國立臺灣大學重點科技研究學院奈米工程與科學學位學程

碩士論文

Program in Nanoengineering and Nanoscience

Graduate School of Advanced Technology

National Taiwan University

Master's Thesis

二維材料的合成掃描穿隧顯微影像：深度學習分析流程

Synthetic STM Imaging of 2D Materials: A Pipeline for Deep Learning-Based Analysis

盧偉澤

Vitezslav Luzny

指導教授: 邱雅萍 博士

Advisor: Ya-Ping Chiu, Ph.D.

中華民國 114 年 7 月

July 2025

doi:10.6342/NTU202501885

國立臺灣大學碩士學位論文

口試委員會審定書

二維材料的合成掃描穿隧顯微影像：深度學習分析流程

Synthetic STM Imaging of 2D Materials: A Pipeline for Deep Learning-Based Analysis

本論文係盧偉澤君（R12K45028）在國立臺灣大學奈米工程與科學學位學程完成之碩士學位論文，於民國 114 年 7 月 28 日承下列考試委員審查通過及口試及格，特此證明

口試委員：_____

（指導教授）

_____ _____

_____ _____

_____ _____

_____ _____

所　　長：_____

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

Acknowledgements

I wish to convey my sincere gratitude to my advisor, Ya-Ping Chiu, for creating a welcoming and motivating environment that made this work possible. Her advice and encouragement were vital in shaping this work.

I am also grateful to my seniors and colleagues at Ya-Ping Chiu SPM Lab—Bo-Chao Huang, Yu-Kuan Lin, Yi-Ting Chen and Yu-Chieh Lin for their camaraderie, fruitful discussions, and support.

Finally, I thank National Taiwan University and the Institute of Atomic and Molecular Sciences, Academia Sinica, for fostering an inspiring academic atmosphere and providing the resources necessary to complete this work.

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

摘要

掃描式穿隧顯微鏡（STM）是一種極為有效的技術，可用於成像導電或半導體材料表面的影像，其解析度可達單一原子。然而，STM 仍受限於對人力密集的專家解讀的依賴，以及容易受到探針狀況、掃描參數與硬體相關失真所引入的人為誤差影響。這些因素限制了其速度與可擴展性。

在本研究中，我提出了一套以物理建模為基礎的模擬框架，用於生成合成的 STM 影像。該系統利用緊束縛方法計算電子結構，建構真實的探針幾何形狀，並模擬由 PID 控制迴路與壓電致動器行為所引起的失真。這使得能夠生成反映 STM 成像實際複雜性的龐大標註資料集，為訓練機器學習模型提供基礎。

利用這些合成資料集，我訓練了一個用於鎢二硒化物缺陷定位的深度學習模型，並將其表現與未考慮物理失真的模型進行比較。對真實 STM 測量結果的定性評估顯示，完整模擬所訓練出的模型具有更佳的泛化能力，證明高擬真模擬對於推進 STM 影像分析自動化的潛力。

關鍵字：掃描式穿隧顯微鏡（STM）、合成 STM 影像、深度學習、二維材料、鎢二硒化物，過渡金屬二硫族化合物

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

Abstract

Scanning Tunneling Microscopy (STM) is a highly effective technique for imaging the surfaces of conductive or semiconductive materials with a resolution down to single atoms, but it remains limited by its reliance on labor-intensive expert interpretation and susceptibility to artifacts introduced by tip condition, scanning parameters, and hardware-related distortions. These factors hinder its speed and scalability.

In this work, I present a simulation framework for generating synthetic STM images grounded in physical modelling. The system calculates electronic structure using the tight-binding method, models realistic tip geometries, and simulates distortions arising from PID control loop and piezoelectric actuator behaviour. This allows for the generation of large labelled datasets that reflect the real-world complexity of STM imaging, providing a foundation for training machine learning models.

Using these synthetic datasets, I trained a deep learning model for defect localization in tungsten diselenide, comparing its performance against a model trained without

doi:10.6342/NTU202501885

physical distortions. Qualitative evaluation on real STM measurements reveals superior generalization when using the full simulation, demonstrating the potential of high-fidelity simulations to advance automated STM image analysis.

Keywords: Scanning Tunneling Microscopy (STM), Synthetic STM Imaging, Deep Learning, 2D Materials, Tungsten Diselenide, Transition-metal dichalcogenides

doi:10.6342/NTU202501885

Contents

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

List of Figures

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

Chapter 1 Background

1.1 Related Work

In recent years, there has been a surge of interest in leveraging machine learning for applications in Scanning Tunneling Microscopy (STM). These include denoising [2], measurement workflow automation [3], localizing atomic-scale defects [4], autonomous tip conditioning [5, 6], nematic electronic order identification [7], and nanoparticle detection [8].

Most of these approaches rely on manually annotated STM measurements [3–6]. However, collecting and labeling such datasets is time-consuming, and often needs to be repeated for each material or microscope configuration, limiting scalability and generalizability.

To overcome these challenges, several works have proposed generating synthetic data through simulation [2, 7, 9, 10]. These simulations are based on a range of physical models, including methods such as density functional theory (DFT) [11], heuristic convolution models [9], and tight-binding approaches [2, 7]. Among these, the tight-binding method stands out for its fast speed and reasonable accuracy, making it particularly attractive to use for generating large-scale synthetic datasets.

doi:10.6342/NTU202501885

Despite their advantages, many simulation-based approaches assume idealized conditions, such as constant-height mode, single-atom tip apices with constant local density of states (LDOS), and minimal Gaussian-like distortions. These assumptions often overlook distortions introduced by Proportional-Integral-Derivative (PID) control loops, piezoelectric actuators and tip geometry, which can significantly affect STM measurements.

This work aims to address these limitations by presenting a more comprehensive simulation framework. In addition to generating STM images using tight-binding electronic structure, it incorporates a realistic model of the tip apex, accounts for various scanning distortions and enables dataset generation at scale for machine learning tasks.

doi:10.6342/NTU202501885

1.2 Scanning Tunneling Microscopy (STM)

STM is based on the phenomena of quantum tunneling that allows electrons to pass through a potential barrier.

This effect is used to generate a current through a vacuum gap from a sample to an atomically sharp tip while scanning the surface.

Unlike classical physics, in quantum physics, the electron behaves like a wave. And its behavior in a potential is governed by the Schrödinger's equation:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dz^2}\phi(z) + U(z)\phi(z) = E\phi(z) \tag{1.1}$$

For regions where $U > E$ the solution is equal to

$$\phi(z) = \phi(0)e^{\pm\kappa z}, \qquad \kappa = \frac{\sqrt{2m(U-E)}}{\hbar} \tag{1.2}$$

Where the sign corresponds to the direction of movement. The wave function is always decaying exponentially through the barrier, in order to satisfy the square integrability principle (see Figure 1.1). [12]

doi:10.6342/NTU202501885

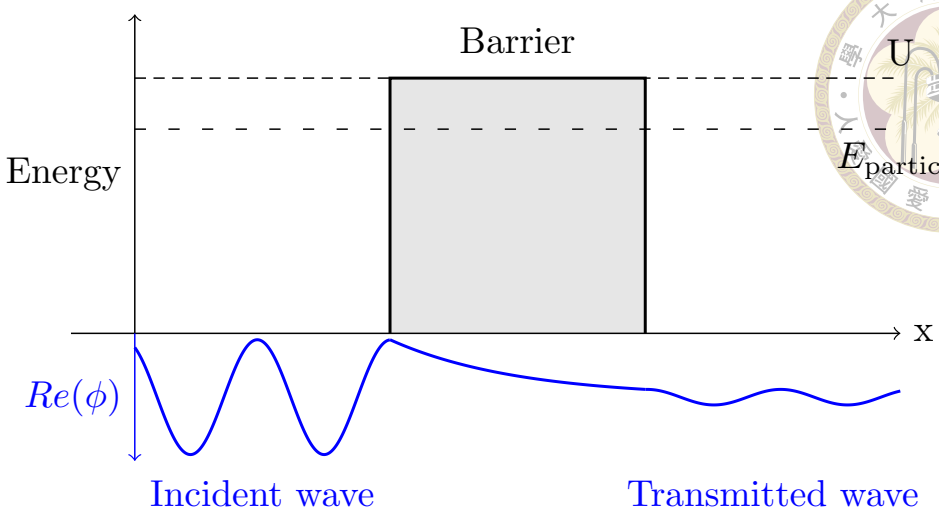


Figure 1.1: Illustration of an exponential decay through a potential barrier.

As we would expect the tunneling current to be proportional to the square of the wavefunction, we can say that it must decay exponentially with respect to distance with a decay constant κ:

$$I(z) \propto e^{-2\kappa z} \tag{1.3}$$

where z is the distance between the tip and the sample and typical value for κ is 11.4nm^{-1}.

Now that we understand how the tunneling current depends on the tip–sample distance, another key aspect to consider is its dependence on the electronic structure of both the sample and the tip.

Under certain assumptions [13], the Tersoff–Hamann model simplifies the tunneling current into:

$$I(\rho_T, \rho_S) \propto \int_0^{eV} \rho_T(E_F - eV + \epsilon)\rho_S(E_F + \epsilon)d\epsilon \tag{1.4}$$

doi:10.6342/NTU202501885

where ρ_T, ρ_S are the local densities of states (LDOS) of the tip and the sample respectively, E_F is the fermi level and V is the applied voltage between the tip and the sample.

This relation shows how the tunneling current depends on the overlap between the tip and sample density of states (DOS) within the applied bias window, as illustrated in Figure 1.2.

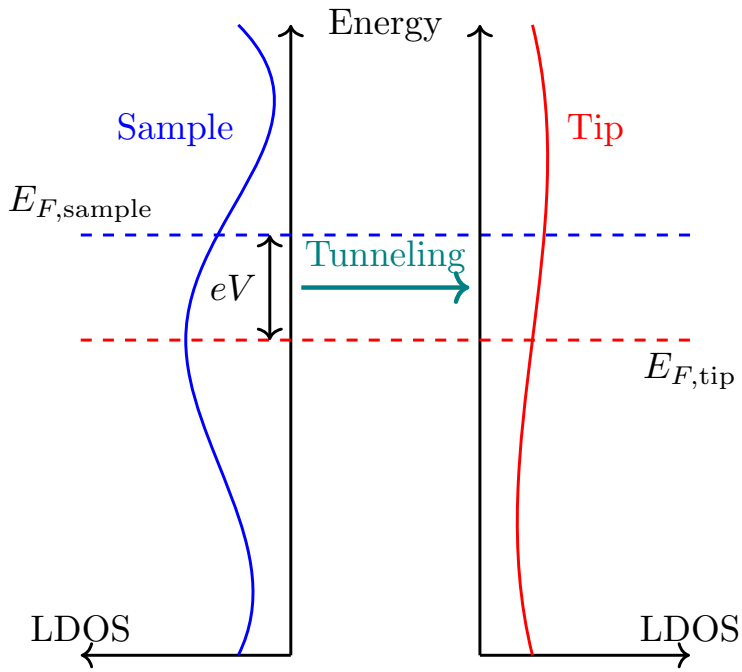


Figure 1.2: Illustration of the quantum tunneling dependence on DOS.

1.2.1 Feedback Control (PID)

The most common operation of STM is the *constant current mode*. In this mode, the tip height is continually adjusted to keep the tunnelling current at a constant value. This results in a more consistent imaging and prevents the tip from crashing into the sample surface.

To maintain a steady current, a feedback control system is required—typically a Proportional-Integral-Derivative (PID) controller is used. This controller calculates the

doi:10.6342/NTU202501885

deviation of the measured current from its target value, then adjusts the tip position based on the following three terms [14]:

- The **proportional** term responds to the immediate deviation.
- The **integral** term smooths out noise by accounting for the accumulated deviation values over time.
- The **derivative** term anticipates future deviations by responding to their rate of change.

$$\Delta z(t) = K_p \cdot e(t) + K_i \cdot \int_{t-t^*}^{t} e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt} \tag{1.5}$$

where $e(t)$ is the error or deviation between the target current and the measured current. This error can either be linear: $e(t) = I_{\text{target}} - I(t)$, however, the most common way is to subtract logarithms of the currents to adjust for the exponential dependency on distance $e(t) = \log(I_{\text{target}}) - \log(I(t))$.

In order to avoid amplification of noise, K_d is often set to zero. Simplifying the system to a PI controller. [14]

1.2.2 Piezoelectric Actuators

In order to perform the PID control, a sub-nanometer positioning system is required. This is typically achieved using **piezoelectric actuators**, which exploit the *inverse piezoelectric effect*—a phenomenon where an applied voltage induces a physical displacement.

Piezoelectric materials are crystalline solids with uniformly aligned unit cells. When

doi:10.6342/NTU202501885

subjected to mechanical strain, their internal ionic displacements generate electric dipoles, resulting in a measurable voltage—an effect known as the piezoelectric effect (see Figure 1.3).

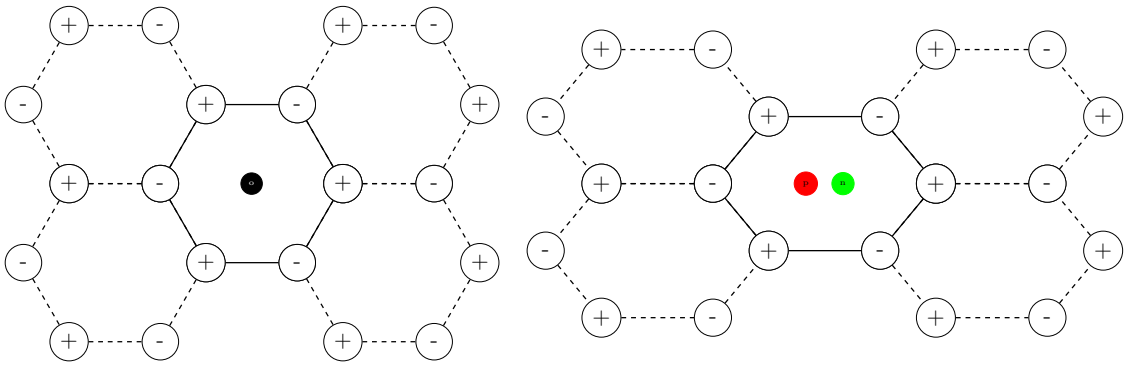


Figure 1.3: Illustration of the piezoelectric effect. (a) Unstrained crystal, charges cancel out. (b) Strained crystal, generating a polarization.

These materials also exhibit the inverse piezoelectric effect, where applied voltage results in a strain [15]:

$$S_{ij} = d_{ijk}E_k + s_{ijkl}T_{kl} \tag{1.6}$$

Where S is the strain tensor, s is compliance, T is stress and d is the piezoelectric tensor which denotes the strength of the piezoelectric effect.

