State-of-the-Art Automated Line-Slope Extraction in Quantum-Dot Charge Stability Diagrams

Letitia-Anda Bulica

Abstract

This work surveys state-of-the-art methods for extracting line slopes from images of charge stability diagrams, using both 1D and 2D approaches. It also reviews strategies for automating region detection in such diagrams through feedback-driven experimental controllers. The aim is to replace manual tuning with an automated loop integrated into our in-house measurement framework.

1 Charge Stability Diagrams

In order to describe quantum dot systems, charge stability diagram (CSD) are an essential tool. These kinds of diagrams are often used in quantum computing. They represent graphically the change configurations of quantum dots, which are determined by external control parameters, usually the voltages applied to the gate electrodes. To investigate the electronic properties of quantum dots and to precisely tune them to the desired operating regimes, it is essential to understand these diagrams [1].

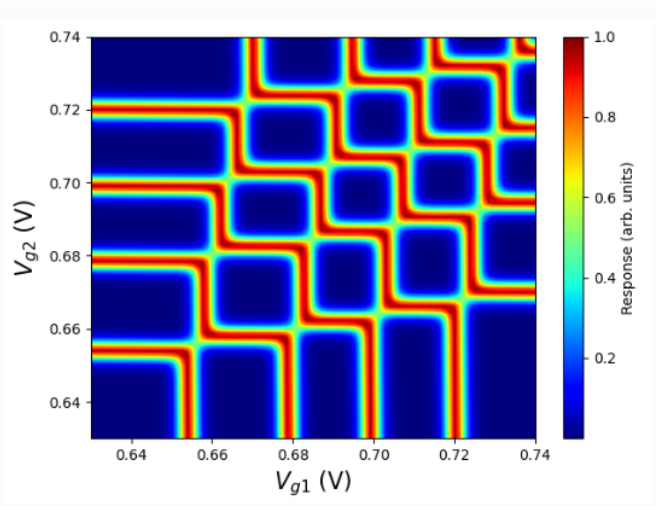

Figure 1: Charge-stability diagram [2]

A **quantum dot** is a nanometric region in a semiconductor material where the movement of electrons or holes is confined in all three spatial dimensions. Because they have separate energy levels and quantized charge states, some people call them "artificial atoms" [3]. A quantum dot is usually between 2 and 10 nanometers wide, which is about

the size of a volume that holds 100 to 100,000 atoms [4]. Quantum dots are very easy to change with outside gate voltages because of the well-defined electronic states that quantum confinement creates. This is an important feature for making and studying charge stability diagrams.

In semiconductor quantum dot systems, metal **gate electrodes** are used to define and control the potential landscape. By applying different voltages to these gates, electrons can be confined in specific regions, allowing for a precise control. There are two primary types of gates:

- **Plunger gates**: They are responsible for controlling the amount of charge in a quantum dot. Adjusting the voltage on a plunger gate changes the energy of the dot, thus controlling the number of electrons it can contain. This is a crucial step in manipulating qubit states in quantum computing applications.
- **Barrier gates**: Control the tunneling (jumping) rates of electrons. A high tunnel rate means that the electrons can pass back and forth quickly. A low tunnel rate affects how blurred the CSD lines look and how fast operations can be performed on the dot [5].

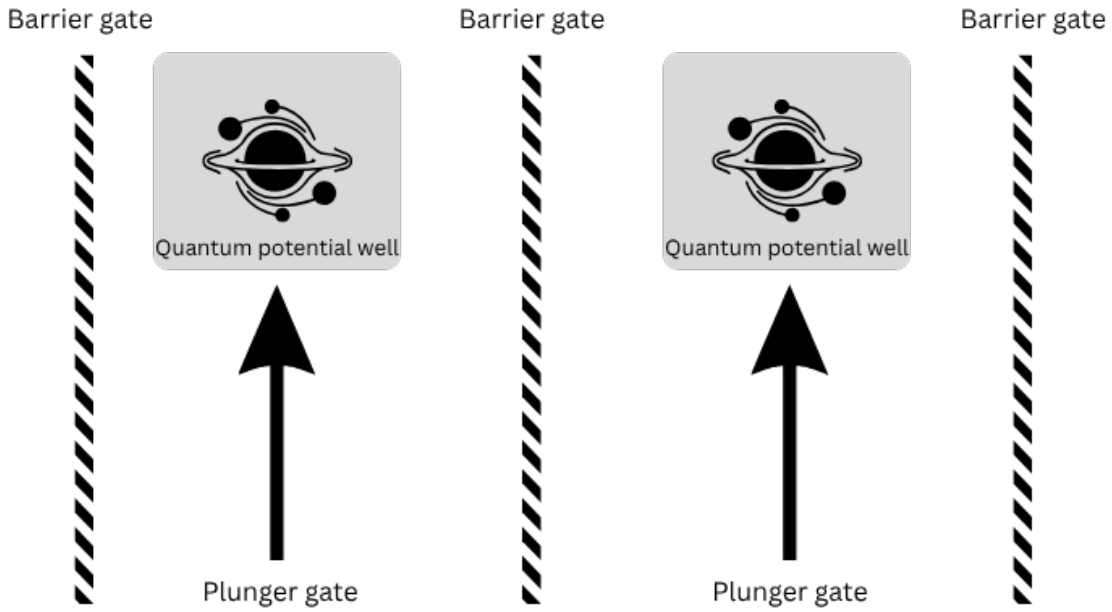

Figure 2: Layout of a Double Quantum Dot with Plunger and Barrier Gates

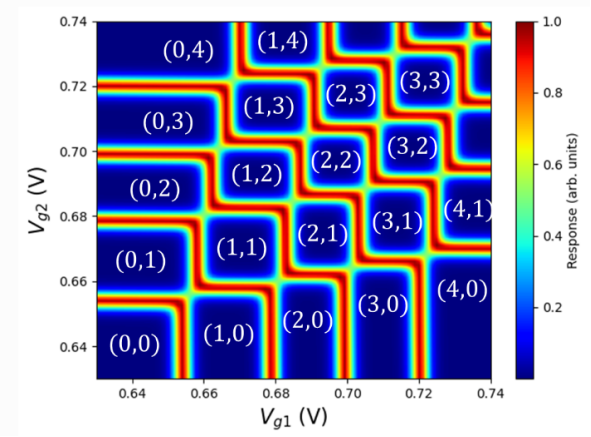

Figure 3: Ideal CSD [2]

