Zentralinstitut für Engineering, Elektronik und
Analytik (ZEA) · Systeme der Elektronik (ZEA-2)

Evaluation of Line Detection Methods for the Analysis of Charge Stability Diagrams

Benedikt Scherer

Jül-4441

Mitglied der Helmholtz-Gemeinschaft

JÜLICH
Forschungszentrum

Zentralinstitut für Engineering, Elektronik und Analytik (ZEA) · Systeme der Elektronik (ZEA-2)

Evaluation of Line Detection Methods for the Analysis of Charge Stability Diagrams

Benedikt Scherer

Berichte des Forschungszentrums Jülich
Jül-4441 · ISSN 0944-2952
Zentralinstitut für Engineering, Elektronik und
Analytik (ZEA) · Systeme der Elektronik (ZEA-2)

Vollständig frei verfügbar über das Publikations-
portal des Forschungszentrums Jülich (JuSER)
unter www.fz-juelich.de/zb/openaccess

Forschungszentrum Jülich GmbH · 52425 Jülich
Zentralbibliothek, Verlag
Tel.: 02461 61-5220 · Fax: 02461 61-6103
zb-publikation@fz-juelich.de
www.fz-juelich.de/zb

Abstract

The various approaches for the implementation of qubits are objects of ongoing research. One approach uses the spin states of electrons that are confined in double quantum dots. For the calibration of such a qubit, charge stability diagrams (CSDs) are measured and analyzed. They constitute the changes in dependence on the voltages of two electrodes and, thus, can be represented as grayscale images. The voltages at which electrons tunnel in or out of the quantum dots correspond to line structures in the CSD images. Therefore, the automatic detection of lines in CSD images plays an important role in the qubit calibration process.

Various line detection methods are explored in this thesis. Since edge detection is a substep employed by several line detectors, a selection of edge detection methods is explored as well. Additionally, postprocessing methods for the validation, grouping, and merging of lines and the segmentation of infinite lines into finite line segments are investigated. This thesis focuses on traditional approaches as opposed to approaches based on machine learning.

Two methods are proposed for the automatic evaluation of the detection results. Both methods function by comparing the detections against a manually labeled ground truth data set. The first method evaluates edge maps by comparing the individual edge pixels with the pixels corresponding to the ground truth lines. The second method benchmarks the detected lines against the ground truth lines. Using these evaluation methods, the free parameters of the edge detectors, line detectors, and postprocessing methods are optimized.

The detection results of the methods are evaluated and compared for four data sets of varying quality. Additionally, the implications of the detection quality for the analysis of CSDs are explained.

Kurzfassung

Die verschiedenen Möglichkeiten der Implementierung von Qubits sind Gegenstand aktueller Forschung. Eine Art der Implementierung nutzt die Spinzustände von Elektronen, welche in Doppelquantenpunkten eingefangen sind. Für die Kalibrierung solcher Qubits werden Ladungsstabilitätsdiagramme gemessen und analysiert. Diese Diagramme stellen die Änderungen des Leitwertes in Abhängigkeit von zwei an Elektroden angelegte Spannungen dar und können dementsprechend als Grauwertbilder dargestellt werden. Diejenigen Spannungsbereiche, in welchen sich Elektronen in einen Quantenpunkt hinein oder aus einem Quantenpunkt heraus bewegen, entsprechen Linienstrukturen innerhalb der Ladungsstabilitätsdiagramme. Daher spielt die automatische Erkennung von Linien innerhalb der Ladungsstabilitätsdiagramme eine wichtige Rolle für den Kalibrierungsprozess der Qubits.

Verschiedene Methoden zur Liniendetektion werden in dieser Arbeit untersucht. Da mehrere Liniendetektoren die Detektion von Kantenpixeln als Teilschritt einsetzen, wird ebenfalls eine Auswahl an Kantendetektoren betrachtet. Des Weiteren werden Nachverarbeitungsmethoden für das Validieren, Gruppieren und Verschmelzen von Linien sowie für die Segmentierung von unendlichen Linien in endliche Liniensegmente untersucht. Diese Arbeit konzentriert sich dabei auf traditionelle Ansätze, die nicht auf Machine Learning basieren.

Zwei Methoden für die automatische Bewertung der Detektionsergebnisse werden vorgestellt. Beide Methoden basieren darauf, die Detektionsergebnisse mit einem manuell erzeugten Ground-Truth-Datensatz zu vergleichen. Die erste Methode bewertet die Kantenbilder, indem die individuellen Kantenpixel mit den Pixeln der Ground-Truth-Linien verglichen werden. In der zweiten Methode werden die detektierten Linien direkt mit den Ground-Truth-Linien verglichen. Unter der Nutzung dieser Bewertungsmethoden werden die freien Parameter der Kanten- und Liniendetektoren sowie Nachverarbeitungsmethoden optimiert.

Die Detektionsergebnisse werden auf vier Datensätzen mit unterschiedlicher Qualität bewertet und verglichen. Zusätzlich werden die Implikationen, welche die Detektionsqualität für die Analyse von Ladungsstabilitätsdiagrammen hat, diskutiert.

Contents

List of Figures

Abbreviations

CSD charge stability diagram

ED Edge Drawing

ELSD elliptical arc and line segment detector

ELSED enhanced line segment drawing

FLD fast line detector

FSG fast segment grouping

GMT gradient magnitude thresholding

gPb globalized probability of boundary

KHT kernel-based Hough transform

LSD line segment detector

NFA number of false alarms

Pb probability of boundary

PPHT probabilistic progressive Hough transform

SHT standard Hough transform

SSWMS slice sampling and weighted mean shift

1 Introduction

This chapter gives an overview of the application that motivates this thesis. First, the institute this thesis is written at is introduced. Afterward, an implementation of qubits that uses double quantum dots is described in Section 1.2. For the calibration of this type of qubit implementation, a charge stability diagram (CSD) has to be analyzed. Section 1.3 describes the properties of the CSDs and Section 1.4 explains the importance of the automatic detection of line segments in CSDs for the qubit calibration process.

1.1 Central Institute of Engineering, Electronics, and Analytics

The Central Institute for Engineering, Electronics, and Analytics (ZEA) develops devices, experiments, methods, and computer-aided tools for cutting-edge research. For these developments, the ZEA collaborates with other institutes at the Forschungszentrum Jülich as well as external research institutions internationally. The ZEA is divided into the following three sections:

1. Engineering and Technology (ZEA-1)
2. Electronic Systems (ZEA-2)
3. Analytics (ZEA-3)

This thesis is written at ZEA-2. The ZEA-2 develops electronic system solutions. One of the research projects the ZEA-2 is involved in is *Semiconductor Quantum Bit Control* (*SemiQuBiC*). The goal of *SemiQuBiC* is the development of control and read-out electronics for the usage in future quantum computers.

1.2 Qubits in Double Quantum Dots

A quantum computer is a computer that uses quantum effects to perform calculations. For some problems, the calculations performed by a quantum computer have the potential to be exponentially faster than those performed by a classical computer. An example is the Shor algorithm for prime factorization [1].

In a quantum computer, the fundamental unit of information is a quantum bit (qubit). Its analog on a classical computer, the classical bit, is always in one specific state, either 0 or 1. However, the state $|\psi\rangle$ of a qubit is a superposition of two base states $|0\rangle$ and $|1\rangle$[1]:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \text{ with } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1. \tag{1.1}$$

When measured, the qubit collapses into one of its base states. The probabilities for the resulting state are $|\alpha|^2$ for $|0\rangle$ and $|\beta|^2$ for $|1\rangle$. Since computations on a qubit are based on these states $|0\rangle$ and $|1\rangle$, they are also called the computational base states of the qubit.

A set of n connected qubits, called a quantum register, has 2^n base states. Equation (1.2) shows the formula for the state of a quantum register which consists of two qubits.

$$|\psi_2\rangle = \alpha_1\,|00\rangle + \alpha_2\,|01\rangle + \alpha_3\,|10\rangle + \alpha_4\,|11\rangle \text{ with } \alpha_1, \ldots, \alpha_4 \in \mathbb{C} \text{ and } \sum_{i=1}^{4} |\alpha_i|^2 = 1 \tag{1.2}$$

When an operation on a quantum register is performed, the coefficients $\alpha_1, \ldots, \alpha_n$ are manipulated while the equation remains fulfilled. Since the number of base states of a quantum register grows exponentially with the number of qubits, the number of coefficients manipulated by a quantum operation also grows exponentially. This explains why quantum computers can outperform classical computers for certain calculations.

In addition to superposition, quantum entanglement is another quantum effect that plays an important role in quantum computing. Qubits are entangled if their measurement outcomes depend on each other. For example, two qubits could be entangled in such a way that, upon being measured, they will both collapse into the same state:

$$|\psi_e\rangle = \alpha_1\,|00\rangle + \alpha_2\,|11\rangle\,. \tag{1.3}$$

Possible implementations of qubits are an object of ongoing research. The implementation that is researched in the *SemiQuBiC* project uses the spin states of two electrons to define the computational base states of a qubit. The spin of a single electron is in a superposition of two base states, designated as $|\downarrow\rangle$ for a spin down and $|\uparrow\rangle$ for a spin up. A system of two electrons, therefore, has $|\downarrow\downarrow\rangle$, $|\downarrow\uparrow\rangle$, $|\uparrow\downarrow\rangle$, and $|\uparrow\uparrow\rangle$ as its base states. The states $|\uparrow\uparrow\rangle$ and $|\downarrow\downarrow\rangle$, in which both electrons have the same spin, are eliminated by applying a magnetic field to the double quantum dot [2, Section 2.4.1]. This leaves the two states $|\downarrow\uparrow\rangle$ and $|\uparrow\downarrow\rangle$, from which the computational base states of the qubit are derived.

To form a qubit, the two required electrons first need to be trapped in a double quantum dot. A quantum dot is a nanostructure that confines electrons in all three dimensions [2, Section 2.3]. A double quantum dot consists of two coupled quantum dots that can exchange electrons. In the *SemiQuBiC* project, the quantum dots are

[1] The Dirac notation $|\cdot\rangle$ designates quantum states.

composed of two layers of semiconductors between which a two-dimensional electron gas is located. A schematic representation of a double quantum dot using aluminum gallium arsenide (AlGaAs) and gallium arsenide (GaAs) as semiconductors is displayed in Figure 1.1.

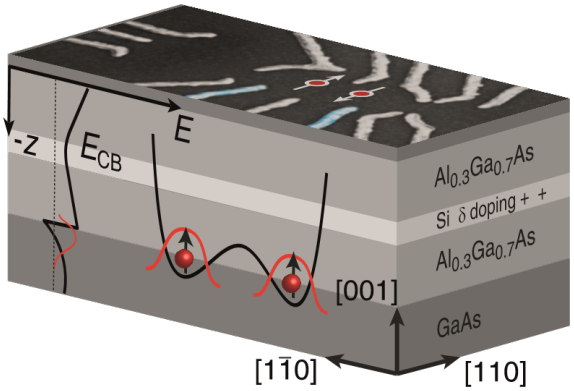


Figure 1.1: Schematic representation of a double quantum dot [2, Section 2.2].

On top of the area where the double quantum dot is formed, several electrodes, called gates, are located. Figure 1.2 shows the gate layout of the used sample. For this thesis, the most important gates are the so-called plungers P_1 and P_2. The plungers are responsible for controlling the number of electrons in the respective dots. A nearby structure, the sensor dot, reacts to changes in its electrostatic environment. This gives information on changes in the number of electrons contained in each of the quantum dots. The sensor dot itself is also a quantum dot, but it is only used for reading out the double quantum dot and not for computation.

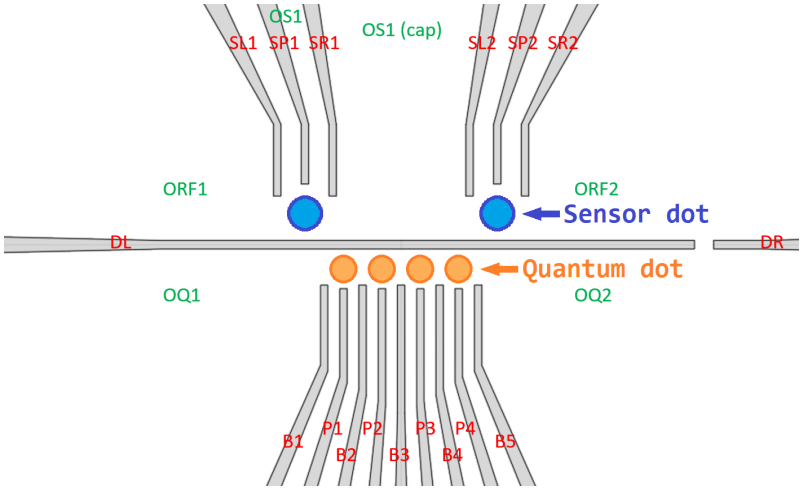


Figure 1.2: Two double quantum dots and their corresponding sensor dots and gates [3].

To trap the desired number of electrons, typically one, in each of the two quantum dots, the correct voltage ranges for the plunger gates P_1 and P_2 must be determined. This is done by measuring the conductance of the sensor dot while varying the gate voltages, which results in two-dimensional measurement data. A plot of these data is called a charge stability diagram (CSD).

1.3 Charge Stability Diagrams

As explained in the previous section, CSDs are the result of plotting the sensor dot measurement, which is proportional to the conductance, against the voltages of the plunger gates P_1 and P_2. The analysis of CSDs aids in determining the optimal voltage ranges for these gates, at which each of the two quantum dots contains the desired number of electrons. The voltage ranges with a specific number of electrons in the left and right quantum dot can be identified as a honeycomb-shaped structure inside the CSD. Figure 1.3 shows a schematic CSD. The number of electrons in the left and right quantum dot is marked as a tuple within the honeycomb-like structure. The lines between the segments signify that an electron is tunneling in or out of the quantum dots. At the longer lines, exemplarily marked with the red and green arrows in the image, the electron tunnels between the leads and the dot system. Therefore, such transitions of electrons are called lead-to-dot transitions. At the shorter lines, like the one marked by the blue arrow, an electron is tunneling from one quantum dot into the other. Such a transition is called interdot transition.

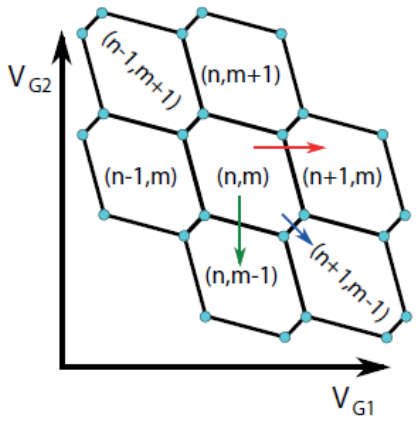


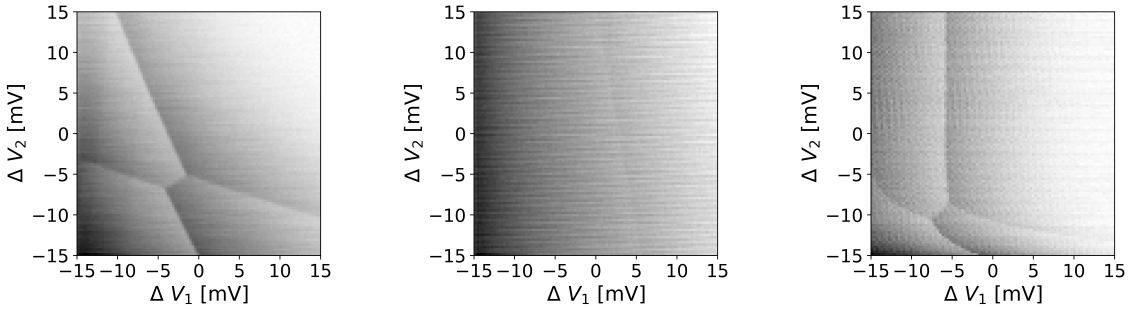
Figure 1.3: Schematic CSD [2, Section 2.4].

Currently, the optimal voltage ranges at which the desired number of electrons is contained in each quantum dot are usually determined manually by an experimenter. However, this approach is not scalable. For a scalable solution, automatic detection of the honeycomb-shaped segments is required. For this purpose, the application of line detection algorithms to CSD images is explored in this thesis, to automatically detect the voltage ranges, at which tunneling of electrons in and out of the quantum dots occurs.

1.4 Motivation

The CSDs are disturbed by several kinds of noise and artifacts. Figure 1.4 shows three CSDs with different line characteristics. In all three images, additive noise and gradual brightness changes within the honeycomb-shaped structures can be observed. Additionally, the CSDs contain horizontal stripes. These are caused by

the measurement process, which is performed as a series of 1D sweeps. While the lines in Figure 1.4 (a) are distinct and straight, the line in (b) is much weaker and the lines in (c) are slightly curved. These characteristics make the automatic detection of the lines challenging. The gate voltages used during the measurement of the CSD are not given as absolute values, but as changes ΔV_1 and ΔV_2 from a given start voltage.



(a) (b) (c)

Figure 1.4: CSDs with (a) distinct, straight lines, (b) a weak straight line, and (c) curved lines.

Initial experimentation on CSD data shows that line detectors often fail in areas where the lines are not distinct, and that strong noise leads to false positive detections. Other problems, such as the detection of too short lines or the erroneous detection of several short or overlapping lines instead of one contiguous line, do occur, but they present a much less critical issue for the analysis of CSDs. Therefore, the aim of this thesis is to evaluate different approaches of line detection methods, with the main goal of determining which approaches are most robust for detecting many lines at least partially and without too many false positives.

The explored methods are described in Chapter 2. In Chapter 3, two evaluation methods are proposed to quantify the quality of a given detection result according to the goals described above. Using these evaluation methods, the free parameters of the detection methods can be optimized, which improves the detection quality and enables a more accurate comparison of the results than a manual parameter selection. This optimization process is described in Chapter 4. Finally, Chapter 5 contains quantitative and qualitative evaluations of the detection methods.

2 Line Detection

The detection of lines in digital images plays an important role in various applications, and there is a vast number of line detection methods in the literature. An overview of the methods this thesis focuses on is given in Section 2.1. Additionally, the contained submethods are subdivided into edge detection, line detection, and postprocessing. Splitting the line detectors into their submethods enables, where applicable, the evaluation of whether a new combination of submethods from different line detectors leads to improved results. Sections 2.2 to 2.5 give a more detailed description of the methods.

All described methods are assumed to receive a grayscale image as input.