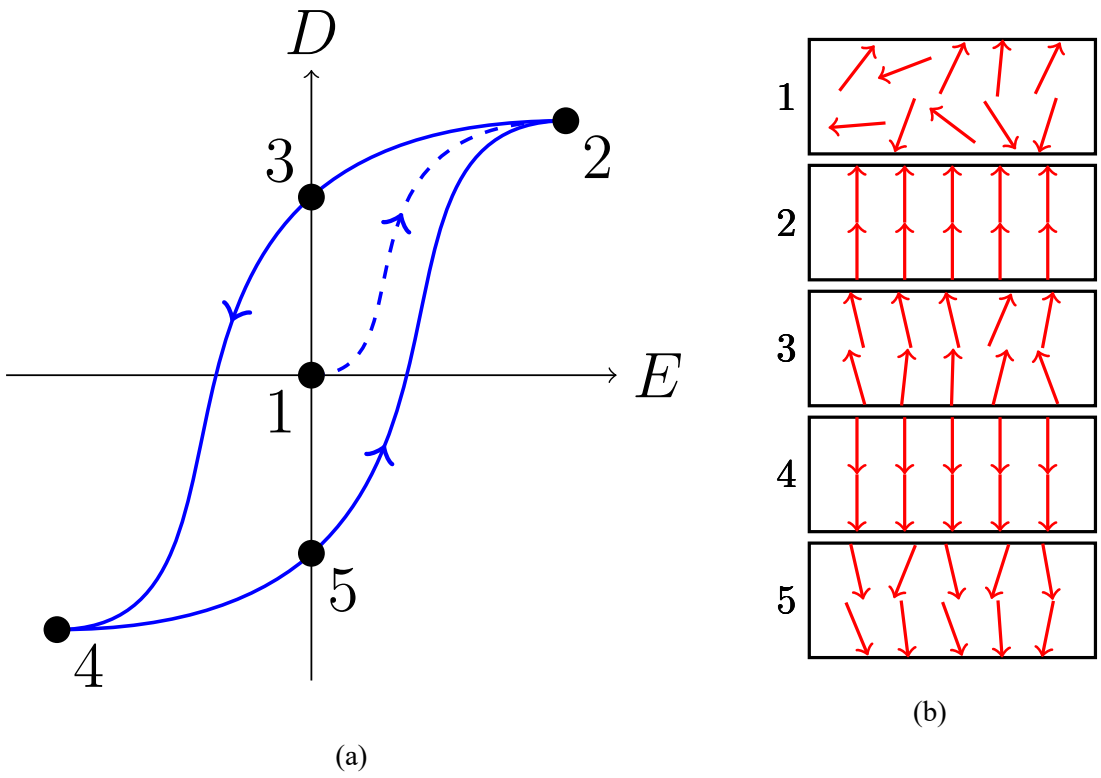
For typical piezoelectric plates used in actuators under no external stress, the strain simplifies to a form that implies the applied voltage and displacement relation to be linear:

$$y(t) \propto V(t) \tag{1.7}$$

where $y(t)$ is the displacement at time t and $V(t)$ is the applied voltage at time t.

doi:10.6342/NTU202501885

Hysteresis However, the linear relationship 1.7 holds only approximately. The piezoelectric tensor d depends on the net polarization of the material, which can change in a path-dependent manner as the internal domains reorient in response to varying electric fields. This hysteretic behavior introduces nonlinearities, as illustrated in Figure 1.4.



(a) (b)

Figure 1.4: Illustration of the domain orientation on the applied electric field. (a) Electric displacement field change ($D = \epsilon_0 E + P$). (b) Domain reorientation.

This path-dependent behavior gives rise to hysteresis—a non-linear relationship between the applied voltage and resulting displacement, contrary to the idealized linear dependence described in Equation 1.7 [16].

Several models exist to capture this hysteresis behavior, one of which is the **Bouc–Wen model** [17], which describes the displacement as a function of both the input voltage and an internal hysteresis state governed by an ordinary differential equation:

doi:10.6342/NTU202501885

$$y(t) = \alpha V(t) + \beta H(t)$$
$$\dot{H}(t) = A\dot{V}(t) - B\left|\dot{V}(t)\right| H(t) - C\dot{V}(t)\,|H(t)| \tag{1.8}$$

where $H(t)$ is the internal hysteresis state and α, β, A, B, C are parameters of the hysteresis.

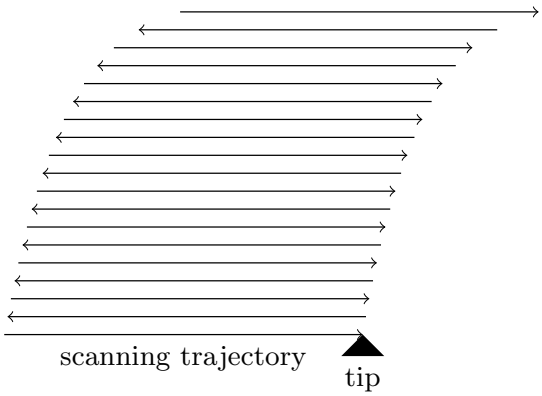
This model has been successfully applied to piezoelectric actuators exhibiting hysteresis [18]. Due to its simplicity and the relatively small number of parameters it requires, I have adopted the model for this work.

Piezoelectric Creep Another source of distortion from ideal behavior is *piezoelectric creep*. This phenomenon occurs when the applied voltage changes too quickly, and the resulting displacement does not respond instantaneously. Instead, it gradually approaches the new target logarithmically over time [19]:

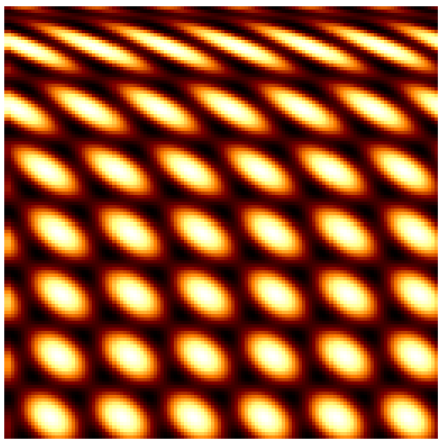
$$y(t, V) = y_0 \left[1 + \gamma(V) \log\left(\frac{t}{t_0}\right)\right] \tag{1.9}$$

In STM imaging, creep manifests as a logarithmic drift of the probe tip during scanning, causing each scan line to be shifted (see Figure 1.5).

doi:10.6342/NTU202501885



(a)



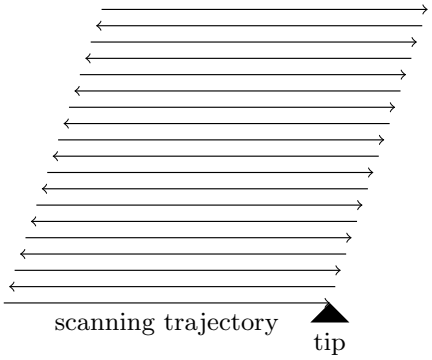
(b)

Figure 1.5: Creep effect in the fast scanning axis. (a) Scanning trajectory of the STM tip. (b) Resulting image.

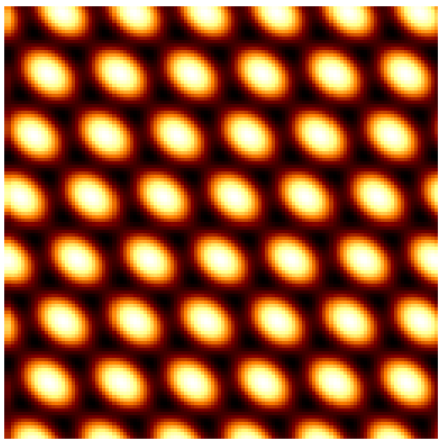
Thermal Drift Another distortion affecting the tip's movement is thermal drift. This distortion does not come just from the piezoelectric actuator itself, but rather from other components of STM that expand or shrink as a response to a change in temperature. This affect is usually linear with time [20]:

$$y(t) = y_0 + \alpha t \tag{1.10}$$

where α is the speed of the drift.



(a)



(b)

Figure 1.6: Illustration of the thermal drift effect in the fast scanning axis. (a) Scanning trajectory of the STM tip. (b) Resulting image.

doi:10.6342/NTU202501885

Closed loop control To mitigate these effects, **closed-loop control** is often employed in piezoelectric actuator systems. In this approach, external sensors measure the actual displacement, and a control system corrects the applied voltage in real time to match the desired position [21].

However, implementing closed-loop control in STM is challenging due to the extreme precision required—often below one angstrom. Current sensor technologies do not provide the sensitivity needed at this scale, which makes open-loop control the default in most STM systems despite its susceptibility to hysteresis, creep and drift [22].

doi:10.6342/NTU202501885

1.3 Electronic Structure

1.3.1 Tight-Binding Method

The tight-binding (TB) approximation is a method for modeling the electronic structure of solids. It assumes that electrons are tightly bound to their corresponding atomic sites. By constructing a basis of localized orbitals and considering electron hopping between them, the tight-binding model provides a physically intuitive and computationally efficient framework. [23]

In its simplest form, the tight-binding Hamiltonian is expressed as [24]:

$$\hat{H} = \sum_i \varepsilon_i \, c_i^\dagger c_i + \sum_{i \neq j} t_{ij} \, c_i^\dagger c_j, \tag{1.11}$$

where ε_i denotes the on-site energy of an electron localized on the orbital site i, t_{ij} is the hopping integral between orbital sites i and j, $c_i^\dagger$ are fermionic creation operators and c_j are annihilation operators.

The hopping integral t_{ij} represents the energy associated with an electron moving between two atomic orbitals, determined by their wavefunction overlap:

$$t_{ij} = \langle \phi_i | \hat{H} | \phi_j \rangle \equiv \int \phi_i^*(\mathbf{r}) \hat{H} \phi_j(\mathbf{r}) \, d\mathbf{r}, \tag{1.12}$$

where ϕ_i and ϕ_j are atomic-like orbitals centered on sites i and j, respectively. The on-site energy is then $\varepsilon_i = \langle \phi_i | \hat{H} | \phi_i \rangle$.

The Hamiltonian can now be conveniently discretized into a matrix using the chosen

doi:10.6342/NTU202501885

basis of localized orbitals:

$$H_{ij} = \begin{cases} \varepsilon_i, & \text{if } i = j, \\ t_{ij}, & \text{if } i \neq j \text{ and sites } i, j \text{ are connected,} \\ 0, & \text{otherwise.} \end{cases} \tag{1.13}$$

This leads to a sparse Hermitian matrix **H**, where each row and column corresponds to a basis orbital.

Solving the time independent Schrödinger equation,

$$\hat{H}|\psi_n\rangle = E_n|\psi_n\rangle, \tag{1.14}$$

reduces to a matrix eigenvalue problem:

$$\mathbf{H}\psi_n = E_n\psi_n, \tag{1.15}$$

where ψ_n is the discretized wavefunction represented as a vector of amplitudes on the atomic orbitals, and E_n is the corresponding energy eigenvalue. This problem can be solved using standard linear algebra techniques, such as direct diagonalization or sparse eigensolvers, depending on the system size.

Construction of a Tight-Binding model To construct tight-binding models, one requires knowledge of the geometric structure of the material and, critically, the values of the hopping integrals t_{ij}. These parameters are often obtained from *ab initio* calculations through a process known as Wannierization, which projects the delocalized Bloch states onto a localized orbital basis. [23]

doi:10.6342/NTU202501885

Alternatively, the Slater–Koster method provides a framework for computing hopping integrals based on tabulated parameters that account for orbital symmetries and interatomic directions. These parameters can themselves be derived from *ab initio* calculations or empirically fitted to experimental data. [25]

1.3.2 Local Density of States (LDOS)

The local density of states (LDOS) quantifies the distribution of available electronic states at a given energy and location. The LDOS at a site i can then be described as:

$$\rho_i(E) = \sum_n |\langle i|\psi_n\rangle|^2 \, \delta(E - E_n), \tag{1.16}$$

where ψ_n is the n-th eigenstate of the Hamiltonian, with energy E_n.

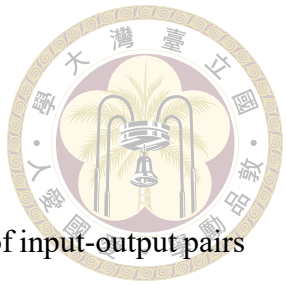
Numerically, we can evaluate this by approximating the dirac delta function using the Lorentzian function:

$$\delta(E - E_n) \approx \frac{1}{\pi} \frac{\eta}{(E - E_n)^2 + \eta^2} \tag{1.17}$$

where η is a fitted broadening parameter.

doi:10.6342/NTU202501885

1.4 Deep Learning



In this work, I use the supervised learning paradigm, where a set of input-output pairs is provided (i.e., a labeled dataset) and a model is built with the goal of predicting output values of previously unseen inputs.

1.4.1 Loss functions

To assess a model's performance on a given dataset, we use a **loss function** that measures how the model's predictions deviate from the correct values. The loss is averaged across all samples in the dataset to obtain a single performance metric.

For regression problems, where the task is to predict real numbers, a commonly utilized loss function is the **mean squared error** (MSE):

$$\text{MSE}(\hat{\mathbb{Y}}, \mathbb{Y}) = \frac{1}{N}\sum_{i=0}^{N}(\hat{y}_i - y_i)^2 \tag{1.18}$$

where $\mathbb{Y} = \{y_i\}_{i=0}^{N}$ are the correct output values and $\hat{\mathbb{Y}} = \{\hat{y}_i\}_{i=0}^{N}$ are the predicted output values.

However, MSE can yield excessively large gradients in the presence of outliers, potentially causing instability during training. To address this, the smooth L1 loss is commonly employed [26]:

$$\text{smooth\_L}_1(\hat{y}, y) = \begin{cases} 0.5 \cdot (\hat{y}_i - y_i)^2, & \text{if } |\hat{y}_i - y_i| < 1 \\ (\hat{y}_i - y_i) - 0.5, & \text{otherwise} \end{cases} \tag{1.19}$$

doi:10.6342/NTU202501885

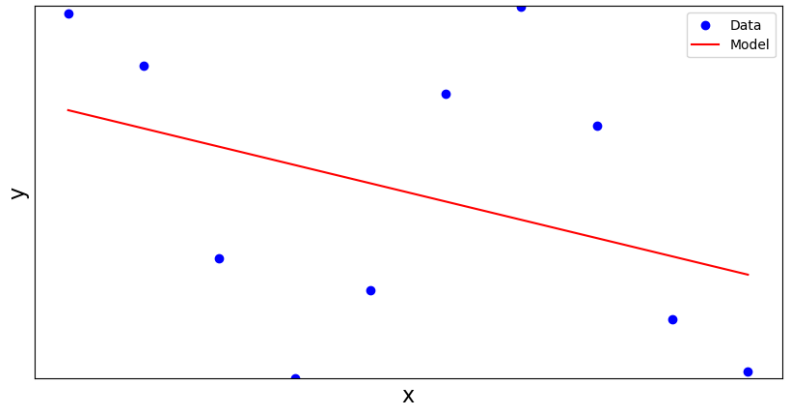
1.4.2 Model assessment

A deep learning model consists of a sequence of layers, where each layer receives inputs from the previous layer and passes processed outputs to the next. The computations within each layer depend on a set of parameters. The process of adjusting these parameters to minimize the loss function on a dataset is called training. The model's ability to precisely fit a large amount of samples is known as its capacity, and it depends on factors such as the number of layers, the number of parameters in those layers, and any constraints applied during training.

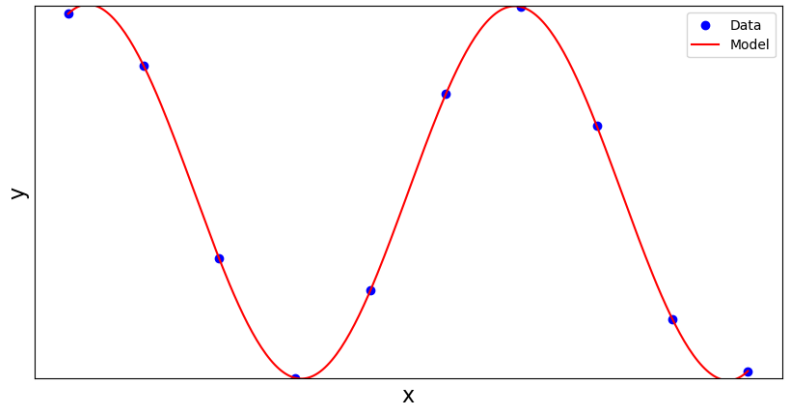
To evaluate how much does the model generalize to previously unseen data, the dataset is typically split into training and validation sets. The training set is used to update the model parameters, while the validation set guides decisions about the model's capacity and training strategy. A separate test set is often reserved for final evaluation, ensuring an unbiased estimate of the model's real-world performance.

If a model performs well on a training set but poorly on the test set, it is said to overfit. If it fails to perform well even on the training set, it is called underfitting. See Figure 1.7.

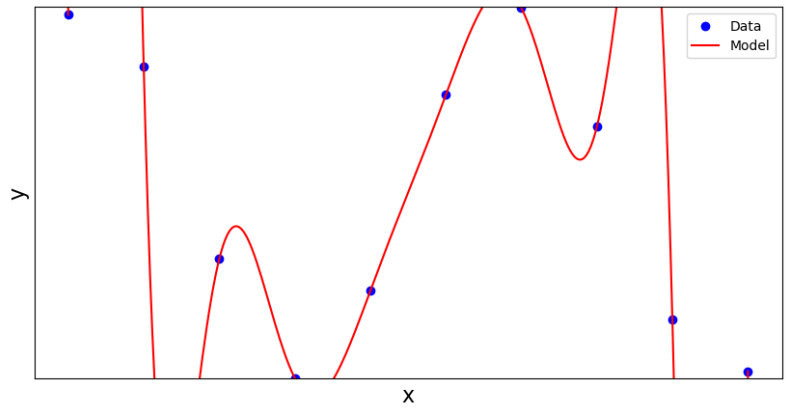
doi:10.6342/NTU202501885



(a)



(b)



(c)

Figure 1.7: Illustration of underfitting (a), appropriate capacity (b) and overfitting (c).

