 21, 22] The <span class="math-inline">\(x,y\)</span> location of the extremum gives the blob's center, and the <span class="math-inline">\\sigma</span> value at which the extremum occurs provides an estimate of the blob's characteristic scale (size).

    *   **Scale-Normalized Laplacian:** For the response of the LoG operator to be truly comparable across different scales and to accurately reflect the presence of a blob whose size matches the scale, it is often scale-normalized. The scale-normalized Laplacian is typically defined as <span class="math-inline">\\sigma^2 \\nabla^2 G_\\sigma</span> or <span class="math-inline">t \\nabla^2 L</span> where <span class="math-inline">L\(x,y;t\) \= G\(x,y;t\)*I\(x,y\)</span> and <span class="math-inline">t\=\\sigma^2</span>.[20, 21, 24] The maxima/minima of this scale-normalized Laplacian in scale-space are robust indicators of blob presence and scale. The formula for the normalized LoG can be written as [24]:
        <span class="math-block">\\text\{LoG\}_\{\\text\{normalized\}\}\(x,y;\\sigma\) \= \\sigma^2 \\cdot \\text\{LoG\}\(x,y;\\sigma\) \= \\frac\{1\}\{\\pi\\sigma^2\} \\left\(\\frac\{x^2\+y^2\}\{2\\sigma^2\} \- 1 \\right\) e^\{\-\\frac\{x^2\+y^2\}\{2\\sigma^2\}\}</span>
    *   **Difference of Gaussians (DoG) Approximation:** The LoG operator can be efficiently approximated by the Difference of Gaussians (DoG), which involves subtracting two Gaussian-blurred versions of the image obtained with slightly different scales (<span class="math-inline">\\sigma_1</span> and <span class="math-inline">\\sigma_2</span>, where typically <span class="math-inline">\\sigma_1 \= k\\sigma_2</span> for some constant <span class="math-inline">k</span>, e.g., <span class="math-inline">k\=\\sqrt\{2\}</span>).[21, 23] This approximation is computationally less expensive than direct LoG convolution, especially when building a scale-space.
    *   **Relevance:** The LoG method is a cornerstone of blob detection due to its mathematical elegance and its ability to detect blobs of varying sizes via scale-space analysis. It is particularly good at finding isolated, roughly circular regions. The use of second-order derivatives makes it sensitive to the rapid change in gradient that occurs around the center of a blob.

B. Hessian-based Methods:
  The Hessian matrix of an image captures information about its local second-order structure (curvature). It is a matrix of second-order partial derivatives.
  <span class="math-block">H\(I\) \= \\begin\{bmatrix\} I_\{xx\} & I_\{xy\} \\\\ I_\{yx\} & I_\{yy\} \\end\{bmatrix\}</span>
  where <span class="math-inline">I_\{xx\} \= \\frac\{\\partial^2 I\}\{\\partial x^2\}</span>, <span class="math-inline">I_\{yy\} \= \\frac\{\\partial^2 I\}\{\\partial y^2\}</span>, and <span class="math-inline">I_\{xy\} \= I_\{yx\} \= \\frac\{\\partial^2 I\}\{\\partial x \\partial y\}</span>. These derivatives are usually computed on an image smoothed by a Gaussian <span class="math-inline">I_\\sigma \= I *

G_\\sigma</span> to ensure stability and define the scale of analysis.
1. Determinant of Hessian (DoH):
The determinant of the Hessian matrix (DoH) is a scalar measure that can be used to detect blob-like structures.[25, 26, 27]
* **Formulation:**
<span class="math-block">\\text\{det\}\(H\(I_\\sigma\)\) \= I_\{xx\}I_\{yy\} \- I_\{xy\}^2</span>
* **Blob Identification:** Blobs correspond to locations where the DoH is maximal (for bright blobs on dark backgrounds if <span class="math-inline">I_\{xx\}</span> and <span class="math-inline">I_\{yy\}</span> are negative, or minimal for dark blobs if <span class="math-inline">I_\{xx\}</span> and <span class="math-inline">I_\{yy\}</span> are positive, assuming a convention where the Laplacian is positive for dark blobs). A scale-normalized version, often <span class="math-inline">\\sigma^4 \\text\{det\}\(H\)</span>, is used when searching across scales to ensure that the response magnitude is independent of the blob's scale.[21] Maxima of this scale-normalized DoH in scale-space indicate blob centers and their characteristic scales.
* **Relevance:** The DoH operator is computationally efficient and robust. It responds strongly to regions with high curvature in multiple directions, which is characteristic of blobs. It can detect both bright and dark blobs.[26] However, it may not be as accurate for very small blobs.[26]
2. Eigenvalue Decomposition of the Hessian Matrix:
Analyzing the eigenvalues of the Hessian matrix provides more detailed information about the local image structure and can be used to discriminate between blobs, lines, and other shapes.[27, 28, 29, 30, 31]
* **Eigenvalues and Eigenvectors:** For the 2D Hessian matrix <span class="math-inline">H</span>, there are two eigenvalues, <span class="math-inline">\\lambda_1</span> and <span class="math-inline">\\lambda_2</span> (ordered, e.g., <span class="math-inline">\|\\lambda_1\| \\le \|\\lambda_2\|</span>), and corresponding eigenvectors <span class="math-inline">\\mathbf\{v\}_1, \\mathbf\{v\}_2</span>. The eigenvectors indicate the directions of principal curvatures, and the eigenvalues indicate the magnitude of these curvatures.
* **Blob Characterization [29, 31]:**
    * For an ideal circular blob, the curvature is high and roughly equal in all directions around its center. This translates to <span class="math-inline">\|\\lambda_1\|</span> and <span class="math-inline">\|\\lambda_2\|</span> being large and of similar magnitude.
    * If the blob is bright on a dark background, both <span

class="math-inline">\\lambda_1</span> and <span class="math-inline">\\lambda_2</span> will be large and negative (assuming the intensity function is like an inverted bowl).

        *   If the blob is dark on a bright background, both <span class="math-inline">\\lambda_1</span> and <span class="math-inline">\\lambda_2</span> will be large and positive (intensity function is like a bowl).