2.1 Overview

Traditional line detection approaches use hand-crafted algorithms for which few or even no parameters need to be fine-tuned. More recent approaches incorporate machine learning models, like neural networks, into the line detection process. Such models require a large labeled data set to train their parameters, which is not available to the author of this thesis. Additionally, traditional approaches and their parameters are easier to understand than machine learning models. This can be beneficial when the CSD data, that a detector is applied to and adapted for, changes, for example when there is a change in the hardware implementation of the quantum dots. Especially since implementation of the detection in hardware is required, an approach of low complexity is preferable. Therefore, this thesis will focus on a selection of promising traditional approaches.

An important submethod occurring in some line detection methods is the detection of edge pixels. Although some line detectors contain methods for the detection of edge pixels, other line detectors expect a binary edge map as input. Table 2.1 displays a selection of methods to create such binary edge maps. Some of them are submethods of a larger line detection method. For example, CannyPF was published as a submethod to be used in a line detection method named CannyLines [4]. As with line detectors, many recently published edge detectors are based on machine learning. This thesis will focus on hand-crafted algorithms for the same reasons as mentioned above for line detectors.

Some methods detect lines without defined starting and end points. In the following, such lines are referred to as infinite lines. Lines with defined starting and end

Table 2.1: Methods for the detection of edge pixels.

Method name	Method category	Section
GMT	Differentiation-based	Section 2.2.1
Canny	Differentiation-based	Section 2.2.2
CannyPF	Differentiation-based	Section 2.2.3
ED	Differentiation- and linking-based	Section 2.2.4
gPb	Based on several features	Section 2.2.5

points are referred to as (finite) line segments. Table 2.2 gives an overview of the line detectors evaluated in this thesis and displays their submethods. Submethods marked as "external" are not part of the given line detector but are expected to be already applied to the image which the line detector receives as its input. The line detection methods can be grouped mostly into voting-based approaches and region-based approaches. In voting-based methods, a line is defined by a parameter pair (e. g. y-axis-intercept and slope), and detections are made by voting for specific parameter pairs. Methods based on regions connect sets of pixels into the final line segments. Many line detectors also contain postprocessing steps like validation, in which those detected lines that are likely to be false positives are removed. It should be noted that this list of line detection methods is by no means complete. However, it contains the most widely used methods and also represents the different groups of algorithmic approaches.

2.2 Edge Detection

Edge pixels are pixels at which the image brightness changes significantly [5]. The detection of edge pixels plays an important role in line detection because edge pixels are potentially part of a line. The result of an edge detector is either a binary edge map, classifying each pixel as an edge or non-edge pixel, or a different data structure containing the same information, like a list containing the coordinates of the edge pixels.

2.2.1 Gradient Magnitude Thresholding

A simple method for the detection of edge pixels is to select pixels at which the local gradient is the strongest. This method is referred to as gradient magnitude thresholding (GMT) in the following. Image gradients are usually determined by convolving the input image with a filter. One filter that is often used for this purpose is the Sobel operator [6]. First, the gradients G_x in x-direction and G_y in y-direction of the grayscale image I are computed:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \text{ and } G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I. \tag{2.1}$$

Table 2.2: Methods for the detection and postprocessing of lines.

Method name	Edge detection method	Line detection method	Postprocessing method
SHT	External.	Voting-based, see Section 2.3.1.	No.
KHT	External.	Voting-based, see Section 2.3.2.	No.
PPHT	External.	Voting-based, see Section 2.3.3.	No.
PCLines	External.	Voting-based, see Section 2.3.4.	No.
LSD	No.	Region growing, rectangular approximation, validation, see Section 2.4.1.	No.
EDLines	Edge Drawing, see Section 2.2.4.	Fitting, see Section 2.4.2.	Validation, see Section 2.5.1.
ELSED	No.	Combined Edge Drawing and fitting, see Section 2.4.3.	Validation, see Section 2.5.1.
SSWMS	No.	Iterative sampling and growing, see Section 2.4.4.	No.
ELSDc	No.	Region-based, fitting, validation, see Section 2.4.5.	No.
FLD	Canny, see Section 2.2.2.	Region growing, see Section 2.4.6.	No.
Canny-Lines	CannyPF, see Section 2.2.3.	Region-based, fitting, see Section 2.4.7.	Validation, see Section 2.5.1.
Yuan	No.	Region- and voting-based, see Section 2.4.8.	No.
Linelet	No.	Region-based, validation, aggregation, see Section 2.4.9.	No.
FSG	No.	External.	Clustering, see Section 2.5.2.

The gradient magnitude G can then be computed as:

$$G = \sqrt{G_x^2 + G_y^2}. \tag{2.2}$$

By comparing this gradient magnitude to a given threshold, the pixels at which the brightness change is strong, i. e. the edges, are determined.

2.2.2 Canny Edge Detector

The edge detector proposed by Canny [7] is one of the most widely used edge detection algorithms. It first smoothes the image and computes the gradient magnitude

and gradient orientations at each pixel. Then, a threshold is applied to the gradient, and edges are thinned to have a thickness of one pixel using non-maximal suppression. Non-maximal suppression means comparing pixels that are neighboring along their gradient direction and suppressing those pixels with the lower gradient magnitude. Afterward, two thresholds are applied to the gradient magnitude to assign one of three labels to each pixel: non-edge pixel, weak edge pixel, and strong edge pixels. All strong edge pixels will be present in the final edge map. Additionally, weak edge pixels are classified as edge pixels, if they are connected to a strong edge pixel, directly or via a chain of other weak edge pixels. All other pixels are suppressed.

2.2.3 Parameter-Free Canny Edge Detector

A parameter-free version of the Canny edge detector, named CannyPF, adaptively sets the lower and higher thresholds that are applied to the gradient magnitude image. CannyPF was published by Lu et al. as part of the line segment detection method CannyLines [4] which is explained in Section 2.4.7.

In CannyPF, the lower and higher threshold are chosen adaptively based on the input image. This is done by observing the histogram of gradient magnitudes of the image and determining how likely the detection of false edge segments with a certain length would be if the algorithm was run on an image consisting of only white noise[1]. A detailed description of the steps of this computation can be found in [4]. Apart from that, the CannyPF algorithm functions like the standard Canny edge detector described in the previous section.

2.2.4 Edge Drawing

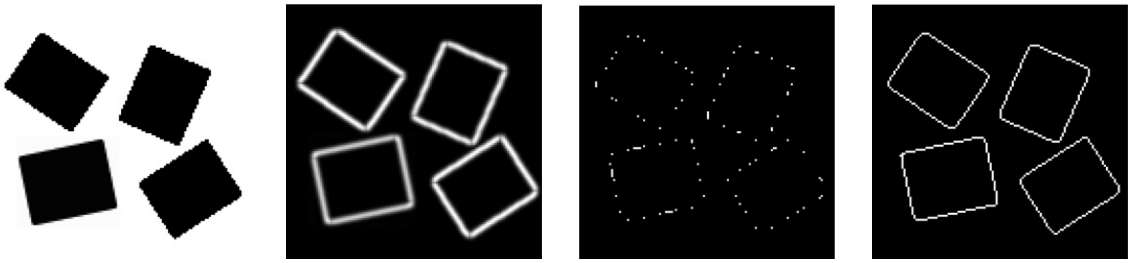
The Edge Drawing (ED) algorithm first selects a set of anchor points, that are most likely edge pixels and then links them into edge segments. Its name derives from the similar linking procedure in boundary completion puzzles, in which a person draws an object by connecting given dots on the object's outline [9]. ED was first published as part of a line segment detection method named EDLines [9], which is explained in Section 2.4.2.

First, the image is smoothed with a Gaussian filter and its gradient directions and gradient magnitudes are computed for every pixel. Then, the anchor points are determined by scanning the image vertically and horizontally. These are the points at which the gradient magnitude reaches local maxima. The scanning procedure allows to adjust the density of the detected anchor points by changing the scanning intervals, i. e. the intervals between the vertical and horizontal lines that are

[1] This idea is based on the Helmholtz principle, which declares that an observed structure is meaningful if its occurrence would be very rare in a random situation [8]. The Helmholtz principle is also used by several line detection methods to validate whether a detection constitutes a false positive. This validation step is explained in Section 2.5.1.

scanned. This way, the number of detected details can be influenced. Finally, isolated anchor points are linked to edge segments. Starting from an anchor point, neighboring pixels in the linking direction are checked and the one with the highest gradient magnitude is selected and added to the edge segment. The linking direction is based on the gradient direction of the currently selected pixel. This process is repeated from the new edge pixel until a pixel is selected which is either already part of an edge segment or whose gradient magnitude is below a chosen global threshold.

Due to this drawing process, ED results are more contiguous than those of some other methods.



(a) (b) (c) (d)

Figure 2.1: Illustration of the ED process: (a) an input image, (b) the corresponding gradient magnitude map, (c) the extracted anchor points, and (d) the edge map created by drawing connections between the anchor points [9].

2.2.5 Globalized Probability of Boundary

The globalized probability of boundary (gPb) algorithm is a boundary detector published by Arbelaez et al. [10]. It takes both local and global information into account. Its result is a probability map, designating the probability of a pixel belonging to a boundary. A boundary represents the change of ownership of a pixel in an image from one object to another. While boundary detection is different from edge detection, the resulting map can be used similarly. Therefore, gPb is grouped together with edge detection methods in this thesis.

GPb is an expansion of the probability of boundary (Pb) detector published by Martin et al. [11], which combines several local image features to create a probability map. Four feature channels are taken into account by the Pb algorithm:

1. oriented energy[2],
2. brightness gradient,
3. color gradient[3], and

[2]Oriented energy [12] combines several local filters to define an energy function. The peaks of this energy function correspond to edge-like discontinuities in brightness.

[3]The color gradient feature is not used in this thesis, as the CSDs are given as grayscale image.

4. texture gradient[4].

For every pixel position (x, y), the features are computed along a line through the center of a circular image patch with radius σ, having (x, y) as its center. This is done with eight equally spaced values for the angle of this line. In the gPb algorithm, this procedure is expanded by observing not only one circular image patch around every pixel, but three circular patches with $\frac{\sigma}{2}$, σ and 2σ as their respective radius. This enables the method to detect coarse as well as fine differences in the respective feature channels. The distributions of the features channel values along each half of the circular image patch are compared using a statistical dissimilarity measure. At points that are likely to be located in a boundary, the different distributions lead to higher dissimilarity values. A probability map *Pb* is obtained as a linear combination of these values.

Additionally, gPb takes global image properties into account. For this purpose, the computed features are represented as a weighted undirected graph. The nodes of the graph are the points in the feature space. Between every pair of nodes, a weighted edge is formed. The weight of these edges is a function of the similarity of the nodes, i. e. the similarity of the represented features. In the gPb algorithm, the intervening contour cue is used. This cue is defined as the maximum value of *Pb* along a line connecting two pixels. An optimal partition of the graph is obtained by minimizing the normalized cuts criterion [13]. This criterion measures the quality of an image partition, i. e. the similarity between the different subgraphs and the dissimilarity within subgraphs. The minimization of this criterion is formulated as an eigenvalue problem. A second probability map *sPb*, which takes global information into account, is obtained by using the determined eigenvectors. Through a linear combination of *Pb* and *sPb*, the final probability map *gPb* is obtained.

A boundary map can be used for various applications. In [10], it is used to segment natural images. In this thesis, a binary map is obtained from the probability-of-boundary map by applying a threshold.

2.3 Voting-Based Line Detectors

The following methods define parameterized representations of lines and vote for specific parameter pairs based on a given binary edge map.

The standard Hough transform (SHT) is the first of such methods, proposed by Hough in 1962. An improved version was proposed by Duda and Hart in 1972 [14] and is presented in Section 2.3.1. Since then, many variations on the SHT have been published. They mainly expand the original idea by proposing different parameterizations and voting schemes. In this thesis, the kernel-based Hough transform (KHT) and probabilistic progressive Hough transform (PPHT) are chosen as

[4]The texture gradient used in the Pb and gPb measures the degree to which the texture varies at a specific image location along a given direction.

a representatives of these variations and described in Sections 2.3.2 and 2.3.3.

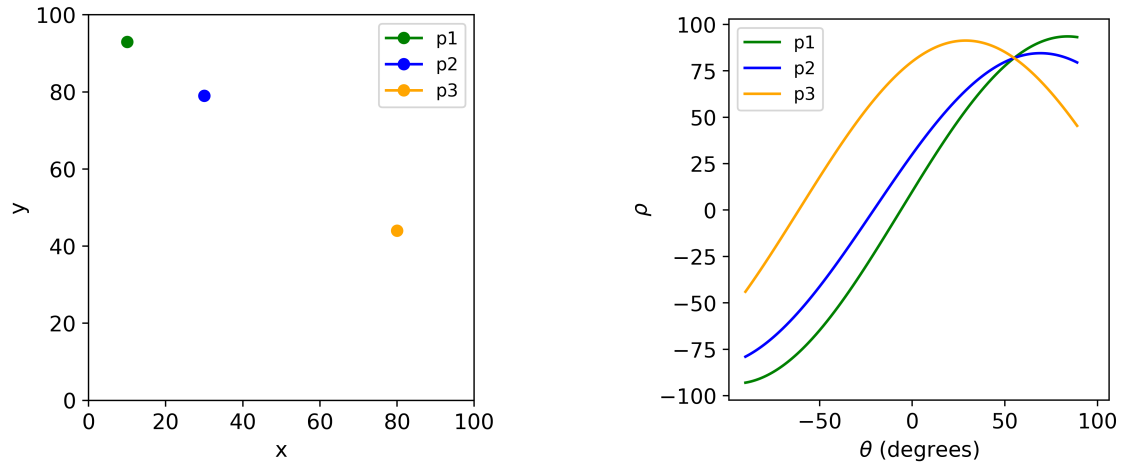
The algorithm PCLines (see Section 2.3.4) is a special case as it uses parallel coordinates but can be seen as an SHT variant.

2.3.1 Standard Hough Transform

The original SHT proposed by Hough uses a parameterization of lines by their slope and intercept. It was shown by Duda and Hart [14] that the so-called normal parameterization

$$x \cdot \cos(\theta) + y \cdot \sin(\theta) = \rho \tag{2.3}$$

is advantageous because it allows every possible line in an image to be associated with a point in a bounded parameter space, while the parameter space for slope and intercept is unbounded. In this normal parameterization, θ is the angle between the line's normal and the x-axis. The parameter ρ is the line's distance to the coordinate center. Every point in the parameter space corresponds to a line in the image space. For every edge pixel in the input image, several points in the parameter space are associated as several potential lines could pass through that edge pixel. The points associated with an individual point in the image form a sinusoidal curve in the parameter space. These relations between the image and parameter space are illustrated in Figure 2.2.



(a) (b)

Figure 2.2: Example of three colinear points $p1$, $p2$, and $p3$ in (a) cartesian coordinates and in (b) Hough space. Note that the curves in the Hough space intersect in a single point because the three points in the image space are located on a single line.

The SHT receives a binary edge map as input and votes for the parameter pairs associated with each edge pixel. This is done by increasing the counts in an accumulator image that represents the parameter space. The peaks in this accumulator correspond to the parameter pairs of the detected lines. Visually, these peaks correspond to the intersections of the sinusoidal curves. This accumulation process is also

called voting as the edge pixels "vote" for specific line parameters. By changing the quantization of the parameter space, the allowed colinearity error of the detection can be adjusted.

2.3.2 Kernel-Based Hough Transform

The KHT is a variation of the SHT published by Fernandes et al. [15]. It is much more efficient than the SHT and more robust to the detection of spurious lines, because of the use of an improved voting scheme. While a pixel votes for every potential line passing through it in the SHT, the KHT identifies clusters of approximately colinear edge points, associates each cluster to an elliptical kernel in the parameter space, and votes only for selected lines according to the kernels.

The clusters of edge points are determined as follows. First, neighboring edge points are linked together into chains of edge points. The chains are then subdivided into smaller chains that correspond better to line segments. This subdivision is done by recursively splitting a chain at the point which is most distant to the line segment defined by the beginning and end point of the chain. When a minimum chain length is reached, or when the created sub-chains do not provide a better fit to the line segments than the parent chain itself, the recursive splitting process is stopped. At this point, every created chain constitutes a cluster of approximately colinear edge points.

For every cluster C, the best-fitting line l is computed using linear regression. The authors provide a way of computing the parameters ρ and θ of the line l (corresponding to the line parameterization given in equation (2.3)) as well as their empirical variances σ_ρ^2 and σ_θ^2, which represent the uncertainty associated with fitting l onto C. C is associated to an elliptical-Gaussian kernel in the parameter space, using the best fitting parameters (ρ, θ) as its center, σ_ρ^2 and σ_θ^2 to define the shape and extension, and the covariance $\sigma_{\rho,\theta}$ to define the orientation. The exact definition of the kernel can be found in [15]. This is the kernel that gives the KHT its name. In a culling step, some kernels are discarded if their center's value is below a given threshold. The authors show that these discarded kernels do not affect the detection results much, but have a strong impact on the algorithm's efficiency, as fewer points in the parameter space need to be taken into account for the peak detection which determines the final lines.

Since a kernel's values outside of a neighborhood of its center (ρ, θ) are negligibly small, parameter pairs associated with a value below a given threshold do not cast a vote. The other parameters cast votes weighted according to the kernel values. The voting map is smoothed using a 3×3 Gaussian kernel, to consolidate adjacent peaks in the parameter space as single lines in the image space. Then, a list of all points in the voting map that received at least one vote is created and sorted in descending order. The peaks of the voting map are determined by visiting parameter pairs following the order of this list. Parameter pairs with completely unvisited eight neighbors so far are identified as the detected lines in the image space.

2.3.3 Probabilistic Progressive Hough Transform

The PPHT is one of the most widely used adaptions of the SHT and was published by Matas et al. [16]. It builds on other variations of the SHT. The probabilistic Hough transform (PHT) performs SHT on a fraction of the input and can be viewed as a Monte Carlo estimation of the SHT. Expanding on the PHT, the adaptive PHT (APHT) monitors the accumulation process and terminates it when the desired structures appear to have been detected. The PPHT is a specific form of APHT. It is also a more computationally efficient version of the PHT as it allows a premature termination of the voting process.

For long lines, only a small fraction of the supporting points need to vote for the line to be detected, while a short line requires almost all supporting points to be detected. The PPHT exploits this difference in the fractions of votes required to detect lines with a different number of supporting points. This is done by dynamically controlling the number of votes necessary to detect a line, as a function of the number of cast votes. When a line is detected, supporting points that voted retract their votes, and all supporting points are removed from the set of points that can still cast a vote. This is done for the finite underlying line segment, not the infinite line described by the given parameters. The finite line segment is extracted by analyzing the supporting points on the infinite line and selecting the largest segment of edge points that is either contiguous or does not exhibit a gap that exceeds a given threshold.

A consequence of this procedure is that the PPHT detects the most salient features first. This can be useful if an early termination of the algorithm is desired for efficiency.