1.4.3 Training

Minibatch Stochastic Gradient Descent (SGD) A common approach of minimizing functions is the gradient descent, where we iteratively use the gradient of the function to

doi:10.6342/NTU202501885

update its parameters in the direction of the largest descent.

$$\mathbf{w}_{n+1} \leftarrow \mathbf{w}_n - \alpha \nabla_{\mathbf{w}} \left(\frac{1}{N} \sum_{i=0}^{N} L(f(x_i; \mathbf{w}_n), y_i) \right) \tag{1.20}$$

where L is the loss function, f is the model, and $\mathbf{w}$ are its parameters. The dataset consists of input-output pairs $\{(x_i, y_i)\}_{i=0}^{N}$ and α is the size of the step, also called as the *learning rate*.

However, computing the gradient of a loss function with respect to the whole dataset is computationally expensive, especially for large datasets. At the same time, introducing some randomness to our parameter updates helps to prevent the model from getting stuck in local minimums. Both of these drawbacks are addressed by the *minibatch stochastic gradient descent (SGD)* algorithm, where we approximate the gradient only by accounting for a random subset of samples from the dataset [27]:

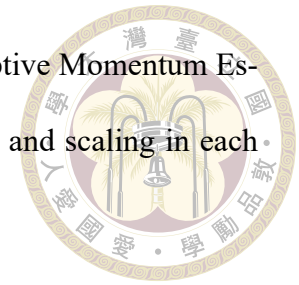
$$\nabla_{\mathbf{w}} \left(\frac{1}{N} \sum_{i=1}^{N} L(f(x_i; \mathbf{w}_n), y_i) \right) \approx \frac{1}{|\text{Batch}|} \sum_{i \in \text{Batch}} \nabla_{\mathbf{w}} L(f(x_i; \mathbf{w}_n), y_i) \tag{1.21}$$

where the set Batch $\subset (0, \ldots, N)$ defines a random subset of the dataset.

The smaller the batch, the more random the updates are.

Adam optimizer In some scenarios, such as steep gradients and saddle regions, standard SGD may converge slowly or be unstable. To address this issue, modifications for SGD were developed, these modifications often incorporate scaling of gradients and accounting for gradient momentums (decaying weighted average of a certain number of past minibatch gradients).

doi:10.6342/NTU202501885



The most widely used optimization algorithm today is the Adaptive Momentum Estimation (*Adam optimizer*) [28] that utilizes the following moments and scaling in each update iteration:

$$
\begin{gathered}
m_1 = 0 \qquad v_1 = 0 \\
\mathbf{m}_{n+1} \leftarrow \beta_1 \cdot \mathbf{m}_n + (1-\beta_1) \cdot \mathbf{g}_n \qquad \hat{\mathbf{m}}_{n+1} \leftarrow \frac{\mathbf{m}_{n+1}}{1-\beta_1^{n+1}} \\
\mathbf{v}_{n+1} \leftarrow \beta_2 \cdot \mathbf{v}_n + (1-\beta_2) \cdot \mathbf{g}_n^2 \qquad \hat{\mathbf{v}}_{n+1} \leftarrow \frac{\mathbf{v}_{n+1}}{1-\beta_2^{n+1}} \\
\mathbf{w}_{n+1} \leftarrow \mathbf{w}_n - \alpha_{n+1} \cdot \frac{\hat{\mathbf{m}}_{n+1}}{\sqrt{\hat{\mathbf{v}}_{n+1}} + \epsilon}
\end{gathered}
\tag{1.22}
$$

where $\mathbf{g}_n$ is the loss gradient of the batch for the n-th iteration, $\hat{\mathbf{m}}_n, \hat{\mathbf{v}}_n$ are the unbiased estimates of the first and second gradient moment, $\{\alpha_i\}_{i=0}^{M}$ is a sequence of learning rates, usually constant or decaying, $\beta_1, \beta_2 \in [0, 1)$ are the decaying rates of gradient moments, $\epsilon > 0$ is a small constant to prevent divison by zero.

1.4.4 Types of layers

Feed forward layer Feed forward layer is based on a statistical method called linear regression, where we try to fit data using a linear function:

$$
\mathbf{Y} = \mathbf{X}\mathbf{W} + \mathbf{1}_n\mathbf{B} \tag{1.23}
$$

where $\mathbf{Y} \in \mathbb{R}^{p \times n}$ are the output values of dimension p corresponding to n samples, $\mathbf{X} \in \mathbb{R}^{m \times n}$ are input values of dimension m and $W \in \mathbb{R}^{m \times p}, B \in \mathbb{R}^{1 \times p}$ are trainable parameters.

Stacking multiple feedforward layers without any nonlinear modifications would be equivalent to a single linear transformation, because a composition of linear functions is

doi:10.6342/NTU202501885

still linear. To model nonlinear relationships and increase the network's capacity, we apply nonlinear functions known as activation functions. A common example is the Rectified Linear Unit (ReLU):

$$\mathrm{ReLU}(y) = \begin{cases} y, & \text{if } y > 0 \\ 0, & \text{otherwise} \end{cases} \tag{1.24}$$

Convolutional layer Feed forward networks usually do not perform well on dimensional data, such as images. For these scenarios, convolutional layers are usually used.

A two-dimensional convolution of a tensor $X \in \mathbb{R}^{I \times J \times C}$, where $I \times J$ is its resolution and C its number of channels, with stride equal to one would be equal to:

$$(K * X)_{i,j,o} = \sum_{m,n,c} X_{i+m,j+n,c} K_{m,n,c,o} \tag{1.25}$$

where $K \in \mathbb{R}^{M \times N \times C \times O}$ is the kernel we convolve the input with, values inside this kernel are the trainable parameters of the convolutional layer.

For an illustration, see Figure 1.8.

The number of pixels we move in each step of the convolution operation is called stride.

doi:10.6342/NTU202501885

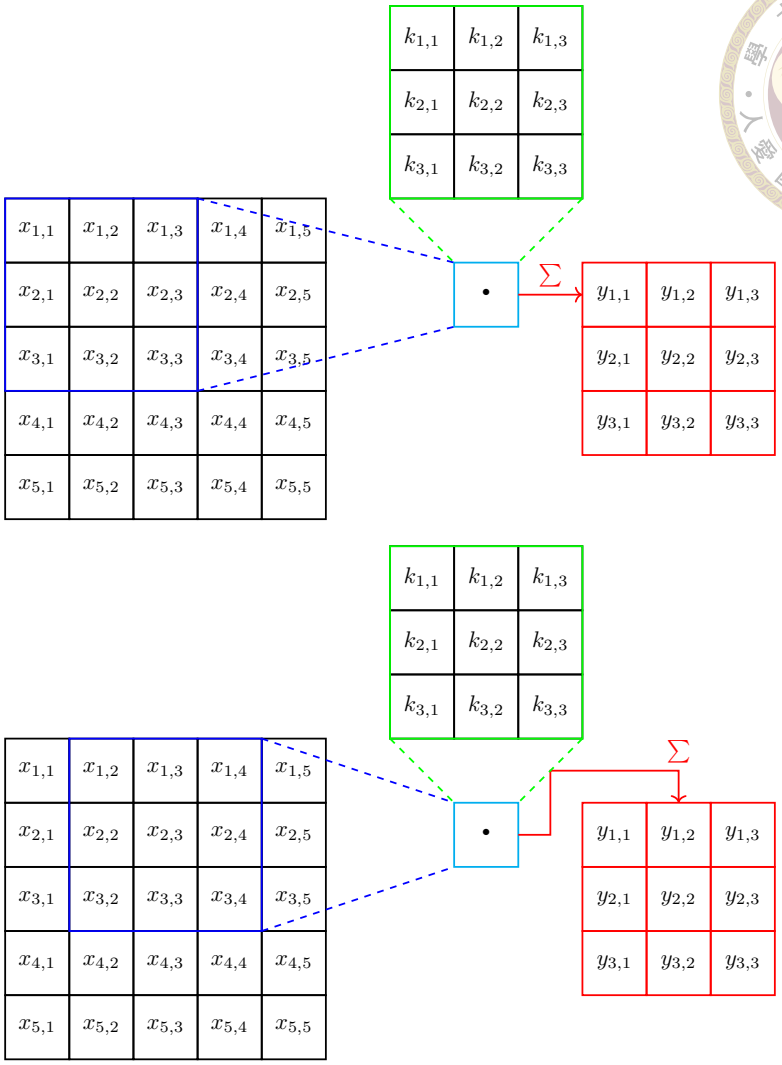


Figure 1.8: Illustration of a convolution with a stride of one pixel.

Transposed convolutional layer As illustrated, convolutional layers will produce an output with the same or smaller resolution than its input. However, sometimes we might want to upscale the output compared to the input. In these cases, we will use the transposed convolutional layer that is analogous to the convolutional layer, but its stride has an inverse effect by adding padding around input pixels, see Figure 1.9.

doi:10.6342/NTU202501885

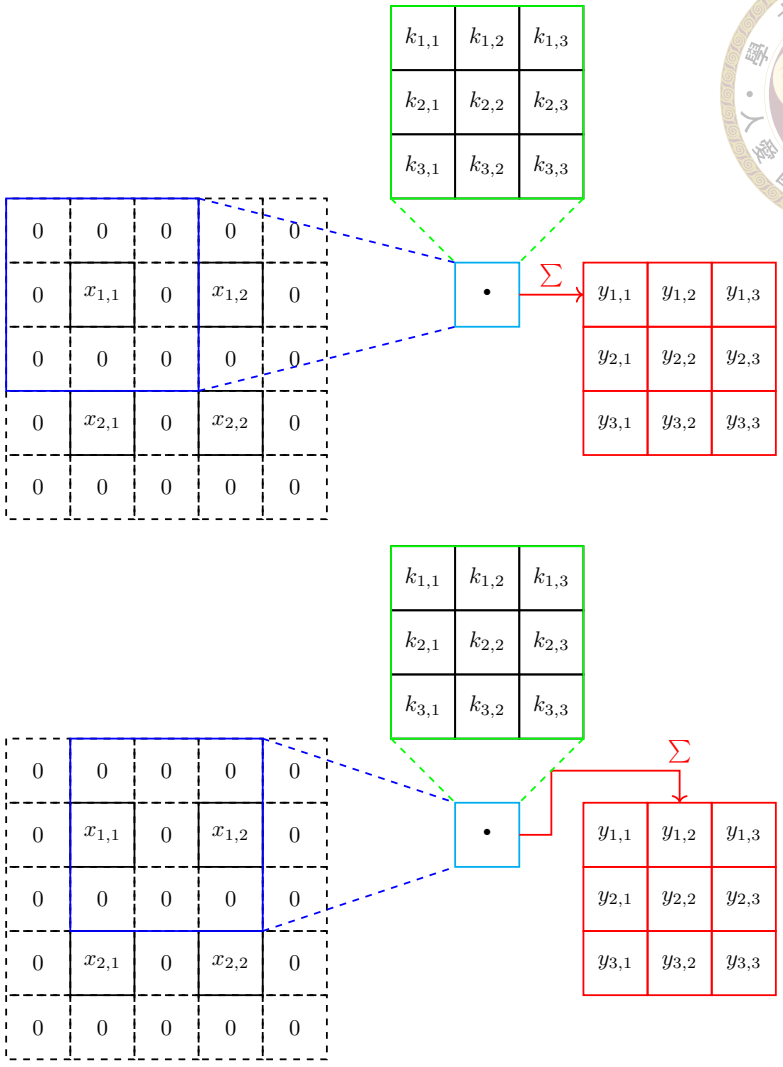


Figure 1.9: Illustration of a transposed convolution with a stride of two pixels.

An example application of the transposed convolutional layer is the decoder branch of the *U-Net* architecture, first introduced in [29]. It was developed for the task of image segmentation, where the model predicts, for each pixel, whether it belongs to a particular object. U-Net employs an encoder-decoder structure: a pre-trained Convolutional Neural Network (CNN) is used as the encoder to capture features through progressive downsampling, while the decoder generates spatial predictions with progressive upsampling. The model also features connections between the corresponding encoder and decoder layers to preserve spatial information (see Figure 1.10).

doi:10.6342/NTU202501885

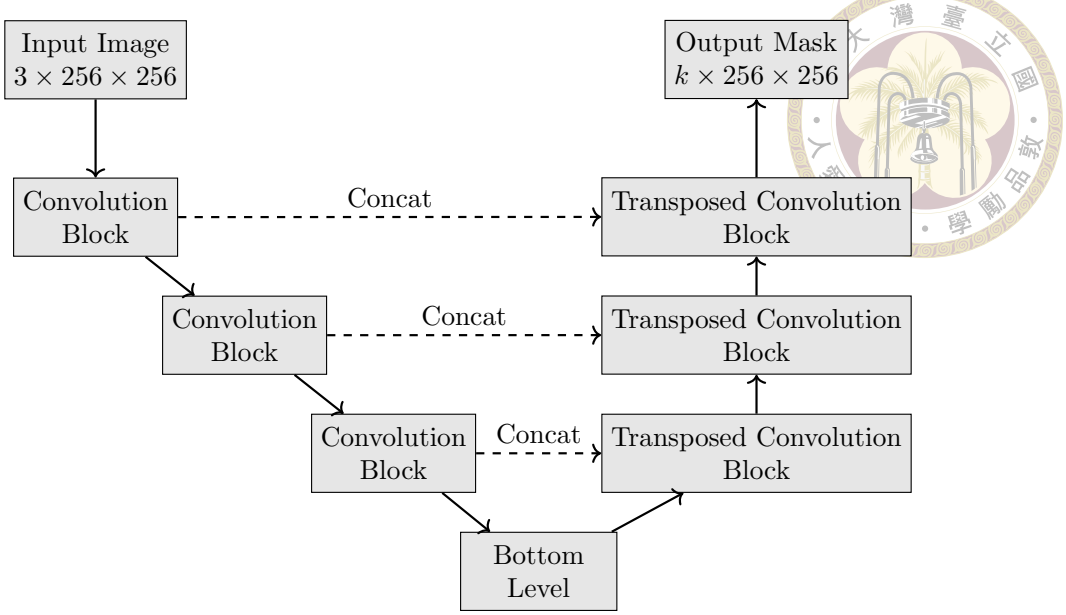


Figure 1.10: Illustration of the U-Net architecture.

Batch normalization Batch normalization layers are used to stabilize gradients by normalizing and rescaling inputs to another layer [30]:

$$\mathrm{BN}(x_i) = \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta \tag{1.26}$$

where σ_B^2, μ_B are the variance and mean calculated for the current batch. γ, β are learnable scaling parameters.

If the network switches to an inference mode, then σ_B^2, μ_B are replaced by their running estimates computed during the training.

Residual connection and ResNet People observed that stacking many convolutional layers in a network results in a suboptimal training as shown in the Figure 1.11.

The reason for this is that increasing the amount of convolutional layers increases the roughness of the loss function [31], making the model stuck in local minimums during

doi:10.6342/NTU202501885

optimization.

To prevent this behavior we can add *residual connections* to our model, where we add the input values to the output values of a layer or a sequence of layers [1]:

$$\text{Output} = \text{CNN}(x) + x \tag{1.27}$$

where $CNN(x)$ is a transformation by a sequence of convolutional layers and x is the input.

Applying residual connections after each block of two or three convolutional layers greatly smooths the loss function [31] and empirically leads to better training results with increasing depth; see Figure 1.11.