        The condition <span class="math-inline">\|\\lambda_2\| \\neq 0</span> and <span class="math-inline">\|\\lambda_3\|</span> (in 3D, implying both <span class="math-inline">\|\\lambda_1\|</span> and <span class="math-inline">\|\\lambda_2\|</span> for 2D) being significant indicates a structure with strong gradients in multiple directions, characteristic of blobs or vessels/nodules in medical imaging contexts.[31]

    *   **Shape Analysis:** The ratio of eigenvalues, such as <span class="math-inline">\|\\lambda_1\|/\|\\lambda_2\|</span>, can be used to assess the anisotropy of the structure. For a circular blob, this ratio will be close to 1. For an elongated structure (like a scratch segment), one eigenvalue will be significantly larger in magnitude than the other. This provides a mechanism to distinguish digs from elongated scratches, which is crucial for the user's task.

    *   **Relevance:** Eigenvalue analysis of the Hessian offers a powerful way to characterize local image structures. It is more computationally intensive than simply calculating the DoH, as it requires eigenvalue decomposition at each pixel (and potentially at multiple scales). However, the richer information it provides can lead to more robust blob detection and shape discrimination. This method is particularly valuable for distinguishing true blob-like digs from other features that might also elicit a response from simpler operators.

The successful application of these methods for dig detection often relies on a multi-scale approach to handle the variability in defect sizes. The mathematical foundation in second-order derivatives captures the characteristic intensity profiles of blob-like structures. Furthermore, the ability of Hessian eigenvalue analysis to provide shape information is a significant advantage in differentiating digs from other types of defects or image features.

Table III.A: Comparison of Blob Detection Methods

Method	**Mathematic**	**Key**	**Strengths**	**Weaknesse**	**Computatio**

	al Principle	**Parameters**	**for Dig Detection**	**s**	**nal Aspects (Manual/Simulation)**
Laplacian of Gaussian (LoG)	Scale-space extrema of ∇2(Gσ∗I) or σ2∇2(Gσ∗I)	σ (scale), scale-space levels, response threshold	Good size estimation, detects bright/dark blobs, robust to noise (due to Gaussian)	Can be slow for many scales/large σ, may respond to edges/lines as well as blobs	Moderate to High (multiple convolutions)
Determinant of Hessian (DoH)	Scale-space extrema of det(H(Gσ∗I)) or σ4det(H)	σ (scale), scale-space levels, response threshold	Computationally faster than LoG for some implementations, detects bright/dark blobs	May not detect small blobs accurately [26], can be sensitive to noise without proper smoothing	Moderate
Hessian Eigenvalue Analysis	Analysis of eigenvalues (λ1,λ2) of H(Gσ∗I) across scales	σ (scale), criteria on λ1,λ2 (magnitudes, ratios, signs)	Excellent shape discrimination (blobs vs. lines vs. sheets), good size estimation	Most computationally intensive (eigenvalue decomposition on per pixel/scale), complex criteria	High

IV. Segmentation and Region Delineation

Once potential defect locations (scratches or digs) have been identified by the detection methods outlined in Sections II and III, the next critical step is to accurately delineate their boundaries. This process, known as segmentation, aims to precisely isolate the pixels belonging to each defect from the background. The output of detection algorithms is often a likelihood map or a set of candidate points/regions, which require further processing to yield a clean segmentation.

A. Mathematical Morphology for Shape Analysis and Refinement:
Mathematical morphology is a powerful non-linear image processing framework based on set theory and topology. It probes an image with a small pattern called a structuring element (SE) to modify its geometrical structures.[32, 33] It is particularly useful for post-processing binary or grayscale images resulting from initial detection steps.
* **Set-Theoretic Definitions [32, 33]:**
Let <span class="math-inline">A</span> be the input image (set of pixels) and <span class="math-inline">B</span> be the structuring element.
  * **Dilation (<span class="math-inline">A \\oplus B</span>):** Expands the boundaries of regions in <span class="math-inline">A</span>. Defined as the set of all points <span class="math-inline">z</span> such that the reflection of <span class="math-inline">B</span>, <span class="math-inline">\\hat\{B\}</span>, translated by <span class="math-inline">z</span>, has a non-empty intersection with <span class="math-inline">A</span>:
<span class="math-block">A \\oplus B \= \\\{z \\mid \(\\hat\{B\}\)_z \\cap A \\neq \\emptyset \\\}</span>
  * **Erosion (<span class="math-inline">A \\ominus B</span>):** Shrinks the boundaries of regions in <span class="math-inline">A</span>. Defined as the set of all points <span class="math-inline">z</span> such that <span class="math-inline">B</span> translated by <span class="math-inline">z</span> is entirely contained within <span class="math-inline">A</span>:
<span class="math-block">A \\ominus B \= \\\{z \\mid B_z \\subseteq A \\\}</span>
  * **Opening (<span class="math-inline">A \\circ B</span>):** Defined as an erosion followed by a dilation with the same structuring element:
<span class="math-block">A \\circ B \= \(A \\ominus B\) \\oplus B</span>
Opening generally smooths contours, removes small isolated objects (noise), breaks narrow connections (isthmuses), and eliminates thin protrusions, without significantly changing the size of larger objects.[32]
  * **Closing (<span class="math-inline">A \\bullet B</span>):** Defined as a dilation followed by an erosion with the same structuring element:
<span class="math-block">A \\bullet B \= \(A \\oplus B\) \\ominus B</span>
Closing tends to fill small holes within objects, fuse narrow breaks and long thin gaps, and smooth contours, without significantly changing the size of larger objects.[32]
* **Structuring Element (SE):** The choice of the SE's shape and size is critical and depends on the geometry of the features being processed. For roughly circular digs, a disk-shaped SE would be appropriate. For linear scratches, a line-shaped SE,