2.3.4 PCLines

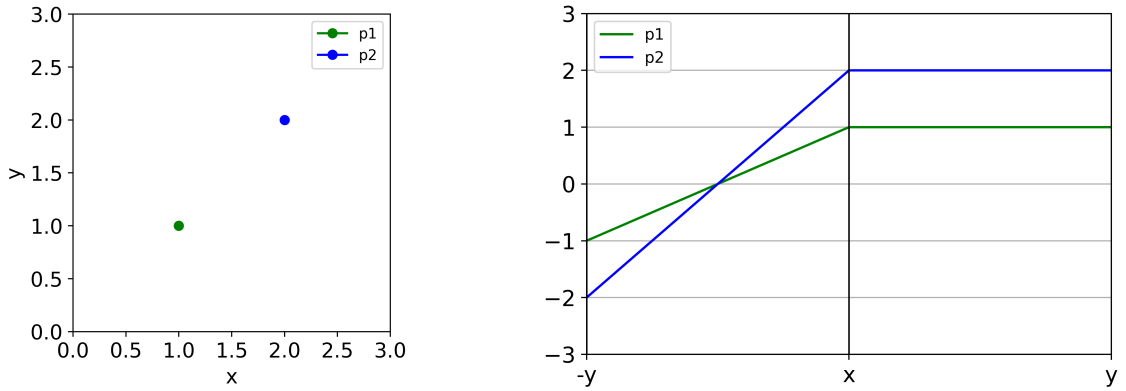
PCLines is a method for the detection of infinite lines published by Dubskà et al. [17]. It is a voting-based method that uses a representation of lines in parallel coordinates. Parallel coordinates are an alternative to the commonly used Cartesian coordinates. In a parallel coordinate system, the components of a vector correspond to points on mutually parallel axes, with each axis corresponding to one dimension. Therefore, an n-dimensional vector is represented by $n-1$ connected lines in the parallel coordinate system[5].

For the two-dimensional case, points that are colinear in Cartesian coordinates correspond to lines that intersect in one point in parallel coordinates. However, an intersection in the parallel coordinates, which corresponds to a detected line in the image space, can potentially be located outside of any bounded area[6]. To mitigate

[5]For this reason, parallel coordinate systems are often employed for the visualization of high-dimensional spaces, which would be unintuitive in Cartesian coordinates.

[6]In the parallel coordinate space, lines can also be parallel. In this case, they technically never intersect. However, in [17] such lines are considered as lines which intersect in a point that is infinitely far away from the axes of the parallel coordinate system.

this, a third axis $-y$, which corresponds to the flipped y-axis, is added to the parallel coordinate system. In this new system, containing three axes, every possible line in the image space corresponds to at least one intersection point that lies within a bounded area of the parallel coordinate space. Figure 2.3 displays an example containing two points $p1$ and $p2$ in (a) cartesian coordinates and (b) parallel coordinates. The intersection of the lines in Figure 2.3 (b) corresponds to the line that passes through the points $p1$ and $p2$.



(a) (b)

Figure 2.3: Example of two points $p1$ and $p2$ in (a) cartesian coordinates and (b) parallel coordinates. Note that the addition of the $-y$-dimension is necessary to determine an intersection.

Since lines in the image space correspond to points in the parallel coordinate space, the latter can be used as an accumulator space, similar to the space spanned by θ and ρ used in the SHT. It is quantized into a 2D image and every edge point in the original image votes for the corresponding points in the accumulator image. Lines are then detected as peaks in this accumulator image. A simple function for determining the intersections of the detected line and the borders of the original image, based on the detected line's representation in the accumulator space, is provided by the authors.

2.4 Region-Based Line Detectors

In contrast to the voting-based methods, the detectors described in the following sections are based on determining regions of pixels that belong to an underlying line segment, and then extracting the beginning and end point of the line segment from this region.

2.4.1 Von Gioi's Line Segment Detector

LSD is a method for detecting line segments published by von Gioi et al. [18]. In this thesis, this method is referred to as LSD while the term line segment detector

is used for all methods that detect (finite) line segments. The algorithm works by growing pixels into regions and then validating or rejecting these regions.

The authors recommend scaling the input image to 80% of its original size. Scaling the input image to a size that is smaller than the original leads to different results because of artifacts caused by aliasing and quantization effects.

The regions that are proposed for the subsequent validation step are grown as follows. First, the gradient magnitude and level-line angle[7] are calculated for all pixels. Pixels whose gradient magnitude is below a given threshold are ignored. The other pixels are pseudo-ordered by assigning them to one of b bins depending on their gradient magnitude. Pseudo-ordering via bins is advantageous because it is possible in linear time while sorting algorithms usually require $\mathcal{O}(n \log(n))$ operations to order n elements. Starting with a pixel from the highest bin, all unused neighboring pixels that are not already part of a different region are added to the region if their level-line angle matches the level-line angle θ_{region} of the region up to a given tolerance. θ_{region} is continuously updated when new pixels are added to the region. This process is then repeated for the neighboring pixels of the pixels that were newly added to the region until no new pixels can be added anymore.

The created regions do not yet form straight line segments with distinct beginning and end points but are just sets of pixels. To more accurately represent the underlying line segment, a rectangle approximation is associated with every region. This is done by interpreting the region as a solid object and the gradient magnitude of each pixel as the mass of that point. The rectangle center is then set to the center of mass of the region and the main direction of the rectangle is set to the first inertia axis of the region[8]. Finally, the length and width are set to the minimum values such that the rectangle still contains all pixels of the region. Figure 2.4 shows an exemplary region of pixels with its associated rectangle approximation.

Figure 2.4: A region of pixels with its associated rectangle approximation [18].

In the final step, the created regions and their associated rectangular approximations are validated. This validation step is based on the Helmholtz principle. A validation

[7] The level-line angle is the angle orthogonal to the local gradient angle.

[8] This corresponds to the axis around which rotating the object at a given angular velocity would require the least force. For example, the first inertia axis of a straight rod would be a line going along the rod's center.

decision is made depending on the size of the rectangle and the number of pixels in the rectangle whose level-line angle matches the main direction of the rectangle. Other line segment detectors also contain a postprocessing validation step based on the Helmholtz principle, which is described in more detail in Section 2.5.1. Since the validation step in LSD is tightly linked to the data objects of the previous steps, it is not considered as postprocessing step in this thesis. Every validated region corresponds to a detected line having the intersections

The authors assert that an advantage of LSD is that the provided default parameters work on a variety of different images without requiring any fine-tuning.

2.4.2 EDLines

EDLines is a line segment detector published by Akinlar et al. [9]. Its name stems from the Edge Drawing method it employs (see Section 2.2.4).

For the extraction of a line segment, or even several line segments, from a given chain of edges, lines are fitted onto the edge pixels using the least squares line fitting method[9]. Starting with a short line consisting of a few edge pixels, new pixels are added until the line fitting error exceeds a given threshold. At that point, a new line segment is created and the extraction is recursively continued with the remaining edge pixels. Note that this process requires the contiguous edge segments as a list of pixel coordinates instead of a binary image. Therefore, ED cannot easily be exchanged with another edge detector.

The detected line segments are validated in a final step based on the Helmholtz principle (see Section 2.5.1).

The authors provide recommended default parameters for EDLines that were empirically chosen based on a set of natural images but also state that other parameters can be advantageous for some images.

2.4.3 Enhanced Line Segment Drawing

ELSED is a line segment detector published by Suàrez et al. [19]. It works similarly to the EDLines algorithm described in the previous section, but includes line fitting and drawing connections between anchor points into a combined method called Enhanced Edge Drawing (EED). This method produces line segment candidates without using an edge map as an intermediary result. Therefore, ELSED contains no dedicated edge detection step which could be viewed as an isolated sub-method. In the final validation step, false positives are filtered out of the line segment candidates.

[9]Least squares line fitting means that for a given chain of edge points, a line is chosen such that the average squared deviation of the points from the line is minimized.

The EED algorithm begins by smoothing the input image with a Gaussian filter and computing the gradient orientations and gradient magnitudes for all pixels. Gradient magnitudes below a given threshold are set to 0. Gradient orientations are quantized into either horizontal or vertical orientation. Then, anchor points are extracted by scanning the pixels horizontally and vertically and determining pixels, with a local maximum in the gradient magnitude and with matching gradient orientation and scanning direction.

Drawing connections between the extracted anchor points is performed in a way that combines following neighboring pixels with a high gradient magnitude, like in the original ED algorithm, and fitting lines onto the pixels that have been drawn so far. Therefore, EED creates straight line segments directly from the anchor points. In comparison, ED creates chains of edge pixels that may change direction instead of following a straight line. Consequently, ED requires an additional step to extract the line segments from the edge chains.

ELSED uses Helmholtz validation (see Section 2.5.1) as a postprocessing step.

2.4.4 Line Segment Detection Using Slice Sampling and Weighted Mean Shift Procedures

SSWMS is a line segment detector published by Nieto et al. [20]. It iteratively proposes candidate pixels and then grows line segments out of these candidates.

In the first step, the likelihood $\hat{p}(x, y)$ belonging to a line segment is determined for each pixel coordinate (x, y). This is done by computing a pixel's gradient structure tensor T and analyzing the eigenvalues of this tensor. The gradient structure tensor T of a pixel is defined as:

$$T = \begin{pmatrix} g_x^2 & g_x g_y \\ g_x g_y & g_y^2 \end{pmatrix}, \tag{2.4}$$

where g_x and g_y are the partial derivatives in x- and y-direction at the given pixel. An explanation of the general relationship between the tensor eigenvalues of the gradient structure and the local structures in the image, such as lines and corners, is given in [21]. The function used to determine the probability that a pair of eigenvalues corresponds to a line is proposed in [20].

In the sampling stage, pixel candidates are selected for the line segment generation process. For this purpose, an efficient strategy that sequentially draws samples for the horizontal and vertical coordinates is used by the authors. It is based on a general sampling algorithm named slice sampling (the SS in SSWMS). The sample selection follows the probability distribution computed in the previous step, i. e. pixel coordinates that are more likely to correspond to a line are more likely to be proposed as candidates in this stage. Samples with an estimated likelihood lower than the mean are ignored.

After a pixel candidate has been selected, it is passed to the line segment generation

stage. In this stage, the pixel candidate is first refined, i. e. the candidate is replaced with a nearby pixel that better represents the underlying line segment. The authors propose a novel weighted mean shift procedure (the WMS in SSWMS) to determine this pixel. Mean shift is an iterative algorithm to locate the maximum of a density function from the sample data. In the WMS procedure, a twofold target is maximized in a window surrounding the initial candidate pixel: the probability of the pixel belonging to a line segment and the homogeneity of the estimated angle orientation of the underlying line segment. This orientation is determined based on the pixel's gradient information. Pixels belonging to the same line are expected to have similar orientations.

Starting from the refined point that was determined with the WMS procedure, nearby pixels are connected following the mentioned estimated orientation. Each of the added points is also refined using the WMS procedure. When the orientation of a newly added refined pixel deviates too strongly from that of the other connected pixels, the line segment generation is terminated and all the pixels belonging to the line segment are marked as visited. After that, a new candidate pixel for the next line segment is selected, ignoring the already visited pixels.

The algorithm can potentially be continued until all pixels have been visited. To reduce the computational cost, it can also be terminated earlier, for example after some iterations.

2.4.5 Joint Elliptical Arc and Line Segment Detector

ELSD is a method for the detection of both elliptical arcs and line segments published by Pătrăucean et al. [22]. It works by proposing a set of candidate regions, then either accepting or rejecting these candidates during a validation step and finally selecting a model that best describes the underlying line segment or elliptical arch for each candidate. The method builds upon LSD, which is described in Section 2.4.1.

The proposition of candidate line segment regions is done the same way as in the LSD algorithm: starting from points with high gradients, regions that represent the line segments are grown and a rectangle is fit around each of them. For the proposition of elliptical arc regions, the authors propose a curve-growing algorithm in which region growing and region chaining are alternated. This means that, instead of growing a single straight line segment region, smaller straight segments are connected into a curved segment. The curve-growing also works similarly to LSD, but the end point of one grown straight region is used as a starting point for the next. An elliptical shape is ensured by only chaining regions if the result is convex and roughly smooth, i. e. the line segments constituting the arc have angles that do not differ too strongly. Similar to the selection of an associated rectangle for line segments in LSD, ELSD fits elliptical arcs around the regions. Additionally, circular arcs are proposed in a similar way to the elliptical arcs.

The validation of candidate regions is also done similarly to LSD: the expected number of occurrences on an image consisting of only white noise is estimated and the candidate is only accepted if this estimate is below a given threshold. This expectation is called the number of false alarms (NFA). While a formula to compute the NFA was given for line segments and their associated rectangles in LSD, ELSD also provides formulas for candidate regions that correspond to circular or elliptical arcs.

It can occur that more than one model is proposed for the same region. For example, a region could be represented by both a non-circular elliptical arc and by a circular arc. In this case, the model that best fits the underlying structure is selected. This is done by choosing the model which possesses the smallest NFA value.

In [23], the authors propose a similar algorithm named ELSDc. It differs from ELSD by using a continuous distribution for the noise model during the computation of the NFA, rather than a discrete one. The authors showed that this continuous model leads to better model selections on synthetic images of circles and polygons, altered by different levels of Gaussian noise.

To compare the results of ELSDc with those of other methods, straight line segments are extracted from the elliptical arcs in this thesis. This is done by choosing the longest line segment that fits the elliptical arc up to a given error.

2.4.6 Fast Line Detector

FLD[10] is a line segment detector published by Lee et al. [24]. It uses an edge map as a starting point to grow edge pixels into line segments. In [24], the Canny algorithm (see Section 2.2.2) is used for the creation of the edge map.

Closely located edge pixels are grown into a chain of pixels if they fit onto a line up to a given error. This is done until all edge pixels have been visited. To close gaps between associated chains, they are merged into final line segments if they are overlapping or closely located and their orientation is sufficiently similar.

2.4.7 CannyLines

CannyLines is a line segment detector published by Lu et al. [4]. It begins by extracting an edge map using CannyPF, which is described in Section 2.2.3. The edge pixels are linked and split into line segments, which are then extended and merged into longer, more complete line segments. In the final step, the line segments are validated.

[10] The name "Fast Line Detector" is not used in the paper [24] itself but for the class `cv::ximgproc::FastLineDetector` which provides an interface to the algorithm in the OpenCV library.

Before the linking stage, all edge points are first sorted by their gradient magnitude. Starting with the edge pixel that has the highest gradient magnitude, neighboring edge pixels are linked if their gradient orientation matches. If a linked region is larger than a given threshold, it is split at the point that maximally deviates from the line segment. This splitting is done to ensure that regions belonging to different underlying line segments remain separate. Least squares line fitting is then applied to the linked region to obtain line segments with more accurate beginning and end points. The lines are extended by searching for edge pixels that are closely located to the end points and especially closely located in the direction orthogonal to the line. If, during this extension process, an edge point of another line is met, the two lines are merged if their angles are sufficiently similar and the least squares fitting error of the merged line is sufficiently small. Thus, regions that were erroneously split in the previous step can still be detected as a connected line segment.

The detected lines are validated using the Helmholtz principle. This step is described in Section 2.5.1.

2.4.8 Yuan's Line Segment Detector

Yuan et al. [25] published a line segment detection method that combines the formation of regions based on the direction of the gradient with a voting-based approach.

First, the gradient direction and magnitude are computed for each pixel. For pixels with a gradient magnitude greater than a given threshold, the gradient directions are sorted into n bins which are equally divided between $0°$ and $360°$. Using these bins, contiguous regions with similar gradient orientations are formed. These are called line support regions.

For each line support region, the gradient structure tensors (see Equation (2.4) in Section 2.4.4) of all pixels are summed, and the eigenvectors of this sum are computed. The eigenvector corresponding to the smaller of the eigenvalues indicates the orientation θ of the line segment that underlies the line support region. This orientation θ is the same parameter as in the parameterization of lines used by the SHT in equation (2.3). The second parameter ρ, which corresponds to the distance between the line and the coordinate center, is determined using a voting procedure similar to the SHT. Each pixel in the line support region votes for a value of ρ with the vote weighted by the local gradient magnitude of the pixel. The value with the highest score is selected and the determined parameters define an infinite line. Finally, a finite line segment is extracted by selecting the part of the infinite line that overlaps with the associated line support region.

2.4.9 Linelet-Based Detection

Cho et al. [26] published a line segment detector based on linelets. A linelet is a horizontal or vertical group of connected edge pixels. The discretization in a digital image causes line segments to have a staircase shape, and a linelet can be considered as an individual stair in such a staircase. This concept is illustrated in Figure 2.5. The linelet-based detector works by first detecting the individual linelets, then grouping the closely located linelets and assigning a line segment to each group. Afterward, the line segments are validated, and disjoint line segments are merged if they are more likely to belong to a single line segment.

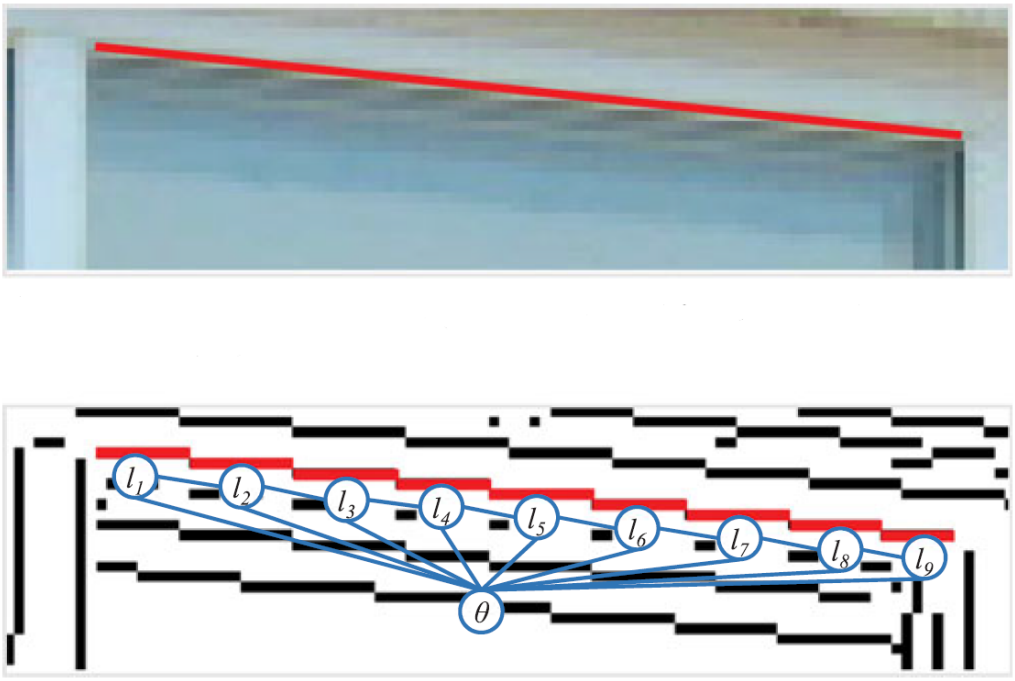


Figure 2.5: A line segment is broken into linelets l_1 to l_9 due to the discretization of the image [26].