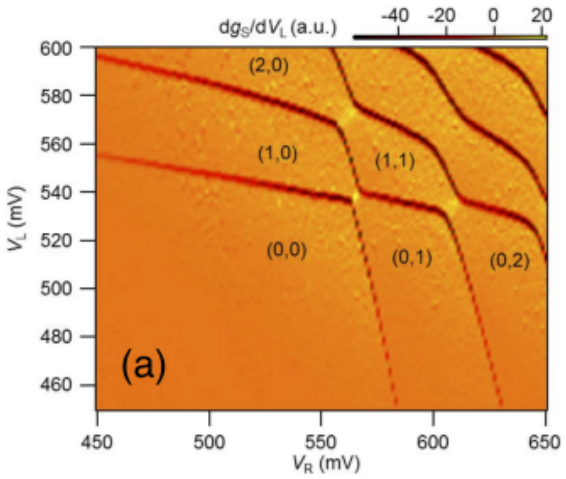

Figure 4: Real CSD [1]

Charge stability diagrams provide a visual representation of stable charge configurations in a quantum dot system, which depends on the voltages that are applied to the gate electrodes. As seen in figure 3, each delimited region in the diagram corresponds to a fixed configuration in the two quantum dots denoted: **(N1, N2)**. N1 is the number of electrons tuned mainly by Gate1.

In the left image these regions look like a perfect "honeycomb" pattern, with clear straight lines separating the areas. These transition lines indicate the moments when an electron is added or removed from a quantum dot. The transition occurs when the energy level of the dot aligns with the chemical potential of the electrode, allowing the electron to tunnel [1]. The colors of the lines offer us information about the tunneling properties:

- Red color: The transition is very easy, electrons hop on and off.
- Dark blue color: The transition is very weak or even nonexistent, electrons can't tunnel efficiently.

The figure on the right represents an actual, real diagram. The lines are more blurred and irregular due to noise. Despite all of these, the transitions can still be recognized and analyzed.

The slopes of the lines in a charge stability diagram actually tell us a lot about how the quantum dot system works. They show how much each gate affects the dots, depending on how the voltages are applied. Basically, the slope depends on how strongly each gate is connected to the dots through capacitance. For example, if a line is really steep, it might mean one gate has more control over a dot than the other. In double dot systems, the angles between the lines can also hint at how strongly the two dots are connected or how easily electrons can move between them. So by looking at these slopes, we can figure out things like how the system is wired up, how the dots interact, and how to tune everything to make the quantum dot behave the way we want.

2 Line Extraction Techniques

In charge stability diagrams, the key features are the transition lines that indicate changes in the charge configuration of the quantum dots. In order to automate the analysis of these diagrams, it's essential to extract these lines reliably. There are two main approaches for doing this: 1D techniques, which analyze slices or projections of the image, and 2D techniques, which treat the entire diagram as an image and apply computer vision methods. Each has its own advantages and trade-offs depending on the quality of the data and the specific use case.

2.1 1D Line Detection

One simple and intuitive way to find lines in an image is by looking at it one row or column at a time. This is called 1D line detection. Instead of treating the image as a whole, we treat each row or column like a separate signal. Each pixel in that line has a brightness value, so when we look at how those values change, we can often spot where a line might be.

To detect those spots, one option is to use peak detection. This means we look for points where the brightness suddenly increases or decreases, these jumps often happen along edges or transitions in the image. Libraries like SciPy make this easy with functions like find_peaks().

Another method is to look at how fast the brightness is changing. For this, we calculate the first derivative of the signal, which gives us spikes where the change is strongest. These spikes usually show where edges are located [6]. This idea is used in popular edge detectors, like the Canny Edge Detector, which looks at both the gradient and direction of changes to find edges in a more accurate way [6]. While Canny is often used in full 2D analysis, the logic behind it also applies when working slice by slice in 1D. After detecting where the signal jumps in each row or column, we collect those positions. If they follow a consistent path (like gradually shifting in one direction) we can connect them and fit a straight line. The slope of that line tells us how it's angled in the image. This is usually done using linear regression.

What makes 1D detection nice is that it's simple to understand and runs very fast. It's especially useful when the lines in the image are straight, not too noisy, and go in the same direction as the slices we're scanning. But it also has its limits. If the lines are tilted, curved, or the image is messy, this method might miss them or give confusing results. Also, we need to decide ahead of time whether to scan rows or columns, and that doesn't always match the orientation of the lines we want [7]. Even with those downsides, 1D methods are a solid starting point, especially when working with cleaner or more ideal data.

2.2 2D Line Detection

Unlike 1D methods that look at individual rows or columns, 2D line detection analyzes the entire image as a whole. This approach is inspired by tools from computer vision and is especially useful when the lines in the image are diagonal, curved, noisy, or overlapping: situations where 1D methods often struggle. In 2D, the idea is to look for edges: sharp changes in intensity, no matter which direction they go. These edges are often the first step to finding the full lines in the image.