possibly oriented along the scratch direction (if known or estimated), could be beneficial.
    *   **Application to Defect Isolation and Cleaning:**
        After an initial thresholding of a defect likelihood map (e.g., from LoG or Hessian responses), morphological operations can refine the result:
        *   **Noise Removal:** Opening can remove small, isolated high-intensity pixels (false positives) if their size is smaller than the SE.
        *   **Gap Filling/Region Consolidation:** Closing can fill small gaps within a detected defect region or connect nearby components of a fragmented defect.
        *   **Boundary Smoothing:** Opening and closing can smooth the jagged boundaries of detected regions.
        *   **Boundary Extraction:** The boundary of a set <span class="math-inline">A</span> can be obtained by subtracting its erosion from itself: <span class="math-inline">\\text\{Boundary\}\(A\) \= A \- \(A \\ominus B\)</span>.[32]
        *   **Region Filling:** Iterative application of dilation, complementation, and intersection can fill bounded regions starting from an interior seed point.[32]
    *   **Relevance:** Morphological operations are indispensable for post-processing the raw outputs of detection algorithms. They help in refining the shape of detected defects, removing noise artifacts from binary masks, and preparing the segmented regions for further analysis (e.g., measurement of size and shape). Their non-linear nature makes them powerful for shape-based manipulations.

B. PDE-Based Segmentation Approaches:
    Partial Differential Equation (PDE)-based methods evolve curves or surfaces dynamically based on forces derived from image data and internal constraints (e.g., smoothness), until they conform to the boundaries of objects of interest. These methods often integrate detection and segmentation.
    1. Active Contour Models (Snakes):
        Active contour models, or "snakes," are energy-minimizing deformable splines that are influenced by image forces pulling them towards object contours and internal forces resisting deformation.[34]
        *   **Energy-Minimizing Spline Formulation [34]:** A snake is represented as a parametric curve <span class="math-inline">\\mathbf\{v\}\(s\) \= \(x\(s\), y\(s\)\)</span>, where $s \in $. The total energy of the snake is minimized:
            <span class="math-block">E_\{snake\}^* \= \\int_0^1 \(E_\{internal\}\(\\mathbf\{v\}\(s\)\) \+ E_\{external\}\(\\mathbf\{v\}\(s\)\)\) ds</span>
        *   **Internal Energy (<span class="math-inline">E_\{internal\}</span>) [34]:** This term controls the snake's intrinsic properties, like elasticity and stiffness, ensuring smoothness and continuity. It is typically defined as:

<span class="math-block">E_\{internal\}\(\\mathbf\{v\}\(s\)\) \= \\frac\{1\}\{2\} \\left\(\\alpha\(s\) \\left\| \\frac\{d\\mathbf\{v\}\}\{ds\} \\right\|^2 \+ \\beta\(s\) \\left\| \\frac\{d^2\\mathbf\{v\}\}\{ds^2\} \\right\|^2 \\right\)</span>

Here, <span class="math-inline">\\alpha\(s\)</span> controls the tension (resistance to stretching), and <span class="math-inline">\\beta\(s\)</span> controls the rigidity (resistance to bending).

* **External Energy (<span class="math-inline">E_\{external\}</span>) [34]:** This term couples the snake to the image data, guiding it towards salient features. It can be formulated to attract the snake to lines, edges, or other features. A common form for edge attraction is <span class="math-inline">E_\{external\} \= \-w_\{edge\} \|\\nabla I\(\\mathbf\{v\}\(s\)\)\|^2</span>, where <span class="math-inline">I</span> is the image intensity and <span class="math-inline">w_\{edge\}</span> is a weight.
* **Evolution:** The snake evolves iteratively to minimize its total energy. This is often achieved by solving the Euler-Lagrange equations derived from the energy functional, which results in a set of force balance equations that govern the snake's motion.
* **Initialization:** Snakes require an initial contour to be placed near the target object boundary.[34, 35] The quality of this initialization significantly affects convergence and accuracy. The output of a preliminary detection stage could provide this initialization.
* **Relevance:** Snakes can provide accurate and smooth segmentation of defect boundaries. The internal energy terms help in dealing with noise and irregularities in the detected boundaries. The "edge" method available in tools like `activecontour` [35] is directly applicable for segmenting features with strong gradients.

2. Level Set Methods:

Level set methods offer a more flexible framework for curve evolution, particularly in handling topological changes such as splitting and merging of contours.[36]

* **Implicit Contour Representation [36]:** Instead of explicitly parameterizing the contour, the level set method embeds the evolving contour as the zero level set (i.e., <span class="math-inline">\\phi\(x,y,t\)\=0</span>) of a higher-dimensional function <span class="math-inline">\\phi\(x,y,t\)</span>, called the level set function.
* **Evolution PDE [36, 37]:** The evolution of the level set function <span class="math-inline">\\phi</span> is governed by a PDE:

<span class="math-block">\\frac\{\\partial \\phi\}\{\\partial t\} \+ F\|\\nabla \\phi\| \= 0</span>

Here, <span class="math-inline">F</span> is the speed function, which dictates the velocity of the contour in its normal direction. <span class="math-inline">F</span> is derived from image properties and can include terms based on image gradient (to

stop at edges), regional statistics (to segment regions of similar intensity), and curvature (to maintain smoothness). For example, an edge-based speed function might be <span class="math-inline">F \= g\(\|\\nabla I\|\) \(1 \- \\epsilon \\kappa\)</span>, where <span class="math-inline">g</span> is a decreasing function of gradient magnitude (e.g., <span class="math-inline">g\(s\) \= 1/\(1\+s^2\)</span>) and <span class="math-inline">\\kappa</span> is the curvature of the level set.
    *   **Advantages:** The primary advantage is the ability to handle complex topological changes naturally. The contour can split into multiple parts or merge without requiring explicit re-parameterization. This is beneficial if multiple digs are close together or if a scratch is fragmented.
    *   **Initialization:** Similar to snakes, level set methods require an initial level set function <span class="math-inline">\\phi_0\(x,y\)</span> (e.g., the signed distance function from an initial closed curve). The evolution starts from this initial state.
    *   **Relevance:** Level set methods are very powerful for segmentation. Region-based variants, like the Chan-Vese model [35], are less reliant on strong gradients and can segment objects based on intensity homogeneity within the object and background, making them potentially robust to variable defect intensities. The implicit representation and ability to handle topology make them suitable for complex defect morphologies.