For the detection of the linelets, the gradient magnitude and orientation are computed for each pixel, and non-maximal suppression is applied in such a way that the resulting image only contains pixels that have the maximum gradient magnitude to either their horizontal or vertical neighbors. From this image, linelets are extracted as neighboring pixels having non-suppressed values in either horizontal or vertical direction. Neighboring linelets are then grouped if they have the same direction and their length difference is below a given threshold. For each linelet group, the underlying line segment is identified.

During the validation step, the probability that a detected line segment belongs to a true line is estimated. This process integrates the gradient information at the linelets contained in the line segment and makes a final decision using a mixture of expert models.

In a final aggregation step, separate line segments are connected if their aggregate is more plausible than the individual line segments. For each line segment, aggregation

candidates are determined based on the difference in line angle, the shortest distance between end points, and the similarity of the lengths of the contained linelets.

Since the validation and aggregation steps make use of the linelets a given line segment consists of, they are not considered as separate postprocessing steps in this thesis.

2.5 Postprocessing

After the lines have been detected, a subsequent postprocessing step can improve the results. Several such steps have already been described in the context of their associated line detection methods in the previous sections. The following postprocessing methods have been selected to be treated as individual methods as they allow a combination with any other line detection step without significant overhead.

2.5.1 Validation Using the Helmholtz Principle

The Helmholtz principle states that a structure is perceptually meaningful if its occurrence is very rare in a random situation [8]. Desolneux et al. [8] published a general procedure that applies this principle to validate whether a set of pixels in a binary image is likely to be meaningful. Various authors introduced specific line segment validation algorithms based on this general procedure. The LSD algorithm uses a version of this procedure that requires knowledge of a specific intermediary result, the rectangle described in Section 2.4.1. However, versions used by EDLines, ELSED, and CannyLines only require knowledge of the line segment to be validated and the gradient of the original image.

In general, the validation procedure first computes the expected number of occurrences for the given structures in a random situation, i. e. a binary white noise image. This expectancy is called the NFA. By comparing the NFA to a given threshold, a validation decision is made.

For the line segment detection methods above, the NFA is computed by analyzing the number k of line segment pixels with a gradient roughly orthogonal to the line direction, in the context of the total number n of line segment pixels. In a random environment, the probability for any line segment of n' pixels containing exactly k' aligned points is given by a binomial distribution:

$$\mathbb{P}(k = k' \mid n = n') = \binom{n'}{k'} p^{k'} (1-p)^{n'-k'}. \tag{2.5}$$

Since a line segment is defined by two points, an $N \times N$ image contains approximately N^4 possible line segments. The exact number of possible line segments in an image depends on factors like whether a swapping of beginning and end points leads to the

same or two different line segments. In the context of this validation procedure, N^4 is accurate enough as an approximation, especially since a different approximation can be accounted for by comparing the NFA to a different threshold. Therefore, the expectancy of line segment occurrences of length n with at least k aligned pixels can be computed as:

$$\mathrm{NFA}(n,k) = N^4 \sum_{i=k}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{2.6}$$

The value p in equations (2.5) and (2.6) depends on the maximum deviation that a pixel's gradient angle is allowed to have before it is considered to be not aligned with the underlying line segment. The line segment is accepted if its NFA is below a given threshold. If a large number of line segments or line segments from many different images with the same dimensions need to be validated, the computational efficiency can be increased by creating a lookup table that saves the smallest acceptable value of k for every value of n .

In [18], it was shown that the fine-tuning of the threshold parameter is practically not necessary because a wide range of thresholds will lead to the same or very similar results for many images. Nonetheless, the selection of an optimal threshold value for the CSD data set used in this thesis is described in Section 4.2.

The formula given in equation (2.6) is the one used in the EDLines algorithm [9]. LSD and CannyLines compute the NFA differently. LSD computes the NFA for the rectangle that surrounds the line's region and only takes those aligned pixels into account whose gradient magnitude is above a threshold [18]. CannyLines takes both the NFA of the line segment and the NFA of the underlying edge segment into account [4]. The ELSED algorithm does not compute the NFA but simply checks whether at least 50% of the pixels in a line segment are aligned correctly [19]. Therefore, the validation step used by the ELSED algorithm does technically not make use of the Helmholtz principle but functions very similarly.

2.5.2 Fast Segment Grouping

Fast Segment Grouping (FSG) is an algorithm for merging groups of line segments published by Suàrez et al. [27]. In the original publication, FSG was used to improve the results of the LSD algorithm, but it can also be used to postprocess the results of any other line segment detector. The algorithm consists of two iterative steps. First, a greedy proposer selects sets of line segments to be merged. Then, a subsequent validation step checks whether the group's line segments are likely to correspond to a single underlying line segment.

Before grouping candidates are selected, the line segments are partially ordered by their length, and a histogram of the orientations of the line segments is created. Starting with the longest segment s that is not yet assigned to a group as the base line segment, a circle of uncertainty is defined around the base line segment's beginning and end point. The base line segment and the corresponding circles of

uncertainty are updated with each new addition to the group as described below. These circles delineate cones in which the beginning or end point of an extension candidate must be located. From the set of line segments fulfilling this condition with an angle sufficiently similar to s, each element is considered a candidate. Then, the probabilistic check described below is applied to determine whether a candidate should be added to the group. This process is repeated until each line segment has been assigned to a group (even if the group consists of only the segment itself).

The probabilistic validation used above considers the smallest bounding box that encloses the candidate as well as every line segment in the group. Then the expected number of occurrences of a box that encloses the same amount of line segments with the same lengths is computed for a random situation, i. e. an image in which the beginning and end points of the line segments have completely random locations. This is similar to the computation of the NFA and a formula for this expectation is provided in [27]. The decision is made by comparing this expectancy to a given threshold.

After every addition of a new line segment to a group, the base line segment is recalculated as follows. Using all beginning and end points of the line segments in the group, an infinite line is fit using least squares. Onto this line, the furthest beginning and end point in the group are projected. These two projected points define the new base line segment of the group. A change in the base line segment also leads to a change in the cones used to determine new candidates in subsequent iterations. From the final groups, the respective base line segments are used as an improved detection result. The original line segments are discarded.

2.5.3 Segmentation of Infinite Lines

Given an associated binary edge map, a simple method for the segmentation of infinite lines is used in the PPHT algorithm. It analyzes the edge points in the binary edge map that are located on the infinite line and selects the longest segment that is either contiguous or whose gaps have a length that does not exceed a given threshold.

In PPHT, this line segmentation step cannot be separated from the voting process because only the points which are located on the line segment are excluded from subsequent voting iterations. However, it can easily be applied to other infinite line detectors as long as a binary edge map is provided. All infinite line detectors discussed in this thesis use such an edge map.

A limitation of this approach is that it extracts a single line segment from every infinite line, even though an infinite line could be composed of several separate finite line segments. This does not pose a problem in the case of PPHT because the edge points located on the smaller segments can vote again in subsequent iterations. However, it can lead to the wrongful exclusion of shorter line segments if the method is applied as a separate postprocessing step after the detection of infinite lines has

already been completed. Therefore, an adapted version is proposed here. After extracting the longest segment, shorter segments are extracted if they fulfill the same conditions and their length is above a given threshold.

3 Methods for the Evaluation of Edge and Line Segment Detectors

A method for quantifying the quality of the detection results is required to enable the automatic optimization of the parameters and an accurate comparison of the results achieved by the different detectors. In the following sections, two different evaluation methods are presented. Both methods work by comparing the detection results to ground truth data for a given input image.

A description of the ground truth data set is given in Section 3.1. The software interfaces for the detection and postprocessing methods are described in Section 3.2. In Section 3.3, a method for the evaluation of edge detector results is presented. This method is based on comparing the individual pixels of the ground truth data and the detected edge map. The method presented in Section 3.4 evaluates the results of line detectors and is based on comparing individual line segments.

3.1 Ground Truth Data Set

The data set used in this thesis consists of 53 manually labeled CSD images, having an 8-bit data type and 100x100 pixels. These images were categorized into four classes, ranging from very high-quality data with weak noise and distinct lines to low-quality data with strong noise and very weak lines. In the following, these classes will be referred to as “very high quality”, “high quality”, “mediocre quality” and “low quality”. An example CSD for each class is given in Figure 3.1.

From each class, four CSDs were randomly selected for the parameter optimization. This was done to ensure that the optimization data set represents the different levels of quality present in the underlying data. The remaining data constitutes the validation data set, which is used to evaluate the final results in Chapter 5.

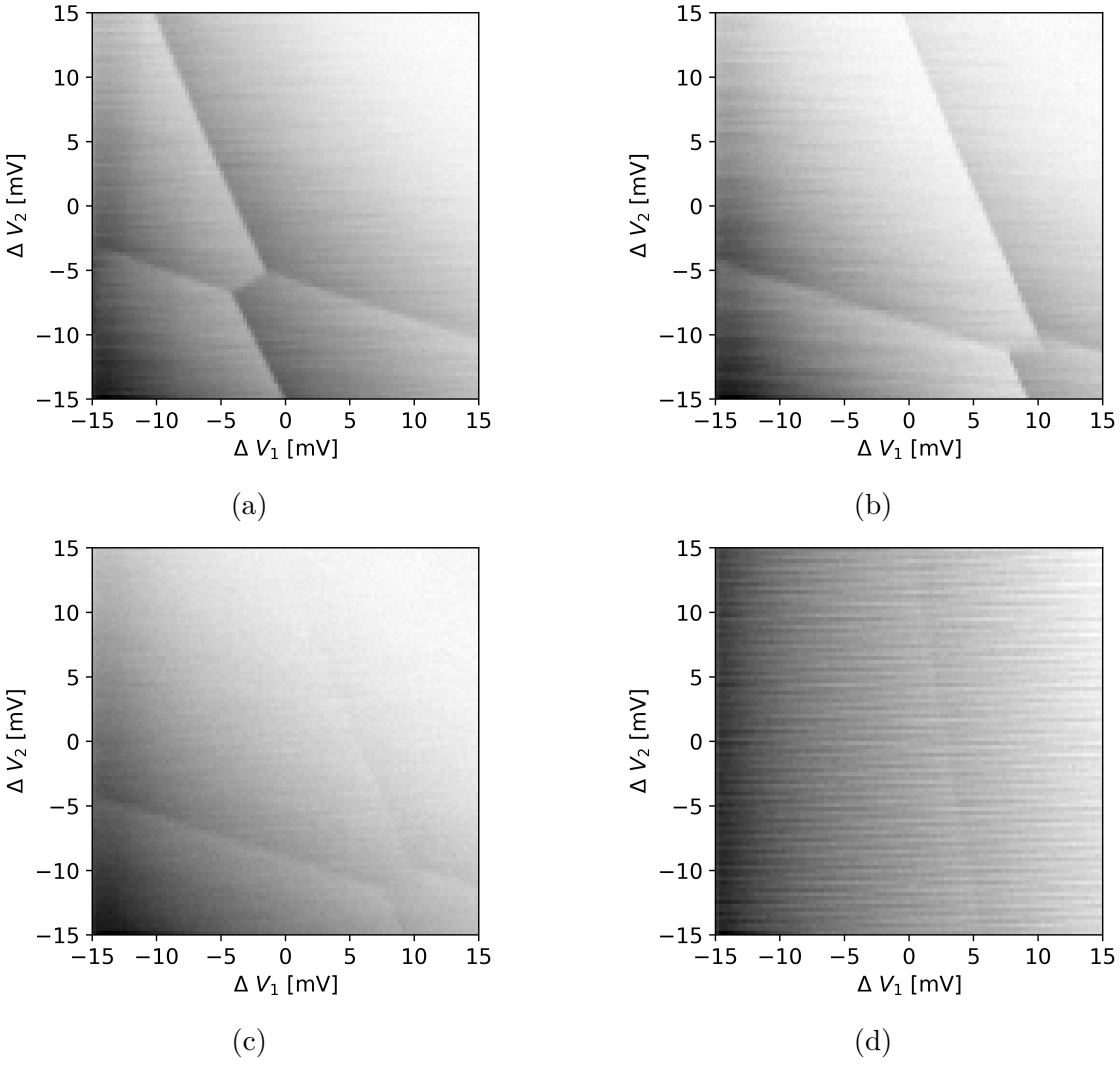


Figure 3.1: Examples of CSDs from the optimization data sets of (a) very high, (b) high, (c) mediocre, and (d) low quality.

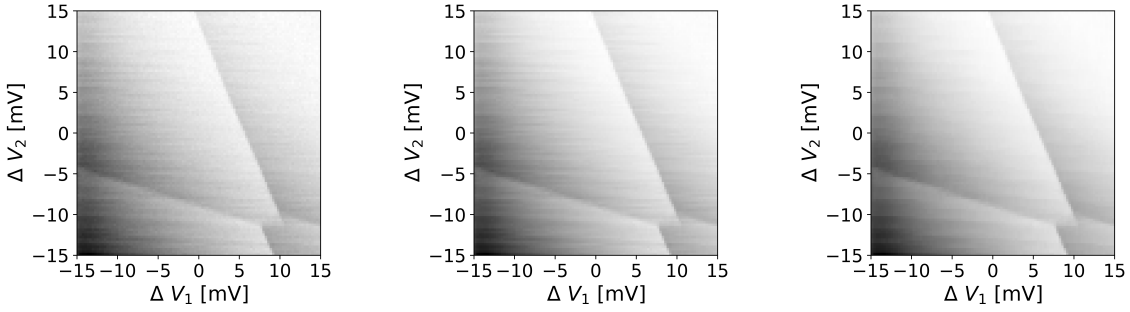


Figure 3.2: Example of a CSD (a) before processing, (b) after applying the BM3D filter, and (c) after additionally applying the median filter.

To improve the quality of the data, each image was processed with edge-preserving smoothing algorithms. First, the BM3D filter is applied. For a CSD data set that

is similar to the one used in this thesis, Fleitmann et al. [28] evaluated that BM3D is preferable to other smoothing algorithms and leads to a strong reduction of noise while preserving the edges. However, there are stripes in the CSD images that the evaluation of Fleitmann et al. [28] does not account for. These stripes are due to the measurement process which happens as a series of 1D sweeps over a voltage range while the voltage of the other gate remains constant. A 1D median filter is employed to weaken these stripes. Figure 3.2 displays (a) a CSD before smoothing, (b) after applying the BM3D filter, and (c) after additionally applying the 1D median filter.

Formally, the manually created ground truth labels for each image are given as a list of 4-tuples $l_i = (x_{\mathrm{b},i}, y_{\mathrm{b},i}, x_{\mathrm{e},i}, y_{\mathrm{e},i})$, $i = 1, \ldots, L$, i. e. a list of line segments l_i with the points $(x_{\mathrm{b},i}, y_{\mathrm{b},i})$ and $(x_{\mathrm{e},i}, y_{\mathrm{e},i})$ as the coordinates of their beginning and end points.

3.2 Software Interfaces for the Detectors

To execute the detection methods on the CSD images, implementations by the respective authors or third parties are used. Table 3.1 displays the language and source of these implementations. To evaluate and compare the detection methods in a structured manner, interfaces in the Python programming language are created to wrap the various implementations. The return values of the line segment detectors are unified to represent the line segments as 4-tuples as described in Section 3.1. Similarly, infinite lines are represented as line segments whose beginning and end points are located on the image borders. Line segments that are shorter than five pixels are automatically discarded because all ground truth lines are significantly longer and very short lines are especially often false positives. Additionally, the given interfaces are adapted to use quantiles instead of absolute values for those parameters that threshold the gradient magnitude. This was done to account for the different distributions of the gradient magnitude that are present in different images. For this purpose, the gradient magnitude is computed using the Sobel operator as described in Section 2.2.1.

3.3 Pixel-Based Evaluation

Most edge detectors produce a binary image as output, in which the edge pixels can be scattered or are not contiguous. Therefore, a method for the evaluation of edge detectors has to be able to take scattered individual pixels into account. This method determines the average score of correctly classified edge pixels per ground truth line segment and subtracts an error term based on the amount of false positive edge pixels.

The ground truth data is first transformed into a multiclass image

$$I_{\mathrm{T}} \colon \{0, \ldots, N\} \times \{0, \ldots, M\} \to \mathbb{N}_0,$$

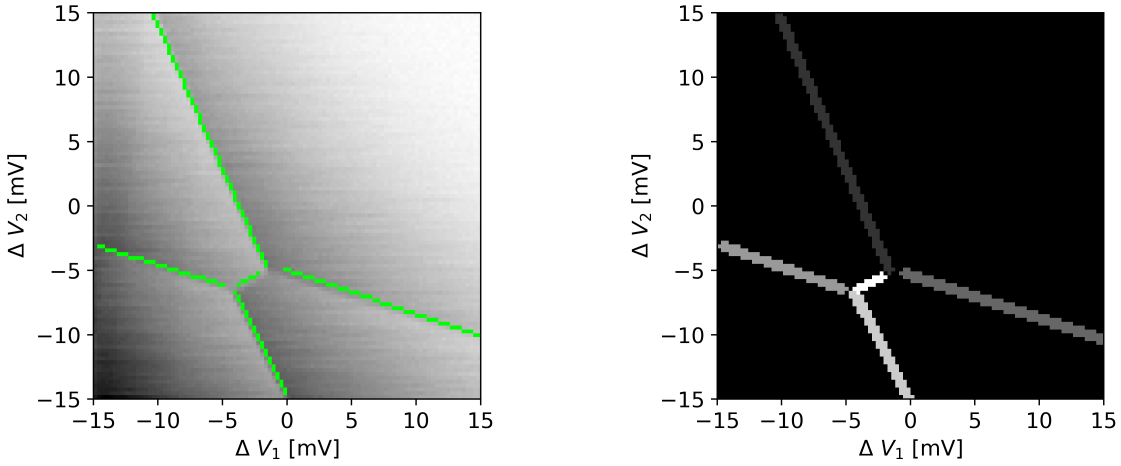
Table 3.1: Interfaces for the edge detectors, line detectors, and postprocessing methods.