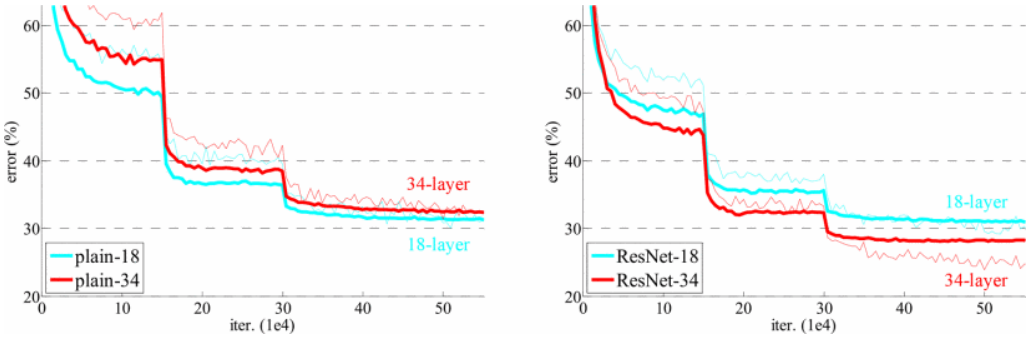


Figure 1.11: Comparison of training errors for models of different depths. The left figure shows that, without residual connections, error increases with depth. In contrast, the right figure shows that models containing residual connections achieve lower training error with greater depth. Taken from [1] © 2016 IEEE.

The paper [1] was the first to introduce an architecture based on convolutional layers with residual connections, referred to as *ResNet*.

doi:10.6342/NTU202501885

1.5 Numerical Analysis

Many of the equations encountered in this work are too complex to solve analytically. Numerical methods are utilized to overcome this problem, such as Newton's method for root-finding problems and Runge-Kutta method for ordinary differential equations (ODE).

1.5.1 Root finding problem - Newton's method

Consider a non-linear equation:

$$f(x) = 0 \qquad \text{where } f : \mathbb{R} \to \mathbb{R}, x \in \mathbb{R} \tag{1.28}$$

To numerically solve this equation, we iteratively generate a sequence of approximations $\{x_n\}$ intended to approach the root x. Assuming $f \in C^2([x, \hat{x}_n])$, then we can rewrite the equation using the Taylor expansion with Lagrange remainder:

$$\exists \xi_n \in [x, \hat{x}_n] : f(x) = f(\hat{x}_n) + f'(\hat{x}_n)(x - \hat{x}_n) + \frac{1}{2} f''(\xi_n)(x - \hat{x}_n)^2 \tag{1.29}$$

Based on this relation, Newton's method makes an approximation of f as a linear function with a slope equal to its derivative and updates the guess accordingly (for illustration see Figure 1.12):

$$\begin{aligned} 0 = f(x) &\approx f(\hat{x}_n) + f'(\hat{x}_n)(x - \hat{x}_n) \\ \Rightarrow \boxed{\hat{x}_{n+1} &= \hat{x}_n - \frac{f(\hat{x}_n)}{f'(\hat{x}_n)}} \end{aligned} \tag{1.30}$$

doi:10.6342/NTU202501885

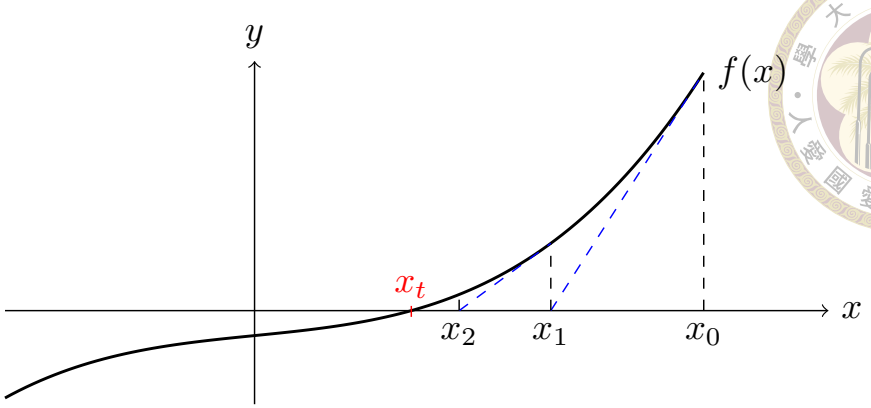


Figure 1.12: Illustration of the Newton's method. Iterations x_0, x_1, x_2 approach the root x_t.

Using the formula 1.29, we can derive the error of the updated estimate ($\epsilon_{n+1} \equiv x - \hat{x}_{n+1}$) should be equal to:

$$\epsilon_{n+1} = \frac{-f''(\xi_n)}{2f'(\hat{x}_n)}\epsilon_n^2 \tag{1.31}$$

This shows that under appropriate conditions Newton's method is at least quadratically convergent. These conditions are [32]:

$$\begin{aligned} &f \in C^2(I), \text{ where } I \text{ is an open interval around the root } x \\ &f'(x) \neq 0 \\ &\hat{x}_0 \text{ sufficiently close to } x \end{aligned} \tag{1.32}$$

1.5.2 Solving ordinary differential equations (ODE)

Here, we aim to find mesh points $\{(\hat{y}_i, t_i)\}_{i=1}^N$ (where $t_i = a + ih$ and $\hat{y}_i \approx y(t_i)$) close to the solution ($y(t)$) of an ODE:

doi:10.6342/NTU202501885

$$\frac{dy}{dt} = f(t, y), \qquad t_1 \leq t \leq t_N, \qquad y(t_1) = \alpha \tag{1.33}$$

where $y : [t_1, t_N] \to \mathbb{R}$, α is the boundary value and $[t_1, t_N]$ is the interval of interest.

Euler's method We assume that $y \in C^2([t_1, t_N])$, so that we can apply the Taylor expansion:

$$\exists \xi_i \in [a, b] : y(t_{i+1}) = y(t_i) + (t_{i+1} - t_i)y'(t_i) + \frac{(t_{i+1} - t_i)^2}{2} y''(\xi_i) \tag{1.34}$$

Euler's method assumes that the Lagrange remainder is close to zero, which then simplifies the formula 1.34 into:

$$\frac{(t_{i+1} - t_i)^2}{2} y''(\xi_i) \approx 0 \qquad \Rightarrow \qquad y(t_{i+1}) \approx y(t_i) + hf(t_i, y(t_i)) \tag{1.35}$$

Based on this, we can iteratively construct the mesh point values starting from the boundary value:

$$\begin{aligned} \hat{y}_1 &\equiv y(t_1) = \alpha \\ \hat{y}_{i+1} &= \hat{y}_i + hf(t_i, \hat{y}_i) \end{aligned} \tag{1.36}$$

Local truncation error refers to the error made in a single step of a method. For Euler's method this error equal to the Lagrange remainder (as shown in the equation 1.34):

doi:10.6342/NTU202501885

$$\epsilon = \frac{(t_{i+1} - t_i)^2}{2} y''(\xi_i) = \frac{h^2}{2} y''(\xi_i) \propto h^2 \tag{1.37}$$

Since the error is proportional to the square of the step size, we say that Euler's method has a local truncation error of order 2.

Classical Runge Kutta method The Runge-Kutta method improves upon Euler's method by estimating the slope over an internal between subsequent t_i as a weighted average of slopes within the interval:

$$\begin{aligned}
\hat{y}_0 &\equiv y(t_0) = \alpha \\
k_1 &= hf(t_i, \hat{y}_i) \\
k_2 &= hf\left(t_i + \tfrac{h}{2}, \hat{y}_i + \tfrac{1}{2}k_1\right) \\
k_3 &= hf\left(t_i + \tfrac{h}{2}, \hat{y}_i + \tfrac{1}{2}k_2\right) \\
k_4 &= hf\left(t_{i+1}, \hat{y}_i + k_3\right) \\
\hat{y}_{i+1} &= \hat{y}_i + \tfrac{1}{6}(k_1 + k_2 + k_3 + k_4)
\end{aligned} \tag{1.38}$$

This method has a local truncation error of the order of 5. [32]

doi:10.6342/NTU202501885

Chapter 2 Simulation

In this chapter, we simulate an STM image by first computing the tunneling current based on the Local Densities of States (LDOS) of the sample and the tip. From this we generate the ideal image in the constant current mode using a numerical method. Finally, a distorted image is created by simulating piezoelectric actuator and PID controller.

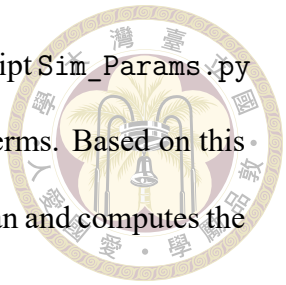
2.1 Input

The primary inputs to the simulation are:

- The lattice structure of the sample.
- The **local density of states (LDOS) of the tip atoms**.

Tip Structure The **tip structure** must be generated externally, and its LDOS provided via `.npy` files located in the `tip_ldos` directory. Each `.npy` file should contain a list of arrays, where each array has four elements: the first three specify the spatial coordinates of a tip atom, and the fourth gives its LDOS value at the relevant energy level. The middle value is expected to correspond to the Fermi level and the range and resolution must be consistent between both the tip and the sample.

doi:10.6342/NTU202501885



Sample Surface The function `get_lattice_definition` in the script `Sim_Params.py` must be implemented to define the sample's geometry and hopping terms. Based on this input, the `PyBinding` [33] package constructs the system's Hamiltonian and computes the LDOS for the 2D defective surface.

Sample Defects Vacancies are simulated by directly removing atoms from the system, while neglecting structural relaxation effects.

Dopants are modeled within the tight-binding framework as additional on-site potential energies at the dopant locations. This approximation offers a fast way to capture their electronic influence without requiring computationally expensive structural modifications.

As input, the user specifies the expected density of vacancies and dopants (per square nanometer), as well as the range of potential energy values for dopants.

These parameters can be interactively adjusted through a graphical user interface to match the characteristics of real experimental samples.

2.2 Tunneling Current Calculation

The tunneling current can be calculated using the formulas 1.3 1.4.

To use it in the simulation we can rewrite it in a discrete form, summing over the contributions of each pair of sample and tip atoms:

doi:10.6342/NTU202501885

$$\rho_{ij}^{\text{sum}}(V) \equiv \sum_{\epsilon=0}^{eV} \rho_i(E_F - eV + \epsilon)\rho_j(E_F + \epsilon)$$
$$D_{ij}(x,y,z) \equiv \sqrt{(x_i - (x_j + x))^2 + (y_i - (y_j + y))^2 + (z_i - (z_j + z))^2} \tag{2.1}$$
$$I(x,y,z,V) \approx \sum_{i \in \text{sample}} \sum_{j \in \text{tip}} e^{-2\kappa(D_{ij}(x,y,z) - \epsilon_D)} \rho_{ij}^{\text{sum}}(V)$$

where (x, y, z) denotes the position of the tip relative to the sample. The positions $(x_i, y_i, z_i)_{i \in \text{sample}}$ refer to the atomic coordinates of the sample, while $(x_j, y_j, z_j)_{j \in \text{tip}}$ are those of the tip atoms. The term ρ_{ij}^{sum} represents the integrated product of the local densities of states (LDOS) of sample atom i and tip atom j over the energy window defined by the applied bias V. The factor $e^{-2\kappa D_{ij}}$ describes the exponential decay of the tunneling probability with the interatomic distance D_{ij}, where κ is the decay constant related to the work function and ϵ_D is a constant for improving stability.

Stability Constant ϵ_D As we are only interested in the relative distribution of current, rather than exact absolute values and because the exponential term becomes numerically unstable for very large or very small exponents, we introduce a constant shift to the exponent to improve numerical stability.

This constant, denoted as ϵ_D, should be chosen to match the minimum value of $D_{ij}(x, y, z)$ across all relevant i, j, x, y, and z. This ensures that the computation maintains high precision at positions where the tip and sample are closest.

Constant Height Mode The constant-height mode can be directly computed using Equation 2.1. In this mode, the tip height is fixed at $z = z_{\text{initial}}$, and the scan is performed parallel to the surface. Each pixel p_{mn} corresponds to a tip position $(x, y) = (x_{\text{initial}} + mh, y_{\text{initial}} +$

doi:10.6342/NTU202501885

nh), where h is the pixel step size.

$$p_{mn} \equiv I(x_{\text{initial}} + mh, y_{\text{initial}} + nh, z_{\text{initial}}, V_{bias}) \tag{2.2}$$

2.3 Constant Current Mode

While some previous works, such as [2], have focused exclusively on simulating the constant-height mode, the constant-current mode is the most commonly used in experimental STM due to its ability to compensate for surface topography variations.

In this section, we focus on simulating the constant-current mode in two stages: first, under idealized conditions using a numerical solver; and then with the inclusion of realistic distortions caused by the PID controller and piezoelectric actuator behavior.

2.3.1 Numerical solution

Current Derivative To compute the surface in constant-current mode, we first examine the derivative of the tunneling current with respect to the tip height z:

$$\frac{dI(x,y,z,V)}{dz} \approx \sum_{i\in\text{sample}} \sum_{j\in\text{tip}} \left(-2\kappa \frac{z_i - (z_j + z)}{D_{ij}(x,y,z) + \epsilon} \cdot e^{-2\kappa(D_{ij}(x,y,z) - \epsilon_D)} \cdot \rho_{ij}^{\text{sum}}(V) \right) \tag{2.3}$$

here, we added a very small constant $\epsilon \approx 10^{-10}$ to ensure stability when $D_{ij} \to 0$

doi:10.6342/NTU202501885

Logarithmic Smoothing We would expect the logarithm of the tunneling current to behave more linearly, at least for large z. Let us examine its derivative too:

$$\frac{d\log\left(I(x,y,z,V)+\epsilon_I\right)}{dz} = \frac{1}{I(x,y,z,V)+\epsilon_I}\cdot\frac{dI(x,y,z,V)}{dz} \tag{2.4}$$

here, we added another stability constant ϵ_I to improve stability when $I(x,y,z,V)\to 0$.

Convergence of the Newton's Method Since the logarithm of the tunneling current is expected to vary approximately linearly with height, Newton's method is a suitable choice, as it assumes local linearity for convergence.

Now we examine the function $\log\left(I(x,y,z,V)+\epsilon_I\right)$ for conditions 1.32 of the quadratic convergence of the Newton's Method:

- $f \in C^2$

  Since $I(x,y,z,V)\to[0,\infty)$, we can see that in this form the function $\log\left(I(x,y,z,V)+\epsilon_I\right)\in C^2$ as its derivative is continuous and continuously differentiable for the whole range of I.

- $f' \neq 0$

  The derivative in Equation 2.4 equals zero only when all atoms of the tip lie in a single plane that coincides with the sample's mirror plane. In some cases, this configuration may cause instability due to excessively large gradients. To address this, derivative clipping can be introduced as a workaround.

- **Sufficiently close initial guess**

doi:10.6342/NTU202501885

If we choose the target current values as

$$I_{\text{target}} = \text{median}_{mn}\left(I(x_{\text{initial}} + mh, y_{\text{initial}} + nh, z_{\text{initial}}, V_{bias})\right) \tag{2.5}$$

then we should expect this initial guess z_{initial} to be close to its target values (z_{mn}) :

$\forall m, n\ I(x_{\text{initial}} + mh, y_{\text{initial}} + nh, z_{mn}) = I_{\text{target}}$

We see that we have mostly fulfilled all the conditions for the Newton Method to converge at least quadratically. We will use it to calculate the ideal case of constant current mode image (see Algorithm 1).

Algorithm 1 Newton's Method for Height Estimation

Require: $\epsilon, \epsilon_I > 0$ ▷ Stability constants

Require: $\tau > 0$ ▷ Clipping constant

Require: $z_{\text{initial}} \in \mathbb{R}$ ▷ Initial z-position

Require: $\mathbf{grid}_x, \mathbf{grid}_y \in \mathbb{R}^{r \times r}$ ▷ x, y scan axis positions

Require: $M \in \mathbb{N}$ ▷ Maximum number of iterations

$I_t \leftarrow \text{median}(\texttt{calculate\_current}(z_{\text{initial}}, \mathbf{grid}_x, \mathbf{grid}_y))$ ▷ Formula 2.1

$\mathbf{z} \leftarrow z_{\text{initial}} \cdot \mathbf{1}_r \mathbf{1}_r^\top$

for iter $= 1$ to M **do**

  $\mathbf{I}, \mathbf{I}' \leftarrow \texttt{calculate\_current\_and\_derivative}(\mathbf{z}, \mathbf{grid}_x, \mathbf{grid}_y)$ ▷ Formula 2.3

  $\mathbf{I}'_{\text{clipped}} \leftarrow \min\left(\max\left(\mathbf{I}', -\tau\right), \tau\right) + \epsilon$ ▷ Derivative clipping

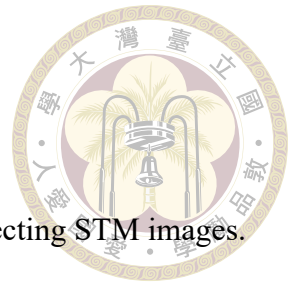
  $\mathbf{z} \leftarrow \mathbf{z} + \dfrac{\mathbf{I} + \epsilon_I}{\mathbf{I}'_{\text{clipped}}} \cdot \left(\log(I_t + \epsilon_I) - \log(\mathbf{I} + \epsilon_I)\right)$ ▷ Formula 1.30

end for

In practice, I observed that Algorithm 1 becomes unstable when the initial height satisfies $z_{\text{initial}} < 1\text{Å}$. This instability is likely due to a breakdown in the local linearity assumption, as contributions from lateral distances x and y begin to dominate the exponential term.

doi:10.6342/NTU202501885

2.3.2 Distortions



Now, we will move to simulate the characteristic distortions affecting STM images.