The process usually starts with preprocessing, which can include smoothing or denoising the image. This helps reduce small fluctuations that could confuse the edge detector later on. Next, we apply an edge detection filter. One option is the Sobel filter, which looks at how brightness changes in both the horizontal and vertical directions. Another popular choice is the Canny edge detector, which uses a more advanced method. It calculates the gradient, applies thresholds, and performs edge thinning and noise suppression to produce clean and accurate edge maps [6]. After this step, we're left with a binary image showing only the pixels where edges are found. These edges often trace out parts of the lines we're trying to detect. The next step is to apply the Hough Transform, a powerful algorithm that can detect straight lines in an image. It works by mapping each edge point into a new space called Hough space, where each point "votes" for all the possible lines it could belong to [6]. Even if a line is slightly broken or noisy, the Hough Transform still works well because the points will still vote for similar line parameters. Once the most likely lines are found, we can look at their angles to compute slopes, and optionally filter out short or weak ones, or merge lines that are very close together.

The biggest strength of 2D detection is its flexibility. It can handle diagonal or irregular lines and works well in real-world images where noise is unavoidable. It also finds multiple lines at once, making it a good fit for images like CSDs, which usually contain several overlapping features. There are downsides too. The Hough Transform and edge detection algorithms need fine-tuning, and the whole process can be more computationally intensive than simple 1D scanning. There's also a chance it might have false positives, especially in very noisy images. However, 2D methods (especially the combination of Canny + Hough) are one of the most popular solutions in both research and practical applications.

Feature	**1D Line Detection**	**2D Line Detection**
Approach	Analyzes individual rows or columns as 1D signals	Processes the full image as a 2D structure
Detection	Peak finding or first derivative on each slice	Edge detection (like Canny) followed by Hough Transform
Direction Sensitivity	Requires preselected scan direction (horizontal or vertical)	Detects edges in any orientation
Robustness to Noise	Sensitive to noise and artifacts	More robust
Computational Complexity	Lightweight and fast	More computationally intensive
Best Suited For	Clean images with straight, well-separated lines	Noisy, real-world images with diagonal or overlapping lines
Limitations	Fails on curved or tilted lines; requires manual direction choice	Needs tuning of parameters; may detect false positives

Table 1: Comparison of 1D and 2D Line Detection Methods

3 State-of-the-Art: 1D Gradient-Based Line Detection

The method described by the paper "Fast Virtual Gate Extraction For Silicon Quantum Dot Devices" [8] is one of the most efficient 1D techniques available today. It proposes a method to extract these virtual gates quickly by using 1D scans only, without needing to collect or analyze the full 2D charge stability diagram (CSD). In a multi-quantum-dot setup, changing the voltage on one gate doesn't only affect the dot it's supposed to control,it slightly shifts the energy levels of neighboring dots too. This makes precise control very hard and tuning very slow. So, the goal of a virtual gate is to control just one dot independently, while leaving the other dots unchanged. It is linear combination of physical gate voltages.

The main idea of this method is: instead of mapping the entire CSD, they assume that the two main transition lines (for the first electron on each dot) have slopes within known bounds. That means you don't need to search the entire voltage space. You only need to focus on a small triangular region where these transitions are guaranteed to be in. This triangle is defined by choosing two "anchor points" (as they are called inside the paper): one close to the (0,1) transition and one close to the (1,0) transition, based on known approximate behavior or previous data. Then they search within this triangle.

The first step is defining the triangular search region mentioned before. They start by selecting two known or estimated points that are roughly on the transition lines of the two dots. These can be manually chosen or calculated from previous data. A triangle is then formed between these two points and a shared origin. The idea is that the (0,1) and (1,0) transitions must both pass somewhere inside this triangle [8].

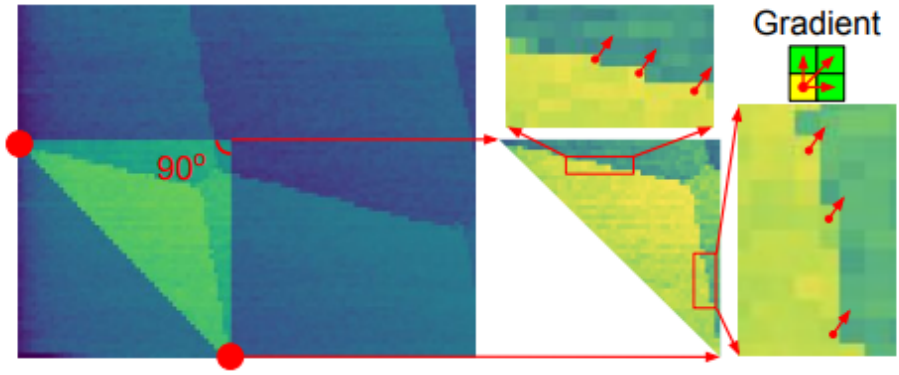

Figure 5: The triangular region [8]

The next step is doing a 1D gradient sweeps along rows and columns. Instead of scanning the whole triangle at once (2D), they sweep across it line by line, both row-wise and column-wise. For each line, they collect a 1D signal of the values and look for sharp transitions. These transitions are found using a gradient-based score [8]. The, the score for each point is calculated. For a given pixel (voltage point), they measure the difference in conductance between that pixel and its neighbor one step forward (to the right or above). After that, they add up these differences in both directions to compute a gradient score. The pixel with the highest gradient score in that row or column is marked as the likely location of a transition line. This process is repeated across every row and column in the triangle.