Method name	Language	Source
GMT	Python	Implemented by the author
Canny	Python	[29]
CannyPF	C++	[30]
Edge Drawing	Python	[31]
gPb	C++	[32]
SHT	Python	[33]
KHT	C++	[34]
PPHT	Python	[35]
PCLines	Python	[36]
LSD	Python	[37]
EDLines	Python	[31]
ELSED	C++	[38]
SSWMS	C++	[39]
ELSDc	C	[40]
FLD	Python	[41]
CannyLines	C++	[30]
Yuan	Matlab	[42]
Linelet	Matlab	[43]
Helmholtz	Python	Implemented by the author
FSG	C++	[44]
Segmentation	Python	Implemented by the author

in which every pixel belonging to line segment l_i has the value i and the pixels not belonging to any line segment have the value 0. For this purpose, the line segments are assumed to not overlap. By changing the width of the line segments to be larger than one pixel, the evaluation method allows a certain error margin. This is especially useful since the position of a line segment can be slightly inaccurate in the manually labeled data. Additionally, the amount of pixels the line segment contains without margin is denoted as a_i. Since this multiclass image needs to be created only once and can be reused, for example in subsequent optimization steps, the overhead caused by this step is negligible. An example of a multiclass image is displayed in Figure 3.3. The edge map created by the detector is given as a binary image

$$I_{\mathrm{D}} \colon \{0, \dots, N\} \times \{0, \dots, M\} \to \{0, 1\},$$

in which edge pixels have the value 1 and non-edge pixels have the value 0.



(a) (b)

Figure 3.3: Example of (a) the original CSD with the ground truth lines marked in green, and (b) the corresponding multiclass image with a margin of one pixel on each side of the line.

From these images, the percentage p_i of correctly detected pixels is determined for every ground truth line segment l_i:

$$p_i = \min\left\{\frac{|\{(x,y)\colon I_{\mathrm{D}}(x,y) = 1 \wedge I_{\mathrm{T}}(x,y) = i\}|}{a_i}, 1\right\}, \tag{3.1}$$

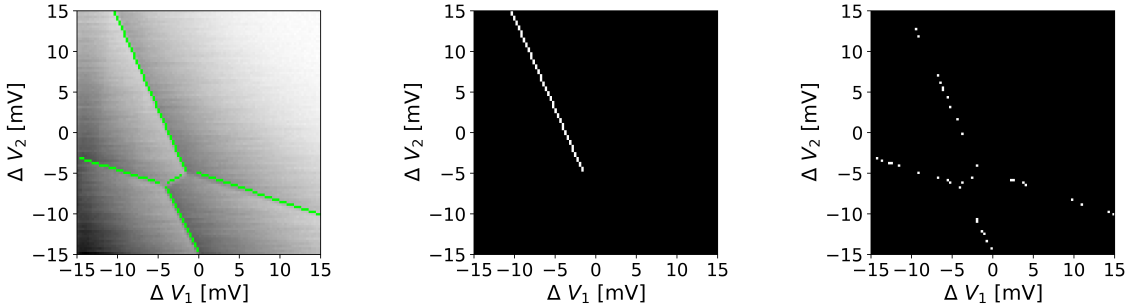
where $|X|$ denotes the cardinality, i. e. the number of elements, of the set X. Taking the minimum of the percentage and 1 is necessary because an edge detector could "correctly" classify more pixels than the line segment contains due to the margin in the ground truth image. A limitation of this approach is that p_i can potentially be close to 1 due to the margin, even though parts of a line segment are missed. Qualitative analysis of the edge detection results on the optimization data set shows that this does practically not occur. However, a consequence of this limitation is that this evaluation method is slightly biased towards detectors that create thick edges wider than one pixel, such as GMT, as opposed to detectors that create thin edges having a width of one pixel, such as Canny.

The percentages p_i are then non-linearly transformed because the usefulness of an edge detector does not scale linearly with the percentage of correctly identified edge pixels. For example, detecting only half of all edge pixels in a given line segment can be sufficient to create acceptable results in a subsequent line detection step, and should therefore be assigned a higher score than 0.5. The square root is chosen for this non-linear transformation and the mean $\bar{p}$ of the transformed percentages is computed:

$$\bar{p} = \frac{1}{L}\sum_{i=1}^{L}\sqrt{p_i}. \tag{3.2}$$

An example of the benefit of this transformation is given in Figure 3.4. Two different edge maps for the same CSD are displayed. The five ground truth lines are marked in green in Figure 3.4 (a). In the edge map in Figure 3.4 (b), the edges corresponding

to one line are detected perfectly, leading to a value of $p_i = 1$, while the edges corresponding to the other four lines are not detected at all, leading to values of $p_j = 0$. This results in a score of $\bar{p} = 0.2$. In the edge map in Figure 3.4 (c), approximately 20% of the edge pixels corresponding to every line are detected. This leads to values of $p_i \approx 0.2$ and a score of $\bar{p} \approx \sqrt{0.2} \approx 0.447$. Since the second edge map is more useful to a subsequent line detection step, the higher $\bar{p}$ score it receives is justified. However, without the non-linear transformation of the p_i values, both edge maps would receive approximately the same score.



(a) (b) (c)

Figure 3.4: Example demonstrating the benefit of non-linearly transforming the p_i values: (a) a CSD with the five ground truth lines marked in green, (b) a corresponding edge map with the edges of one line detected perfectly and the other four lines not detected at all, and (c) an edge map in which roughly 20% of edge pixels were detected for every line. Without applying the square root to the p_i values, both edge maps would receive approximately the same evaluation score despite the edge map in (c) being more useful for a subsequent line detection.

The value $\bar{p}$ takes the true positive and false negative pixels into account but not the false positive pixels. For n_{TP}, n_{FN}, and n_{FP} as the respective numbers of detections in the edge map I_{D}, an error term E_{FP} is defined as

$$E_{\mathrm{FP}} = \min\left\{\bar{p}, \frac{n_{\mathrm{FP}}}{2(n_{\mathrm{TP}} + n_{\mathrm{FN}})}\right\}. \tag{3.3}$$

The final evaluation score is then calculated as

$$\bar{p} - E_{\mathrm{FP}}. \tag{3.4}$$

This score is 1 for a perfect edge map, in which all edge pixels were detected and no false positive detections are present, and 0 for an unusable edge map, in which no edge pixels were correctly classified or in which several false detections for every correct detection exist.

The efficiency of the method could be improved by computing only a single percentage p for all line segments combined. However, such an evaluation would strongly favor long line segments over shorter line segments as they contain more pixels. Favoring longer lines is not desired for the application that motivates this thesis.

Since a binary image I_{D} could be created from a given list of detected line segments, the presented method could also be used to evaluate a line segment detector.

3.4 Segment-Based Evaluation

As explained in Section 1.4, the main goal of this thesis is to determine which approaches are most robust for detecting many lines at least partially and without too many false positives. Detecting a line that is too short or too long, up to a given threshold, is therefore not punished by the evaluation method presented in this section. Instead, each line segment will receive a binary evaluation: either it is detected correctly, or it is not. The method determines the numbers of true positive (TP), false positive (FP), and false negative (FN) line segments and quantifies the quality of the detection results using the F1 score. The following description of the method is aimed at finite line segments. However, it can also be applied to the evaluation of infinite lines. For this purpose, the infinite lines are simply treated as finite line segments whose beginning and end points are located on the image borders, and the ground truth line segments are extended to also have their beginning and end points located on the image borders.

First, it needs to be defined under which conditions a detected line segment $d_j = (x_{\mathrm{b},j}^{(d)}, y_{\mathrm{b},j}^{(d)}, x_{\mathrm{e},j}^{(d)}, y_{\mathrm{e},j}^{(d)})$ matches a line segment l_i in the ground truth data. Such a match occurs if the following conditions are fulfilled:

1. the angles α_{d_j} and α_{l_i} of d_j and l_i differ by at most r radians,

2. in the direction orthogonal to l_i, the beginning and end points of d_j have a maximum distance of k pixels from l_i,

3. in the direction parallel to l_i, the beginning and end points of d_j deceed those of l_i by at most $n\%$, and

4. in the direction parallel to l_i, the beginning and end points of d_j exceed those of l_i by at most $m\%$.

For conditions 3. and 4., the maximum allowed distances are $n\%$ and $m\%$ of the length of the ground truth segment l_i. To check the first condition, the angle is computed as

$$\alpha_{d_j} = \arctan\left(\frac{y_{\mathrm{e},j}^{(d)} - y_{\mathrm{b},j}^{(d)}}{x_{\mathrm{e},j}^{(d)} - x_{\mathrm{b},j}^{(d)}}\right) \tag{3.5}$$

for every detected line segment and similarly for the ground truth line segments. The result is a value between $-\pi$ and π. The angles α_{d_j} and α_{l_i} match if either:

1. $|\alpha_{d_j} - \alpha_{l_i}| \leq r$, or

2. $|\pi - |\alpha_{d_j} - \alpha_{l_i}|| \leq r$.

For all matching line segments, the first case will be assumed in the following. The first case can always be produced from the second one by flipping the beginning and end points in one of the segments. It makes sense to check this condition first for any given possible match because it is more efficient to check than the other conditions.

The other conditions are checked by analyzing the vector

$$\vec{v}_{i,j} = \begin{pmatrix} x_{b,j}^{(d)} - x_{b,i} \\ y_{b,j}^{(d)} - y_{b,i} \end{pmatrix} \tag{3.6}$$

spanned from the beginning point of l_i to the beginning point of d_j. The relative distance of the beginning points in the direction parallel to l_i is computed as the scalar projection:

$$s_{i,j} = \frac{v_{i,j} \cdot w_{l_i}}{\|w_{l_i}\|_2^2} \text{ with } \vec{w}_{l_i} = \begin{pmatrix} x_{e,i} - x_{b,i} \\ y_{e,i} - y_{b,i} \end{pmatrix}. \tag{3.7}$$

In this equation, $\vec{w}_{l_i}$ represents the vector spanned from the beginning to the end point of l_i. The distance $s_{i,j}$ is relative to the length of the line segment l_i. For example, if the detected beginning point was directly in the middle of the ground truth line segment, $s_{i,j}$ would be 0.5. The value of $s_{i,j}$ can be positive, negative, or zero.

1. Case $s_{i,j} > 0$: this means a part of the ground truth line segment l_i failed to be detected. In this case, a match occurs if $s_{i,j} < n\%$.

2. Case $s_{i,j} < 0$: this means the detected beginning point exceeds the ground truth line. A match occurs if $s_{i,j} > -m\%$.

3. Case $s_{i,j} = 0$: the detected beginning point does neither deceed nor exceed the ground truth line.

Additionally, the distance $o_{i,j}$ of the detected beginning point in the direction orthogonal to the ground truth line must be taken into account This distance is computed as:

$$o_{i,j} = \|\vec{v}_{i,j} - s_{i,j} \cdot \vec{w}_{l_i}\|_2. \tag{3.8}$$

This is the scalar rejection. An illustration of an example ground truth line l_1 and two detected lines d_j, with the resulting vectors and distances used in the evaluation method, is given in Figure 3.5.

The vector spanned from the end point of l_i to the end point of d_j is checked likewise. If a detected line segment fulfills all conditions for one of the ground truth line segments, it is considered to be a TP. Several detected line segments that match the same ground truth line segment are counted as one single TP. Detected line segments that do not have a matching line segment in the ground truth data are considered as FP. Every ground truth line segment without a detected line segment match is counted as FN. From these values, the quality of the detection can be quantified via precision, recall, and the F1 score.

The precision of a detector is the percentage of correctly detected lines within the set of all detections. It can be computed as:

$$P = \frac{TP}{TP + FP}. \tag{3.9}$$

A detector's recall is defined as the percentage of correctly detected lines within the set of all lines in the ground truth. This value can be computed as:

$$R = \frac{TP}{TP + FN}. \tag{3.10}$$

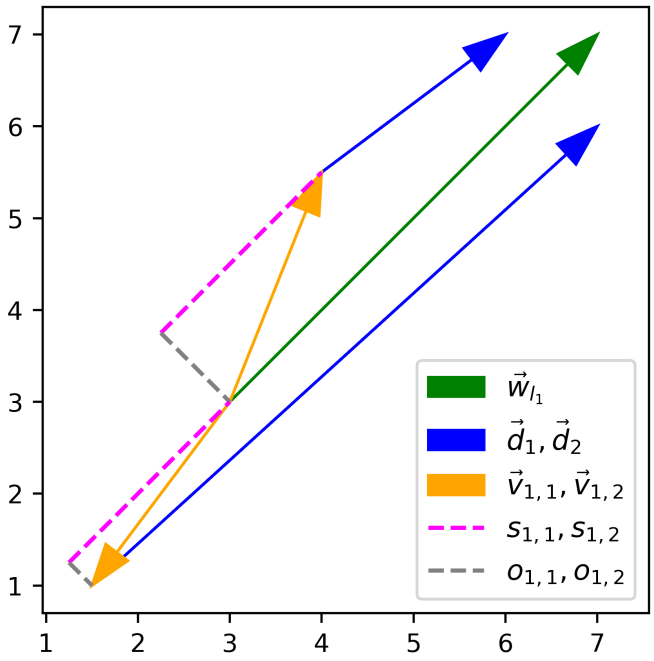


Figure 3.5: Example of the vectors and distances used in the segment-based evaluation algorithm. The green vector $\vec{w}_{l_1}$ represents a true line. The vectors $\vec{d}_1$ and $\vec{d}_2$, representing two detected lines, are marked in blue. Marked in orange are the vectors $\vec{v}_{1,j}$ spanned from the beginning point of the true line to the beginning point of the detected lines. From these vectors, the distances between the true beginning and both the detection in the direction parallel to the true line ($s_{1,j}$, magenta) and orthogonal to the true line ($o_{1,j}$, gray) are determined.

Both precision and recall are important metrics for the evaluation of a detector. It is insufficient to evaluate only one of them because precision does not take into account the FN, i. e. lines that were not detected, and recall does not take into account the FP, i. e. false detections. Therefore, a compromise between precision and recall has to be made. The F1 score quantifies such a compromise as a scalar value between 0 and 1. It is defined as:

$$F1 = 2\frac{P \cdot R}{P + R}. \tag{3.11}$$

In the worst case, all detected line segments have to be compared to all ground truth line segments. This can make the evaluation inefficient on some images. For the CSD data set, the number of ground truth line segments is always small[1]. Therefore, the efficiency of the algorithm is acceptable for the data used in this thesis.

[1]The highest number of line segments for an image in the used data set is eight. Most images contain even less line segments.

4 Parameter Optimization

The edge and line detection methods presented in Chapter 2 have several free parameters. Manually selecting the optimal parameters for the CSD data set can be very time-consuming. Therefore, optimal parameters are automatically determined in this thesis. This was done using the differential evolution algorithm [45] and the evaluation methods described in Sections 3.3 and 3.4 to maximize the goodness on a manually labeled data set.

Differential evolution is a minimization method that does not require the objective function to be differentiable and therefore does not require a gradient of the objective function. It works by stochastically generating trial parameter vectors by mutating the elements within a population of vectors. Since differential evolution is a minimization method, it is applied to the negative evaluation scores. Each parameter requires real-valued constraints. Differential evolution is defined for real-valued vectors, but some of the described methods use integer parameters. In this thesis, this is solved by rounding these parameters to the nearest integer before applying them to the detection method. Consequently, the objective function is not continuous in these cases.

The results for the parameter optimization process on the edge detectors, line detectors, and postprocessing methods are given in Sections 4.1 and 4.2.

4.1 Optimization Results for Edge and Line Detectors

The free parameters for the edge detection methods are optimized by maximizing the average score of the pixel-wise evaluation, as described in Section 3.3, over the optimization data set. For the ground truth lines in the multiclass image I_{T}, a line thickness of three pixels is chosen. Therefore, detected edge pixels are allowed to deviate by a distance of one pixel from the ground truth lines. The resulting optimized parameters are given in Table 4.1.

The ranges within which the parameters are optimized are determined experimentally.

For the gPb, only the quantile for the probability threshold is optimized because of the high run time of the algorithm. The default parameters suggested by the authors are used for the radius length σ and for the coefficients used to combine the feature channels.

Table 4.1: Edge detector parameters with their data types, ranges, and optimized results.

Method	Parameter	Type	Range	Result
GMT	Quantile for gradient threshold	real	$[0.7; 1.0]$	0.99
Canny	Quantile for lower gradient threshold	real	$[0.7; 1.0]$	0.91
	Quantile for higher gradient threshold	real	$[0.8; 1.0]$	0.9996
CannyPF	Vision-meaningful parameter[1]	real	$[200; 700]$	463
ED	Quantile for gradient threshold	real	$[0.7; 1.0]$	0.74
	Quantile for anchor threshold	real	$[0.8; 1.0]$	0.93
	Scan interval	integer	$[1; 5]$	1
gPb	Quantile for probability threshold	real	$[0.7; 1.0]$	0.99

Similarly, free parameters for the line detection methods are optimized by maximizing the average F1 score, as described in Section 3.4, over the optimization data set. For this purpose, the values $r = 0.15$ radians, $k = 3$, $n = 70\%$, and $m = 15\%$ are used.

Detectors that require an edge map as input are combined with the CannyPF edge detector. CannyPF is chosen because it leads to better results than the other edge detectors evaluated in this thesis, as explained in Section 5.1. An exception is the EDLines algorithm, which uses ED, as it requires its input edges to be grouped into edge chains.

The resulting optimized parameters for the voting-based detection methods are listed in Table 4.2. It can be observed that the numbers of discrete angles used by KHT and PPHT are much higher than those of SHT and PCLines. This shows that the improved voting schemes of these methods allow voting within a more detailed parameter space.

Table 4.2: Parameters of voting-based line detectors with their data types, ranges, and optimized results.

Method	Parameter	Type	Range	Result
SHT	Number of discrete angles	integer	$[50; 100]$	70
	Voting threshold	integer	$[40; 90]$	76
KHT	Minimal cluster size	integer	$[5; 20]$	12
	Minimal kernel height	real	$[10^{-4}; 0.1]$	0.05
	Number of discrete angles	integer	$[90; 1800]$	760
	Voting threshold	integer	$[40; 90]$	76
PPHT	Maximum gap between pixels of a line	integer	$[5; 20]$	5
	Number of discrete angles	integer	$[90; 1800]$	1214
	Voting threshold	integer	$[5; 20]$	12
PCLines	Quantile for gradient threshold	real	$[0.8; 1]$	0.97
	Accumulator resolution	integer	$[64; 256]$	234

[1]The vision-meaningful parameter of the CannyPF algorithm is used for the validation of edge segments. Its exact purpose is beyond the scope of this thesis.

For the region-based line detection methods, the results of the optimization process are listed in Table 4.3. It can be observed that the optimized scaling factor of 81% for the LSD algorithm is very close to the factor of 80% which the authors recommend.

Table 4.3: Parameters of region-based line segment detectors with their data types, ranges, and optimized results.