The choice between morphological post-processing and PDE-based segmentation depends on the complexity of the defects and the desired accuracy. Morphological operations are generally simpler and faster for basic refinement tasks. PDE-based methods, while more computationally intensive and requiring careful initialization, offer more sophisticated control over the segmentation process and can adapt to complex boundary shapes and image characteristics. The quality of segmentation from PDE methods is highly dependent on the initial contour placement; thus, outputs from earlier detection stages (Sections II and III) can serve as crucial starting points.

Table IV.A: Comparison of Segmentation Methods

Method	**Principle**	**Key Parameters**	**Initialization Needs**	**Strengths for Defect Delineation**	**Weaknesses**	**Topological Flexibility**
Mathematical	Set-theoretic	SE shape and size,	None for basic	Simple, fast, good	Results highly	Fixed (operates

Morphology	operations with a structuring element (SE)	sequence of operations (erosion, dilation, opening, closing)	operations on a binary image; input is typically a thresholded detection map.	for noise removal, hole filling, separating/connecting components, boundary refinement	dependent on SE choice; may distort true shape if SE is not well-matched to feature geometry	on existing regions)
Active Contours (Snakes)	Energy-minimizing deformable spline	Internal energy weights (α,β), external energy weights (wline,wedge), iteration parameters	Requires an initial contour close to the target object boundary.	Smooth and continuous boundaries, can incorporate prior shape knowledge via energy terms.	May get stuck in local minima, struggles with topological changes (splitting/merging), sensitive to initialization.	Limited (basic snake)
Level Set Methods	Implicit surface evolution governed by a PDE	Speed function F (image term, curvature term), time step Δt, number of iterations	Requires an initial level set function $\phi0$ (e.g., from an initial curve).	Handles topological changes naturally (splitting, merging), robust to initialization (compared to snakes sometimes), can use region/edge informatio	Computationally more intensive than basic morphology, can be complex to implement and tune.	High

				n		

V. Numerical Implementation Considerations
Many of the advanced mathematical methods discussed, particularly those involving image smoothing via diffusion (Heat Equation, Perona-Malik) and PDE-based segmentation (Active Contours, Level Sets), are defined by partial differential equations. To apply these continuous models to a discrete pixel grid, numerical discretization and solution techniques are essential.

A. Discretization of Partial Differential Equations:
    The process of converting a continuous PDE into a system of algebraic equations that can be solved on a discrete grid is known as discretization. The Finite Difference Method (FDM) is a common approach for this in image processing.
    *   **Finite Difference Method (FDM):** This method approximates the partial derivatives in a PDE using difference quotients calculated from pixel values at neighboring grid points.
        *   **Spatial Derivatives:** For an image <span class="math-inline">I\(x,y\)</span> defined on a grid with spacing <span class="math-inline">\\Delta x</span> and <span class="math-inline">\\Delta y</span> (often assumed to be 1 for pixel units):
            *   First-order derivatives (e.g., gradient components):
                Central difference: <span class="math-inline">I_x\(i,j\) \\approx \\frac\{I\(i,j\+1\) \- I\(i,j\-1\)\}\{2\\Delta x\}</span> (using <span class="math-inline">j</span> for x-index, <span class="math-inline">i</span> for y-index as per matrix notation, or vice-versa if image coordinates are used)
                Forward/Backward difference: <span class="math-inline">I_x\(i,j\) \\approx \\frac\{I\(i,j\+1\) \- I\(i,j\)\}\{\\Delta x\}</span>
            *   Second-order derivatives (e.g., Laplacian, Hessian components):
                <span class="math-inline">I_\{xx\}\(i,j\) \\approx \\frac\{I\(i,j\+1\) \- 2I\(i,j\) \+ I\(i,j\-1\)\}\{\(\\Delta x\)^2\}</span>
                The 2D Laplacian <span class="math-inline">\\nabla^2 I \= I_\{xx\} \+ I_\{yy\}</span> can be approximated using a 5-point stencil:
                <span class="math-inline">\\nabla^2 I\(i,j\) \\approx \\frac\{I\(i\+1,j\) \+ I\(i\-1,j\) \+ I\(i,j\+1\) \+ I\(i,j\-1\) \- 4I\(i,j\)\}\{h^2\}</span> (assuming <span class="math-inline">\\Delta x \= \\Delta y \= h</span>).
        *   **Temporal Derivatives:** For time-dependent PDEs like <span class="math-inline">\\frac\{\\partial I\}\{\\partial t\} \= \\mathcal\{L\}\(I\)</span> (where

<span class="math-inline">\\mathcal\{L\}</span> is a spatial operator), the time derivative is often approximated using a forward Euler scheme:
    <span class="math-block">\\frac\{I^\{n\+1\}\(i,j\) \- I^n\(i,j\)\}\{\\Delta t\} \\approx \\mathcal\{L\}\(I^n\(i,j\)\)</span>
    where <span class="math-inline">I^n\(i,j\)</span> is the intensity at pixel <span class="math-inline">\(i,j\)</span> at time step <span class="math-inline">n</span>, and <span class="math-inline">\\Delta t</span> is the time step size. This leads to an explicit update rule: <span class="math-inline">I^\{n\+1\}\(i,j\) \= I^n\(i,j\) \+ \\Delta t \\mathcal\{L\}\(I^n\(i,j\)\)</span>.
  *   **Stability Considerations:** Explicit schemes like forward Euler are easy to implement but have stability constraints. For diffusion-like equations, the time step <span class="math-inline">\\Delta t</span> must be chosen sufficiently small relative to the spatial grid spacing <span class="math-inline">\(\\Delta x, \\Delta y\)</span> to prevent numerical instability (e.g., the Courant-Friedrichs-Lewy or CFL condition). For the 2D heat equation discretized with forward Euler and central differences, stability typically requires <span class="math-inline">\\Delta t \\le \\frac\{h^2\}\{4D_\{max\}\}</span>, where <span class="math-inline">D_\{max\}</span> is the maximum diffusion coefficient. Implicit schemes (e.g., backward Euler or Crank-Nicolson) are often unconditionally stable but require solving a system of linear equations at each time step. Understanding these stability limits is crucial for obtaining meaningful results from simulations. Discretization choices can also introduce "discretization artifacts," as mentioned in the user query, where the discrete approximation does not perfectly represent the continuous model, potentially leading to orientation biases or inaccuracies.