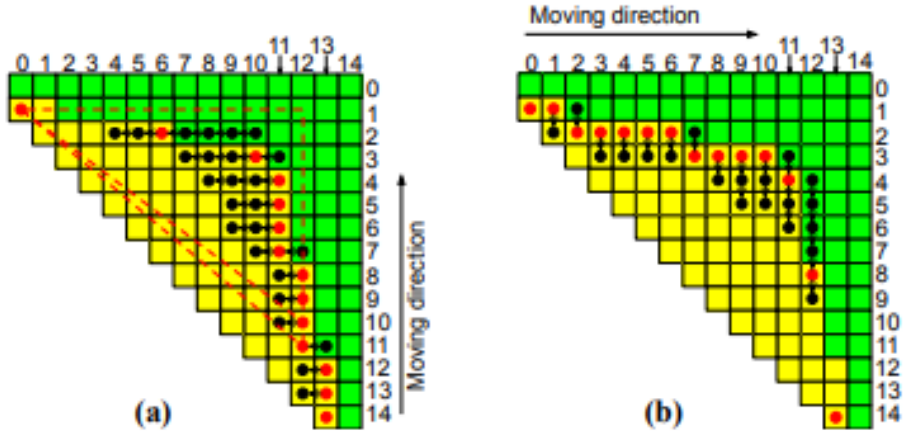

Figure 6: (a) - Row Sweeps, (b) - Column Sweeps [8]

As seen in Figure 6, each black dot represents the pixel with the highest gradient score in its corresponding row or column. Red squares are the anchor points. The yellow region shows the area being scanned, while green cells are ignored.

The third step is filtering and merging the detected points. After the row and column scans, there are two sets of transition points: one from the horizontal sweep and one from the vertical one. But not all these points are accurate and useful. Some maybe are outside the expected path due to noise or weak transitions [8]. Thus, they applied a filtering step. Firstly, the points that have a very low gradient score are discarded. Then, they ensure that the remaining points form a mostly straight line, based on the distance. In the end, the row and column detections are merged, and a linear fit is applied to each line. A linear fit means drawing a straight line that best matches that set of points.

Using the two line slopes, they find the intersection point and calculate the matrix that transforms physical gate voltages into virtual gates. This matrix makes it so that tuning one dot no longer affects the others. This virtual gate matrix is the final goal of the method. It can be used in future scans and experiments to simplify the dot's control.

The results were extremely good. The method was tested on a real dataset called qFlow, which contains CSDs from silicon double-dot devices [8]. The method was compared against a method that uses a traditional 2D image-processing: edge detection followed by a Hough Transform, which is a standard way of finding lines in noisy images. The propsed method successfully extracted the correct virtual gate transformation in 10/12 cases, while the 2D version only succeeded in 9/12 cases. Success means that the detected transition lines correctly matched the actual truth. So, the 1D method was at least as accurate as the 2D method, and slightly better overall. Because the 1D method only scans rows and columns in a small triangular region, it avoids scanning the full 2D diagram. That leads to big gains in time and data usage. It scanned only 10–20% of the total pixels that the 2D method used. As a result the total runtime was reduced as well.

The evaluation proves that the 1D gradient method is not just a simplification, but it's actually more efficient. It extracts useful virtual gates using far fewer resources and with less sensitivity to noise. This makes it a practical choice for real-time tuning in experimental quantum dot setups. Table 2 shows how the proposed 1D method outperforms the more traditional Hough-based approach in both speed and reliability and how useful this method actually is.

Metric	1D Gradient Method	2D Hough Transform Baseline
Successful Extractions (out of 12)	10	9
Average Runtime	30–100 seconds	950 seconds (scales with image size)
Speedup Factor	5× to 19× faster	—
Data Sampled	10–20% of pixels	100% of pixels
Slope Extraction Method	1D row/column gradient sweeps	Full-image edge + Hough
Post-processing	Filtering + linear fit	Hough parameter filtering

Table 2: Comparison between the proposed 1D method and a 2D baseline

4 State-of-the-Art: 2D-Based Slope Detection Techniques

When working with charge stability diagrams, another approach is to analyze the image as a full 2D object, instead of breaking it into 1D slices. These methods consider the structure and patterns in both directions simultaneously, which makes them more robust to noise, curved lines, or irregularities that might be confusing for simpler techniques. In this section, I'll focus on three recent works that take very different but effective approaches to 2D analysis.

The first method "Computer-automated tuning procedures for semiconductor quantum dot arrays" is a quite classical by Zwolak et al. [9], where edge detection is combined with the Hough transform to extract charge transition lines and compute their slopes. This is one of the earliest examples of using 2D tools to perform automatic line extraction in CSDs. The second method is called QDA^2, proposed by Weber and Zwolak [10]. Instead of directly detecting slopes, it focuses on identifying geometric features like bias triangles and Coulomb diamonds, using clustering and region labeling. It doesn't compute slopes directly, it offers a clean and fast way to detect structured regions in the CSD. And the last method "Robust quantum dots charge autotuning using neural network uncertainty" propose a more modern solution that uses a neural network to detect line transitions. Their method outputs confidence-weighted predictions and applies post-processing heuristics to improve the reliability of line detection, especially in noisy data [11].

4.1 Classical 2D Edge and Hough Detection

The goal of the paper is to detect the charge transition lines, get their slopes, and use them to define the virtual gate matrix. It uses classical image processing tools, no machine learning, just filters and transformations, which makes the method easy to reproduce.

The process starts with a 2D image of the CSD, where each pixel corresponds to a measurement (like the current) taken at a specific pair of gate voltages. Charge transitions appear in this image as lines where the signal intensity changes quickly. The first step is the edge detection. In order to make these features easier to detect, the authors apply

a simple image filter (a 2D derivative) that highlights the edges [9]. This is similar to applying a Sobel filter in computer vision, which finds areas in the image where the intensity changes the most.