Lateral Scanning Distortions In this section, we model slow, accumulative effects in the lateral (XY) scanning process, including thermal drift, creep, and random positional offsets. These distortions cause the actual physical scanning positions to deviate from the intended target positions used for pixel assignment (see Figure 2.1).

As described in the Background chapter, **drift** introduces a linear positional deviation over time, while **creep** introduces a logarithmic one due to the viscoelastic response of the piezoelectric material. These effects, along with normally distributed positional noise, are incorporated in Algorithm 2.

Algorithm 2 Distorted Scanning Grid Generation

Require: $R \in \mathbb{R}$ ▷ Image resolution

Require: $c_{\text{drift},f}, c_{\text{drift},s} \in \mathbb{R}$ ▷ Drift parameters

Require: $c_{\text{creep},f}, c_{\text{creep},s}, \nu_f, \nu_s \in \mathbb{R}$ ▷ Creep parameters

Require: $\sigma_f^2, \sigma_s^2 \in \mathbb{R}$ ▷ Random offset parameters

Require: $h \in \mathbb{R}$ ▷ Scanning step size

1: $\mathbf{drift}_f \leftarrow c_{\text{drift};f} \cdot \left(\frac{i}{R}\right)_{i=0}^{R}$ $\mathbf{drift}_s \leftarrow c_{\text{drift},s} \cdot \left(\frac{i}{R}\right)_{i=0}^{R}$ ▷ $\mathbb{R}^{1\times R}$, Formula 1.10

2: $\mathbf{random}_f \sim \mathcal{N}(0, \sigma_f^2)^{1\times R}$ $\mathbf{random}_s \sim \mathcal{N}(0, \sigma_s^2)^{1\times R}$ ▷ $\mathbb{R}^{1\times R}$

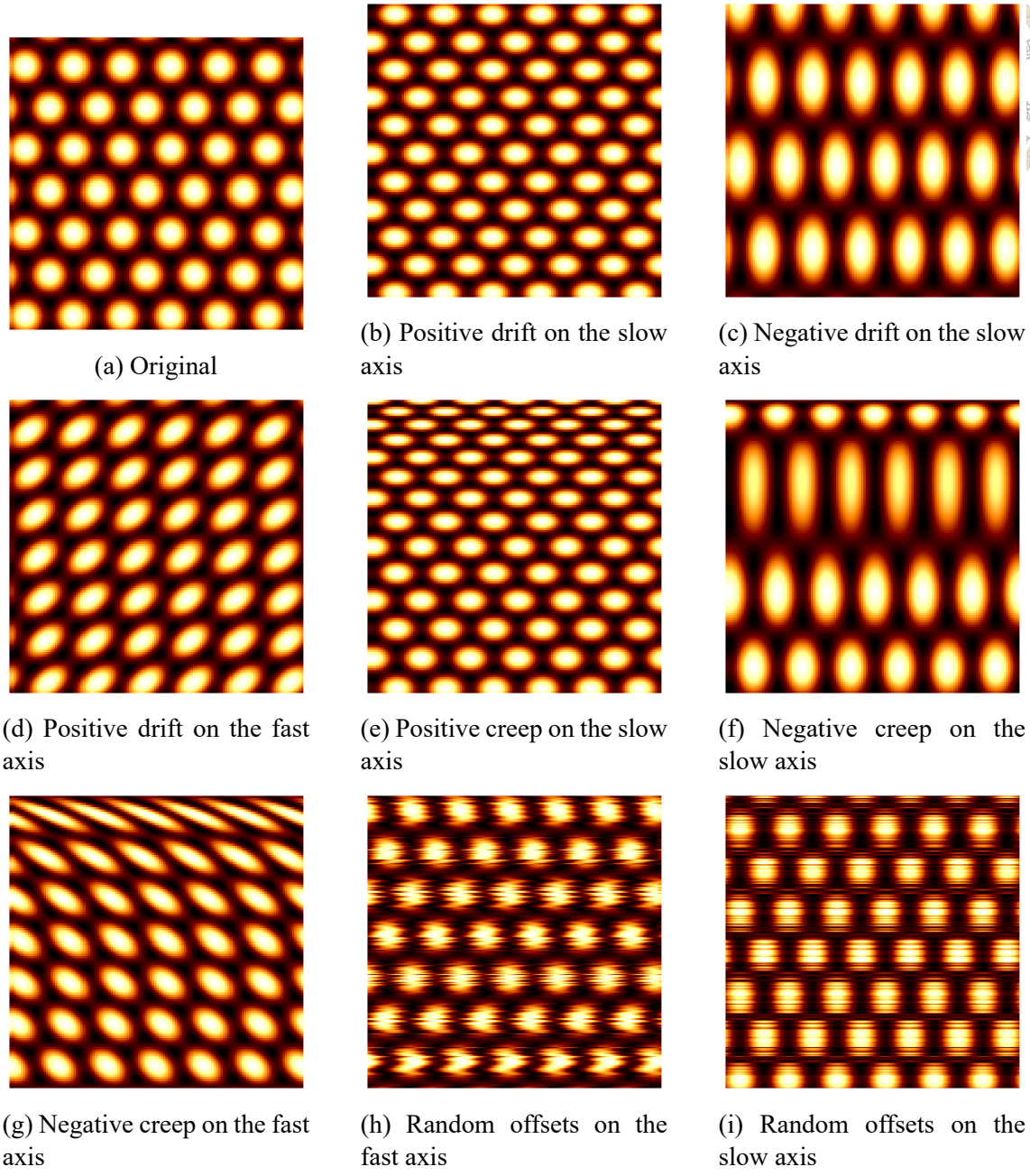
3: $\mathbf{creep}_f \leftarrow c_{\text{creep},f} \cdot \left(\log\left(\frac{1}{\nu_f} + \frac{i}{R}\left(1 - \frac{1}{\nu_f}\right)\right)\right)_{i=0}^{R}$ ▷ $\mathbb{R}^{1\times R}$, Formula 1.9

4: $\mathbf{creep}_s \leftarrow c_{\text{creep},s} \cdot \left(\log\left(\frac{1}{\nu_s} + \frac{i}{R}\left(1 - \frac{1}{\nu_s}\right)\right)\right)_{i=0}^{R}$ ▷ $\mathbb{R}^{1\times R}$, Formula 1.9

5: $\mathbf{grid}_f \leftarrow \left(h \cdot \left(\frac{i}{R}\right)_{i=0}^{R}\right)^T \mathbf{1}_R^T + \mathbf{1}_R\left(\mathbf{drift}_f + \mathbf{creep}_f + \mathbf{random}_f\right)$ ▷ $\mathbb{R}^{R\times R}$

6: $\mathbf{grid}_s \leftarrow \mathbf{1}_R\left(h \cdot \left(\frac{i}{R}\right)_{i=0}^{R} + \mathbf{drift}_s + \mathbf{creep}_s + \mathbf{random}_s\right)$ ▷ $\mathbb{R}^{R\times R}$

doi:10.6342/NTU202501885



(a) Original

(b) Positive drift on the slow axis

(c) Negative drift on the slow axis

(d) Positive drift on the fast axis

(e) Positive creep on the slow axis

(f) Negative creep on the slow axis

(g) Negative creep on the fast axis

(h) Random offsets on the fast axis

(i) Random offsets on the slow axis

Figure 2.1: Types of lateral scanning distortions.

Vertical Hysteresis Distortion In constant current mode STM, the pixel values represent the tip height. However, this height is measured indirectly by recording the voltage applied to the piezoelectric actuator.

The rapid movement of the STM tip in the z-axis is affected by **hysteresis**. Due to this effect, the relationship between applied voltage and resulting displacement is nonlin-

doi:10.6342/NTU202501885

ear and history-dependent. As a result, even when the tip physically returns to a previous z-position, the voltage required may differ. This mismatch leads to observable artifacts: contrast variation between scan lines (see Figure 2.2) and trailing effects after rapid movements, such as after encountering a vacancy (see Figure 2.3).

The Algorithm 3 simulates actuator movement under the **Bouc-Wen model** using the classical Runge-Kutta method.

The unit of displacement provided to this algorithm should be scaled in such a way that $1V$ change in voltage will approximately correspond to a unit diplacement of 1. However, this scaling factor depends on specific values of $A, \beta,$ and γ.

To address this, I run a calibration loop after setting the hysteresis parameters, where the actuator moves between voltages of $+0.5V$ and $-0.5V$ and the resulting displacement range is measured. This range is then used to scale the target displacement in the PID Control loop.

doi:10.6342/NTU202501885

Algorithm 3 Piezoelectric Actuator Movement with Bouc-Wen Model

Require: $\Delta x \in \mathbb{R}$ ▷ Desired displacement

▷ Units chosen such that a $1V$ change yields 1 displacement unit

Require: $T \in \mathbb{R}$ ▷ Duration of the movement

Require: $N \in \mathbb{N}$ ▷ Number of time steps

Require: $A, \beta, \gamma \in [0, 1]$ ▷ Actuator parameters

Require: $v_0, z_0 \in \mathbb{R}$ ▷ Initial voltage and hysteresis state

$t_0 \leftarrow 0$

function VOLTAGE(t) ▷ Voltage change linear with time

return $\begin{cases} v_0, & \text{if } t < 0 \\ v_0 + \frac{\Delta x}{T} \cdot t, & \text{otherwise} \end{cases}$

end function

function DZ_DT(t, z) ▷ $\dot{z}$ calculation

$\dot{v} \leftarrow \frac{\text{voltage}(t+h/2) - \text{voltage}(t-h/2)}{h}$

return $A \cdot \dot{v} - \beta \cdot |\dot{v}| \cdot z - \gamma \cdot \dot{v} \cdot |z|$ ▷ Formula 1.8

end function

for $i = 0$ to $N - 1$ **do** ▷ Formula 1.38

$k_1 \leftarrow h \cdot \texttt{dz\_dt}(t_i, z_i)$ $k_2 \leftarrow h \cdot \texttt{dz\_dt}(t_i + \frac{h}{2}, z_i + \frac{k_1}{2})$

$k_3 \leftarrow h \cdot \texttt{dz\_dt}(t_i + \frac{h}{2}, z_i + \frac{k_2}{2})$ $k_4 \leftarrow h \cdot \texttt{dz\_dt}(t_i + h, z_i + k_3)$

$z_{i+1} \leftarrow z_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$

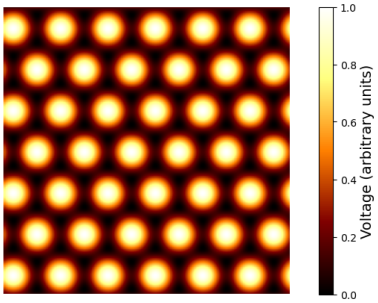
$t_{i+1} \leftarrow t_i + h$

$v_{i+1} \leftarrow \texttt{voltage}(t_{i+1})$

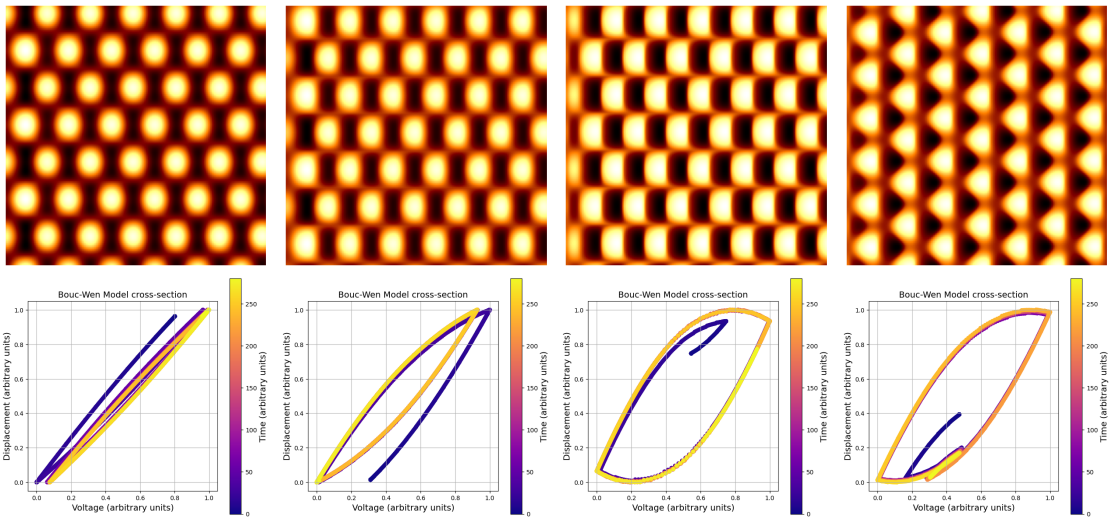
end for

$x \leftarrow v_N - z_N$ ▷ Resulting position

doi:10.6342/NTU202501885

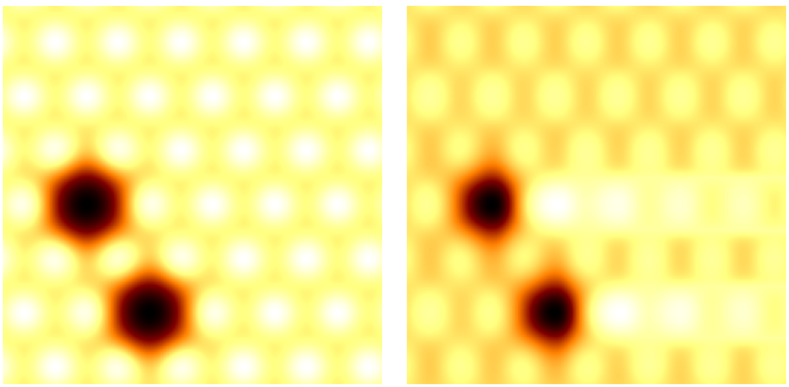


(a)



(b) (c) (d) (e)

Figure 2.2: Hysteresis effect on a non-defective lattice. Hysteresis parameters $\beta = 0.6, \gamma = 0.6$ are common. (a) No hysteresis: $A = 0$, (b) Moderate hysteresis: $A = 0.4$, (c) Strong hysteresis: $A = 0.7$, (d) Very strong hysteresis: $A = 0.9$. (e) Very strong hysteresis, sample rotated by 30°.



(a) (b)

Figure 2.3: Effect of hysteresis on a defective lattice. (a) No hysteresis. (b) With hysteresis, showing characteristic trailing artifacts.

doi:10.6342/NTU202501885

2.3.3 PID Control loop

Here, we simulate the behavior of a Proportional-Derivative (PD) controller.

The integral term K_I is omitted, as I have observed it to lead to the same characteristic distortions as the proportional term.

We begin by computing the ideal height values $\mathbf{z}_{\text{ideal}}$ for constant current mode using Newton's method (Algorithm 1). We also compute the corresponding tunneling current values in constant height mode $\mathbf{I}_{\text{chm}}$, their derivatives $\mathbf{I}'_{\text{chm}}$, and set the target current I_t based on the same median heuristic as before (Algorithm 1).

To set the proportional gain K_P, we scale the input parameter k_P by the median ratio of $\mathbf{I}_{\text{chm}}/\mathbf{I}'_{\text{chm}}$:

$$K_P = k_P \cdot \text{median}\left(\frac{\mathbf{I}_{\text{chm}}}{\mathbf{I}'_{\text{chm}}}\right)$$

This ensures that $k_P = 1$ should be close to the ideal value as it corresponds to a gain roughly matching the behavior of Newton's method, which is known to converge well (see Section 2.3.1).

