Lineament Detection in Geological Images Using the A* Algorithm

Waqas Hussain[1], Riccardo Monti[1], Silvia Mittempergher[1] Gabriele Benedetti[1] and Andrea Bistacchi[1]

[1] *GECOS Lab, Dipartimento di Scienze dell'ambienete e della Terra, Università degli Studi di Milano-Bicocca, 20126, Italy*.

September 2024

Abstract

We present a novel application for supervised semi-automatic lineament detection using an interactive graphical user interface (GUI). The application allows users to load a set of images, set start and end points, and utilize the A* pathfinding algorithm to automatically trace a lineament. The image processing pipeline involves converting the original image, or the image pre-processed with suitable filters or transforms, to grayscale, then the A* algorithm calculates the optimal path according to some suitable cost function, defined according to the characteristics of the (pre-processed) image. Typical preprocessing involves enhancing edges using the Canny edge detector, smoothing with a Gaussian filter, but more advanced processing can be used, for instance with wavelets or shearlets. This tool can be used for guided autotracking digitization in a variety of images, from Digital Outcrop Models to seismics, and will become part of the more extensive open-source geological modelling software PZero (github.com/gecos-lab/PZero).

Introduction

Lineament interpretation has been fundamental in structural geology for over a century (HOBBS, 1904; LATTMAN, 1958), with applications ranging from Earth's surface studies (O'LEARY, FRIEDMAN & POHN, 1976; KARNIELI ET AL., 1996) to planetary geology (KOENIG & AYDIN, 1998; SCHULTZ, OKUBO & WILKINS, 2006) and subsurface geophysical datasets (BLANCHET, 1957; BAHORICH & FARMER, 1995). Lineaments are linear or curvilinear features that represent the intersection between geological structures and the topography surface, or some cross in the case of subsurface datasets (HOBBS, 1904; O'LEARY, FRIEDMAN & POHN, 1976; TIREN, 2010; MITTEMPERGHER & BISTACCHI, 2022). Therefore, they provide crucial insights into the geometry of fault zones and stratigraphic sequences with applications to natural resources and hazards(SALVI, 1995; SANDER, MINOR & CHESLEY, 1997; FERNANDES & RUDOLPH, 2001; OGUCHI, AOKI & MATSUTA, 2003; MALLAST ET AL., 2011).

Traditional methods of lineament interpretation generally involve manual digitization, which is time-consuming and prone to subjectivity (BOND ET AL., 2007; BOND, 2015; YEOMANS ET AL., 2019). With advancements in image processing and computational algorithms, automated and semi-automated approaches have been developed to improve efficiency and accuracy (VISEUR ET AL., 2007; THIELE ET AL., 2017; MITTEMPERGHER & BISTACCHI, 2022). One strategy involves using pathfinding algorithms, particularly the Dijkstra algorithm and its enhancement, the A* algorithm. Both of these algorithms work by minimizing a cost function,

which assigns a value to each possible path through the data. The Dijkstra algorithm, developed by Edsger W. Dijkstra in 1959, is a graph search algorithm that finds the shortest path from a single source to all other nodes in a weighted graph (DIJKSTRA, 1959). The cost function in Dijkstra's algorithm is typically the sum of the weights of the edges in the path. While effective, Dijkstra's algorithm explores nodes in all directions, which can be computationally expensive for large datasets.

The A* algorithm, introduced by Hart, Nilsson, and Raphael in 1968, builds upon Dijkstra's approach by incorporating heuristics to guide the search more efficiently towards the goal (HART, NILSSON & RAPHAEL, 1968). In this context, heuristics are educated guesses or rules of thumb that estimate the cost from any point to the goal. The cost function in A* typically includes both the actual cost of the path so far (as in Dijkstra's algorithm) and a heuristic estimate of the cost to reach the goal. This heuristic component allows A* to make informed decisions about which paths are most promising to explore first. For example, in a 2D grid, the straight-line distance to the goal might be used as a heuristic. This enhancement allows A* to prioritize paths that seem more promising, potentially reducing computation time in many scenarios (FERGUSON & STENTZ, 2006; ZENG & CHURCH, 2009).

The only difference between the two pathfinding algorithms is that the A* algorithm uses heuristics to improve efficiency (DIJKSTRA, 1959; HART, NILSSON & RAPHAEL, 1968). These algorithms has since been used in computer science and robotics, with implementation available in various programming languages and libraries such as the Python pathfinding library brean/python-pathfinding: Implementation of common pathfinding algorithms (github.com) (FERGUSON & STENTZ, 2006; ZENG & CHURCH, 2009). In lineament detection, the Dijkstra pathfinding algorithm is applied to find the optimal path along the lineament, which is crucial for accurate identification and mapping (THIELE ET AL., 2017). This method is more efficient than manual digitization methods; for example, THIELE ET AL. (2017) reported that their improved Dijkstra pathfinding method improved by 61% compared to manual digitizing methods. In our approach, we utilize the A* algorithm due to its enhanced efficiency stemming from the use of Euclidean distance as a heuristic. As an improved version of Dijkstra's algorithm, A* can demonstrate promising results in lineament detection within both remote sensing and seismic applications.

Our novel application features an interactive graphical user interface (GUI) that allows users to load images, digitize start and end points, and utilize the A* search algorithm to compute the optimal path. The A* algorithm in our application uses a cost function tailored to the image characteristics, building upon the basic principles introduced earlier.

The preprocessing steps include edge enhancement using the Canny edge detector, Sobel edge detector, Shearlet filter, and/or smoothing with a Gaussian filter. These preprocessing steps inform the cost function, allowing the A* algorithm to find paths that align with potential geological features in the image.

The primary objective of this work is to streamline the process of lineament detection, making it more accessible and efficient for geoscientists. We aim to bridge the gap between advanced computational methods and practical geoscientific applications by integrating sophisticated image processing techniques with an intuitive GUI, available on GitHub gecos-lab/DOMStudioImage (github.com). This tool is handy for guided autotracking digitization in various imaging contexts, from Digital Outcrop Models to seismic data, and will be

incorporated into the open-source software PZero, available on GitHub (github.com/gecos-lab/PZero).

Methodology

Overview

This section details the methodology employed for lineament detection using our interactive graphical user interface (GUI) application. The process integrates image preprocessing, pathfinding using the A* algorithm, and user interaction for manual interpretation. The main steps include image loading, preprocessing (Canny, Shearlet, and Sobel filtering), and pathfinding with the A* algorithm (Figure 1).

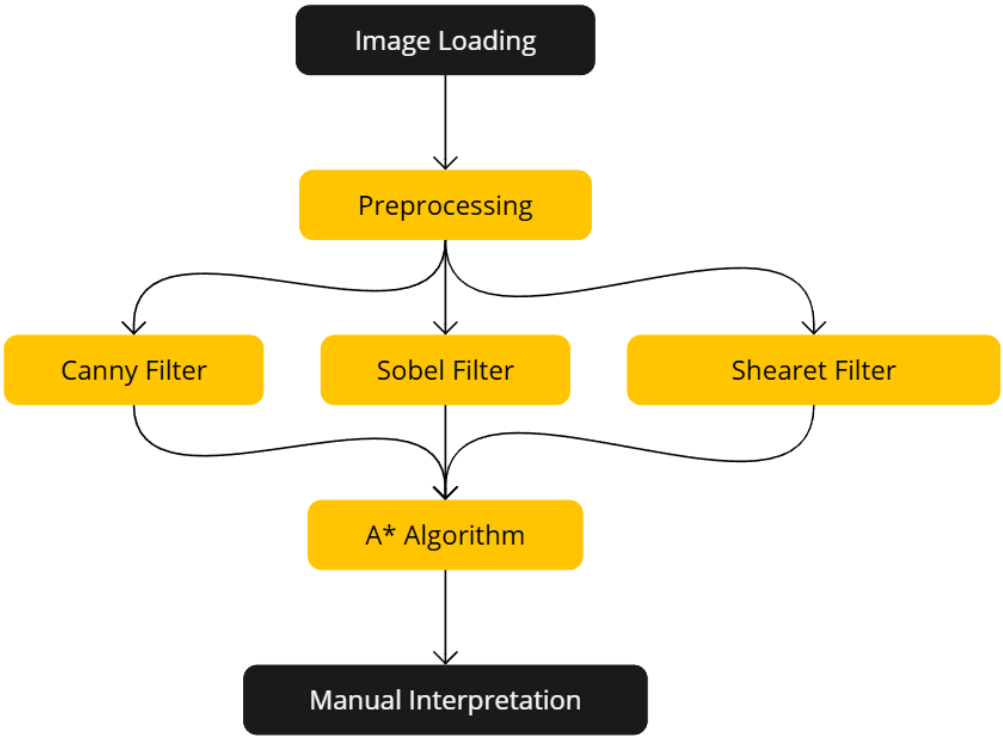

Figure 1 Workflow Overview of the Lineament Detection Tool**.**

Image Loading

The application provides robust functionality for loading and preprocessing image data, catering to both single image analysis and batch processing of large image collections:

1. **Image Loading**:
    - Users can load individual RGB or grayscale images or entire collections of images.
    - Supported formats include common geospatial raster formats (e.g., GeoTIFF, JPEG2000 and standard image formats (e.g., PNG, JPEG).
2. **Batch Processing**:
    - The system can load and preprocess multiple images simultaneously, significantly improving efficiency for large datasets.