Once the image has been filtered to highlight the edges, the Hough Transform is applied to detect straight lines. Instead of finding lines directly in the image, the Hough Transform maps each edge point to a curve in a new space called *Hough space*, defined by the line parameters (θ, ρ), where θ is the angle of the line and ρ is the distance from the origin. If many pixels in the image lie along the same line, they all vote for the same point in Hough space, forming a peak. From these peaks, the most likely lines can be identified, and their slopes will be calculated [12].

After detecting a set of lines, the next step is to convert their slopes into a form that reflects physical parameters. In CSDs, the slope of a charge transition line corresponds to a capacitance ratio, it is how changes in one gate's voltage compare to another's when trying to keep the dot at the same charge state. These slopes are used to calculate a virtual gate matrix, which transforms the original gate voltages into a new set of virtual gates [9]. Each virtual gate is designed to control a specific aspect of the quantum dot device, such as the occupation of a particular dot, with minimal interference from the others.

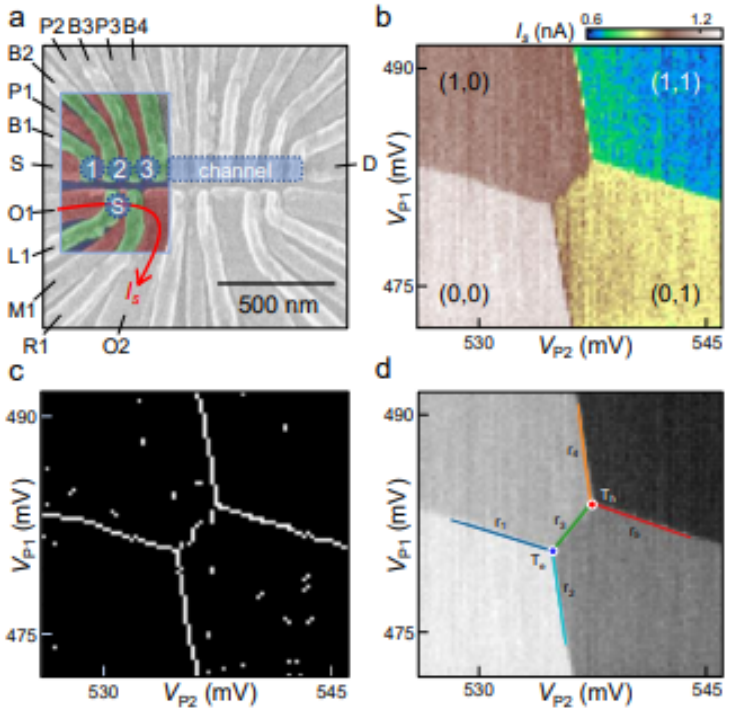

Figure 7: (a) Microscope image (b) Raw charge stability diagram (c) Edge-detected step (d) Detected lines using the Hough Transform [9]

Figure 7 showcases the full process proposed inside the paper. Panel (a) shows a scanning electron microscope (SEM) image of the actual device. The gates labeled P1, P2 are used to define quantum dots, and the voltages applied to these gates are what generate the CSDs. Panel (b) is the raw CSD image, where the color represents the current measured as two gate voltages. Panel (c) shows the data after applying the first step: derivative filter. This enhances the contrast of the transition lines by highlighting where the signal changes most rapidly. White pixels indicate likely edges [9]. Panel (d)

shows the result of applying the Hough Transform to the image. The straight transition lines were identified and overlaid on the original image.

This method is quite simple, which in my opinion it's its greatest strength. It doesn't require training data, and all the processing steps are based on standard tools from image processing. However, the method assumes that transition lines are straight and clearly visible. If the diagram contains overlapping, curved, or noisy features, the Hough Transform may not perform well. The accuracy also depends on setting good parameters for things like edge detection thresholds and Hough resolution. To summarize, this is a clean and classical method that provides a reliable baseline for automated slope extraction in clean CSDs.

4.2 QDA²

QDA² (it is short for Quantum Dot Annotator version 2) proposes a geometric way of analyzing charge stability diagrams (CSDs). Unlike image-processing or machine-learning-based methods, this one doesn't rely on manually extracted features or neural networks. Instead, it starts from the assumption that CSDs are structured, polygonal shapes that can be described using physical rules. QDA² takes a fully 2D approach to interpreting CSDs, by analyzing the full structure of the CSD as a spatial pattern made up of geometric regions [10].

In most charge stability diagrams, there are well-defined regions (often hexagonal or triangular) separated by straight-line transitions. These transitions correspond to the addition or removal of an electron in one of the quantum dots. What QDA² does is try to reconstruct the hidden structure of these regions by fitting a set of polygons to the diagram that are consistent with known charge states and transitions.

At the start we have a raw CSD image (usually from a double quantum dot system) and then we apply a basic edge detection filter like the Sobel operator to highlight the transition lines (just like the first method did). Then, instead of just detecting individual lines (through Hough Transform), the algorithm builds a graph structure of vertices and edges that could describe the charge transitions in the image [10].

Once the graph structure is built, QDA² tries to find the set of polygons that best explain the observed edge pattern. It does this by grouping edges together to form closed regions. Because these regions are supposed to be arranged in a grid-like pattern, the algorithm checks how well the current graph layout fits that expected shape. This step is important, since edge detection alone can result in gaps or broken segments, and the geometry-based fitting helps to fill in the missing parts and clean up.

After the structure is fitted, the next step is to assign labels to the regions. QDA² starts by choosing a known region,usually one assumed to be (0,0) [10], and then uses the orientation of the edges to figure out how the charge state changes as you move from one region to the next.

The final result is a CSD where every region is labeled, and the entire diagram is interpreted in terms of its underlying quantum dot structure. This is useful because it doesn't just find lines, it gives a full, interpretable mapping of the device's behavior.