The PID loop is then performed. At each step, the error between the current and target tunneling currents is used to compute a correction in height, which is then modified by the actuator model to simulate hysteresis (Algorithm 3, scaling factors are omitted to keep the formulation concise). To simulate a tilt between the scanning plane and the sample, the sample atoms are slightly rotated using a matrix $\mathbf{R}_{\text{rot}}$. Additionally, measurement noise is simulated by the addition of Gaussian noise to the tunneling current.

doi:10.6342/NTU202501885

Algorithm 4 PID Control

Require: $k_P \in \mathbb{R}$ ▷ Proportional term

Require: $K_D \in \mathbb{R}$ ▷ Differential term

Require: $z_{\text{initial}} \in \mathbb{R}$ ▷ Initial z position

Require: $\mathbf{R}_{\text{rot}} \in \mathbb{R}^{3\times3}$ ▷ Rotation matrix to simulate sample tilt

$\mathbf{grid}_f, \mathbf{grid}_s \leftarrow$ `get_distorted_grids()` ▷ Algorithm 2

$\mathbf{z}_{\text{ideal}}, \mathbf{I}_{\text{chm}}, \mathbf{I}'_{\text{chm}}, I_t \leftarrow$ `newton_method`$(z_{\text{initial}}, \mathbf{grid}_f, \mathbf{grid}_s)$ ▷ Algorithm 1

$K_P \leftarrow k_P \cdot \text{median}\left(\dfrac{\mathbf{I}_{\text{chm}}}{\mathbf{I}'_{\text{chm}}}\right)$ ▷ $k_p = 1 \rightarrow$ PID control analogous to Newton's method

$\mathbf{de} \leftarrow \mathbf{0}_R$

$(\mathbf{I}_{\text{temp}})_{0,:} \leftarrow (\mathbf{I}_{\text{chm}})_{0,:}$

$(\mathbf{z}_{\text{final}})_{0,:} \leftarrow (\mathbf{z}_{\text{ideal}})_{0,:}$

$(\mathbf{V})_{0,:} \leftarrow$ `actuator_initialize`$((\mathbf{z}_{\text{final}})_{0,:})$

for i = 0, 1, …, R-1 **do**

  $\mathbf{e} \leftarrow \log(I_t) - \log\left((\mathbf{I}_{\text{temp}})_{i,:}\right)$

  $\Delta\mathbf{z} \leftarrow K_P \cdot \mathbf{e} + K_D \cdot \mathbf{de}$

  $(\mathbf{V})_{i+1,:}, (\mathbf{z}_{\text{final}})_{i+1,:} \leftarrow$ `actuator_move`$(\Delta\mathbf{z})$ ▷ Algorithm 3

  $(\mathbf{I}_{\text{temp}})_{i+1,:} \leftarrow$ `calculate_current` $\left((\mathbf{z}_{\text{final}})_{i+1,:}, (\mathbf{grid}_f)_{i+1,:}, (\mathbf{grid}_s)_{i+1,:}, \mathbf{R}_{\text{rot}}\right)$ ▷ Formula 2.1

  $(\mathbf{I}_{\text{temp}})_{i+1,:} \leftarrow (\mathbf{I}_{\text{temp}})_{i+1,:} + \mathbf{noise}$, where $\mathbf{noise} \sim \mathcal{N}(0, \sigma^2)^{1\times R}$

  $\mathbf{de} \leftarrow \log\left((\mathbf{I}_{\text{temp}})_{i+1,:}\right) - \log\left((\mathbf{I}_{\text{temp}})_{i,:}\right)$

end for

2.4 Summary and parameter selection

While the simulation has been optimized to minimize the number of user-defined parameters, it still involves over 30 adjustable values.

Once the electronic structures of the sample and tip are set, a graphical user interface (GUI) can be used to explore how different parameter configurations affect the resulting

doi:10.6342/NTU202501885

simulated image (see Figure 2.4).

The user can assign fixed values to the parameters or define them as uniformly distributed within specified bounds.

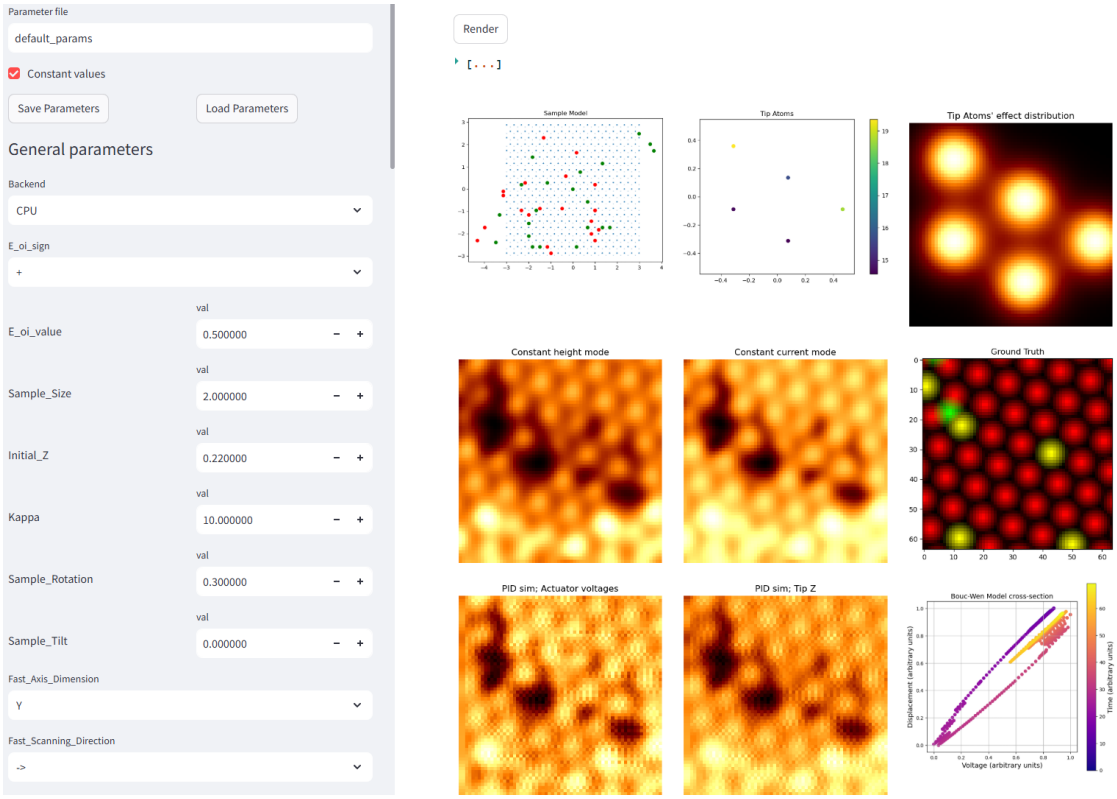


Figure 2.4: Graphical User Interface for parameter selection.

The selected parameters can then be saved and used to generate a training dataset for a machine learning task of interest.

doi:10.6342/NTU202501885

Chapter 3 Application

The primary intended use of the developed simulation framework is to generate training data for machine learning models. However, a critical question is how well a model trained exclusively on synthetic dataset will be able to generalize to real experimental measurements.

In this section, I address this question by training defect localization models on two different datasets of simulated tungsten diselenide (WSe_2) images:

- The first dataset consists of ideal constant-current mode images computed using Newton's method (Algorithm 1), with only lateral distortions (Algorithm 2) and Gaussian noise applied.
- The second dataset includes images generated using the full PID control loop simulation (Algorithm 4) developed in this work, incorporating additional realistic scanning distortions.

By comparing model predictions on real STM images, I assess whether incorporating PID control effects into the simulation framework improves the generalization ability of the model.

doi:10.6342/NTU202501885

3.1 Tight binding models

The main inputs to the simulation are the tight-binding models of tungsten to simulate the tip and tungsten diselenide to simulate the sample.

3.1.1 Sample model

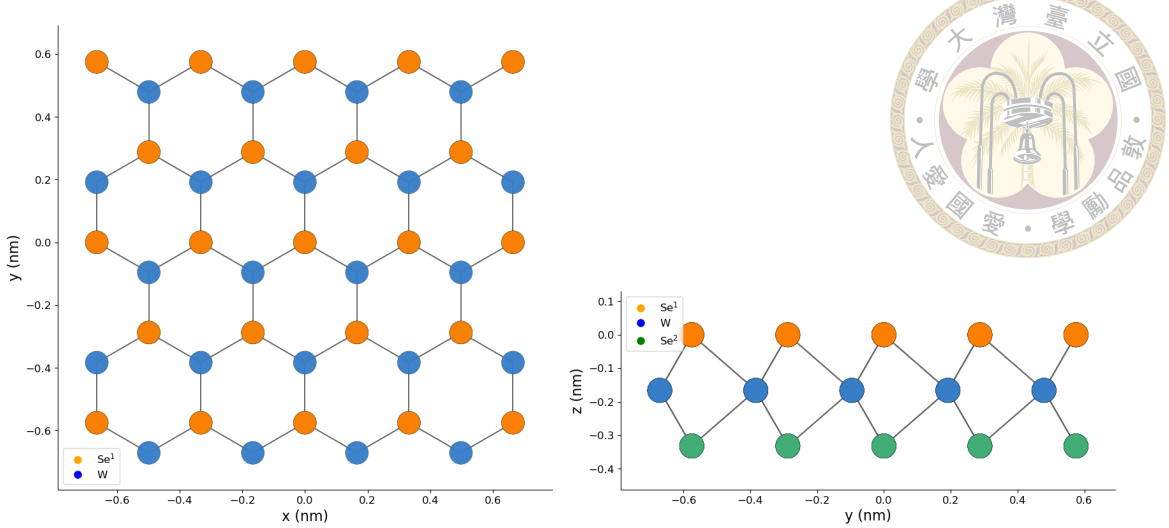
Tungsten Diselenide lattice Tungsten Diselenide (WSe_2) has a hexagonal lattice with a lattice constant $a = 0.332\,\text{nm}$. The primitive cell of monolayer WSe_2 contains three atoms positioned at:

$$\begin{aligned} \mathbf{r}_{\text{W}} &= a \cdot (0, \ \frac{1}{\sqrt{3}}, \ -\frac{1}{2}) \\ \mathbf{r}_{\text{Se}^1} &= a \cdot (0, \ 0, \ 0) \\ \mathbf{r}_{\text{Se}^2} &= a \cdot (0, \ 0, \ -1) \end{aligned} \tag{3.1}$$

The lattice vectors of the monolayer unit cell are given by:

$$\begin{aligned} \mathbf{a}_1 &= a \cdot (1, \ 0, \ 0) \\ \mathbf{a}_2 &= a \cdot (\frac{1}{2}, \ \frac{1}{2\sqrt{3}}, \ 0) \end{aligned} \tag{3.2}$$

doi:10.6342/NTU202501885



(a) (b)

Figure 3.1: WSe_2 lattice. (a) Top-down view. (b) Side view.

Hopping parameters To construct the tight-binding model, I used results from [34], which showed that the dominant contributions to the hopping integrals arise from interactions between orbitals of matching parity—specifically, between the odd d orbitals of tungsten and the odd p orbitals of selenide, and between the even d orbitals of tungsten and the even p orbitals of selenide.

They calculated the exact values of these hopping parameters using density functional theory simulation. I used these values to construct the tight-binding hamiltonian.

From this Hamiltonian we can construct the time-independent Schrödinger equation 1.15 and solve it. The local density of states (LDOS) at site i and energy E is then computed using the formulas 1.16 and 1.17.

Validation of the model To validate the constructed tight-binding model, I compare the simulated local density of states (LDOS) with those reported in the literature.

We see that the calculated densities of states (Figure 3.2) closely match previously

doi:10.6342/NTU202501885

published results [35] [36] [37].

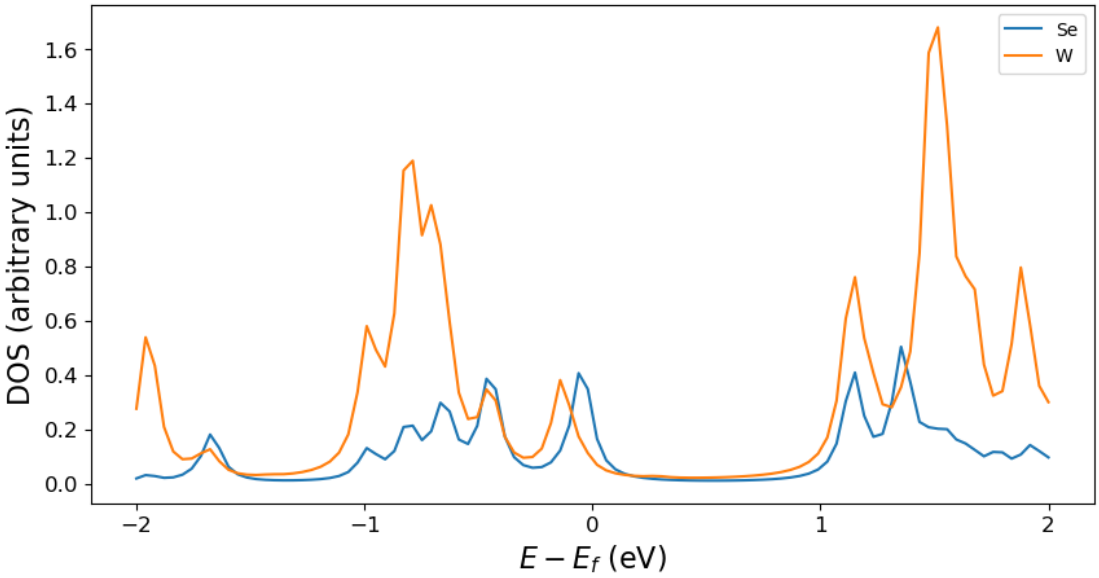


Figure 3.2: Simulated density of states for tungsten and selenide atoms in monolayer WSe_2.

3.1.2 Tip model

Tungsten lattice Tungsten has a body-centered cubic lattice with a lattice constant $a = 0.3615\,\text{nm}$. Its primitive cell contains a single atom and the lattice vectors are:

$$\begin{aligned} \mathbf{a}_1 &= a \cdot (-0.5,\ \ 0.5,\ \ 0.5) \\ \mathbf{a}_2 &= a \cdot (0.5,\ \ -0.5,\ \ 0.5) \\ \mathbf{a}_3 &= a \cdot (0.5,\ \ 0.5,\ \ -0.5) \end{aligned} \tag{3.3}$$

doi:10.6342/NTU202501885

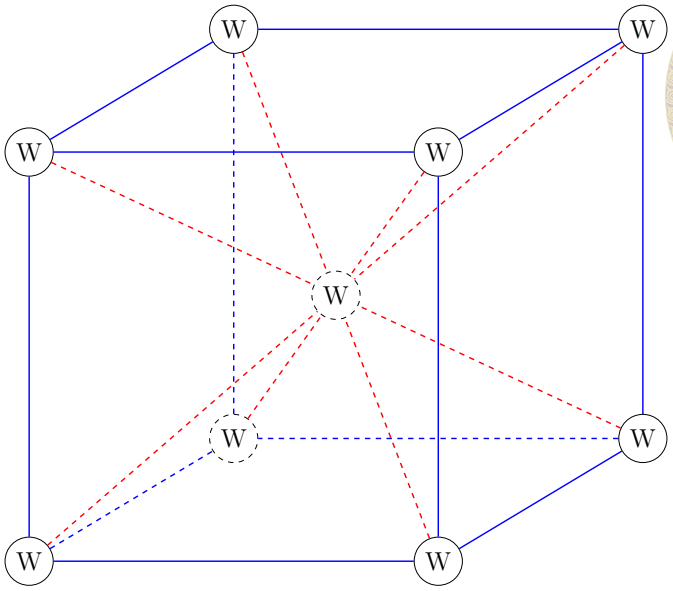


Figure 3.3: Tungsten lattice.

Slater-Koster method Single-element systems such as bulk tungsten can be modelled using the Slater-Koster method. This approach uses sets of parameters to represent the strength and angular dependence of overlaps between orbitals of neighbouring atoms. [25]

For tungsten, these sets of parameters can be found in [38] and hopping integrals can then be calculated using standard formulas implemented in a python package PySKTB [39].