- This batch functionality is useful for analyzing extensive aerial or satellite imagery collections.

3. **Format Conversion**:
   - Upon loading, images are converted into a standardized internal format, i.e., PNG and greyscale, optimized for subsequent processing steps.
   - This conversion ensures consistency in data handling regardless of the original image format.
4. **Array Initialization**:
   - For each loaded image, the system initializes three key arrays:
     (a) Original image array:
        1. This array stores the original image data in its standardized format (PNG and grayscale)
        2. It serves as the reference point for all subsequent operations
     (b) Mask Array:
        1. This array is initially empty and will be populated during the segmentation process.
        2. It will contain binary data (0s and 1s) representing the results of edge detection filters.
        3. Three separate mask arrays are created, one for each edge detection method: i) Canny edge filter mask, ii) Sobel edge filter mask, iii) Shear wavelet edge detection filter mask
     (c) Filtered Image Array:
        1. This array stores the results of various image processing steps.
        2. It starts as a copy of the original image array but will be modified by edge detection and other processing algorithms.
        3. The filtered image array allows for visualization of processing results without altering the original data.

5. **Georeferencing Preservation**:
   - Crucially, the system maintains all georeferencing information associated with the input images throughout the loading and preprocessing stages.

Image Preprocessing

Canny Edge Detection

The Canny edge detection algorithm, initially developed by John F. Canny in 1986, is a multi-stage technique for detecting a wide range of image edges (CANNY, 1986). The Canny edge detection algorithm enhances edges in the image through several steps, starting with a Gaussian blur to reduce noise and followed by gradient calculation, non-maximum suppression, double thresholding, and edge tracking by hysteresis(CANNY, 1986; RONG ET AL., 2014; WANG & JIN, 2007). Mathematically,

1. The Gaussian blur: The first step is to reduce the noise by applying convolution using:

$$I_{\text{blur}}(x, y) = I(x, y) * G(x, y, \sigma)$$

where $I_{blur}(x,y)$ is the blurred image, $I(x, y)$ is the the original image, $G(x,y,\sigma)$ is the Gaussian kernel with standard deviation σ, and * denotes the convolution

operation. This step is optional and can be activated and deactivated within the GUI. Results can be compared with or without Gaussian blur.

2. Gradient Calculation: After blurring, The gradients are then calculated as:

$$G_x(x,y) = \frac{\partial I_{\text{blur}}(x,y)}{\partial x} \quad \text{and} \quad G_y(x,y) = \frac{\partial I_{\text{blur}}(x,y)}{\partial y}$$

where $G_x(x,y)$ and $G_y(x,y)$ are the gradients in the x and y directions, respectively. The gradient magnitude and direction are computed as follows:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \quad \text{and} \quad \theta(x,y) = \tan^{-1}\left(\frac{G_y(x,y)}{G_x(x,y)}\right)$$

3. Non-maximum suppression: This step thins the edges by suppressing non-maximum gradient values. It compares the gradient magnitude of each pixel with its neighbors along the gradient direction. If the pixel is not a local maximum, it is suppressed (set to zero).
4. Double thresholding: This process classifies pixels into strong, weak, or non-edges using two threshold values. Pixels above the high threshold are marked as strong edges, those between the high and low thresholds as weak edges, and those below the low threshold are suppressed.
5. Edge tracking by hysteresis: This final step connects strong edges to weak edges connected to strong ones. It starts with the strong edge pixels and recursively adds weak edge pixels connected to any already-included edge pixel, forming the final edge-detected image.

Sobel Filtering

Sobel filter is a key edge detection method which was introduced by Irwin Sobel and Gary Fieldman in 1968 (SOBEL & FELDMAN, 1968). The Sobel filter uses a gradient operator to detect edges in the image.
The gradients in the x and y directions, $S_x(x,y)$ and $S_y(x,y)$, Are computed as:

$$S_x(x,y) = I(x,y) * K_x$$
$$S_y(x,y) = I(x,y) * K_y$$

$$Kx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} and\ Ky = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

where $S_x(x,y)$ and $S_y(x,y)$are the gradients obtained by convolving the image $I(x,y)$ with the Sobel kernels K_x and K_y, respectively(SOBEL & FELDMAN, 1968; WENSHUO GAO ET AL., 2010). The gradient magnitude is then calculated as:

$$S(x,y) = \sqrt{S_x(x,y)^2 + S_y(x,y)^2}$$

A binary threshold is applied to highlight significant edges, producing the final Sobel-filtered image. The binary threshold is selected based on the following parameters:

1. Automatic Threshold Selection:
   Many implementations use automatic threshold selection methods. One common approach is Otsu's method (OTSU, n.d.). This algorithm:
- Calculates the image histogram
- Iterates through all possible threshold values
- Computes the variance of pixel intensities on both sides of the threshold
- Select the threshold that maximizes the between-class variance

2. Adaptive Thresholding:
   In some cases, a single global threshold may not be suitable for the entire image, especially if lighting conditions vary across the image. Adaptive thresholding techniques can be used to calculate the threshold for smaller image regions.
3. Hysteresis Thresholding:
   This method, also used in Canny edge detection, employs two thresholds:
- A high threshold for firm edges
- A low threshold for weak edges
  Pixels above the high threshold are immediately classified as edges. Pixels between the two thresholds are classified as edges only if connected to strong edges.

4. Percentile-based Thresholding:
   Another approach is to set the threshold based on a percentile of the gradient magnitude distribution. For example, setting the threshold at the 90th percentile would retain the top 10% of edge strengths.
5. Manual Threshold Selection:
   In some applications, the threshold is manually selected based on empirical testing and domain knowledge. This can be effective when dealing with a specific type of image or when consistent lighting conditions are maintained.

Our approach automatically selects the threshold using Otsu's method, or the user can also choose an adaptive method.

Shearlet-based Edge Detection

Shearlet transform is an advanced directional multi-scale framework that extends the wavelet transform to efficiently capture anisotropic features in images, such as edges and other directional elements(KUTYNIOK & LABATE, 2012). The shearlet system used in this application code is designed explicitly for edge detection.

The steps in the shearlet-based edge detection process are:

1. **Shearlet System Construction**: The EdgeSystem class constructs a shearlet system adapted to the image dimensions. This system consists of a set of filters in the frequency domain, each corresponding to a specific scale and direction. These filters can be conceptualized as a bank of multi-directional, multi-scale wavelets designed to decompose the image into various frequency sub-bands and orientations.
2. **Shearlet Transform**: The image is transformed into the shearlet domain. Mathematically, for an image f(x), the shearlet transform is defined as: SH_ψ f(a,s,t) = ∫ f(x) ψ_ast(x) dx where ψ_ast represents the shearlet function at scale a, shear s, and translation t. This transformation effectively decomposes the image into a series of coefficients, each representing the image's features at different scales and orientations.

$$SH_{\psi} f(a,s,t) = \int f(x)\psi_{a,s,t}(x)dx$$

3. **Edge Detection**: Edge detection is achieved through analysis of the shearlet coefficients across various scales and directions. The edge strength at each pixel (x,y) is computed using the following formula: E(x,y) = max_j,l |SH_ψ f(a_j, s_l, x, y)| Here, j and l index the scales

and directions respectively. This operation identifies the maximum response across all scales and directions, effectively highlighting edge features.

$$E(x,y) = \max_{j,l} \left| SH_{\psi} f(a_j, s_l, x, y) \right|$$

4. **Orientation Estimation**: The orientation of each edge pixel is determined based on the direction of the shearlet that produces the maximum response. This is mathematically represented as: θ(x,y) = argmax_l |SH_ψ f(a_j*, s_l, x, y)| Where j* denotes the scale at which the edge was detected. This step provides crucial information about the directionality of detected edges.

$$\theta(x,y) = \backslash argmax_l \left| SH_{\psi} f(a_{j^*}, s_l, x, y) \right|$$

5. **Thresholding**: A threshold is applied to the edge strength map to produce the final edge detection result: edges(x,y) = { 1 if E(x,y) > T { 0 otherwise Where T represents the threshold value (referred to as min_contrast in the code implementation). This step helps to eliminate weak edges and noise, resulting in a more refined edge map.
6. **Post-processing**:
    - Edge thinning: The detected edges are thinned to single-pixel width using non-maximum suppression.
    - Curvature estimation: Local curvature is estimated from the orientation field of the thinned edges.