B. Iterative Solvers for Linear Systems Arising from Discretization:
  When implicit FDM schemes are used for PDEs, or in certain optimization contexts related to segmentation, a large system of linear algebraic equations of the form <span class="math-inline">Ax\=b</span> must be solved at each time step or iteration. Here, <span class="math-inline">x</span> typically represents the unknown pixel values of the image at the new time step. For an image of size <span class="math-inline">M \\times N</span> (e.g., <span class="math-inline">30 \\times 1152 \= 34560</span> pixels), the matrix <span class="math-inline">A</span> can be very large (<span class="math-inline">34560 \\times 34560</span>). Direct methods for solving such systems (e.g., Gaussian elimination or LU decomposition, which compute <span class="math-inline">A^\{\-1\}b</span>) are computationally prohibitive both in terms of operations and memory, especially for manual simulation or basic implementations.
  Iterative solvers provide a more feasible alternative by starting with an initial guess

for <span class="math-inline">x</span> and successively refining it until a desired level of accuracy is reached.[38, 39, 40, 41, 42] The matrix <span class="math-inline">A</span> arising from FDM discretization of local operators (like the Laplacian) is typically sparse (most entries are zero) and often has a regular, banded structure, which iterative methods can exploit.
  1. Jacobi Method:
    The Jacobi method is a fundamental iterative technique.[40, 42]
    *   **Iterative Formula (element-wise):** For the <span class="math-inline">i</span>-th equation <span class="math-inline">\\sum_j a_\{ij\}x_j \= b_i</span>, the update rule for <span class="math-inline">x_i</span> at iteration <span class="math-inline">\(m\+1\)</span> is:
      <span class="math-block">x_i^\{\(m\+1\)\} \= \\frac\{1\}\{a_\{ii\}\} \\left\(b_i \- \\sum_\{k \\neq i\} a_\{ik\}x_k^\{\(m\)\} \\right\)</span>
      All components of <span class="math-inline">x^\{\(m\+1\)\}</span> are computed based entirely on the values from the previous iteration <span class="math-inline">x^\{\(m\)\}</span>.
    *   **Matrix Form:** Decompose <span class="math-inline">A</span> into its diagonal (<span class="math-inline">D</span>), strictly lower triangular (<span class="math-inline">L</span>), and strictly upper triangular (<span class="math-inline">U</span>) parts, such that <span class="math-inline">A \= D\+L\+U</span>. The Jacobi iteration can be written as:
      <span class="math-block">x^\{\(m\+1\)\} \= D^\{\-1\}\(b \- \(L\+U\)x^\{\(m\)\}\) \= \-D^\{\-1\}\(L\+U\)x^\{\(m\)\} \+ D^\{\-1\}b</span>
    *   **Characteristics:** Simple to implement. The computation of each component <span class="math-inline">x_i^\{\(m\+1\)\}</span> can be performed in parallel because they only depend on <span class="math-inline">x^\{\(m\)\}</span>. Convergence is guaranteed if the matrix <span class="math-inline">A</span> is strictly diagonally dominant (i.e., <span class="math-inline">\|a_\{ii\}\| \> \\sum_\{k \\neq i\} \|a_\{ik\}\|</span> for all <span class="math-inline">i</span>).[40] However, its convergence rate is generally slower than that of the Gauss-Seidel method.[42]
  2. Gauss-Seidel Method:
    The Gauss-Seidel method is an improvement over the Jacobi method that typically converges faster by using the most recently updated values of <span class="math-inline">x</span> within the same iteration.[39, 40, 42]
    *   **Iterative Formula (element-wise):**
      <span class="math-block">x_i^\{\(m\+1\)\} \= \\frac\{1\}\{a_\{ii\}\} \\left\(b_i \- \\sum_\{k < i\} a_\{ik\}x_k^\{\(m\+1\)\} \- \\sum_\{k \> i\} a_\{ik\}x_k^\{\(m\)\} \\right\)</span>
      When computing <span class="math-inline">x_i^\{\(m\+1\)\}</span>, the values

<span class="math-inline">x_k^\{\(m\+1\)\}</span> for <span class="math-inline">k < i</span> (already computed in the current iteration) are used, while values <span class="math-inline">x_k^\{\(m\)\}</span> for <span class="math-inline">k \> i</span> (from the previous iteration) are used.
    *   **Matrix Form:**
        <span class="math-block">x^\{\(m\+1\)\} \= \(D\+L\)^\{\-1\}\(b \- Ux^\{\(m\)\}\) \= \-\(D\+L\)^\{\-1\}Ux^\{\(m\)\} \+ \(D\+L\)^\{\-1\}b</span>
    *   **Characteristics:** Generally converges faster than the Jacobi method if it converges (often about twice as fast for problems like the discretized Poisson equation [42]). Updates are sequential, making parallelization less straightforward than Jacobi. Convergence is guaranteed if <span class="math-inline">A</span> is strictly diagonally dominant or if <span class="math-inline">A</span> is symmetric and positive definite.[39]
*   **Application Example: Discretized Poisson Equation:** The 2D Poisson equation <span class="math-inline">\\nabla^2 u \= f</span>, when discretized using the 5-point stencil, results in a linear system <span class="math-inline">Au\=b</span>. For example, at grid point <span class="math-inline">\(i,j\)</span>, the equation is often of the form <span class="math-inline">4u_\{i,j\} \- u_\{i\-1,j\} \- u_\{i\+1,j\} \- u_\{i,j\-1\} \- u_\{i,j\+1\} \= \-h^2 f_\{i,j\}</span> (assuming <span class="math-inline">u</span> represents the unknown potential and <span class="math-inline">f</span> is the source term).[41] This system has a sparse, structured matrix <span class="math-inline">A</span> that is often diagonally dominant (or can be made so), making it suitable for solution by Jacobi or Gauss-Seidel methods.