After constructing the tight-binding Hamiltonian, the Fermi level was adjusted to reach a consistent behaviour.

Validation of the tungsten tight binding model To validate this model, I computed the DOS for an atom inside a $3 \times 3 \times 3\,\mathrm{nm}^3$ tungsten bulk. The result is then compared with reference distributions reported in the literature.

As shown in Figure 3.4, the simulated DOS vaguely resembles the expected features reported in [40] [41] [38].

doi:10.6342/NTU202501885

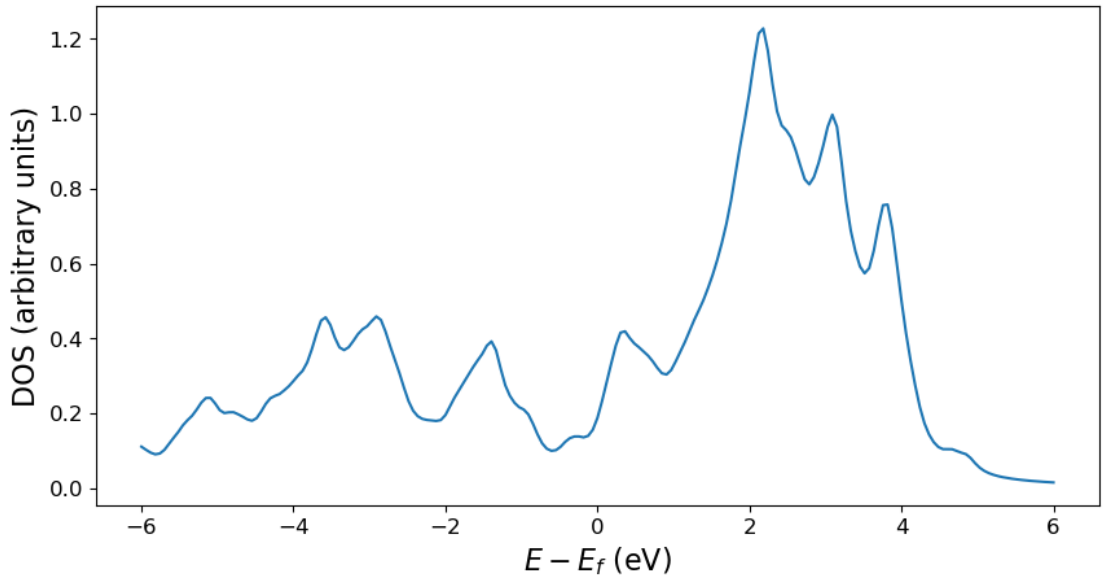


Figure 3.4: Simulated density of states for a tungsten atom in bulk.

3.1.3 Single-orbital approximations

Accurate multi-orbital tight-binding models, as described earlier, are computationally intensive, with complexity increasing quadratically with the number of orbitals. While these models capture the *energy-dependent* distribution of local densities of states (LDOS) more precisely, I have observed that simplified single-orbital models still reproduce the *spatial* distribution of LDOS with reasonable fidelity, producing STM-like images that visually resemble those from full multi-orbital simulations (see Figure 3.5).

To enable the generation of larger datasets within practical time frames, I therefore adopted single-orbital approximations for both tungsten and tungsten diselenide (WSe_2).

For tungsten, I used hopping energies of 2.5eV for first-nearest neighbours and 2.0eV for second-nearest neighbours. For tungsten diselenide, a first-nearest neighbour hopping energy of 1.19eV was applied.

doi:10.6342/NTU202501885

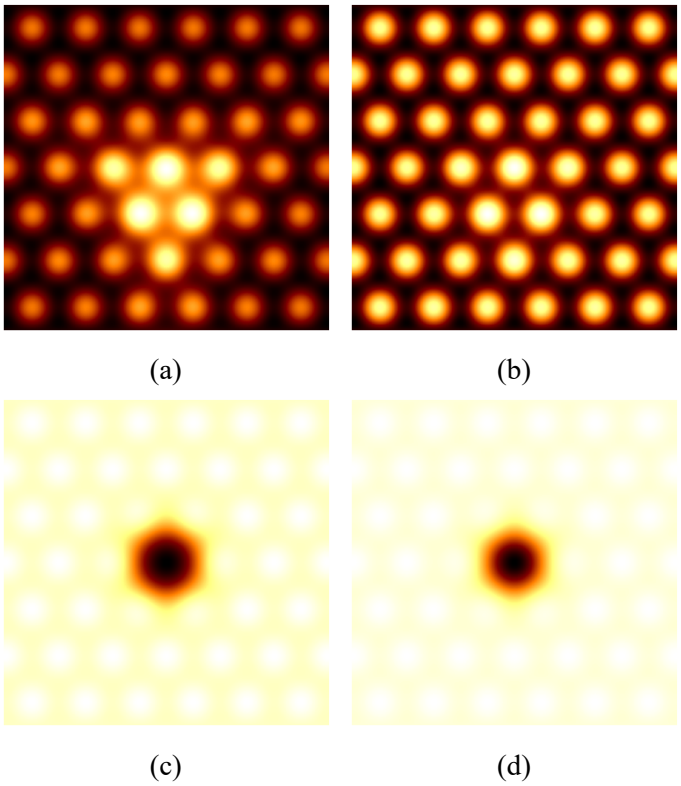


Figure 3.5: Comparison of simulations using different tungsten diselenide models. Initial z position is 0.5nm and lateral dimensions are 2×2nm. (a) Single-orbital model with a tungsten vacancy. (b) Multi-orbital model with a tungsten vacancy. (c) Single-orbital model with a (top) selenide vacancy. (d) Multi-orbital model with a (top) selenide vacancy.

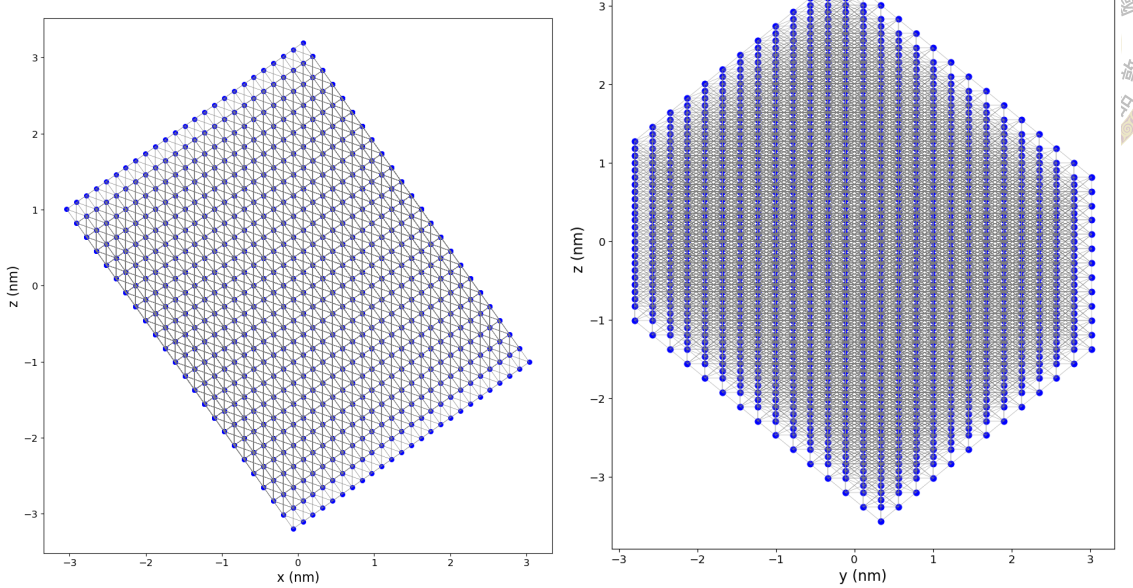
3.1.4 Tip geometry

In this section, I develop models for the tip's geometry. Although these models are inherently simplified and not realistic, they generate diverse LDOS distributions that serve as noise, when calculating the tunneling current.

(111) Tip geometry simulation The most widely used tungsten tip for STM has an apex oriented along the (111) direction.

To approximate this geometry, I modeled a $4 \times 4 \times 4\,\mathrm{nm}^3$ rotated cubic structure and treated its corner as the tip apex (see Figure 3.6).

doi:10.6342/NTU202501885



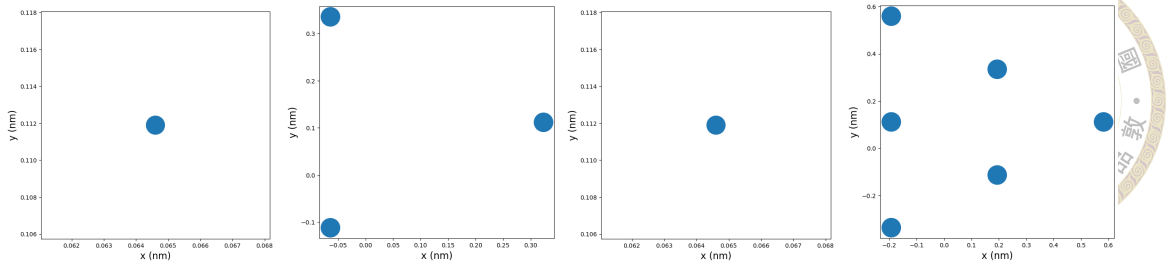
(a) (b)

Figure 3.6: Model of the (111) tip. (a) $x \times z$ side view. (b) $y \times z$ side view.

To incorporate realistic distortions, I introduced defect variations by progressively dulling the apex—removing atoms layer by layer up to four layers deep (see Figure 3.7). For each layer, I considered all possible combinations of atomic vacancies within that layer. This resulted in a total of:

- 1 variation for the pristine (undeformed) tip,
- 7 variations for the second layer (3 variations with single-atom apices),
- 1 variation for the third layer,
- 63 variations for the fourth layer (6 variations with single-atom apices).

doi:10.6342/NTU202501885



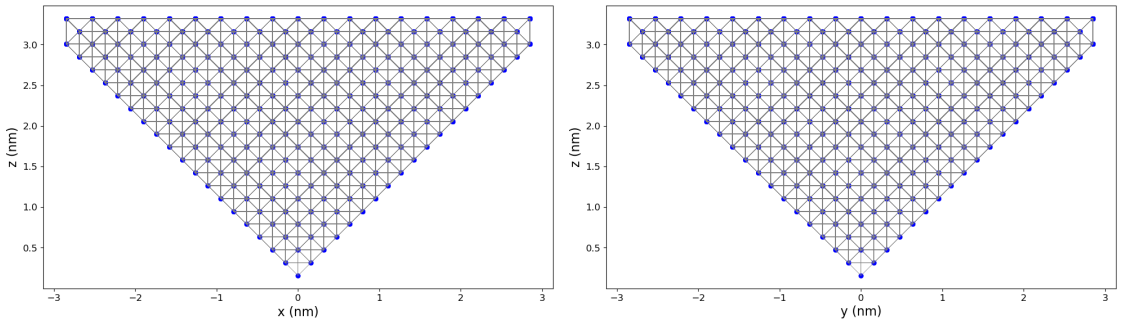
(a) (b) (c) (d)

Figure 3.7: Atom positions in the 4 layers modelling the tip's apex.

Since the contributions to the tunneling current decrease by an order of magnitude with an increase in distance of ~ 0.1nm, I do not consider other layers along the (111) direction, except for the one closest to the sample.

(001) Tip geometry simulation In order to have more variation, I decided to also simulate a (001) tip, although it is less commonly used in practice.

The geometry is simulated as a $6 \times 6 \times 3\,\mathrm{nm}^3$ cone (see Figure 3.8).

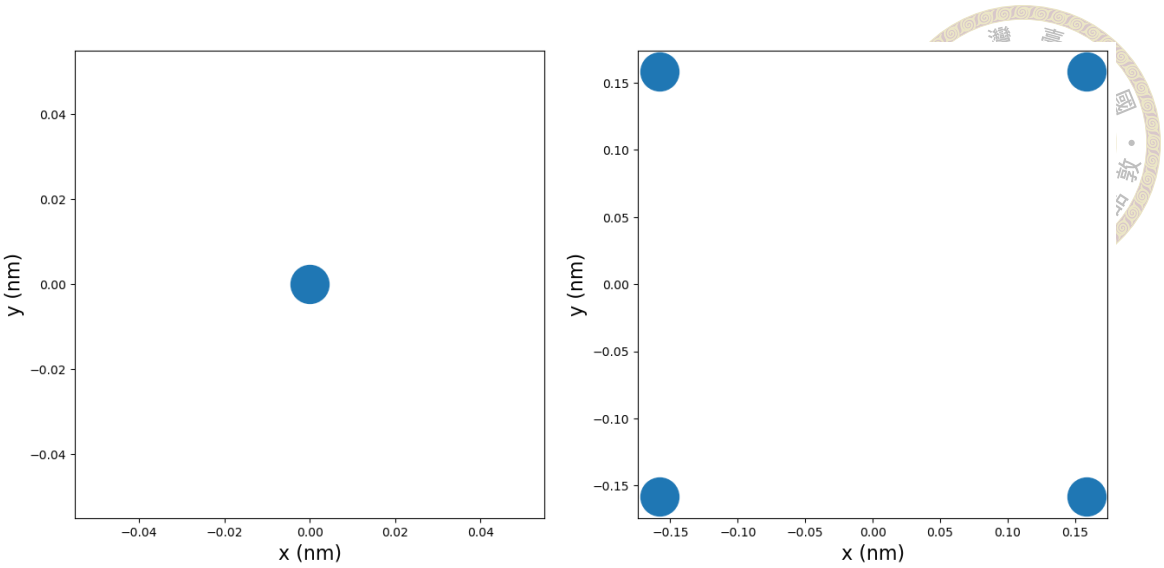


(a) (b)

Figure 3.8: Model of the (001) tip. (a) Side view through the y axis. (b) Side view through the x axis.

Only two layers along the (001) direction have been used for its variations with 1 variation of the undeformed tip and 15 variations of the dull tip (4 single-atom apex variations) (Figure 3.9).

doi:10.6342/NTU202501885



(a) (b)

Figure 3.9: Atom positions in the 2 layers modelling the tip apex.

doi:10.6342/NTU202501885

3.2 Machine Learning Task

3.2.1 Dataset Generation

Overview Two synthetic datasets were generated to train models for the task of defect localization. Each dataset contains 90,000 simulated images for training and 10,000 simulated images for validation. No synthetic test set was created; instead, real STM images are used exclusively for evaluating model generalization.

For clarity, the datasets are referred to as the **main dataset** and the **control dataset**. The main dataset incorporates the full simulation pipeline, including PID control loop distortions, while the control dataset consists of ideal constant-current images corrupted only by Gaussian noise and lateral distortions.

The datasets took approximately 60 hours to generate using an Intel Core i5-13500 CPU.

Tip and Defect Configuration Only single-atom tips were used for both datasets. Multi-atom (defective) tips can produce distorted images where the apparent defect location does not correspond to its true position (see Figure 3.11). This effect complicates model evaluation on real measurements, where the ground truth is unknown.

For the same reason, defects are restricted to selenide (Se) atoms. Defects on tungsten atoms produce more complex features that might be harder to interpret (see Figure 3.5).

Image Parameters Images were generated at a resolution of 64×64 pixels. Simulation parameters were sampled from uniform distributions as follows:

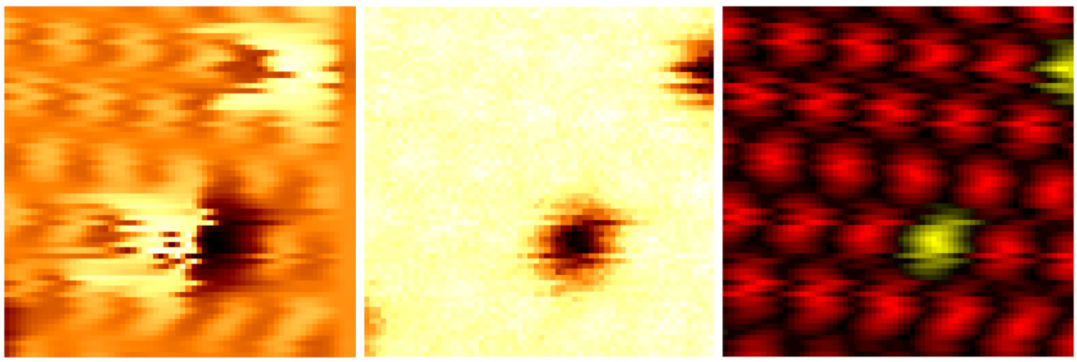
doi:10.6342/NTU202501885

- Bias voltage: $\sim \pm\mathcal{U}(0.2\,\mathrm{V}, 0.5\,\mathrm{V})$
- Sample size: $\sim \mathcal{U}(1\,\mathrm{nm}, 4\,\mathrm{nm})$
- Initial tip height: $\sim \mathcal{U}(0.3\,\mathrm{nm}, 0.6\,\mathrm{nm})$
- Decay constant: $\kappa \sim \mathcal{U}(9\,\mathrm{nm}^{-1}, 11\,\mathrm{nm}^{-1})$

Other parameters were chosen in such a way that defects remain visually identifiable, even at extreme sampled values.