QDA² also has a built-in optimization step. This helps make sure the final polygon map closely follows the real diagram, even if the raw image was noisy or slightly distorted. Because the method uses only the structure of the diagram and basic physical rules, it doesn't need any training data and works even in cases where other methods would struggle.

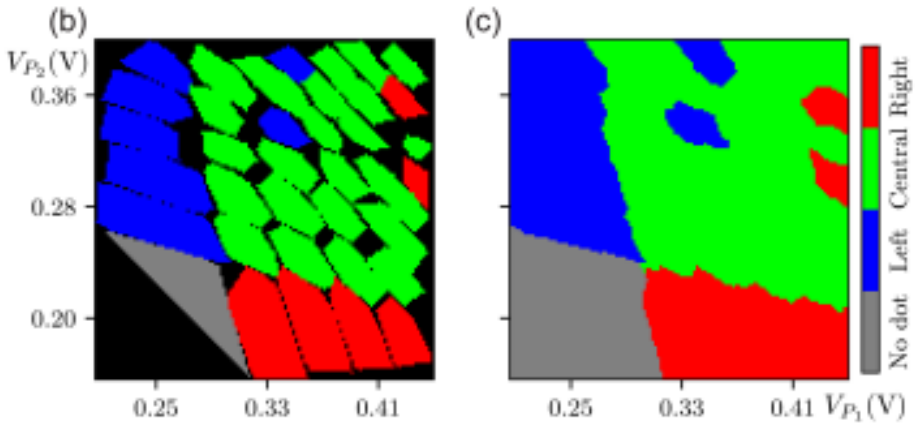

Figure 8: QDA² annotation of a charge stability diagram [10]

Figure 8 shows an example of QDA² applied to a CSD. In panel (b), the algorithm fitted a full geometric structure over the raw data, identifying distinct polygonal regions. Each polygon corresponds to a specific stable charge state. In panel (c), the method goes one step further and assigns a high-level label to each region, such as "Left", "Central", "Right", or "No dot", based on the occupancy of the quantum dots. This demonstrates how QDA² not only recovers the physical structure of the diagram but also provides an interpretable classification of the device behavior. There are a few small patches of blue and red inside the green area. These come from noise or weak transitions [10] in the original data that confuse the model slightly.

QDA² stands out from other approaches because it uses the known structure of charge diagrams to compute the detection algorithm, instead of relying on edge detection or trained models. It might not be suitable for diagrams with curved or very irregular features but it performs very well on the types of diagrams commonly used in double quantum dot experiments.

4.3 Neural Network Approach

This paper proposes a machine learning-based method to automate the interpretation of charge stability diagrams (CSDs). It uses a convolutional neural network (CNN) not just to classify different regions of a diagram into charge states, but also to estimate the uncertainty of its predictions, making it more robust and reliable [11]. The manual tuning of quantum dots: where researchers interpret CSDs and adjust gate voltages by hand becomes harder as device complexity increases. This method treats the entire image as a 2D input, taking advantage of the spatial structure of CSDs.

The input to the model is a usual 2D CSD image.. The CNN is trained to classify each pixel of the diagram as belonging to a certain charge state: for example, "no dot", "single dot", or "double dot". Unlike typical classification networks, the paper's authors do not rely on a single deterministic model. Instead, they use multiple CNNs: independently trained models that process the same input and compare their outputs.

When all the networks agree on a classification, the result is most likely trustworthy. But if the predictions differ widely, the model reports high uncertainty, signaling that the input data is noisy, ambiguous, or out-of-distribution [11]. This uncertainty estimate is computed at each pixel and can be visualized as a separate uncertainty map.

This prediction has two main benefits. First, it enables robust auto-tuning: the system can automatically decide whether it has confidently reached a desired charge configuration. If the network is highly confident (low uncertainty), it proceeds with tuning decisions. But if the uncertainty is high, it flags that part of the diagram as unreliable warning the operator that manual inspection may be required [11]. Secondlt, it acts as a safety net against incorrect classifications. In earlier approaches, the network always outputs a prediction, even if it's unsure. This can lead to errors, especially in regions of the CSD that are noisy, distorted, or contain ambiguous patterns. By incorporating uncertainty, this method avoids making confident decisions when the data doesn't support them.

The model trained and tested on data from various device architectures: single and double quantum dot systems. This showed that the network could be generalized across setups. The architecture used is a relatively simple convolutional neural network, consisting of three convolutional layers followed by two fully connected layers. The network inputs small patches of the CSD image and learns to classify the local charge configuration in that region. This patch-wise processing also helps make the system more robust to noise and differences in image scale or resolution.