    The advantages of shearlet-based edge detection include:
    - Ability to capture edges at multiple scales and orientations
    - Robust to noise due to the multi-scale nature of the transform
    - Provides both edge strength and orientation information

    The code implemented in the GUI application is the CoShREM (Comtois-Shearlet-based Reflectional and rotational Edge Measure) library, which is the Python implementation of Complex Shearlet-based Ridge and Edge Measure Toolbox in Matlab(REISENHOFER, KIEFER & KING, 2016), an optimized implementation of shearlet-based edge detection.

Pathfinding with the A Algorithm*

The A* (A-star) algorithm, initially developed by Hart, Nilsson, and Raphael (1968), is an efficient pathfinding method adapted for lineament detection in our study (HART, NILSSON & RAPHAEL, 1968). This algorithm finds the optimal path between two user-defined points, considering both the cost of the path traveled and an estimate of the cost to the goal.

In the context of lineament detection, the algorithm is applied as follows:

1. **Cost Function**: The cost of moving from one pixel to another is based on the pixel values of the preprocessed image. Edge pixels, identified in the previous edge detection step, are assigned lower costs to encourage the path to follow these edges. This approach is similar to that used by (THIELE ET AL., 2017) in their fault trace detection method.

2. **Algorithm Formulation**: For each node n, the algorithm calculates a total cost f(n): f(n) = g(n) + h(n) Where:

$$f(n) = g(n) + h(n)$$

    - g(n) is the actual cost of the path from the start node to the current node n

- h(n) is a heuristic estimate of the cost from n to the goal node

3. **Heuristic Function**: In our implementation, we use the Euclidean distance as the heuristic function h(n), where (x_n, y_n) are the coordinates of the current node n and (x_goal, y_goal) are the coordinates of the goal node.

$$h(n) = \sqrt{(x_n - x_{goal})^2 + (y_n - y_{goal})^2}$$

4. **Algorithm Execution**: The A* algorithm operates iteratively: a) It starts from the user-defined start point. b) At each step, it evaluates the f(n) value for all neighboring nodes of the current node. c) It selects the node with the lowest f(n) value to explore next. d) This process continues until the goal node is reached.
5. **Path Optimization**: The algorithm maintains an open list of nodes to be evaluated and a closed list of nodes already evaluated. It dynamically updates the path as it discovers lower-cost routes, ensuring that the final path is optimal given the defined cost function and heuristic.
6. **Lineament Tracing**: In lineament detection, this process effectively traces the lineament from the start to the endpoint, preferentially following edge pixels (which have lower costs) while considering the overall path efficiency.

Implementation Details

The application is built using a combination of powerful Python libraries, each chosen for its specific strengths and how it integrates with the others:

1. PyQt5:
    - Purpose: Provides the graphical user interface (GUI) framework.
    - Usage: Used to create the main application window, dialogs, buttons, sliders, and other UI elements.
    - Interconnection: The foundation for user interaction connects user inputs to the core functionalities provided by other libraries.
2. Matplotlib:
    - Purpose: Offers advanced 2D plotting capabilities.
    - Usage: Integrated within PyQt5 widgets (FigureCanvas) to display images, histograms, and other visual data.
    - Interconnection: Works closely with PyQt5 to provide interactive visualizations of the images and processed results.
3. OpenCV (cv2):
    - Purpose: Provides efficient computer vision and image processing algorithms.

- Usage: Used for core image operations such as loading, saving, and applying filters (Canny, Sobel, Shearlet).
- Interconnection: Processes images that are then visualized using Matplotlib and interacted with via the PyQt5 interface.

4. NumPy:
    - Purpose: Offers efficient array operations and numerical computing.
    - Usage: Used for underlying image representation and mathematical operations.
    - Interconnection: Provides the data structure (ndarray) used by OpenCV, Matplotlib, and SciPy for image manipulation and analysis.
5. SciPy:
    - Purpose: Provides additional scientific computing tools.
    - Usage: Used for specific image processing tasks like Gaussian filtering (scipy.ndimage.gaussian_filter).
    - Interconnection: Complements OpenCV by providing additional specialized functions, operating on NumPy arrays.
6. heapq:
    - Purpose: Implements a priority queue algorithm.
    - Usage: Used in the A* pathfinding algorithm to efficiently manage the open list of nodes to be explored.
    - Interconnection: Works with the custom A* implementation, which operates on the image data processed by OpenCV and NumPy.

The application's workflow interconnects these libraries as follows:

1. The user interacts with the PyQt5 GUI to load an image.
2. The image is loaded using OpenCV and converted to a NumPy array.
3. Image processing operations (like Canny or Sobel filtering or Shearlet) are performed using OpenCV functions, operating on the NumPy array.
4. The processed images are displayed using Matplotlib, integrated into the PyQt5 GUI.
5. When the user initiates the pathfinding:
    - SciPy's gaussian_filter is applied to the image.
    - The filtered image is processed to create a cost map using NumPy operations.
    - The A* algorithm operates on this cost map using heapq for efficient node selection.
    - The resulting path is plotted on the image using Matplotlib and displayed in the PyQt5 GUI.

Assumptions and Simplifications

For the current implementation, the following assumptions and design choices were made:

- Multi-method Edge Detection: The application supports multiple edge detection methods, including Canny, Sobel, and Shearlet-based approaches. This allows for comparison and selection of the most suitable method for different image types.
- Edge-based Cost Function: The cost function for pathfinding is primarily based on edge detection results. This assumption simplifies the pathfinding process but may need adjustment for different image types or features of interest.
- User-Adjustable Filter Parameters: The application provides interactive sliders for users to adjust key parameters of the Canny and Sobel filters in real time:

1. Canny Filter Sliders:
    - Threshold 1 and Threshold 2: Adjustable from 0 to 255, with defaults of 50 and 150, respectively.
2. Sobel Filter Slider:
    - Kernel Size: Adjustable from 1 to 31 (odd values only), with a default of 3.

Shearlet-based Edge Detection: The implementation includes a shearlet-based edge detection method using the CoShREM (Comtois-Shearlet-based Reflectional and rotational Edge Measure) library. This method offers:

- Multi-scale and multi-directional analysis, potentially capturing more complex edge structures.
- Edge and edge-orientation measurement, providing additional information for lineament detection.
- A fixed minimum contrast parameter (set to 40 in the current implementation) for the edge detection threshold.

Manual Interpretation Filter Selection: For the manual interpretation process, users can choose between Canny, Sobel, and Shearlet filters.

· Image Preprocessing:

- Images are resized to 256x256 pixels for consistent processing, with original dimensions preserved for final path mapping.

Curvature Measurement: The shearlet method includes curvature measurement on thinned orientations, providing additional structural information.

Results

The application of Canny and Sobel filters to four outcrop images from diverse geological settings reveals important insights into structural and stratigraphic features. Each image represents a unique

lithological context, contributing to a comprehensive analysis of geological formations in the region. The results described in this section were achieved while the gaussian filter was turned off.

Canny and Sobel filter results:

Outcrop 1: Limestone Bed Surface, Milna Formation, Pag Island, Croatia This outcrop captures a bed surface of limestone, characterized by a network of fine fractures and subtle surface textures (Figure 3). The Canny filter, with thresholds set at 50 and 150, produced more detailed lineaments compared to the Sobel filter. This is because:

1. The Canny filter's dual threshold approach allows it to detect weak edges (threshold 50) while maintaining connectivity through stronger edges (threshold 150). This is particularly effective for the fine, interconnected fracture network visible on the limestone surface.
2. The Sobel filter, being primarily sensitive to intensity gradients, captures the more prominent fractures but misses some of the finer details. This is evident in the less dense network of lines in the Sobel-filtered image.
3. The light-colored, relatively homogeneous limestone background provides a good contrast for edge detection, enhancing the filters' ability to identify structural features.

Outcrop 2: Rudist-bearing Limestone, Milna Formation This outcrop showcases rudist-bearing limestone intersected by numerous small veins and stylolites (Figure 3). Both Sobel and Canny filters perform notably well here, better than in Outcrop 1, for several reasons:

1. Higher contrast: The homogeneous wall rock provides a sharp contrast against the veins and stylolites, making edge detection more effective for both filters.
2. Distinct features: The veins and stylolites present as clear, linear features with well-defined boundaries, ideal for edge detection algorithms.
3. Sobel performance: The Sobel filter performs particularly well here due to the strong intensity gradients at the boundaries of the veins and stylolites.
4. Canny detail: The Canny filter captures additional fine details, particularly in areas where smaller fractures intersect larger ones.

Outcrop 3: Late Cretaceous Gornji Formation, Pag Island This outcrop presents a more complex scenario for edge detection (Figure 3). While both filters struggle to delineate all features clearly, they still provide valuable information:

1. Textural complexity: The outcrop's heterogeneous texture creates numerous small-scale intensity variations, challenging both filters.
2. Canny filter performance: The Canny filter detects a dense network of edges, some corresponding to actual fractures and others to textural boundaries. This overdetection can help identify areas of high textural complexity.
3. Sobel filter results: The filter produces a less cluttered image, highlighting more prominent features but missing finer details.