For manual calculation or simulation, iterative solvers are indispensable for handling the large linear systems arising from implicit PDE discretizations. While the number of iterations for convergence can still be substantial, understanding the update rule for a single iteration is straightforward. The choice between Jacobi and Gauss-Seidel often leans towards Gauss-Seidel for its faster convergence, though Jacobi's simpler parallelizable structure might be conceptually easier for initial understanding of the iterative process. The properties of the matrix <span class="math-inline">A</span>, derived from the specific PDE and boundary conditions, determine the convergence behavior of these iterative methods.

Table V.A: Overview of PDE Discretization and Solver Characteristics

PDE Type /	**Common**	**Resulting**	**Suitable**	**Key**

Method	**Discretization Scheme(s)**	**Linear System Type (if implicit)**	**Iterative Solvers**	**Stability/Convergence Factors for Solvers/Schemes**
Heat Equation (Smoothing)	Explicit FDM (Forward Euler, Central Space); Implicit FDM (Backward Euler, Crank-Nicolson)	Sparse, banded, often symmetric (for implicit)	Jacobi, Gauss-Seidel (for implicit)	Explicit: CFL condition ($\Delta t \le h2/(4D)$). Implicit: Often unconditionally stable. Solvers: Diagonal dominance or SPD of system matrix.
Perona-Malik Diffusion	Explicit or Semi-Implicit FDM	Non-linear system, or linear system if c is fixed per iteration	Jacobi, Gauss-Seidel (if linearized per step)	Stability of explicit scheme for diffusion part; careful handling of non-linearity in c. Regularization of c may be needed.[4]
Active Contours (Snakes)	FDM for Euler-Lagrange equations (often explicit time stepping)	Typically not a large linear system per step if explicit. May arise from implicit elasticity.	N/A for typical explicit evolution. If implicit, then Jacobi/GS.	Stability of explicit curve evolution (Δt), parameter choice (α,β,γ).
Level Set Methods	FDM for Level Set PDE (e.g., upwind for $\$	\nabla \phi\	, central for curvature)	Typically explicit time stepping.

VI. Advanced Considerations and Robustness

Achieving reliable defect detection and segmentation requires addressing the inherent variability of defects and the challenges posed by noise and computational artifacts. Robustness, in this context, is multifaceted, encompassing insensitivity to intensity variations, resilience to discretization errors, and the stability of numerical algorithms.

A. Handling Intensity Variations in Defects:
Defects (scratches and digs) may present as darker than, brighter than, or having variable intensity relative to the surrounding fiber optic end face material. This variability poses a challenge for methods relying on absolute intensity thresholds.
* **Gradient-based methods** (Sobel, Prewitt) are inherently sensitive to intensity *changes* rather than absolute levels. The magnitude of the gradient response depends on the *contrast* between the defect and its background. Thus, they can detect both dark and bright defects, provided sufficient contrast exists.
* **Transform-based line detectors** (Hough, Radon) typically operate on binary edge maps produced by an initial edge detection step. Their ability to handle intensity variations is therefore dependent on the robustness of the preceding edge detector to these variations.
* **Laplacian of Gaussian (LoG) and Determinant of Hessian (DoH)** methods can identify both dark blobs (local intensity minima, yielding positive LoG or specific Hessian signatures) and bright blobs (local intensity maxima, yielding negative LoG or corresponding Hessian signatures) by searching for extrema of both signs in their respective response maps.
* **Hessian eigenvalue analysis** characterizes local shape based on curvature patterns, which are less directly dependent on absolute intensity levels than on the relative intensity changes that define the shape.
* **PDE-based segmentation methods** like the Chan-Vese level set model are region-based rather than purely edge-based. They seek to partition the image into regions of statistically homogeneous intensity. This makes them potentially more robust to variations in defect intensity or texture, as long as the defect region is statistically distinguishable from the background.[35]

B. Addressing Discretization Artifacts and Noise in Calculations:
The translation of continuous mathematical models to a discrete pixel grid can introduce artifacts, and numerical computations are susceptible to noise.
* **Discretization Artifacts:**
    * **Orientation Bias:** Simple gradient operators like Sobel and Prewitt can exhibit orientation bias, responding more strongly to edges aligned with their kernel axes (horizontal, vertical, diagonal).
    * **Derivative Approximation Errors:** Finite difference approximations of derivatives are inherently inexact. Using higher-order finite difference schemes can

improve accuracy but at the cost of larger stencils and increased computational complexity.
        *   **Regularization via Smoothing:** Applying Gaussian smoothing prior to derivative computations (as is implicit in the LoG formulation, or can be done explicitly before applying Sobel/Prewitt) helps to regularize the image data, making derivative estimates more stable and less prone to noise and fine-scale discretization errors.[2]
    *   **Numerical Noise and Stability:**
        *   **Floating-Point Errors:** Iterative numerical solvers can accumulate floating-point precision errors over many iterations, although for well-conditioned systems, this is often manageable.
        *   **PDE Solver Stability:** As discussed in Section V.A, numerical solutions to PDEs (e.g., for anisotropic diffusion or level set evolution) are subject to stability constraints (e.g., CFL condition for explicit schemes). Violating these can lead to oscillatory or divergent solutions.
        *   **Model Regularization:** Some PDE models themselves require regularization to ensure well-posedness or desirable behavior. For instance, the original Perona-Malik equation can lead to backward diffusion in noisy regions; regularized versions (e.g., by convolving the gradient term with a Gaussian) address this.[4]