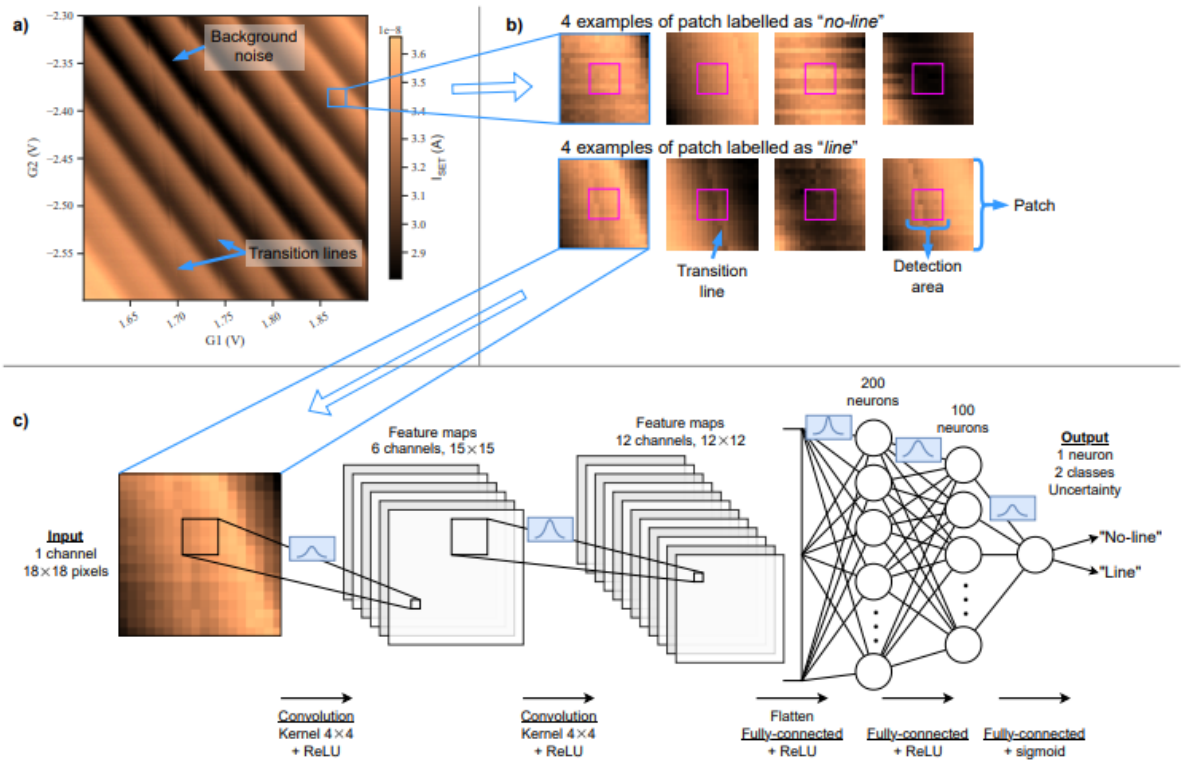

Figure 9: CNN-based classification of CSD patches into "line" or "no-line" regions, including network architecture [11]

Figure 9 shows how the CNN-based method processes a charge stability diagram to detect transition lines. In panel (a) we have a CSD where darker diagonal lines indicate charge transitions, and the background has slight noise. The system doesn't process the entire image at once but instead breaks it down into small square patches (in this case, 18×18 pixels each). Panel (b) shows examples of these patches. The top row includes patches labeled as "no-line", meaning they are entirely within a stable region

(no transitions). The bottom row shows patches that include a charge transition, so they are labeled as "line". Panel (c) shows the network architecture. Each input patch goes through three convolutional layers (with ReLU) to extract spatial features. After that, the feature maps are flattened and passed through fully connected layers. The output layer has two neurons: one for each class (line/no-line) and an extra value representing the network's uncertainty about its decision.

To measure the performance of this method, the authors evaluated how often the method was able to find useful virtual gates. For double quantum dot systems, the success rate was about 81%, and for single dots, it was 95% [11]. They also tested the system under various levels of Gaussian noise and found that even when the noise level was doubled, the model's performance only dropped slightly, showing that the method is quite robust to real-world imperfections.

One useful feature of this work is that the source code is publicly available. This means that researchers and people that are interested in this subject can directly test, modify, or integrate the model into their own frameworks without starting from scratch. The GitHub repository is linked in the paper, making it easier for others to replicate and extend the method.

This method shows the power of introducing modern deep learning within CSD line detections. While it requires training data and more computational resources than the other methods, it offers flexibility, robustness, and self-evaluation, which are crucial for scaling quantum dot control to more complex devices.

5 Comparison between State-of-the-Art 1D and 2D Slope Detection Techniques

All the presented methods help with understanding charge stability diagrams and extracting useful parameters,but they do so in very different ways. Each one shines in a different situation. The classical 2D method (edge detection + Hough transform) is the most straightforward: it finds sharp transitions and then looks for straight lines. It works really well when the data is clean and the lines are clearly visible. It's also very easy to implement using standard image processing tools, which is why it's often used as a baseline.

QDA^2 takes a completely different approach. It doesn't even try to find slopes directly. Instead, it reconstructs the full geometric structure of the diagram based on physical principles. It assumes that the regions in a CSD are shaped like polygons (triangles, hexagons), and tries to fit that structure over the image. This is really useful when you want a complete interpretation of the diagram, not just detecting lines. However, it might not work as well if the CSD is messy or irregular.

The neural network method brings machine learning into the picture. It trains several convolutional neural networks to recognize if a small patch of the image contains a transition line or not. This method also estimates how confident the model is in its prediction. That way, the system knows when it's unsure and can avoid making bad decisions in noisy or unusual parts of the diagram. It's a more modern approach, and even though it requires training data, it's robust and generalizes well to different devices.

Compared to these 2D methods, the 1D gradient-based technique is more focused and efficient. It doesn't try to interpret the whole diagram—instead, it just looks at a small triangle where the transitions are expected to be, and scans it line-by-line. It's very fast,

doesn't need any training or complex processing, and in most cases, it finds the correct transition lines with very little data.

All four methods are valid and state-of-the-art in their own way. The choice depends on what's more important in a given situation: speed, full interpretation, robustness, or simplicity.

6 Planned Work and Methodology

The goal of my dissertation is to automate the process of extracting meaningful features, specifically the slopes of transition lines from charge stability diagrams (CSDs), and to integrate that into an experimental controller. This would allow us to automatically find the right regions in the CSD, replacing the manual tuning that is currently done by hand. To approach this, I plan to begin with simple and robust methods first and then gradually build up to more complex techniques.