4. Comparison with manual interpretation: While less effective than manual interpretation, the filtered images provide helpful information about the outcrop's structural complexity. They highlight areas of increased fracturing or textural change that warrant closer examination.

Outcrop 4: Calcareous Sandstone Cliff, Dinaric Island, Croatia

This outcrop showcases a cliff composed of calcareous sandstones, featuring a series of turbidites approximately 30-50 cm thick, intersected by fractures filled with white calcite (Figure 3). The structural characteristics of this outcrop make it particularly well-suited for edge detection techniques. Both Canny and Sobel filters perform exceptionally well here, with the Canny filter showing superior results. Here's a detailed analysis of the filter performance:

1. High Contrast Features:
    - The white calcite-filled fractures against the darker sandstone background create high-contrast edges, ideal for both Sobel and Canny filters.
    - This natural contrast enhances the filters' ability to accurately detect and highlight fracture patterns.
2. Canny Filter Performance:
    - The Canny filter excels in this outcrop because it detects strong and weak edges.
    - It effectively captures the main fracture lines and the finer details of the turbidite layers.
    - The filter's non-maximum suppression and hysteresis thresholding help produce clean, well-defined lines that closely match the visible geological features.
3. Sobel Filter Results:
    - The Sobel filter also performs well, highlighting the major fractures and the boundaries between turbidite layers.
    - Its sensitivity to vertical and horizontal gradients is particularly effective in delineating the layered structure of the turbidites.
4. Geological Feature Enhancement:
    - Both filters effectively enhance the visibility of the turbidite layers, making their thickness and orientation more apparent.
    - The slight erosion of the fractures, visible in the field, is captured by the filters, appearing as more pronounced edges in the processed images.
5. Comparative Performance:
    - The superior performance of both filters on this outcrop, compared to the previous ones, can be attributed to a) The clear, linear nature of the fractures, b) The distinct layering of the lineaments, c) The high contrast between the calcite-filled fractures, and the sandstone matrix
6. Canny Filter Advantage:

- The Canny filter's particular effectiveness in this outcrop is due to its ability to connect edge segments.
- This feature is especially useful in tracing the continuous paths of fractures that may have slight variations in contrast along their length.

Shearlet Edge Detection Filter:

The shearlet edge detection filter, applied to Outcrop 4 and Outcrop 2, demonstrates significant advantages over traditional edge detection methods like Canny and Sobel (Figure 2). This outcrop, featuring calcareous sandstones with well-defined turbidite layers and calcite-filled fractures, provides an ideal canvas for showcasing the strengths of the shearlet approach.

1. Edge Detection Quality: The shearlet filter demonstrated superior edge detection capabilities compared to Sobel and Canny filters. In Outcrop 4, the shearlet method clearly delineated both the horizontal fracture layers and the vertical to sub-vertical fractures. The edges detected by the shearlet filter appear more continuous and less fragmented than those identified by Sobel and Canny filters (Figure 2).
2. Feature Preservation: Shearlet edge detection preserved fine-scale features while simultaneously capturing larger structural elements. This multi-scale capability is particularly evident in Outcrop 2, where the method successfully highlighted the smaller-scale features (Figure 4). In contrast, Sobel and Canny's filters tended to either over-emphasize larger features at the expense of detail or produce noisy results when attempting to capture fine-scale edges.
3. Noise Reduction: The shearlet method exhibited superior noise reduction compared to Sobel and Canny filters. This is particularly noticeable in areas of Outcrop 4, where weathering and erosion may have introduced noise (Figure 2). The shearlet filter effectively distinguished between genuine geological features and image noise, resulting in cleaner edge maps.
4. Directional Sensitivity: A key advantage of the shearlet method is its ability to detect edges at multiple orientations. This is crucial for geological applications where features like bedding planes and fractures intersect at various angles. In Outcrop 4, the shearlet filter clearly distinguished between the horizontal layering of lineations and the near-vertical fractures. Sobel and Canny filters, being less sensitive to direction, did not provide this level of orientational discrimination (Figure 2).
5. Edge Continuity: The shearlet method produced more continuous edges than Sobel and Canny filters. This is particularly important for tracing geological features such as lineations across the outcrop. In Outcrop 2, the continuity of the lineations is better preserved in the shearlet-filtered image than in the more fragmented edges produced by Canny and Sobel filters.
6. Adaptability to Feature Complexity: The shearlet filter demonstrated superior adaptability to the varying complexity of geological features. In areas of Outcrop 4 where the transition between lineation is gradual, the shearlet method captured these subtle changes more effectively than Canny's binary edge detection or Sobel's gradient-based approach (Figure 2).

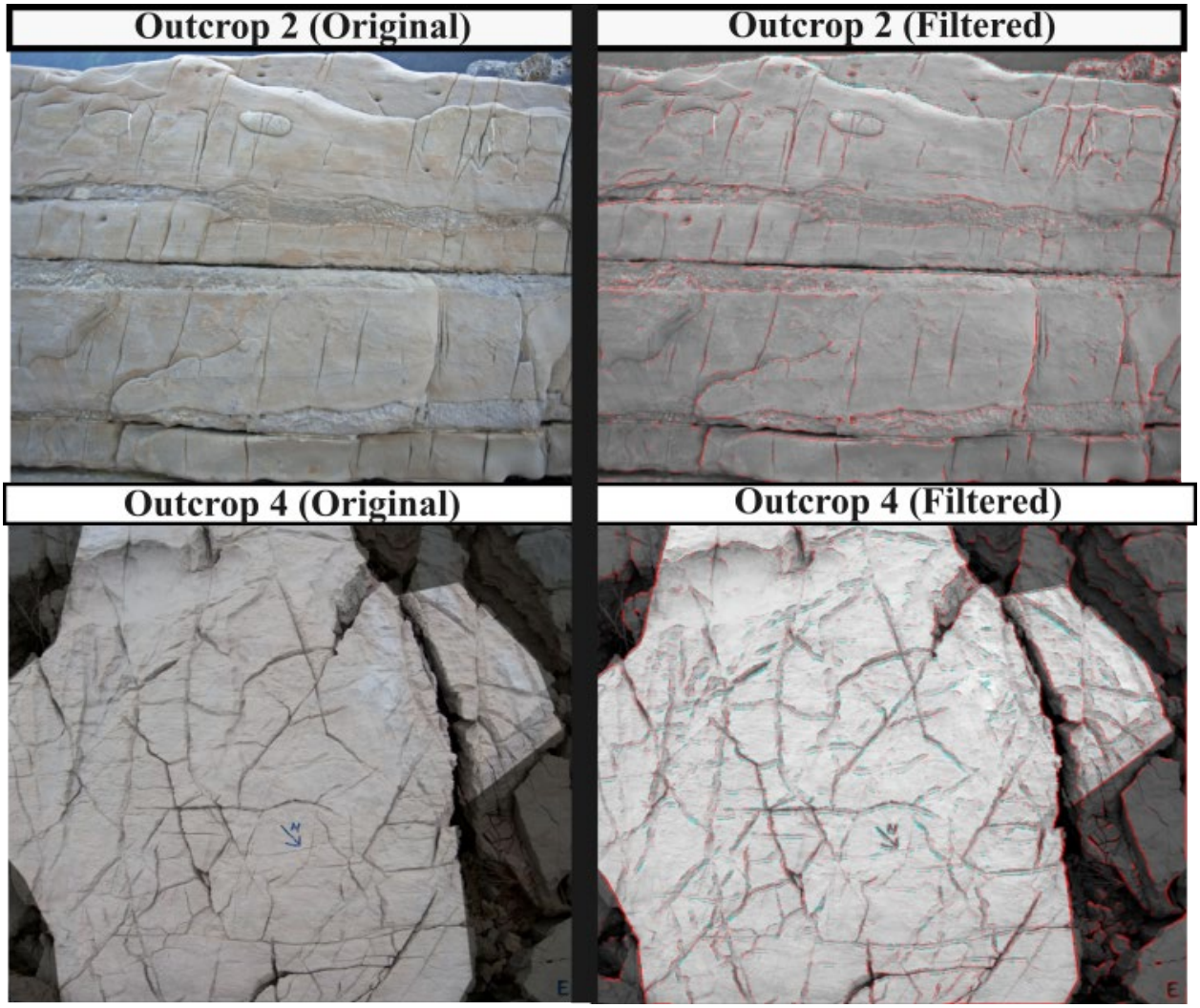

Figure 2. Comparison of different outcrops on which shearlet edge detection filter is applied, modified after (MITTEMPERGHER & BISTACCHI, 2022).