Method	Parameter	Type	Range	Result
LSD	Scaling factor	real	$[0.5; 1]$	0.81
	Angle tolerance for NFA	real	$[1; 50]$	8.3
EDLines	Line fitting error threshold	real	$[0.5; 5]$	1.5
ELSED	Quantile for gradient threshold	real	$[0.5; 1]$	0.90
	Scan interval	integer	$[1; 5]$	4
SSWMS	Accuracy	integer	$[5; 15]$	9
	Maximum number of lines	integer	$[5; 15]$	9
ELSDc	Angle tolerance for NFA (degrees)	real	$[1; 50]$	12.2
FLD	Maximum distance of point to line	real	$[0.5; 2]$	0.78
Yuan	Quantile for gradient threshold	real	$[0.5; 1]$	0.78
Linelet	Quantile for gradient threshold	real	$[0.5; 1]$	0.78

4.2 Optimization of Postprocessing Methods

The parameters for the validation method using the Helmholtz principle, which is described in Section 2.5.1, are not optimized for the validation of the results of any specific line detector. Instead, the parameters are optimized for the validation of a set S_{H} of line segments which contains both ground truth line segments and randomly created false line segments. For each image in the optimization data set, S_{H} contains as many randomly created false line segments as true line segments. These false line segments are created by randomly selecting coordinates for the beginning and end points from a uniform distribution. It is ensured that none of the randomly created line segments match any of the ground truth line segments by the criteria used to match line segments in the segment-based evaluation method (see Section 3.4). An example image with the ground truth lines marked in green and the false, randomly created lines marked in red is displayed in Figure 4.1. The F1 score (see Section 3.4) of the resulting line segments is maximized by the optimization algorithm. The threshold the NFA (see Section 2.5.1) is compared to can be selected from a wide range, including both very large values and values close to 0. Therefore, the threshold is defined in the form 10^t, with t being the parameter that is optimized by the algorithm.

For the parameter optimization of the line segmentation method described in Section 2.5.3, the used binary edge maps are created from the ground truth line segments instead of using any of the edge detection methods. The creation of these edge maps is done by drawing the ground truth line segments with a width of one pixel onto a binary image. Then, $E\%$ of edge pixels were randomly removed to simulate the

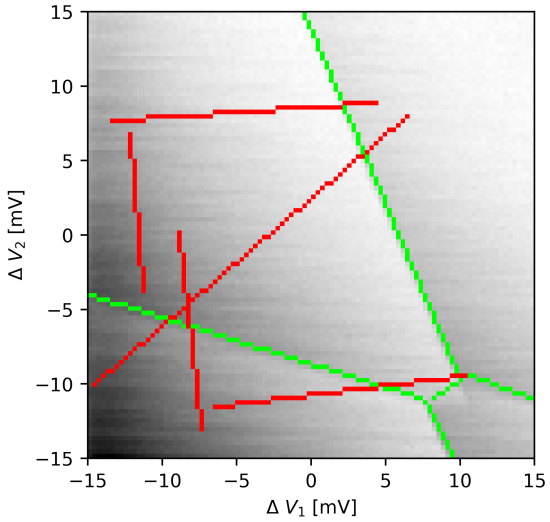


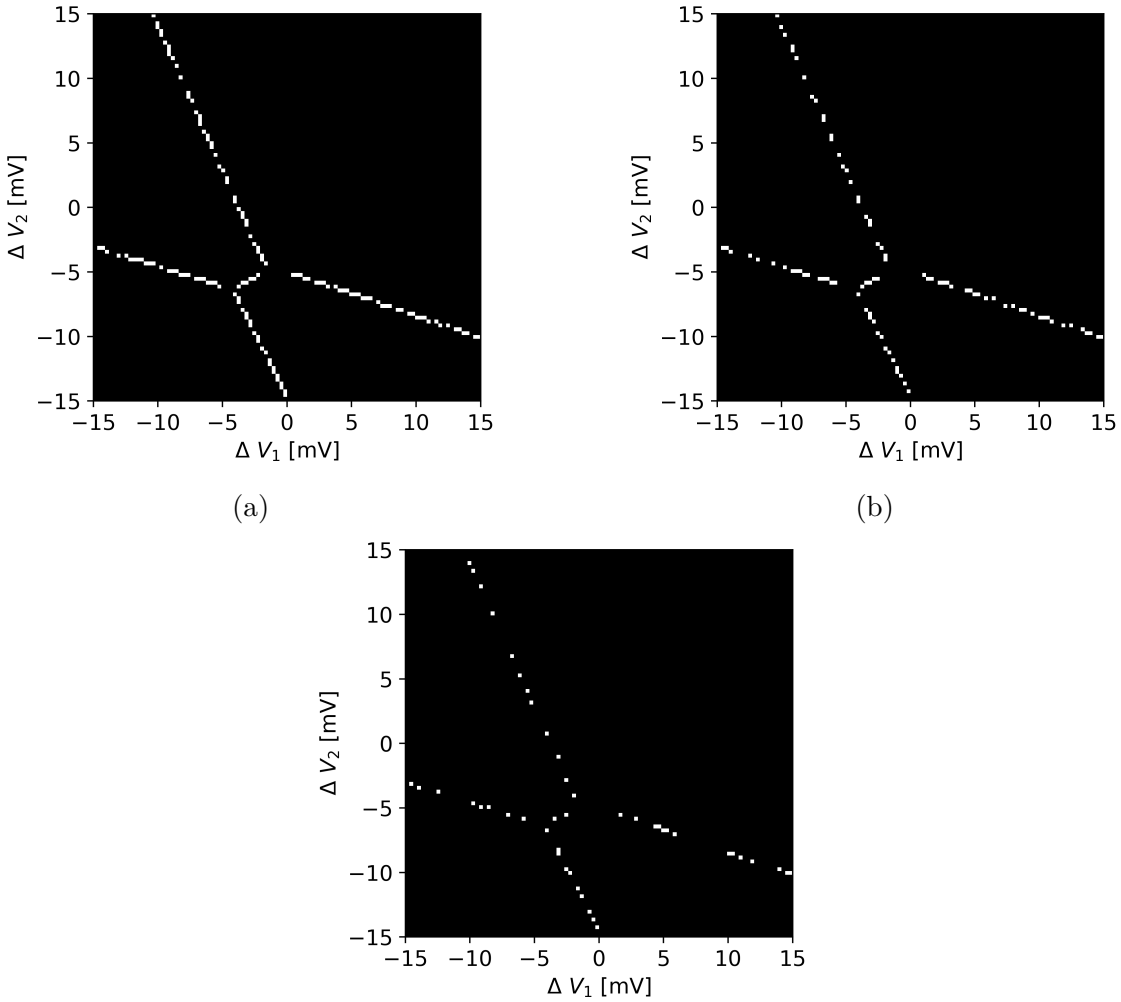
Figure 4.1: Example of a CSD used for the parameter optimization of the Helmholtz validation method. The ground truth lines are marked in green and the false, randomly created lines are marked in red.

imperfect detection quality of edge detection methods. Examples of such edge maps are displayed in Figure 4.2. As an optimal parameter for the maximal gap between edge pixels, the minimal value, which leads to a perfect reconstruction of the finite line segments from their corresponding infinite lines, is chosen. The selection of this value is done manually to ensure that the minimal value is selected.

The results of the parameter optimization for the postprocessing methods are given in Table 4.4. For the line segmentation method, an optimal parameter is determined for $E = 25\%$, $E = 50\%$, and $E = 75\%$, respectively. The optimized parameter for the NFA threshold is $10^{-0.37} \approx 0.43$. The FSG algorithm has no free parameters that need to be optimized.

Table 4.4: Parameters of postprocessing methods with their data types, ranges, and optimized results. No ranges are given for the segmentation parameters because they are chosen manually.

Method	Parameter	Type	Range	Result
Validation	Number of quantized angles	integer	$[4; 16]$	4
	NFA threshold (as exponent)	real	$[-10; 10]$	-0.37
Segmentation	Maximum gap ($E = 25\%$)	integer	n. a.	5
	Maximum gap ($E = 50\%$)	integer	n. a.	6
	Maximum gap ($E = 75\%$)	integer	n. a.	15



(a)

(b)

(c)

Figure 4.2: Examples of binary edge maps with (a) 25%, (b) 50%, and (c) 75% of edge pixels removed.

5 Results

This chapter presents the results produced by the detection and postprocessing methods and discusses their practical implications. In Sections 5.1 to 5.3, the evaluation scores on the optimization and evaluation data sets are analyzed for the edge detectors, line detectors, and postprocessing methods, respectively. Section 5.4 describes the implications for the analysis of CSDs.

5.1 Edge Detection Results

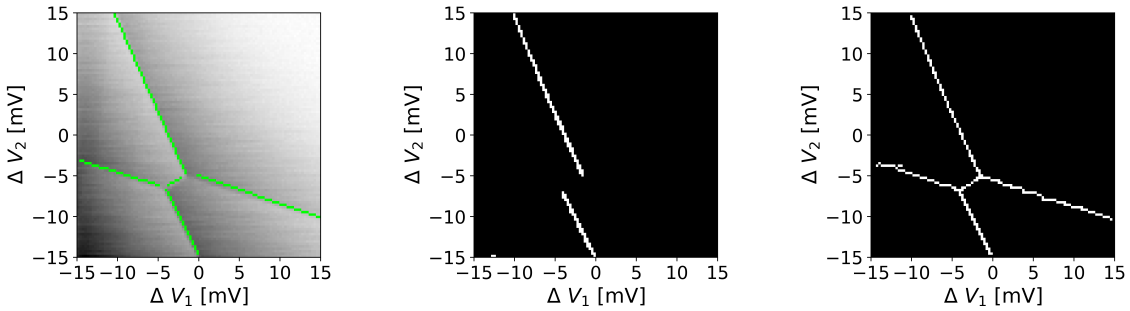
The results for the edge detection methods using optimized parameters on the optimization data set and validation data set are displayed in Table 5.1. The evaluation scores are determined using the method described in section 3.3. In addition to the final evaluation score $\bar{p} - E_{\mathrm{FP}}$, the individual values for the mean $\bar{p}$, which is based on the percentage of correctly detected edge pixels per ground truth segment, and for the error term E_{FP}, which is based on the false positive edge detections, are given.

It can be observed that CannyPF receives the best scores on the optimization data set. On the validation data set, CannyPF is outperformed by other detectors on images of mediocre quality. However, no detector delivers results that are practically usable on images of mediocre or low quality. It should also be noted that the evaluation method is slightly biased towards a method that detects thick edges, such as GMT. Additionally, the individual values for $\bar{p}$ and E_{FP} indicate that CannyPF produces more correct detections at the cost of more incorrect detections than other methods. Such a behavior is desirable for the application that motivates this thesis, because correctly detecting as many lines as possible is critical for the analysis of CSDs, while false detections can be accounted for by subsequent steps. This confirms the choice of CannyPF as the most useful edge detector, even though no detector consistently shows the best results on every data set.

Overall, the results for the optimization data set are roughly equal to those for the validation data set. The five cases in which the optimization data set receives a significantly better (at least 0.1 higher) score are likely to be due to the small number of images in each data set. This also explains both the cases in which the validation data set receives a better score than the optimization data set and in which a data set of lower quality receives a higher score than a data set of higher quality on the same detector.

Table 5.1: Edge detector results for the optimization and validation data sets. The detectors are ordered by the evaluation score on the validation data set.

		Optimization data			Validation data		
Data quality	Method name	$\bar{p}$	E_{FP}	$\bar{p} - E_{\text{FP}}$	$\bar{p}$	E_{FP}	$\bar{p} - E_{\text{FP}}$
Very high	Canny	0.70	0.03	0.67	0.79	0.03	0.76
	CannyPF	0.80	0.06	0.74	0.84	0.09	0.75
	GMT	0.79	0.09	0.70	0.78	0.08	0.70
	gPb	0.47	0.05	0.42	0.59	0.02	0.57
	ED	0.63	0.02	0.61	0.51	0.03	0.48
High	CannyPF	0.84	0.15	0.69	0.78	0.22	0.56
	GMT	0.70	0.18	0.52	0.63	0.10	0.53
	Canny	0.79	0.13	0.66	0.55	0.06	0.49
	ED	0.62	0.01	0.61	0.48	0.03	0.45
	gPb	0.65	0.03	0.62	0.34	0.02	0.32
Mediocre	GMT	0.53	0.38	0.15	0.52	0.26	0.26
	Canny	0.13	0.03	0.10	0.23	0.05	0.18
	ED	0.12	0.05	0.07	0.20	0.06	0.14
	CannyPF	0.22	0.00	0.22	0.27	0.14	0.13
	gPb	0.09	0.09	0.00	0.14	0.11	0.03
Low	CannyPF	0.11	0.11	0.00	0.23	0.15	0.08
	ED	0.00	0.00	0.00	0.10	0.06	0.04
	gPb	0.07	0.07	0.00	0.28	0.26	0.02
	GMT	0.13	0.13	0.00	0.28	0.28	0.00
	Canny	0.04	0.04	0.00	0.09	0.09	0.00



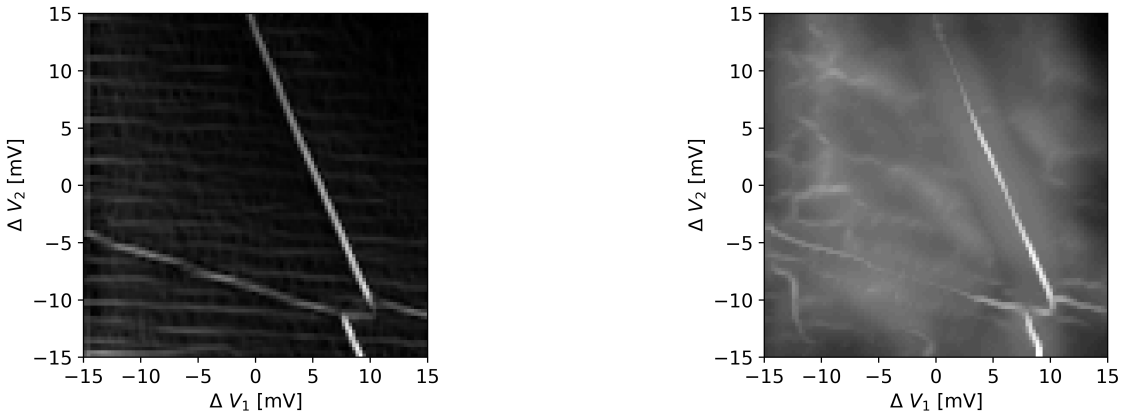
(a) Ground truth (b) GMT (c) Canny

(d) CannyPF (e) ED (f) gPb

Figure 5.1: A CSD with (a) ground truth lines and (b-f) the binary edge maps produced by the edge detectors.

A qualitative analysis confirms the quantitative results. Additionally, it can be observed that the edges detected by Canny, CannyPF, and ED are longer and contiguous, while those detected by GMT and gPb often consist of scattered individual pixels in some areas and thick edge segments in other areas. Figure 5.1 displays the edge detection results for an example CSD.

A comparison between GMT and gPb shows that thresholding the gradient magnitude image produces better results than thresholding the probability map created by gPb. Figure 5.2 displays (a) a gradient magnitude image created using the Sobel operator, and (b) the probability map created using gPb. It can be observed that both maps are disturbed by the noise present in the underlying image, but *gPb* more strongly than the gradient magnitude map.

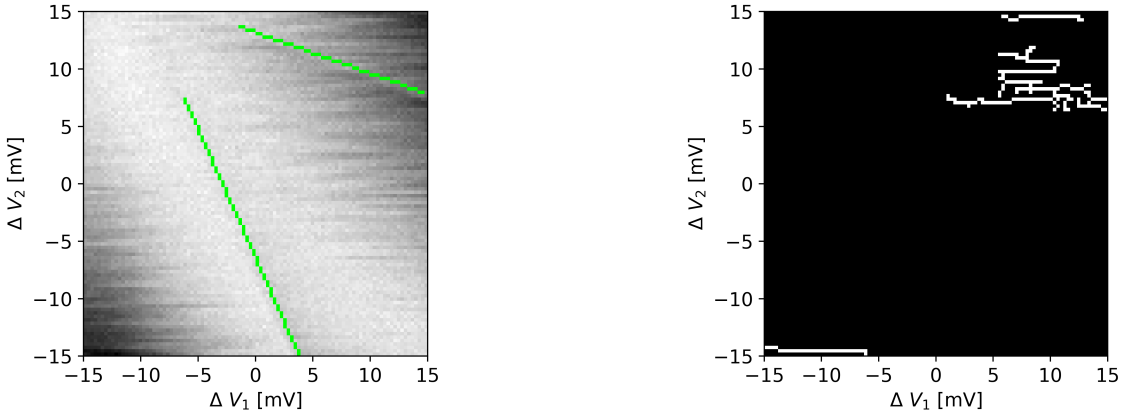


(a) Gradient magnitude

(b) *gPb*

Figure 5.2: Example of (a) a gradient magnitude map created using the Sobel operator, and (b) the probability-of-boundary map created using gPb.

For the images of mediocre and low quality, no edge detector produces usable results, as examplarily shown in Figure 5.3.



(a)

(b)

Figure 5.3: Example of (a) a CSD of low quality and (b) the unusable edge map produced by CannyPF.

5.2 Line Detection Results

The results for the voting-based line detection methods using optimized parameters and the edge map created by CannyPF, are displayed in Table 5.2. The values for precision, recall, and F1 score are determined using the evaluation method described in section 3.4 and averaged over the images in the data sets. For the SHT, KHT, and PCLines algorithms, the results were additionally segmented to acquire finite line segments. A maximum gap of six pixels is used for the segmentation algorithm. As the results in Section 5.3 show, the choice of this parameter has a negligible impact on the quality of the line segmentation.

Table 5.2: Voting-based line detector results for the optimization and validation data sets. The detectors are ordered by the F1 score on the validation data set.