C. Combining Methods for Enhanced Detection and Segmentation:
    No single mathematical method is likely to be universally optimal for all defect types and image conditions. A more robust and effective approach often involves intelligently combining multiple techniques, leveraging their complementary strengths. This aligns with the advanced, PhD-level perspective sought by the user, as the true sophistication often lies in the synergistic application of diverse mathematical tools.
    *   **Sequential Processing Pipelines:** A common strategy is to form a pipeline:
        1. **Pre-processing:** Noise reduction and feature enhancement (e.g., anisotropic diffusion to smooth noise while preserving scratch/dig edges).
        2. **Candidate Detection:** Initial identification of potential defect locations (e.g., Sobel/Prewitt for general edges, LoG for blob-like regions, DoH for points of high curvature).
        3. **Feature-Specific Fitting/Analysis:** Refining candidates based on expected geometry (e.g., Hough/Radon transform on edge points to confirm linear scratches; Hessian eigenvalue analysis on blob candidates to confirm isotropic curvature and differentiate from linear segments).
        4. **Segmentation:** Delineating precise boundaries (e.g., morphological operations on binary maps from thresholded detector responses; active contours or level sets initialized from refined candidates).
    *   **Information Fusion:** Combining the outputs of different detectors. For

example, a region identified by both a strong LoG response and Hessian eigenvalues indicative of a blob is a more confident dig detection than a region identified by only one. Hessian eigenvalues can be used to filter LoG candidates, removing those that are more line-like than blob-like.[31]
    *   **Iterative Refinement:** Using the output of one method as a refined input or initialization for another. For instance, the approximate location and scale of a dig from LoG analysis can provide an excellent initialization for an active contour or level set evolution, leading to faster convergence and more accurate segmentation.[34, 35]

The successful combination of methods requires a deep understanding of the mathematical underpinnings of each technique, including their assumptions, limitations, and sensitivity to parameters. The goal is to create a processing chain where each step optimally prepares the data or refines the results for the subsequent stages, leading to a more robust and accurate final defect characterization.

VII. Concluding Remarks

This report has presented a detailed compendium of sophisticated, non-machine learning mathematical methodologies for the detection and segmentation of scratches and digs on fiber optic end faces, based on the analysis of raw pixel intensity data. The focus has remained strictly on the foundational mathematical principles, encompassing calculus, linear algebra, physics-based models, and matrix equations, with an emphasis on procedures amenable to manual calculation or simulation for enhanced understanding.

A diverse arsenal of techniques has been explored, ranging from fundamental image pre-processing methods like Gaussian smoothing, anisotropic diffusion, FFT-based filtering, and SVD for noise reduction, to specialized algorithms for defect detection. For linear scratches, gradient-based operators (Sobel, Prewitt) and transform-based approaches (Hough, Radon) have been detailed. For blob-like digs, Laplacian-based methods (Laplacian of Gaussian with scale-space analysis) and Hessian-based techniques (Determinant of Hessian, Hessian eigenvalue analysis) have been presented. Furthermore, methods for segmenting and delineating these defects, including mathematical morphology and PDE-based active contour and level set models, have been discussed. Numerical implementation aspects, particularly the discretization of PDEs and the use of iterative solvers like Jacobi and Gauss-Seidel for the resulting linear systems, have also been addressed.

Several inherent challenges and trade-offs emerge from this exploration:

- **Parameter Sensitivity:** Nearly all methods discussed require careful tuning of parameters (e.g., σ in Gaussian smoothing, K in anisotropic diffusion, accumulator resolution in Hough transform, number of singular values in SVD, energy weights in active contours, speed function terms in level sets). Optimal parameter selection is often data-dependent and can be a significant challenge without automated optimization, especially in a manual simulation context.

- **Computational Effort:** While the principles of many algorithms can be understood through manual step-throughs on small examples, their application to realistically sized images (even the 1152×30 grid considered) involves a substantial number of computations. Iterative PDE solvers, for instance, may require many iterations to converge, making full manual calculation impractical. The value for "manual calculation and simulation" lies more in understanding the algorithmic process and the effect of parameters rather than exhaustive computation on full-scale data.
- **Complexity vs. Robustness:** Generally, more sophisticated methods (e.g., anisotropic diffusion, Hessian eigenvalue analysis, level sets) offer greater robustness to noise and defect variability but come with increased mathematical and computational complexity. Simpler methods (e.g., Sobel operator, basic thresholding) are easier to implement and simulate but may be less effective for subtle defects or noisy images.
- **No Single Solution:** The diversity of defect types (scratches vs. digs, varying intensity, jaggedness) and image conditions suggests that no single mathematical tool is universally optimal. A robust solution strategy often involves an intelligent combination of complementary methods, forming a processing pipeline from pre-processing through detection to segmentation.

The mathematical methodologies detailed provide a strong foundation for understanding how defect features can be identified and characterized from first principles. While the "manual calculation" aspect highlights the underlying computational steps, practical application to large datasets would typically involve programming these algorithms. The true sophistication often lies not only in the complexity of individual methods but in the insightful combination and adaptation of these mathematical concepts to solve the specific challenges posed by the defect detection task robustly and accurately. Future advancements in purely mathematical (non-ML) approaches could involve more advanced variational PDE models or refined multi-scale geometric analyses, though these often push the boundaries of what is feasible for basic simulation environments.

Works cited

1. ima12_intensity_with_coords.csv
2. 6.1. Gaussian Convolutions and Derivatives — Image Processing ..., accessed June 12, 2025,