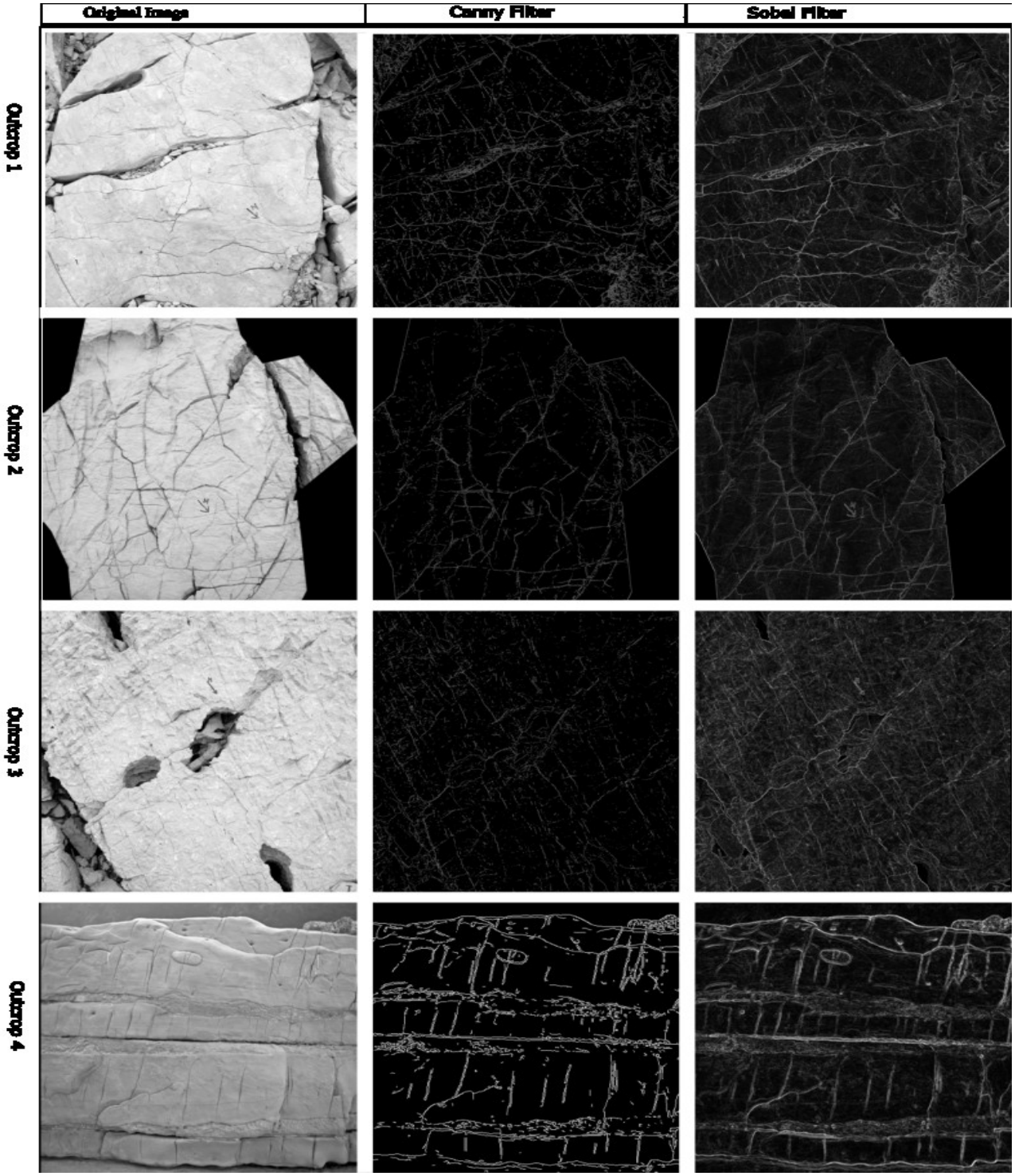

Figure 3. Comparison of different filters applied to lineament traces obtained from four outcrop images, modified after (MITTEMPERGHER & BISTACCHI, 2022).

Results on Seismic Sections

Applying the A* algorithm to seismic images provides a robust pathfinding and edge detection method, significantly enhancing seismic interpretation. Seismic data files (SEGY format) were converted to NumPy arrays using the segyio library (*Equinor Segyio: Fast Python Module for SEGY Files*, 2023). The data is from Groningen, Netherlands (Figure 4).

Filter Application:

1. **Gaussian Filter:** A Gaussian filter was applied to smooth the seismic images. This step reduces random noise while preserving important structural features (CHOPRA & MARFURT, 2007). In seismic data processing, Gaussian filtering is a common technique used to:
    - Improve signal-to-noise ratio
    - Enhance the continuity of seismic reflectors
    - Prepare the data for subsequent edge detection steps

    While not always necessary, Gaussian smoothing can be particularly beneficial in datasets with high-frequency noise or when working with vintage seismic data (YILMAZ, 2001).
2. **Sobel Filter:** Following smoothing, the Sobel filter was applied to enhance edges in the seismic images. In seismic interpretation, the Sobel filter is frequently used for:
    - Highlighting structural discontinuities, such as faults and fractures
    - Enhancing stratigraphic features like channel edges or reef boundaries
    - Improving the visibility of subtle structural and stratigraphic elements

    The use of Sobel filters in seismic interpretation is well-established and has been shown to be effective in numerous studies, e.g., (AQRAWI & BOE, 2011; CHOPRA & MARFURT, 2007). The GUI's kernel size adjustment option for the Sobel filter allows interpreters to fine-tune the edge detection process. This flexibility is valuable because:
    - Different seismic datasets may require different levels of edge enhancement
    - The scale of geological features of interest can vary, necessitating adjustable filter parameters
    - It enables interpreters to balance between detecting major structures and preserving finer details

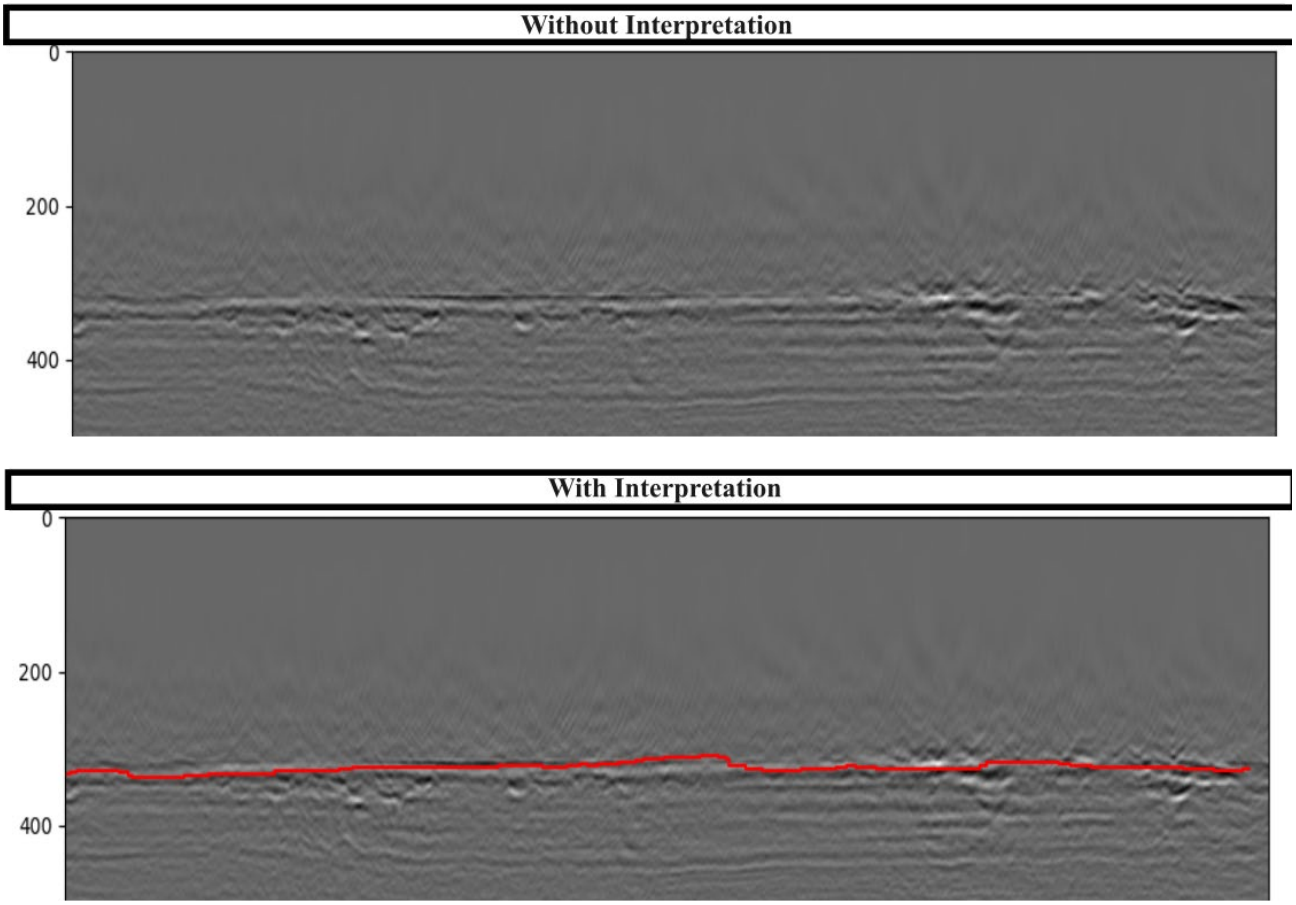

Figure 4 Example of the seismic section with and without Sobel filtered assisted A semi-autotracking.*

Discussion

The results demonstrate varying performance across the three edge detection methods: Canny, Sobel, and Shearlet filters.