		Optimization data			Validation data		
Data quality	Method	P	R	$F1$	P	R	$F1$
Very high	PPHT	0.72	0.57	0.63	0.66	0.61	0.60
	KHT (segmented)	0.94	0.43	0.56	0.86	0.43	0.57
	SHT (segmented)	0.50	0.15	0.23	0.56	0.18	0.26
	PCLines (segmented)	0.50	0.22	0.31	0.42	0.10	0.16
High	PPHT	0.62	0.89	0.65	0.45	0.53	0.44
	KHT (segmented)	0.48	0.53	0.50	0.41	0.29	0.31
	SHT (segmented)	1.00	0.61	0.68	0.40	0.13	0.20
	PCLines (segmented)	0.25	0.06	0.10	0.35	0.15	0.20
Mediocre	PPHT	0.14	0.12	0.13	0.22	0.17	0.15
	KHT (segmented)	0.25	0.09	0.14	0.18	0.15	0.12
	SHT (segmented)	0.25	0.03	0.06	0.17	0.06	0.06
	PCLines (segmented)	0.12	0.03	0.05	0.04	0.02	0.03
Low	PPHT	0.03	0.12	0.04	0.12	0.19	0.12
	KHT (segmented)	0.03	0.25	0.05	0.08	0.32	0.12
	SHT (segmented)	0.00	0.00	0.00	0.00	0.00	0.00
	PCLines (segmented)	0.00	0.00	0.00	0.00	0.00	0.00

In regards to the recall values, the KHT and PPHT algorithms produce better results than the other voting-based methods. This can be explained by the KHT's and PPHT's improved voting schemes, which make the algorithms better at the detection of short lines in the CSD data. The PCLines algorithm produces very poor results in terms of both precision and recall.

For the region-based line detectors, the precision, recall and F1 score values are listed in Table 5.3 using the optimized parameters. The difference to default parameters is discussed below. In terms of the F1 scores, CannyLines produces the best detection results for data of high and very high quality. ELSDc achieves comparable recall values, but has a much lower precision. CannyLines also outperforms all of the voting-based line detection methods. Therefore, CannyLines is the preferable method for the data used in this thesis. For a selection of line detectors, the

detection results on an example CSD are displayed in Figure 5.4.

Table 5.3: Region-based line detector results for the optimization and validation data sets using optimized parameters. The detectors are ordered by the F1 score on the validation data set.

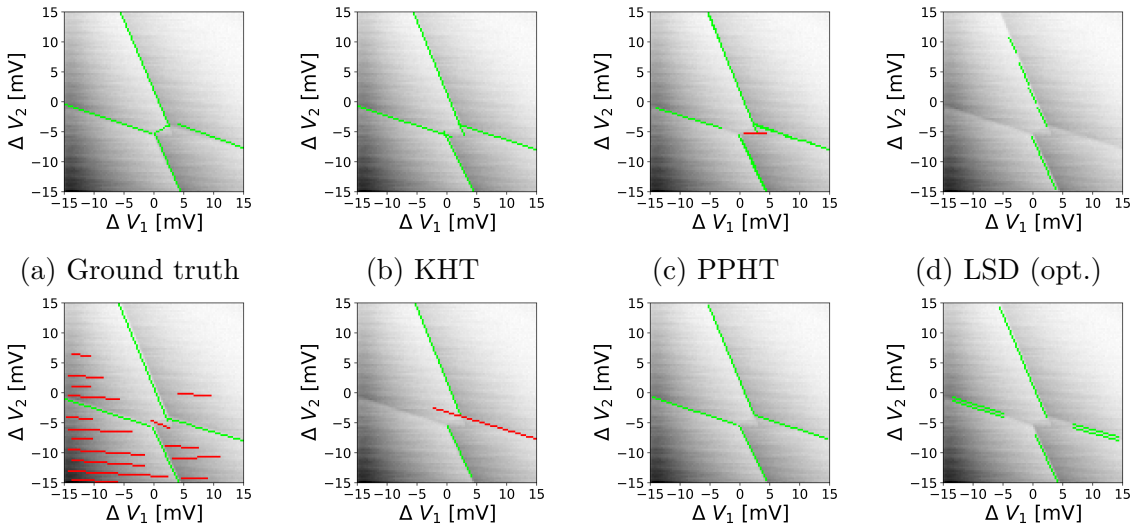
		Optimization data			Validation data		
Data quality	Method	P	R	$F1$	P	R	$F1$
Very high	CannyLines	0.90	0.62	0.73	0.90	0.75	0.80
	LSD	0.88	0.55	0.65	0.87	0.52	0.61
	ELSED	0.61	0.53	0.51	0.56	0.65	0.54
	FLD	0.52	0.25	0.31	0.79	0.43	0.53
	Linelet	0.56	0.47	0.49	0.47	0.43	0.43
	ELSDc	0.63	0.85	0.64	0.21	0.76	0.30
	Yuan	0.23	0.15	0.17	0.18	0.22	0.2
	SSWMS	0.75	0.15	0.25	0.39	0.10	0.15
	EDLines	0.00	0.00	0.00	0.17	0.03	0.06
High	CannyLines	0.94	0.89	0.91	0.67	0.58	0.60
	Linelet	0.67	0.84	0.71	0.42	0.50	0.45
	FLD	0.60	0.53	0.56	0.70	0.35	0.44
	LSD	0.75	0.60	0.64	0.60	0.34	0.42
	ELSED	0.41	0.59	0.37	0.52	0.40	0.38
	ELSDc	0.30	0.95	0.40	0.25	0.51	0.32
	SSWMS	0.38	0.30	0.32	0.40	0.08	0.13
	Yuan	0.15	0.16	0.15	0.09	0.12	0.10
	EDLines	0.25	0.25	0.25	0.15	0.04	0.06
Mediocre	ELSDc	0.17	0.12	0.14	0.28	0.56	0.34
	ELSED	0.25	0.09	0.14	0.42	0.31	0.31
	LSD	0.25	0.06	0.10	0.31	0.19	0.19
	CannyLines	0.12	0.25	0.17	0.28	0.15	0.18
	FLD	0.25	0.09	0.14	0.27	0.15	0.17
	Linelet	0.12	0.09	0.11	0.21	0.17	0.17
	SSWMS	0.00	0.00	0.00	0.33	0.08	0.13
	Yuan	0.10	0.06	0.08	0.07	0.07	0.06
	EDLines	0.00	0.00	0.00	0.08	0.02	0.03
Low	Linelet	0.08	0.25	0.12	0.12	0.30	0.16
	ELSDc	0.02	0.25	0.04	0.09	0.54	0.14
	SSWMS	0.00	0.00	0.00	0.11	0.11	0.11
	ELSDc	0.00	0.00	0.00	0.05	0.27	0.08
	CannyLines	0.00	0.00	0.00	0.06	0.11	0.07
	FLD	0.00	0.00	0.00	0.11	0.02	0.04
	Yuan	0.00	0.00	0.00	0.03	0.11	0.04
	Others	(all 0.00)					

In the publications of the algorithms LSD and EDLines, default parameters are given by the respective authors. The results produced using these default parameters are given in Table 5.4. For LSD, the F1 scores are much lower when using the default

parameters. While the recall values are slightly higher for the default parameters, meaning that more ground truth lines were correctly detected, the precision scores are extremely low, meaning that for every correct detection, there are several false positive detections. This can be explained by the lower angle tolerance parameter in the optimized parameter set, which leads to less acceptance of false detections, but also of correct detections. However, it should be noted that the LSD algorithm using default parameters still performs better than several other line detectors for which the parameters have been optimized. An example that represents the difference between the optimized and standard LSD is displayed in Figure 5.4 (d) and (e). For EDLines, the results are very poor for both the optimized and the default parameters.

Table 5.4: Region-based line detector results for the optimization and validation data sets using default parameters suggested by the authors.

		Optimization data			Validation data		
Method	Data quality	P	R	$F1$	P	R	$F1$
LSD	Very high	0.25	0.75	0.37	0.32	0.82	0.45
	High	0.17	0.62	0.27	0.2	0.51	0.28
	Mediocre	0.06	0.12	0.08	0.14	0.5	0.2
	Low	0.01	0.25	0.02	0.04	0.32	0.06
EDLines	Very high	0.0	0.0	0.0	0.0	0.0	0.0
	High	0.08	0.25	0.12	0.04	0.08	0.05
	Mediocre	0.0	0.0	0.0	0.04	0.02	0.03
	Low	0.0	0.0	0.0	0.0	0.0	0.0



(a) Ground truth (b) KHT (c) PPHT (d) LSD (opt.)

(e) LSD (def.) (f) ELSED (g) CannyLines (h) FLD

Figure 5.4: A CSD with (a) ground truth lines and (b-h) the lines found by the detectors.

For both voting-based and region-based detectors, a qualitative analysis shows that lines with a weak gradient are significantly more problematic than the noise that is found in many images. In images in which the lines have a weak gradient, a correct detection is rarely made by any of the detection methods, even if the image is not

disturbed by strong noise. This effect leads to interdot transitions being missed particularly often because they typically have a weaker gradient than the lead-to-dot transitions. An example is the interdot transition of the CSD in Figure 5.4, which all methods failed to detect. While the noise poses a less critical problem than the weak lines, it also reduces the quality of the detection results. In areas where the noise is too strong, false positive detections occur.

To conclude, CannyLines produces the best results on the data sets of high and very high quality out of all detectors that were evaluated. KHT, PPHT, LSD, FLD, and the linelet-based method achieve relatively robust results, but are outperformed by CannyLines. On the data sets of mediocre and low quality, no detector produces usable results.

5.3 Postprocessing Results

In this section, the results for the postprocessing methods using the optimized parameters are given. Since the result of a postprocessing method depends on the quality of the results of the preceding detection, no absolute values are shown. Instead, the values listed in the tables account for the changes in the averages of precision, recall, and F1 score that are achieved by the postprocessing methods. The changes for the data sets of mediocre and low quality are not listed because no detector produces usable results for them, as shown in the previous section.

The results of the line segmentation method for the infinite line detectors are listed in Table 5.5. As explained in Section 4.2, the preferred values for the maximum gap parameter are 15, 6, and 5 depending on the quality of the edge map. The corresponding methods are referred to as seg15, seg6, and seg5 in the table. It can be observed that the results for different parameters mostly differ marginally. Small improvements are due to some infinite lines not matching the infinite ground truth lines, even though the shorter segmented line segments match the ground truth line segments. This can happen because the absolute distance of a beginning or end point from a ground truth line ($o_{i,j}$ in equation (3.7) in Section 3.4) increases with the length of the lines. On the other hand, false line segmentations, such as a segmentation into a line segment that is too long to match the ground truth, lead to decreases in precision and recall for some images. For the results of the KHT algorithm on the high-quality optimization data set, this happens for a significant number of lines. However, a qualitative analysis shows that these false line segments are still relatively close to the ground truth.

The results of the validation method based on the Helmholtz principle for a selection of line detection methods are given in Table 5.6. For the KHT algorithm, the detected infinite lines are segmented into finite line segments before applying the validation. In terms of precision, the results are mixed because both false and true lines are rejected by the validation. The changes in recall values are in many cases negative and lower than -0.1. This indicates that a significant number of true lines are rejected. Note that a validation method cannot increase the number of correct

Table 5.5: Changes in evaluation scores for applying the line segmentation method using three different parameter values to the ground truth (with lines extend to the image borders), and to the detection results of SHT and KHT. Note that the evaluation works differently for infinite and finite lines because they are compared to different ground truths.

		Optimization data			Validation data		
Method	Data quality	P	R	$F1$	P	R	$F1$
SHT + seg15	Very high	0.0	0.0	0.0	-0.11	-0.03	-0.05
	High	0.0	0.0	0.0	-0.2	-0.04	-0.07
SHT + seg6	Very high	0.0	0.0	0.0	-0.11	-0.03	-0.05
	High	0.0	0.0	0.0	-0.1	-0.02	-0.03
SHT + seg5	(same as above)						
KHT + seg15	Very high	-0.06	-0.05	-0.06	0.0	0.0	0.0
	High	-0.52	-0.36	-0.44	-0.22	-0.2	-0.17
KHT + seg6	Very high	-0.06	-0.05	-0.06	0.0	0.0	0.0
	High	-0.52	-0.36	-0.44	-0.21	-0.18	-0.16
KHT + seg5	(same as above)						

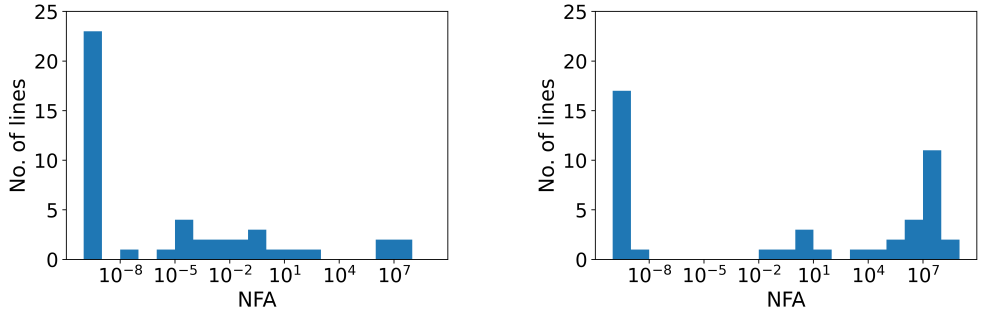
detections and that a positive change in recall is therefore generally impossible for a validation step. The best changes are achieved for ELSED. This is because ELSED detects a relatively large number of false positives, which are rejected by the validation, and fails to detect many true positives that could otherwise be rejected by the validation.

Table 5.6: Changes in evaluation scores for applying the validation method to the ground truth and to a selection of line detectors.

		Optimization data			Validation data		
Method	Data quality	P	R	$F1$	P	R	$F1$
Ground truth + val.	Very high	0.0	-0.12	-0.08	0.0	-0.11	-0.06
	High	0.0	-0.11	-0.06	0.0	-0.19	-0.12
KHT (segm.) + val.	Very high	-0.25	-0.12	-0.17	0.0	0.0	0.0
	High	0.06	-0.05	0.0	0.01	0.0	0.01
PPHT + val.	Very high	-0.25	-0.12	-0.17	0.15	-0.04	0.03
	High	-0.02	-0.05	-0.03	0.01	0.0	0.02
LSD + val.	Very high	0.0	-0.18	-0.14	-0.02	-0.11	-0.09
	High	0.0	-0.11	-0.1	0.1	0.0	0.02
ELSED + val.	Very high	0.09	-0.12	-0.02	0.04	0.0	0.02
	High	0.02	-0.11	-0.07	0.04	0.0	0.01
CannyLines + val.	Very high	-0.21	-0.12	-0.15	0.04	-0.11	-0.05
	High	0.06	-0.05	-0.01	0.04	-0.06	-0.04

The lack of significant improvements after the application of the validation step can be explained by the distribution of the NFA values for the ground truth lines and

the false positive detections. Figure 5.5 displays histograms of the NFA values for (a) ground truth lines in the optimization data, (b) false randomly created lines, and (c) false detections of PPHT. Approximately 16% of the ground truth lines and 58% of the randomly created lines have an NFA value of 1 or higher. Therefore, the validation method rejects random lines much more often than true lines. However, all false detections of the PPHT method[1] have a very small NFA value. Consequently, no false positives are rejected by the validation method. This analysis shows that line validation using the Helmholtz principle is useful for the removal of random lines, but not for the removal of false detections. Therefore, line validation using the Helmholtz principle does not lead to improvements for the data set used in this thesis.



(a) Ground truth lines

(b) Randomly created lines

(c) False PPHT detections

Figure 5.5: Histograms of the NFA values for (a) the ground truth lines, (b) the randomly created lines from the optimization data, and (c) the false detections of PPHT. Note the logarithmic scale of the x-axis.

The changes in evaluation scores when applying FSG to a selection of line detectors are listed in Table 5.7. Additionally, the impact of FSG to the ground truth is given. The results contain small, but significant changes. A qualitative analysis shows that the changes are mostly due to the merging step. Some long lines are detected as separate, short lines. The merging step connects these into longer lines, leading to improvements in the evaluation. Figure 5.6 (b) shows an example of this. Decreases in the evaluation scores also appear in several cases. Some lines are separate but roughly parallel and closely located. These lines are erroneously merged by FSG. This effect occurs even if FSG is applied to the ground truth. An example of this effect is shown in Figure 5.6 (d). For the KHT algorithm, FSG does

[1]The histograms for the other line detectors are similar. PPHT is chosen as an example because it is the worst case, in which all false detections have a very small NFA value.

not lead to any changes in the evaluation scores. This is because KHT often detects long and contiguous lines, as opposed to short and separate ones. In conclusion, FSG produces too many false positives by erroneous merging of lines to be useful for the CSD data set.

Table 5.7: Changes in evaluation scores for applying FSG to the ground truth and to a selection of line detectors.

		Optimization data			Validation data		
Method	Data quality	P	R	$F1$	P	R	$F1$
Ground truth + FSG	Very high	0.0	0.0	0.0	-0.04	-0.07	-0.06
	High	0.0	0.0	0.0	-0.08	-0.12	-0.11
LSD + FSG	Very high	0.12	0.0	0.03	0.05	0.0	0.03
	High	0.0	0.0	0.0	-0.02	-0.02	-0.02
ELSED + FSG	Very high	0.0	0.0	0.0	0.0	0.0	0.0
	High	0.0	0.0	0.0	-0.01	-0.04	-0.02
CannyLines + FSG	Very high	0.0	0.0	0.0	0.02	0.0	0.01
	High	0.0	0.0	0.0	-0.06	-0.04	-0.04
PPHT + FSG	Very high	0.12	0.0	0.03	0.16	0.0	0.05
	High	0.09	0.0	0.08	-0.1	-0.12	-0.08
KHT + FSG	(all 0.00)						

5.4 Implications for the Analysis of CSDs

The results show that the quality of the line detection is strongly dependent on the data quality. For most images, the detection quality is insufficient for an analysis of the CSD. However, there is a significant number of images for which the detection quality is good enough. The validation data sets of high and very high quality contain 16 images. For three of these images, CannyLines (without postprocessing) detects all lines without any false positives. Additionally, there are five images in which all lead-to-dot transitions are detected, but the interdot transition is missed. An example of such a detection is displayed in Figure 5.4 (g). The missing interdot transition could potentially be reconstructed using the detected lead-to-dot transitions and the images contain no or very few false positives. Therefore, for eight out of the 16 validation images with high or very high quality (50%), the detections obtained from CannyLines are sufficient for a full analysis of the structure.