https://staff.fnwi.uva.nl/r.vandenboomgaard/ComputerVision/LectureNotes/IP/LocalStructure/GaussianDerivatives.html
3. Smoothing Filters - math NIST, accessed June 12, 2025, https://math.nist.gov/mcsd/savg/software/filters/smooth/index.html
4. Anisotropic diffusion - Wikipedia, accessed June 12, 2025, https://en.wikipedia.org/wiki/Anisotropic_diffusion
5. Perona–Malik equation - VisualPDE, accessed June 12, 2025, https://visualpde.com/nonlinear-physics/perona-malik.html
6. Fast Fourier Transform - NV5 Geospatial Software, accessed June 12, 2025, https://www.nv5geospatialsoftware.com/docs/FFTReduceBackgroundNoise.html
7. Fast Fourier Transform in Image Processing - GeeksforGeeks, accessed June 12, 2025, https://www.geeksforgeeks.org/fast-fourier-transform-in-image-processing/
8. Noise Reduction in Satellite Imagery Using Singular ... - Informatika, accessed June 12, 2025, https://informatika.stei.itb.ac.id/~rinaldi.munir/AljabarGeometri/2024-2025/Makalah/Makalah-IF2123-Algeo-2024%20(85).pdf
9. Image Denoising using Singular Value Decomposition (SVD) - GitHub, accessed June 12, 2025, https://github.com/kianmajl/Image_Denoising_using_SVD
10. How the Sobel Operator Works - Automatic Addison, accessed June 12, 2025, https://automaticaddison.com/how-the-sobel-operator-works/
11. Sobel operator – Knowledge and References - Taylor & Francis, accessed June 12, 2025, https://taylorandfrancis.com/knowledge/Engineering_and_technology/Artificial_intelligence/Sobel_operator/
12. Prewitt operator – Knowledge and References - Taylor & Francis, accessed June 12, 2025, https://taylorandfrancis.com/knowledge/Engineering_and_technology/Computer_science/Prewitt_operator/
13. Prewitt operator - Wikipedia, accessed June 12, 2025, https://en.wikipedia.org/wiki/Prewitt_operator
14. Line Detection Using Hough Transform, accessed June 12, 2025, https://www.cs.sfu.ca/~hamarneh/ecopy/compvis1999_hough.pdf
15. A Complete Guide on Hough Transform - Analytics Vidhya, accessed June 12, 2025, https://www.analyticsvidhya.com/blog/2022/06/a-complete-guide-on-hough-transform/
16. www.mathworks.com, accessed June 12, 2025, https://www.mathworks.com/help/images/detect-lines-using-the-radon-transform.html#:~:text=The%20Radon%20transform%20is%20closely,used%20to%20detect%20straight%20lines.
17. Detect Lines Using Radon Transform - MATLAB & ... - MathWorks, accessed June 12, 2025, https://www.mathworks.com/help/images/detect-lines-using-the-radon-transform.html

18. www.kaggle.com, accessed June 12, 2025, https://www.kaggle.com/code/mostafaeid/laplacian-of-gaussian-for-blob-detection#:~:text=The%20main%20idea%20of%20using,LoG%20gives%20relatively%20better%20results.
19. Blob Detection — skimage 0.25.2 documentation - Scikit-image, accessed June 12, 2025, https://scikit-image.org/docs/0.25.x/auto_examples/features_detection/plot_blob.html
20. Uncertainty Quantification for Scale-Space Blob Detection - PMC - PubMed Central, accessed June 12, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC11329558/
21. Blob detection - Wikipedia, accessed June 12, 2025, https://en.wikipedia.org/wiki/Blob_detection
22. Assignment 2: Scale-space blob detection (Python), accessed June 12, 2025, http://slazebni.cs.illinois.edu/spring18/assignment2_py.html
23. Edge, Corner, and Blob Detection - Chris Tralie, accessed June 12, 2025, https://www.ctralie.com/Teaching/EdgeCornerBlob/
24. dsp.stackexchange.com, accessed June 12, 2025, https://dsp.stackexchange.com/questions/10647/normalized-laplacian-of-gaussian#:~:text=in%20scale%2Dspace%20related%20processing,y22%CF%832.
25. Blob Detection using OpenCV, accessed June 12, 2025, https://opencv.org/blog/blob-detection-using-opencv/
26. Determinant of Hessian (DoH) - Hands-On Image Processing with ..., accessed June 12, 2025, https://www.oreilly.com/library/view/hands-on-image-processing/9781789343731/08a3a880-ffc8-4e6c-953e-9cad41e3685c.xhtml
27. Hessian matrix - Wikipedia, accessed June 12, 2025, https://en.wikipedia.org/wiki/Hessian_matrix
28. www.spiedigitallibrary.org, accessed June 12, 2025, https://www.spiedigitallibrary.org/conference-proceedings-of-spie/6514/65142K/A-novel-algorithm-for-polyp-detection-using-Eigen-decomposition-of/10.1117/12.709175.full#:~:text=Calculation%20of%20Hessian%20matrix%20on,detecting%20blob%2Dlike%20object%20automatically.
29. Chapter 2, accessed June 12, 2025, https://dspace.library.uu.nl/bitstream/1874/377/18/c2.pdf
30. Shape Decomposition using Modal Analysis - cs.Princeton, accessed June 12, 2025, https://www.cs.princeton.edu/courses/archive/spr11/cos598A/pdfs/Huang09.pdf
31. Eigenvectors of the Hessian matrix for a sheet, a tube, and a blob. A ..., accessed June 12, 2025, https://www.researchgate.net/figure/Eigenvectors-of-the-Hessian-matrix-for-a-sheet-a-tube-and-a-blob-A-sheet-has-one-large_fig1_221625269
32. Mathematical Morphology, accessed June 12, 2025, https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT3/node3.html

33. Mathematical morphology - Wikipedia, accessed June 12, 2025, https://en.wikipedia.org/wiki/Mathematical_morphology
34. Active contour model - Wikipedia, accessed June 12, 2025, https://en.wikipedia.org/wiki/Active_contour_model
35. activecontour - MathWorks, accessed June 12, 2025, https://la.mathworks.com/help/images/ref/activecontour.html
36. Level Set Method: an Explanation - Herve Lombaert, accessed June 12, 2025, https://shape.polymtl.ca/lombaert/levelset/
37. Level set segmentation with machine learning based velocity field - GitHub, accessed June 12, 2025, https://github.com/notmatthancock/level-set-machine-learning
38. Iterative Stencil Loops - Wikipedia, accessed June 12, 2025, https://en.wikipedia.org/wiki/Iterative_Stencil_Loops
39. Gauss-Seidel method | Numerical Analysis II Class Notes | Fiveable ..., accessed June 12, 2025, https://library.fiveable.me/numerical-analysis-ii/unit-8/gauss-seidel-method/study-guide/d5qvlUqGns87MIcj
40. Overview of slides 07, accessed June 12, 2025, https://www.pks.mpg.de/~haque/maynooth_MP468/2021s1_lectureslides/MP468_2021s1_lectureslides07a.pdf
41. Gauss-Seidel Iteration Method For Poisson Equation - Scribd, accessed June 12, 2025, https://www.scribd.com/document/333141038/Gauss-Seidel-Iteration-Method-for-Poisson-Equation
42. Ch 10 Elliptic Partial Differential Equations - INFN Torino, accessed June 12, 2025, https://www.to.infn.it/~mignone/Numerical_Algorithms/ch10_elliptic_pde.pdf