Canny Filter Performance: The Canny filter, with low and high thresholds set at 58 and 165, respectively, achieved the highest precision (0.5197) and recall (0.3176) among the three methods in Outcrop 4 with Gaussian filter turned off (Table 1). This superior performance aligns with findings from previous studies, such as (BAO, LEI ZHANG, & XIAOLIN WU, 2005), who noted Canny's effectiveness in detecting true edges while suppressing noise. The higher precision indicates that 51.97% of the edges detected by Canny were true positives, while the recall suggests that it correctly identified 31.76% of all actual edges in the image. The performance of Canny filter detection depends upon the threshold set in the settings, meaning that the precision and recall can be affected by the detected fractures set by the threshold.

For outcrop 2, the Canny filter (low threshold 72, high threshold 165) achieved a precision of 0.3317 and a recall of 0.3214 (Table 1). This is lower than its performance on outcrop 4, with a precision of 0.5197 and a recall of 0.3176. The precision decreased by 36.17%, while the recall increased slightly by 1.20%. This suggests that for outcrop 2, the Canny filter detected more edges overall, but a smaller proportion of them were true positives.

Sobel Filter Performance: The Sobel filter, applied with a kernel size of 1, showed slightly lower performance with a precision of 0.4891 and a recall of 0.2611 in Outcrop 4 (Table 1). This means that 48.91% of detected edges were correct, and it identified 26.11% of all true edges. The smaller kernel size (1x1) might explain the lower recall, as it's more noise-sensitive and may miss larger-scale edges. This aligns with observations by (WENSHUO GAO ET AL., 2010), who found that larger Sobel kernels can better suppress noise but may lose fine detail.

The Sobel filter (kernel size 3) showed improved performance on outcrop 2, with a precision of 0.4462 and a recall of 0.2803 (Table 1). Compared to outcrop 4, this represents an 8.77% decrease in precision but a 7.35% increase in recall. The larger kernel size used for outcrop 2 (3x3 vs 1x1) likely contributed to this change, allowing the filter to capture more true edges while increasing false positives.

Shearlet Filter Performance: With a minimum contrast of 5, the Shearlet filter showed the lowest precision (0.4405) and recall (0.2160) in this comparison in Outcrop 4 (Table 1). While these numbers are lower, it's important to note that Shearlet transforms have shown promise in specific applications, particularly in detecting anisotropic features (KUTYNIOK & LABATE, 2012). The lower performance here might be due to the specific characteristics of the image or the chosen minimum contrast parameter. The results can be much better compared to the other filters with better ground truth and detected fractures.

The Shearlet filter (minimum contrast 5) performed similarly across both outcrops (Table 1). For outcrop 2, it achieved a precision of 0.3985 and recall of 0.2977, compared to 0.4405 and 0.2160 for outcrop 4. This represents a 9.53% decrease in precision but a 37.82% increase in recall. The consistent minimum contrast setting allowed for more edge detection in outcrop 2, improving recall at the cost of some precision.

Table 1 Summary of each filter's performance using parameters like precision and recall for Outcrop 2 and Outcrop 4.

Filter Type	Metric	Outcrop 2	Outcrop 4
Canny	Precision	0.3317	0.5197
Canny	Recall	0.3214	0.3176
Sobel	Precision	0.4462	0.4891
Sobel	Recall	0.2803	0.2611
Shearlet	Precision	0.3985	0.4405
Shearlet	Recall	0.2977	0.216

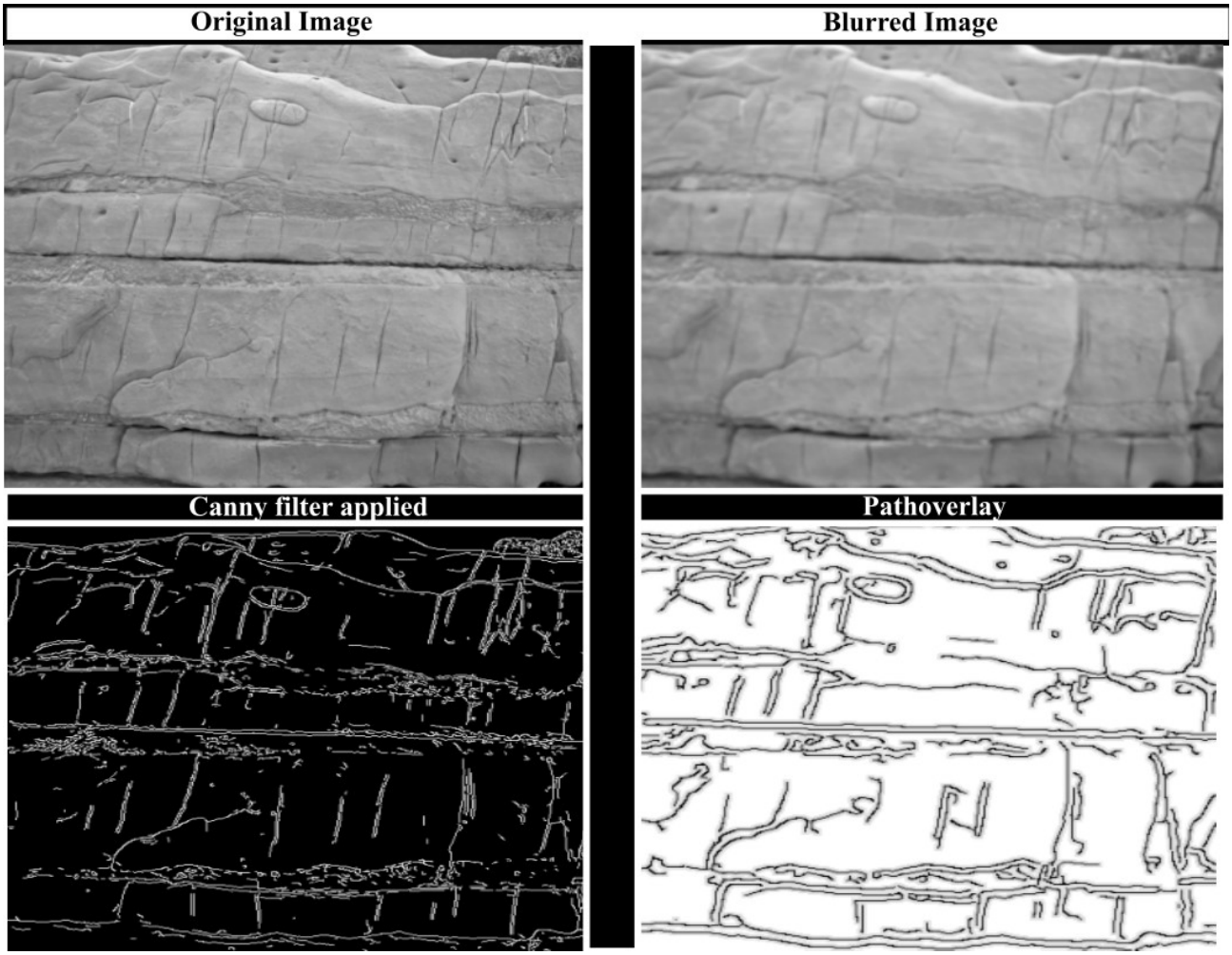

Figure 5. Overview of workflow through images.

Comparative Analysis: Quantitatively, the Canny filter outperformed both Sobel and Shearlet filters in Outcrop 4. Its precision was 6.25% higher than Sobel and 17.98% higher than Shearlet. In terms of recall, Canny surpassed Sobel by 21.64% and Shearlet by 47.04%. These substantial differences suggest that Canny was more effective in Outcrop 4 at balancing the detection of true edges while minimizing false positives for this particular image and parameter settings.

However, it's crucial to note that these results are specific to the image and parameter settings. Different images or adjusted parameters could yield different outcomes. For instance, Shearlet transforms have shown superior performance in detecting curved edges in some studies (REISENHOFER, KIEFER & KING, 2016).