For the other eight validation images of high or very high quality, CannyLines always finds one or several lead-to-dot-transitions, but never misses all of them. The interdot transitions are missed in most cases. Nevertheless, these detection results can still be useful for subsequent CSD analysis methods. Higher quality data is necessary to reliably detect all charge transitions. Improving the signal-gradient-to-noise ratio is a promising option to increase the number of correctly detected charge transitions. For this purpose, a method to automatically tune sensor dots to

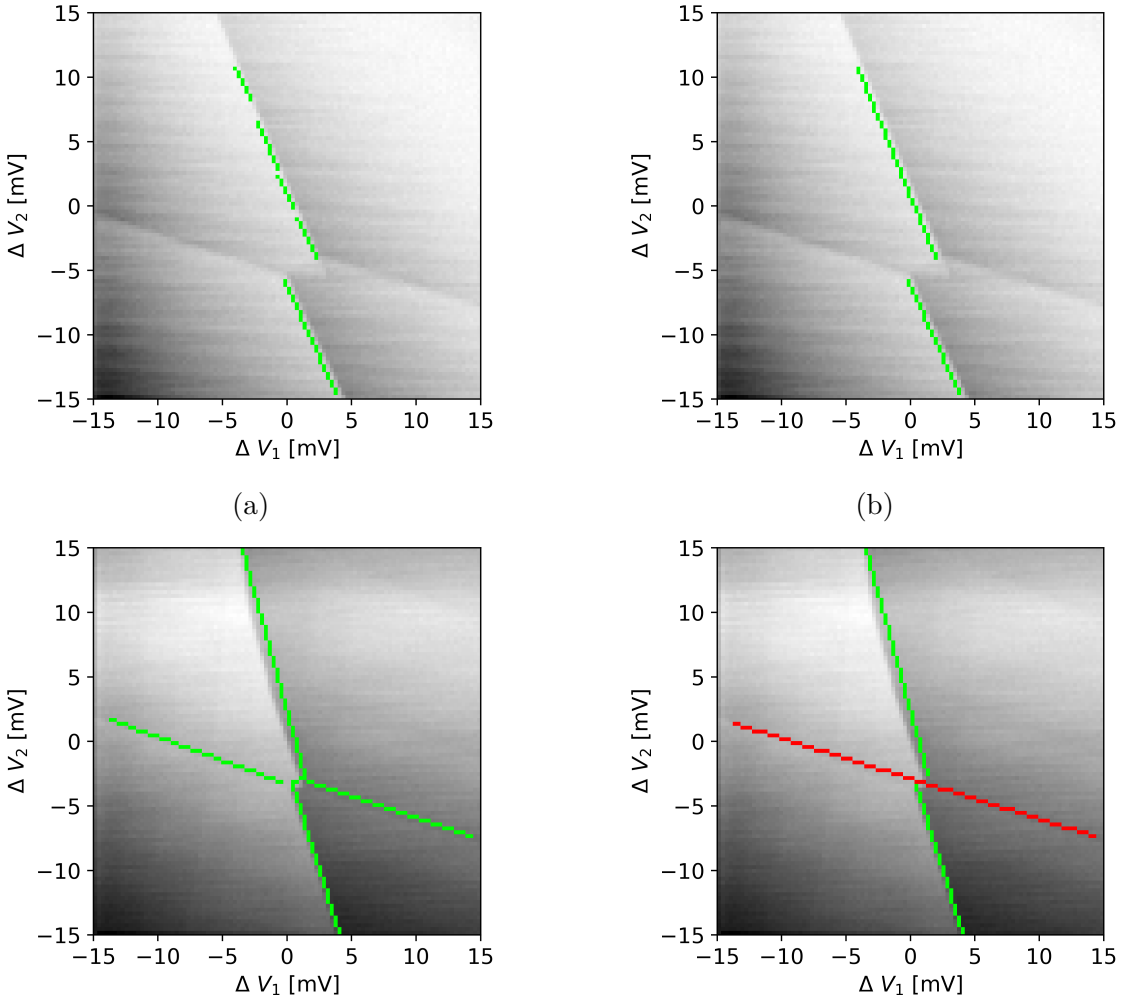


Figure 5.6: Examples of results produced by FSG: a CSD with (a) a single line detected as several short lines, (b) the reconstruction of the longer line using FSG, (c) two correctly detected lines that are parallel and closely located, and (d) an erroneous merging of the two separate lines.

be optimally sensitive to changes in its electrostatic environment was proposed by Hader et al. [46].

Since the hardware and processes used to measure CSDs are subject of ongoing development, the quality of the CSDs is expected to improve in the future. Therefore, the poor line detection results on the CSDs of mediocre and low quality pose a less critical problem for the project.

6 Summary

The analysis of a specific type of diagram, called CSD, plays an important role during the calibration of qubits in double quantum dots. CSDs are grayscale images that contain line segments. For the automatic analysis of CSDs, automatic detection of these line segments is required. However, the CSDs are disturbed by noise and many of the line segments have very weak edges. This makes a robust automatic detection of the line segments challenging.

A variety of different line detection methods were explored in this thesis representing both voting-based and region-based approaches. Since some line detection methods require a binary edge map as input, edge detection methods were analyzed as well. Additionally, some sub-methods of the detectors, which are exchangeable, were analyzed as separate methods for edge detection or postprocessing. To enable a structured evaluation and comparison, uniform software interfaces using the same programming language were defined for the methods.

Two methods for the automatic evaluation of the detection results were proposed. One of them evaluates binary pixel maps, like the ones produced by edge detectors, while the other method evaluates the list of detected line segments. Using these evaluation methods, free parameters of the edge detection, line detection, and postprocessing methods were optimized. Using automatically optimized parameters leads to better results and enables a more objective comparison of the detection qualities of the various methods.

The proposed evaluation methods compare the detection results to manually created ground truth data. This ground truth data consists of CSDs with contained line segments, defined by their beginning and end points. For the evaluation of edge detection methods, the detected edge pixels were compared to the pixels that belong to the line segments in the ground truth data. Thereby, a good representation of many lines is preferred over a perfect representation of few lines, even if both results contain the same amount of correctly detected edge pixels. Both the true positive and false negative edge detections on every ground truth line segment are taken into account. An additional error term punishes the number of false positive edge detections. For the evaluation of line detection methods, the detected line segments were compared to the ground truth line segments. A set of conditions is defined to decide whether a detected line segment matches a ground truth line segment. As some methods only detect infinite lines, they are treated in this context as finite line segments whose beginning and end points are located on the image borders. Based on the determined matches, the numbers of true positive, false positive, and false negative line segments are established. Using these numbers, the quality of

the detection result can be quantified via the F1 score. This evaluation method is intentionally designed to ignore some properties of the detection results like slight displacements of the lines or slight errors in the detected line segments' lengths. This is because such errors are of minor importance to the analysis of CSDs than completely failing to detect some of the lines.

Free parameters of the detection methods were optimized using the differential evolution algorithm to maximize the average score of the evaluation methods mentioned above. This parameter optimization was done on a data set that is separate from the one used in the final evaluation and comparison of the methods.

The CannyPF algorithm produces the best results for the selected edge detectors and CannyLines for the selected line detection methods. For both edge and line detection, the quality of the results is strongly dependent on the quality of the CSD images. Lines with a very weak gradient almost always fail to be detected. Noise and artifacts in the CSD images also pose a problem but are less critical than the weak gradient of some lines.

The postprocessing methods were evaluated based on the changes they cause on the evaluation scores. One postprocessing method is the segmentation of infinite lines into finite line segments. This method mostly works with acceptable quality. However, there are cases in which an infinite line with the correct position and angle is segmented into a line segment that is too short or too long. The validation method using the Helmholtz principle has been shown to not be useful for the data used in this thesis as it mostly cannot distinguish between true positives and false positives. The FSG method for merging and grouping line segments that belong together produces mixed results. There are both small negative and small positive changes in detection quality after applying FSG to the data sets. However, the negative effects outweigh the positive ones.

Within the validation data sets of high and very high quality, the line detection quality of CannyLines is sufficient to perform a full analysis of the structures for half of the images. The other images of high and very high quality are partly analyzable. For the data sets of mediocre and low quality, no line detector produces usable results.

7 Outlook

The detectors evaluated in this thesis do not include approaches based on machine learning. However, such approaches have led to excellent edge and line detection results in recent years [5], [47]. Since machine learning methods use many more free parameters, the size of the optimization data set must be increased to train them. For this purpose, synthetic data [48] can be included in the optimization data set, since they are much easier to obtain and label than real measurement data.

Almost all methods evaluated in this thesis make use of a gradient magnitude map. In this thesis, the Sobel operator is used for this purpose. The quality of the results could potentially be improved by using other local features to enhance the quality of the map. Additionally, more research into edge-preserving smoothing could improve the results.

For many cases, the gradient magnitude map can also be replaced by a probability of boundary map, such as the one produced by gPb. Although thresholding of the gPb underperforms thresholding of the gradient magnitude in this thesis, the quality of the gPb could be further improved as not all free parameters of the gPb were optimized for the CSD data set. Additionally, the features used by the gPb are designed for natural RGB images and can be not or only poorly applicable to CSDs. The selection of CSD specific features could be used to adapt the gPb.

For those methods which use a separate edge detection step, the free parameters of the edge detection step and the line detection step are optimized separately in this thesis. This is suboptimal because the parameter set that maximizes the evaluation score for an edge detector might not lead to the best results in combination with a subsequent line detection step. For example, for a line detector that is good at ignoring false positive edge pixels, an edge map that contains more true positive but also more false positive edge pixels is preferable.

During the parameter selection, a tradeoff between precision and recall has to be made because parameter sets that lead to the detection of more true positives usually also lead to the detection of more false positives. If a reliable validation method for the rejection of false positives was found, more true positives could be detected because a different parameter set could be chosen.

To increase the number of found lines further, missed lines need to be reconstructed from detected lines. For this purpose, a reconstruction method that takes domain related information about the CSDs and charge transitions into account has to be developed. It is to be expected that the reconstruction of an interdot transition

based on the surrounding lead-to-dot transitions works especially well. This fits the results of this thesis, in which the interdot transitions are missed more often than the lead-to-dot transitions.

Finally, the automatic selection of the optimal voltage ranges for the gates must be integrated into the tuning workflow. Therefore, the specific charge distributions for the honeycomb structures have to be determined. Methods that accomplish this task, incorporating the information extracted via line detection, must be developed. For this purpose, a method for the semantic classification of the lines is required. Additionally, the minimal required gradient magnitude per line detection method can be estimated.

Bibliography

[1] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM J. Comput.* 26.5 (1997), 1484–1509.

[2] Tim Botzem. "Coherence and high fidelity control of two-electron spin qubits in GaAs quantum dots". PhD thesis. RWTH Aachen, Jan. 2017.

[3] Fabian Hader. *Noise Analysis and Estimation in Sensor Dot Fine Scans for Automated Tuning of Gate Defined Quantum Dots*. Master's Thesis. 2021.

[4] Xiaohu Lu et al. "CannyLines: A parameter-free line segment detector". In: Sept. 2015, pp. 507–511. DOI: 10.1109/ICIP.2015.7350850.

[5] Junfeng Jing et al. "Recent advances on image edge detection: A comprehensive review". In: *Neurocomputing* 503 (Sept. 2022), pp. 259–271. DOI: 10.1016/j.neucom.2022.06.083.

[6] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).

[7] John Canny. "A Computational Approach To Edge Detection". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* PAMI-8 (Dec. 1986), pp. 679 –698. DOI: 10.1109/TPAMI.1986.4767851.

[8] Agnès Desolneux, Lionel Moisan, and Jean-Michel Morel. "Edge Detection by Helmholtz Principle". In: *Journal of Mathematical Imaging and Vision* 14 (May 2001), pp. 271–284. DOI: 10.1023/A:1011290230196.

[9] Cuneyt Akinlar and Cihan Topal. "EDLines: A real-time line segment detector with a false detection control". In: *Pattern Recognition Letters* 32 (Oct. 2011), pp. 1633–1642. DOI: 10.1016/j.patrec.2011.06.001.

[10] Michael Maire et al. "Using contours to detect and localize junctions in natural images". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on* (June 2008). DOI: 10.1109/CVPR.2008.4587420.

[11] David Martin, Charless Fowlkes, and Jitendra Malik. "Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues". In: *IEEE transactions on pattern analysis and machine intelligence* 26 (June 2004), pp. 530–49. DOI: 10.1109/TPAMI.2004.1273918.

[12] M.C. Morrone and R.A. Owens. "Feature detection from local energy". In: *Pattern Recognition Letters* 6.5 (1987), pp. 303–313. ISSN: 0167-8655. DOI: https://doi.org/10.1016/0167-8655(87)90013-4. URL: https://www.sciencedirect.com/science/article/pii/0167865587900134.

[13] Jianbo Shi and Jitendra Malik. "Normalized Cuts and Image Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22 (May 2002). DOI: 10.1109/34.868688.

[14] Richard O. Duda and Peter E. Hart. "Use of the Hough Transformation to Detect Lines and Curves in Pictures". In: *Commun. ACM* 15.1 (1972), 11–15. ISSN: 0001-0782. DOI: 10.1145/361237.361242. URL: https://doi.org/10.1145/361237.361242.

[15] Leandro Fernandes and Manuel Oliveira. "Real-time line detection through an improved Hough transform voting scheme". In: *Pattern Recognition* 41 (Sept. 2008), pp. 299–314. DOI: 10.1016/j.patcog.2007.04.003.

[16] Jiri Matas, C. Galambos, and J. Kittler. "Robust Detection of Lines Using the Progressive Probabilistic Hough Transform". In: *Computer Vision and Image Understanding* 78 (Apr. 2000), pp. 119–137. DOI: 10.1006/cviu.1999.0831.

[17] Marketa Dubska, Adam Herout, and Jiri Havel. "PClines — Line detection using parallel coordinates". In: *Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (July 2011), pp. 1489 –1494. DOI: 10.1109/CVPR.2011.5995501.

[18] Rafael Gioi et al. "LSD: A Fast Line Segment Detector with a False Detection Control". In: *IEEE transactions on pattern analysis and machine intelligence* 32 (Apr. 2010), pp. 722–32. DOI: 10.1109/TPAMI.2008.300.

[19] Iago Suarez, José Buenaposada, and Luis Baumela. *ELSED: Enhanced Line SEgment Drawing*. Aug. 2021.

[20] Marcos Nieto et al. "Line segment detection using weighted mean shift procedures on a 2D slice sampling strategy". In: *Pattern Anal. Appl.* 14 (May 2011), pp. 149–163. DOI: 10.1007/s10044-011-0211-4.

[21] Hongcheng Wang et al. "Gradient Adaptive Image Restoration and Enhancement". In: *Proceedings - International Conference on Image Processing, ICIP* 8-11 (Nov. 2006), pp. 2893 –2896. DOI: 10.1109/ICIP.2006.313034.

[22] Viorica Pătrăucean, Pierre Gurdjos, and Rafael Gioi. "A Parameterless Line Segment and Elliptical Arc Detector with Enhanced Ellipse Fitting". In: vol. 7573. Jan. 2012, pp. 572–585. ISBN: 978-3-642-33708-6. DOI: 10.1007/978-3-642-33709-3_41.

[23] Viorica Patraucean, Pierre Gurdjos, and Rafael Gioi. "Joint A Contrario Ellipse and Line Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39 (Apr. 2016), pp. 1–1. DOI: 10.1109/TPAMI.2016.2558150.

[24] Jin Han Lee et al. "Outdoor Place Recognition in Urban Environments Using Straight Lines". In: *Proceedings - IEEE International Conference on Robotics and Automation* (June 2014). DOI: 10.1109/ICRA.2014.6907675.

[25] Jiangye Yuan and Anil Cheriyadat. "Learning to count buildings in diverse aerial scenes". In: Nov. 2014. DOI: 10.1145/2666310.2666389.

[26] Nam-Gyu Cho, Alan Yuille, and Seong-Whan Lee. "A Novel Linelet-Based Representation for Line Segment Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (May 2017), pp. 1–1. DOI: 10.1109/TPAMI.2017.2703841.

[27] Iago Suarez et al. "FSG: A statistical approach to line detection via fast segments grouping". In: Oct. 2018. DOI: 10.1109/IROS.2018.8594434.

[28] Sarah Fleitmann et al. "Noise Reduction Methods for Charge Stability Diagrams of Double Quantum Dots". In: *IEEE Transactions on Quantum Engineering* 3 (Jan. 2022), pp. 1–1. DOI: 10.1109/TQE.2022.3165968.

[29] *Canny edge detector software interface*. URL: https://scikit-image.org/docs/stable/api/skimage.feature.html#skimage.feature.canny (visited on 02/06/2023).

[30] *CannyPF and CannyLines software interfaces*. URL: https://cvrs.whu.edu.cn/cannylines/#ct1 (visited on 02/06/2023).

[31] *Edge Drawing and EDLines software interfaces*. URL: https://github.com/shaojunluo/EDLinePython (visited on 02/06/2023).

[32] *gPb software interface*. URL: https://github.com/HiDiYANG/gPb-GSoC (visited on 02/06/2023).

[33] *Hough transform software interface*. URL: https://docs.opencv.org/3.4/dd/d1a/group__imgproc__feature.html#ga46b4e588934f6c8dfd509cc6e0e4545a (visited on 02/06/2023).

[34] *Kernel-based Hough transform software interface*. URL: https://github.com/laffernandes/kht (visited on 02/06/2023).

[35] *Progressive probabilistic Hough transform software interface*. URL: https://scikit-image.org/docs/stable/api/skimage.transform.html#skimage.transform.probabilistic_hough_line (visited on 02/06/2023).

[36] *PCLines software interface*. URL: https://github.com/RomanJuranek/pclines-python (visited on 02/06/2023).

[37] *LSD software interface*. URL: https://github.com/hdkai/Line-Segment-Detector (visited on 02/06/2023).

[38] *ELSED software interface*. URL: https://github.com/iago-suarez/ELSED (visited on 02/06/2023).

[39] *SSWMS software interface*. URL: https://github.com/marcosnietodoncel/lswms (visited on 02/06/2023).

[40] *ELSDc software interface*. URL: https://github.com/viorik/ELSDc (visited on 02/06/2023).

[41] *FLD software interface*. URL: https://docs.opencv.org/4.x/df/d4c/classcv_1_1ximgproc_1_1FastLineDetector.html (visited on 02/06/2023).

[42] *Yuan's line detector software interface*. URL: https://github.com/yuanj07/LineExtr (visited on 02/06/2023).

[43] *Linelet-based detector software interface*. URL: https://github.com/NamgyuCho/Linelet-code-and-YorkUrban-LineSegment-DB (visited on 02/06/2023).

[44] *FSG software interface*. URL: https://github.com/iago-suarez/FSG (visited on 02/06/2023).

[45] Rainer Storn and Kenneth Price. "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces". In: (1997), pp. 341–359.

[46] Fabian Hader et al. "On Noise-Sensitive Automatic Tuning of Gate-Defined Sensor Dots". In: *IEEE Transactions on Quantum Engineering* PP (Jan. 2023), pp. 1–19. DOI: 10.1109/TQE.2023.3255743.

[47] Jun-Tae Lee et al. "Semantic Line Detection and Its Applications". In: (Oct. 2017), pp. 3249–3257. DOI: 10.1109/ICCV.2017.350.

[48] Sarah Fleitmann. *Characterization of Distortions in Charge Stability Diagrams and Their Simulation in Modeled Data*. Master's Thesis. 2022.

Jül-4441 • Juli 2023
ISSN 0944-2952

Mitglied der Helmholtz-Gemeinschaft