I'll start by implementing a 1D approach, similar to the one described in the paper "Fast Virtual Gate Extraction For Silicon Quantum Dot Devices" [8]. This method scans the CSD row-by-row and column-by-column inside a triangular search region, looking for sharp changes in the signal using a simple gradient score. I might process the entire image row-by-row and column-by-column and then improve it by searching only in the traingular regions. It's lightweight, doesn't require any machine learning or complex preprocessing, and works surprisingly well even with partial data. The goal here is to first get a baseline and validate that I can extract correct transition slopes and reconstruct the virtual gate matrix.

After that, I'll move on to a 2D approach that uses edge detection followed by a Hough transform to identify straight lines. This method is closer to traditional image processing and will allow me to compare results with the 1D version: both in terms of speed and accuracy. I'll test how both approaches perform on the same datasets to see which one is more reliable under different conditions, such as noisy or incomplete CSDs.

Once I have working slope detection methods, I'll integrate them into an experimental controller. The idea is to connect the detection loop to the actual measurement framework that we use. The controller will take live CSD data, analyze it in real time, and automatically move to the next region of interest (based on finding a corner or a specific charge state). This loop replaces human work and makes tuning faster and more repeatable.

Even though I am reviewing and implementing existing techniques, I also plan to explore ways to improve them or adapt them to my setup. For example:

- In the 1D method, I will try smarter ways of choosing anchor points or dynamically adapting the search triangle.
- For the Hough-based approach, I can experiment with better edge filters or curved line detection for more messy diagrams.
- I also want to explore combining both methods in a hybrid pipeline: using 1D for a fast guess, and 2D to refine the slope, that way I will have the best of both worlds.

This combination of implementation, comparison, and integration into an actual feedback loop will allow me to evaluate what works best not only in theory, but also in practice. My goal is to make the CSD analysis and tuning process fully autonomous and compatible with real-world usage.

7 Conclusions

Charge stability diagrams are critical tools for understanding and tuning quantum dot systems, especially in the context of quantum computing. The ability to automatically interpret these diagrams and extract meaningful features, such as the slopes of transition lines, can significantly accelerate experimental workflows and reduce the need for manual tuning.

This report has reviewed both classical and modern techniques for extracting slopes and interpreting charge stability diagrams. On the 1D side, methods like the gradient-based approach offer simplicity, speed, and surprisingly strong performance with limited data. On the 2D side, edge detection and Hough transforms remain reliable tools, while more recent methods like QDA^2 introduce geometry-based reasoning for structured region labeling. 2D deep learning methods using convolutional neural networks bring in the power of uncertainty-aware classification.

Each approach has unique strengths and limitations: 1D methods are lightweight but sensitive to line orientation, 2D methods offer robustness and structure, while neural networks add flexibility and scalability at the cost of training effort.

Building on this foundation, my work will begin with 1D and 2D implementations, aiming to compare their effectiveness and trade-offs. Eventually, I will integrate the best-performing methods into a fully automated controller loop within the lab's measurement setup. By improving slope extraction and experimenting with hybrid methods, I aim to contribute to making quantum-dot device control more autonomous, reliable, and ready for larger-scale applications.

References

[1] Nathan L. Foulk and Sankar Das Sarma. "Theory of charge stability diagrams in coupled quantum dot qubits". In: *Physical Review B* 110.20 (Nov. 2024). ISSN: 2469-9969. DOI: 10.1103/physrevb.110.205428. URL: http://dx.doi.org/10.1103/PhysRevB.110.205428.

[2] ***. *Quantum transport—Charge stability diagram of a double quantum dot*. 2023. URL: https://docs.nanoacademic.com/qtcad/tutorials/transport/double_dot_stability/.

[3] Leo Kouwenhoven and Charles Marcus. "Quantum dots". In: *Physics World* 11.6 (June 1998), p. 35. DOI: 10.1088/2058-7058/11/6/26. URL: https://dx.doi.org/10.1088/2058-7058/11/6/26.

[4] ***. *Quantum Dots*. URL: https://www.sigmaaldrich.com/RO/en/technical-documents/technical-article/materials-science-and-engineering/biosensors-and-imaging/quantum-dots.

[5] Nathaniel C. Bishop. "Scaling Quantum-Dot-Qubit Systems". In: *APS Physics* 18 (Apr. 2025), p. 88. DOI: 10.1103/Physics.18.88.

[6] Rein van den Boomgaard. *Canny Edge Detector*. 2017. URL: https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20162017/LectureNotes/IP/LocalStructure/CannyEdgeDetector.html.

[7] ***. *Edge Detection Using OpenCV*. URL: https://learnopencv.com/edge-detection-using-opencv/.

[8] Shize Che et al. *Fast Virtual Gate Extraction For Silicon Quantum Dot Devices*. 2024. arXiv: 2409.15181 [cond-mat.mes-hall]. URL: https://arxiv.org/abs/2409.15181.

[9] A. R. Mills et al. "Computer-automated tuning procedures for semiconductor quantum dot arrays". In: *Applied Physics Letters* 115.11 (Sept. 2019). ISSN: 1077-3118. DOI: 10.1063/1.5121444. URL: http://dx.doi.org/10.1063/1.5121444.

[10] Brian Weber and Justyna P. Zwolak. *QDA^2: A principled approach to automatically annotating charge stability diagrams*. 2023. arXiv: 2312.11206 [cond-mat.mes-hall]. URL: https://arxiv.org/abs/2312.11206.

[11] Victor Yon et al. "Robust quantum dots charge autotuning using neural network uncertainty". In: *Machine Learning: Science and Technology* 5.4 (Nov. 2024), p. 045034. ISSN: 2632-2153. DOI: 10.1088/2632-2153/ad88d5. URL: http://dx.doi.org/10.1088/2632-2153/ad88d5.

[12] A. Walker R. Fisher S. Perkins and E. Wolfart. *Hough Transform*. 2003. URL: https://homepages.inf.ed.ac.uk/rbf/HIPR2/hough.htm.