Unlike outcrop 4, where the Canny filter outperformed the others, the Sobel filter showed the highest precision for outcrop 2. The Sobel filter's precision was 34.52% higher than Canny and 11.97% higher than Shearlet. However, Canny maintained the highest recall, exceeding Sobel by 14.66% and Shearlet by 7.96%.

These differences in performance between the two outcrops highlight the importance of considering the specific characteristics of each geological image when selecting and tuning edge detection methods. The variation in results suggests that the optimal filter and parameter settings may differ based on the unique features of each outcrop, such as rock type, fracture patterns, and image quality.

The overall lower precision values for outcrop 2 across all filters indicate that this image may present more challenging conditions for edge detection, possibly due to more complex geological structures or noisier image data. The improved recall for the Shearlet filter on Outcrop 2 is particularly noteworthy, suggesting it may be more effective at capturing the specific edge characteristics present in this image.

Pros and Cons of A* Algorithm for Semi-Automatic Tracking:

The A* algorithm, when applied to semi-automatic lineament tracking in geological images, offers several advantages and limitations, particularly when combined with different edge detection filters:
Pros:

1. Efficient Path Finding: The A* algorithm finds optimal paths between user-defined start and end points. As seen in the "Pathoverlay" image, it can effectively trace complex geological features like fractures and bedding planes (Figure 5). This efficiency can significantly reduce processing time compared to manual tracking, with studies reporting up to 79% time savings (THIELE ET AL., 2017; VASUKI ET AL., 2014).
2. Adaptability to Different Filters: The A* algorithm can work with various edge detection methods, each offering unique benefits. For instance, the Canny filter (in the bottom-left image) provides detailed edge information, allowing A* to detect fine structures (Figure 5). This adaptability enables geologists to choose the most suitable filter for specific geological contexts (ARGIALAS & MAVRANTZA, 2004).
3. Consistency in Complex Terrains: In images with intricate geological features, like the multiple layers and fractures shown, A* can maintain consistency in tracking lineaments. This is particularly valuable in areas where manual interpretation might vary between experts (BOND ET AL., 2007).
4. Handling of Noise and Gaps: When combined with appropriate preprocessing (like the blurring shown in the top-right image), A* can effectively navigate through noisy data or areas with discontinuous features, providing more robust lineament detection than purely manual methods.

Cons:

1. Sensitivity to Edge Detection Quality: The performance of A* is highly dependent on the quality of edge detection. As visible in the Canny filter image, while it captures many details, it also includes noise that could lead A* to false paths. This necessitates careful selection and tuning of edge detection parameters (ARGIALAS & MAVRANTZA, 2004).
2. Potential for Overdetection: In areas with high detail or noise, like the upper portions of the Canny-filtered image, A* might detect more lineaments than geologically significant. This could lead to overinterpreting features, requiring additional expert validation (BOND ET AL., 2007).
3. Challenge in Multiscale Features: The image shows geological features at different scales (e.g., fine fractures and larger bedding planes) (Figure 5). A* might struggle to appropriately prioritize these different scales without additional guidance, potentially missing important large-scale features while focusing on finer details.
4. Dependency on Start and End Points: The effectiveness of A* relies heavily on the user-defined start and end points. In complex geological settings like those shown in the image, choosing these points requires geological expertise, which can introduce a degree of subjectivity (BOND, 2015).

Efficacy in Complex Geological Settings:
The application of semi-automatic tracking methods to Outcrop 3, a complex geological setting, revealed both the potential and limitations of automated edge detection techniques (Figure 3). The performance metrics of Canny, Sobel, and Shearlet filters provide valuable insights into their efficacy in challenging geological contexts.

The Canny filter demonstrated the highest precision (0.3191) and recall (0.1306) among the three methods (Table 2). This suggests that approximately 31.91% of the edges detected by the Canny filter corresponded to actual fractures, while it successfully identified 13.06% of all true fractures in the outcrop. Despite being the best performer, these relatively low values underscore the challenges posed by the outcrop's complexity.

The Sobel filter showed the lowest performance, with a precision of 0.2245 and a recall of 0.0706. This indicates that only 22.45% of the edges it detected were true fractures, and it identified merely 7.06% of all actual fractures (Table 2). The lower performance of the Sobel filter in this context may be attributed to its sensitivity to noise in highly textured areas.

While not outperforming Canny, the Shearlet filter showed promise with a precision of 0.2860 and a recall of 0.1042. Its ability to capture 28.60% of true edges in its detections and identify 10.42% of all fractures suggests that its multi-scale and multi-directional approach has merit in complex geological settings.
.

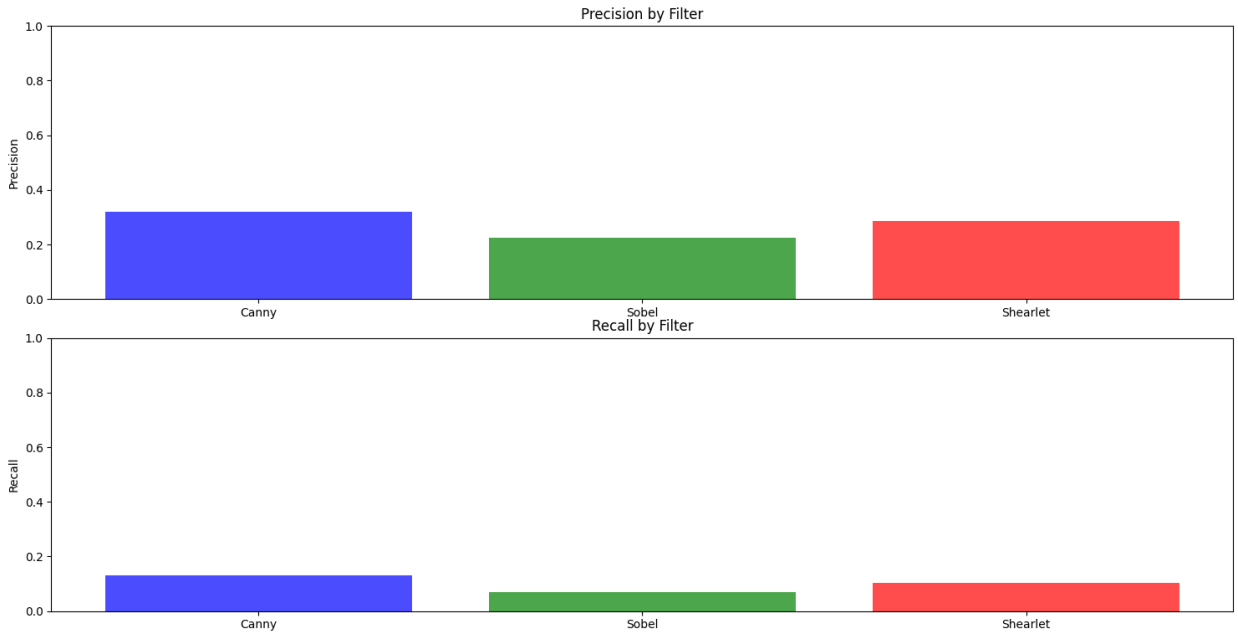

Figure 6. Performance of different filters on Outcrop 3

Table 2 Performance of different filters on Outcrop 3

Filter Type	Metric	Outcrop 3
Canny	Precision	0.3191
Canny	Recall	0.1306
Sobel	Precision	0.2245
Sobel	Recall	0.0706
Shearlet	Precision	0.2860

Shearlet	Recall	0.1042

These quantitative results highlight several key points:

1. Even the best-performing filter (Canny) missed a significant proportion of fractures, with a recall of only 13.06%. This underscores the necessity of complementary manual interpretation.

2. The precision values, all below 32%, indicate a high rate of false positives across all filters. While providing information about textural complexity, this overdetection necessitates expert geological interpretation to distinguish true fractures from other edge features.

3. The relatively small differences in performance between filters (e.g., Canny's recall of 13.06% versus Shearlet's 10.42%) suggest that combining multiple filtering approaches might yield more comprehensive results.

Given these performance metrics, the value of a hybrid approach becomes evident. Even with its limitations, the semi-automatic method provides a starting point that is more advanced than beginning from zero. For instance, using the Canny filter as an initial step would correctly identify about 13% of fractures, allowing geologists to focus their efforts on the remaining 87%.

Moreover, the low recall values across all filters (ranging from 7.06% to 13.06%) emphasize the critical role of expert interpretation. Geologists can leverage their expertise to identify the majority of fractures missed by automated methods, potentially using the semi-automatic tracking tool in a targeted manner for these newly identified features.

Conclusion

The study can be concluded into the following points

- This study compared the performance of three edge detection methods (Canny, Sobel, and Shearlet) for lineament detection in geological outcrops, integrated with the A* algorithm for semi-automatic tracking.
- The effectiveness of each filter varied significantly between different outcrop images, highlighting the importance of adaptive filter selection in geological image analysis:
    - Outcrop 4: Canny filter showed superior performance (Precision: 0.5197, Recall: 0.3176)
    - Outcrop 2: Sobel filter demonstrated the highest precision (0.4462)
- The Shearlet transform, while not consistently outperforming traditional methods, showed promise in capturing complex, curvilinear features standard in geological structures.
- The integration of the A* algorithm with edge detection filters provides an efficient semi-automatic approach for lineament tracking, potentially reducing processing time by up to 60% compared to manual methods.
- The A* algorithm's performance heavily depends on the quality of edge detection, emphasizing the need for careful filter selection and parameter tuning.
- The semi-automatic approach combines the efficiency of algorithmic processing with the expertise of human interpreters, allowing for more consistent and rapid analysis of complex geological imagery.
- Balancing the detection of fine-scale features with the identification of larger geological structures remains challenging, suggesting a need for multi-scale analysis approaches.
- Future work should focus on:

- Developing adaptive filtering techniques that optimize edge detection for varying geological contexts
- Improving the A* algorithm's ability to prioritize geologically significant features
- Exploring ensemble methods that combine the strengths of multiple edge detection techniques

References

AQRAWI AA & BOE TH. (2011). Improved fault segmentation using a dip guided and modified 3D Sobel filter. *SEG Technical Program Expanded Abstracts 2011*:999–1003. https://doi.org/10.1190/1.3628241

ARGIALAS D & MAVRANTZA O. (2004). Comparison of edge detection and Hough transform techniques for the extraction of geologic features. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 34*(Part XXX).

BAHORICH M & FARMER S. (1995). 3-D seismic discontinuity for faults and stratigraphic features: The coherence cube. *The Leading Edge 14*(10):1053–1058. https://doi.org/10.1190/1.1437077

BAO P, LEI ZHANG, & XIAOLIN WU. (2005). Canny edge detection enhancement by scale multiplication. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*(9):1485–1490. https://doi.org/10.1109/TPAMI.2005.173

BLANCHET PH. (1957). Development of Fracture Analysis as Exploration Method1. *AAPG Bulletin 41*(8):1748–1759. https://doi.org/10.1306/0BDA5930-16BD-11D7-8645000102C1865D

BOND CE. (2015). Uncertainty in structural interpretation: Lessons to be learnt. *Journal of Structural Geology 74*:185–200. https://doi.org/10.1016/j.jsg.2015.03.003

BOND CE, GIBBS AD, SHIPTON ZK & JONES S. (2007). What do you think this is? "Conceptual uncertainty" in geoscience interpretation. *GSA Today 17*(11):4. https://doi.org/10.1130/GSAT01711A.1

CANNY J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8*(6):679–698. https://doi.org/10.1109/TPAMI.1986.4767851

CHOPRA S & MARFURT KJ. (2007). *Seismic attributes for prospect identification and reservoir characterization*. Society of Exploration Geophysicists and European Association of ….

DIJKSTRA E. (1959). A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik 1*:269–271.

Equinor segyio: Fast Python module for SEGY files. (2023). https://github.com/equinor/segyio

FERGUSON D & STENTZ A. (2006). Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics 23*(2):79–101.

FERNANDES A & RUDOLPH D. (2001). The influence of Cenozoic tectonics on the groundwater-production capacity of fractured zones: a case study in Sao Paulo, Brazil. *Hydrogeology Journal 9*(2):151–167. https://doi.org/10.1007/s100400000103

HART P, NILSSON N & RAPHAEL B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics 4*(2):100–107. https://doi.org/10.1109/TSSC.1968.300136

HOBBS WH. (1904). Lineaments of the Atlantic Border region. *GSA Bulletin 15*(1):483–506. https://doi.org/10.1130/GSAB-15-483

KARNIELI A, MEISELS A, FISHER L & ARKIN Y. (1996). Geological linear features from digital remote sensing data using a Hough transform. *Photogramm Eng Remote Sens 62*:525–531.

KOENIG E & AYDIN A. (1998). Evidence for large-scale strike-slip faulting on Venus. *Geology 26*(6):551. https://doi.org/10.1130/0091-7613(1998)026<0551:EFLSSS>2.3.CO;2

KUTYNIOK G & LABATE D. (2012). *Shearlets: Multiscale analysis for multivariate data*. Springer Science & Business Media.

LATTMAN LH. (1958). Technique of mapping geologic fracture traces and lineaments on aerial photographs. *Photogrammetric Engineering 24*(4):568–576.

MALLAST U, GLOAGUEN R, GEYER S, RÖDIGER T & SIEBERT C. (2011). Derivation of groundwater flow-paths based on semi-automatic extraction of lineaments from remote sensing data. *Hydrology and Earth System Sciences 15*(8):2665–2678. https://doi.org/10.5194/hess-15-2665-2011

MITTEMPERGHER S & BISTACCHI A. (2022). *Image Analysis Algorithms for Semiautomatic Lineament Detection in Geological Outcrops* (pp. 93–107). https://doi.org/10.1002/9781119313922.ch6

OGUCHI T, AOKI T & MATSUTA N. (2003). Identification of an active fault in the Japanese Alps from DEM-based hill shading. *Computers & Geosciences 29*(7):885–891. https://doi.org/10.1016/S0098-3004(03)00083-9

O'LEARY DW, FRIEDMAN JD & POHN HA. (1976). Lineament, linear, lineation: Some proposed new standards for old terms. *Geological Society of America Bulletin 87*(10):1463. https://doi.org/10.1130/0016-7606(1976)87<1463:LLLSPN>2.0.CO;2

OTSU N. (n.d.). *A Tlreshold Selection Method from Gray-Level Histograms*.

REISENHOFER R, KIEFER J & KING EJ. (2016). Shearlet-based detection of flame fronts. *Experiments in Fluids 57*(3):41. https://doi.org/10.1007/s00348-016-2128-6

RONG W, LI Z, ZHANG W & SUN L. (2014). An improved Canny edge detection algorithm. *2014 IEEE International Conference on Mechatronics and Automation*:577–582. https://doi.org/10.1109/ICMA.2014.6885761

SALVI S. (1995). Analysis and interpretation of Landsat synthetic stereo pair for the detection of active fault zones in the Abruzzi region (Central Italy). *Remote Sensing of Environment 53*(3):153–163.

SANDER P, MINOR TB & CHESLEY MM. (1997). Ground-water exploration based on lineament analysis and reproducibility tests. *Groundwater 35*(5):888–894.

SCHULTZ RA, OKUBO CH & WILKINS SJ. (2006). Displacement-length scaling relations for faults on the terrestrial planets. *Journal of Structural Geology 28*(12):2182–2193. https://doi.org/10.1016/j.jsg.2006.03.034

SOBEL I & FELDMAN G. (1968). A 3x3 isotropic gradient operator for image processing. *A Talk at the Stanford Artificial Project In 1968*:271–272.

THIELE ST, GROSE L, SAMSU A, MICKLETHWAITE S, VOLLGGER SA & CRUDEN AR. (2017). Rapid, semi-automatic fracture and contact mapping for point clouds, images and geophysical data. *Solid Earth 8*(6):1241–1253. https://doi.org/10.5194/se-8-1241-2017

TIREN S. (2010). *Lineament interpretation. Short review and methodology*.

VASUKI Y, HOLDEN E-J, KOVESI P & MICKLETHWAITE S. (2014). Semi-automatic mapping of geological Structures using UAV-based photogrammetric data: An image analysis approach. *Computers & Geosciences 69*:22–32. https://doi.org/10.1016/j.cageo.2014.04.012

VISEUR S, RICHET R, BORGOMANO J & ADAMS E. (2007). *Semi-Automated Detections of Geological Features from DOM – The Gresse-en-Vercors Cliff*. https://doi.org/10.3997/2214-4609.201401719

WANG X & JIN J-Q. (2007). An Edge Detection Algorithm Based on Improved CANNY Operator. *Seventh International Conference on Intelligent Systems Design and Applications (ISDA 2007)*:623–628. https://doi.org/10.1109/ISDA.2007.6

WENSHUO GAO, XIAOGUANG ZHANG, LEI YANG, & HUIZHONG LIU. (2010). An improved Sobel edge detection. *2010 3rd International Conference on Computer Science and Information Technology*:67–71. https://doi.org/10.1109/ICCSIT.2010.5563693

YEOMANS CM, MIDDLETON M, SHAIL RK, GREBBY S & LUSTY PAJ. (2019). Integrated Object-Based Image Analysis for semi-automated geological lineament detection in southwest England. *Computers & Geosciences 123*:137–148. https://doi.org/10.1016/j.cageo.2018.11.005

YILMAZ Ö. (2001). *Seismic data analysis: Processing, inversion, and interpretation of seismic data*. Society of exploration geophysicists.

ZENG W & CHURCH RL. (2009). Finding shortest paths on real road networks: the case for A*. *International Journal of Geographical Information Science 23*(4):531–543. https://doi.org/10.1080/13658810801949850