Ground Truth Format The ground truth labels are 64×64 images, where red denotes a non-defective Se atom, and yellow indicates a defective Se atom (see Figure 3.10(c)).

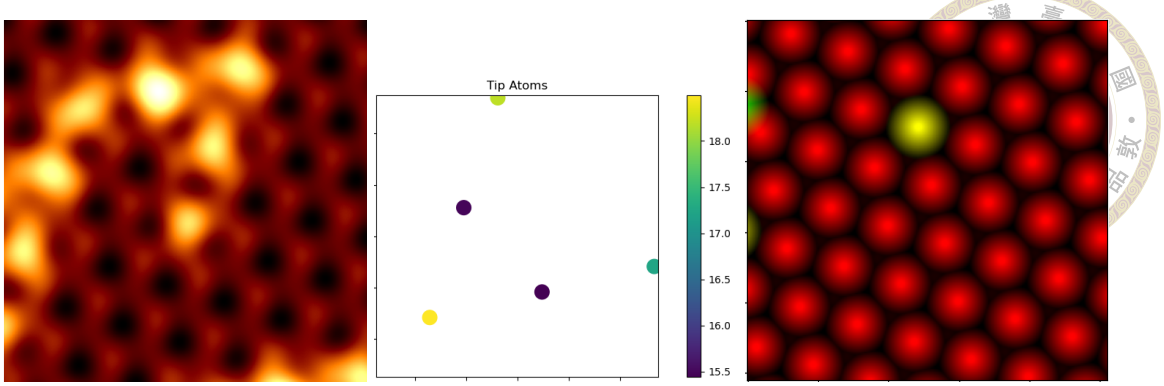
Example Images Figure 3.10 shows sample synthetic images from both datasets alongside the corresponding ground truth. Figure 3.11 illustrates the type of distortions caused by defective tips, which were excluded from defect localization tasks.



(a) (b) (c)

Figure 3.10: Example synthetic images and ground truth for defect localization. (a) Main dataset sample. (b) Control dataset sample. (c) Ground truth mask.

doi:10.6342/NTU202501885



(a) (b) (c)

Figure 3.11: Illustration of defective tip distortion. Such tips were excluded from the defect localization dataset. (a) Synthetic image with tip distortion. (b) Defective tip apex structure. (c) Defect location highlighted.

3.2.2 Model Architecture

A modified **U-Net** architecture (see Figure 3.13) is utilized for the task. This architecture employs a **ResNet** backbone pre-trained to predict the number of defects in each image (see Figure 3.12). Its layers are then use in the encoder path of the U-Net.

doi:10.6342/NTU202501885

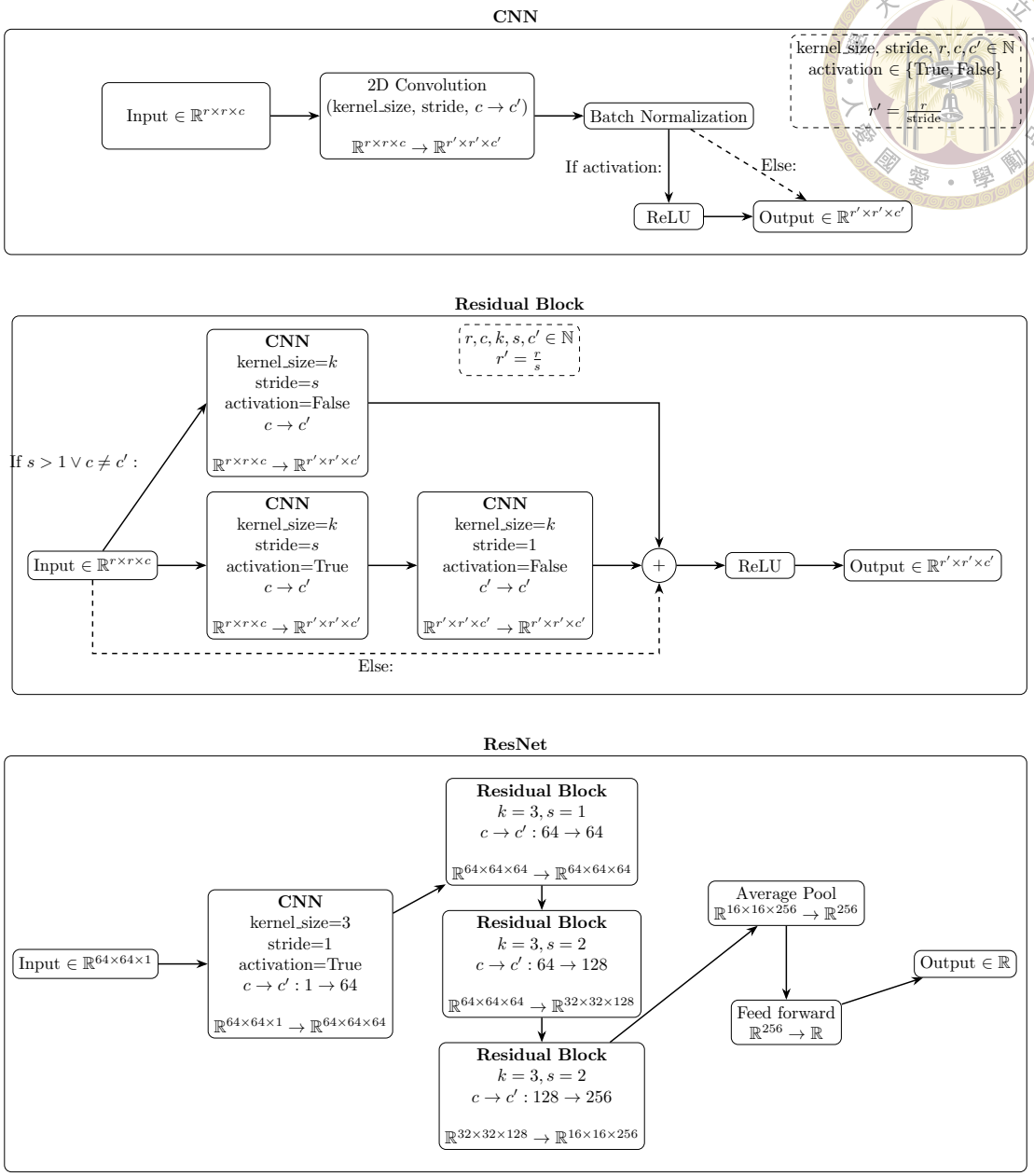


Figure 3.12: Modified ResNet structure and its components.

doi:10.6342/NTU202501885

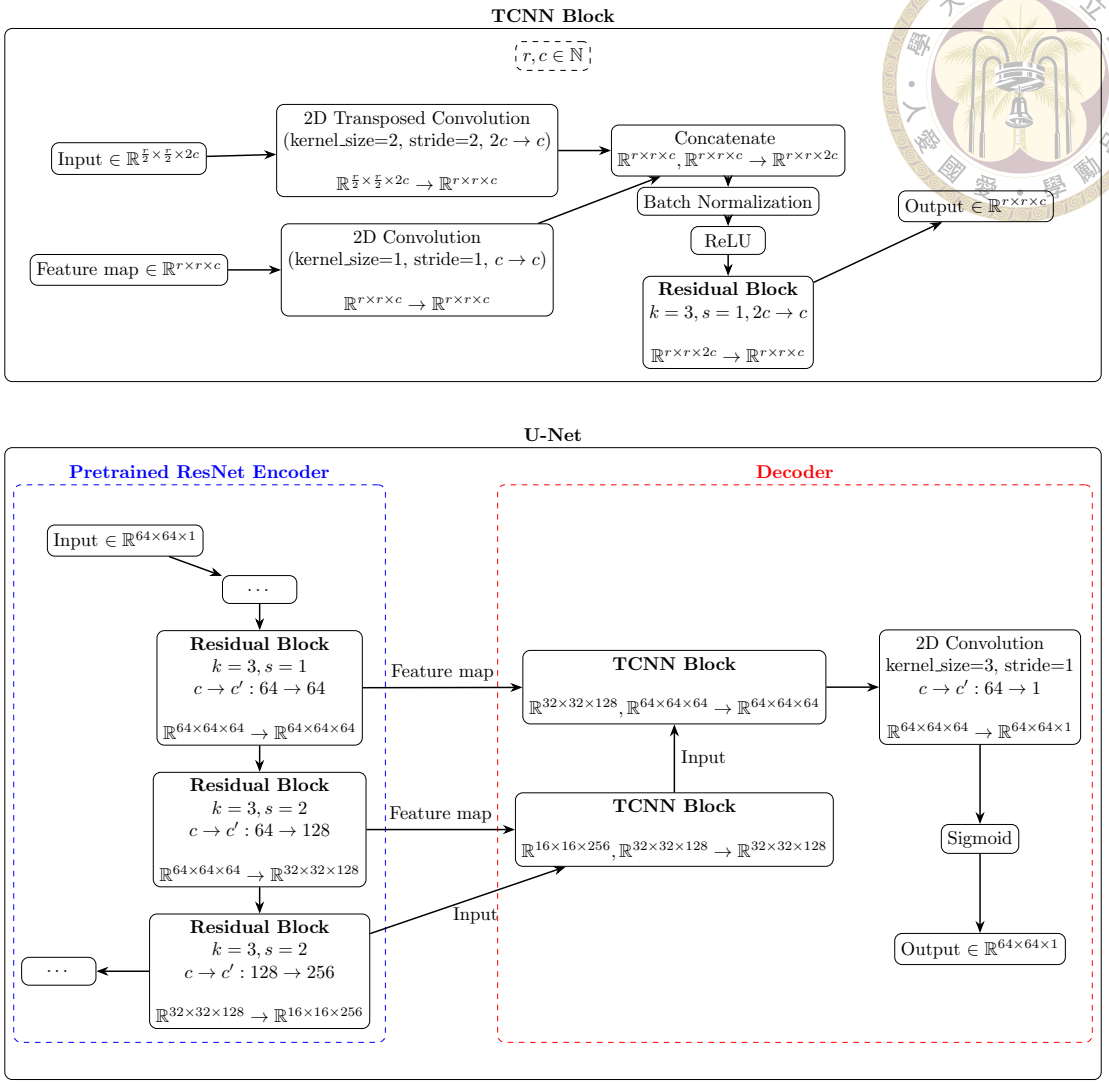


Figure 3.13: Modified U-Net structure and its components.

3.2.3 Training

Environment The models were built and trained using the PyTorch framework (version 2.6.0 + CUDA 12.6) within an Anaconda environment on a Windows 11 machine equipped with an RTX 3060 GPU and an Intel Core i5-13500 CPU. More details about the environment and installation procedure can be found in the project GitHub repository.

doi:10.6342/NTU202501885

Training Procedure Both ResNet backbones and U-Net models were trained by minimizing the smooth L1 loss (see Formula 1.19). Learning rates were selected using the learning rate range test: $7.94 \cdot 10^{-3}$ for ResNets and $1.25 \cdot 10^{-3}$ for U-Nets. These rates were then decayed exponentially during training. Weight decay equal to 10^{-5} was used for the Main ResNet and 10^{-4} for the Control ResNet.ResNets were trained for 50 epochs and U-Nets for 40 epochs (see Figure 3.14), both with a batch size of 16. ResNet training took approximately 1.5 hours, while U-Net training took around 3 hours using an RTX 3060 GPU.

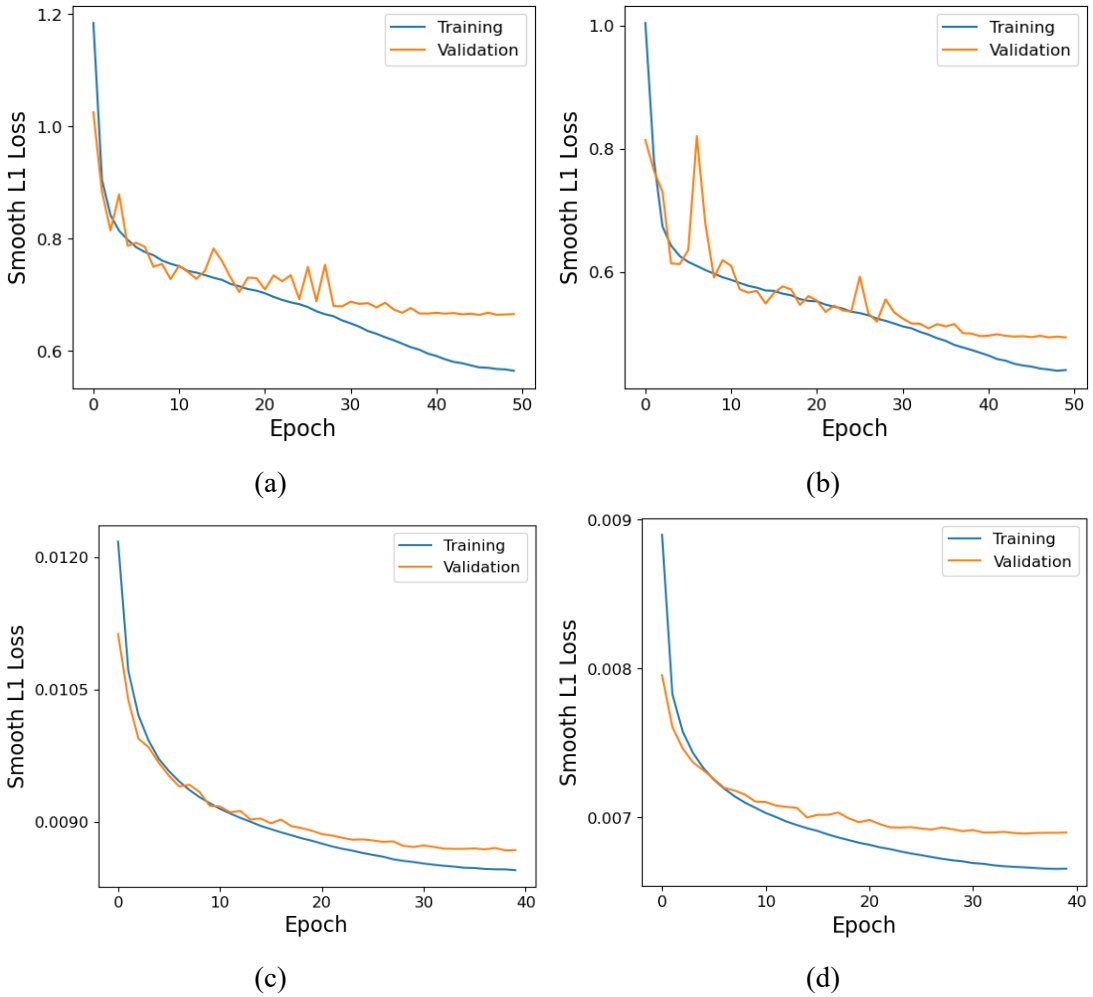


Figure 3.14: Loss over epochs. (a) Main ResNet, (b) Control ResNet, (c) Main U-Net, (d) Control U-Net.

ResNet Training Results

doi:10.6342/NTU202501885

Model	Dataset	MSE	Smooth L1 Loss
Main ResNet	Validation	2.194	0.664
	Training	1.729	0.552
Control ResNet	Validation	1.322	0.473
	Training	1.185	0.422

Table 3.1: ResNet backbone training results. For reference, the defect count variance is approximately 14.

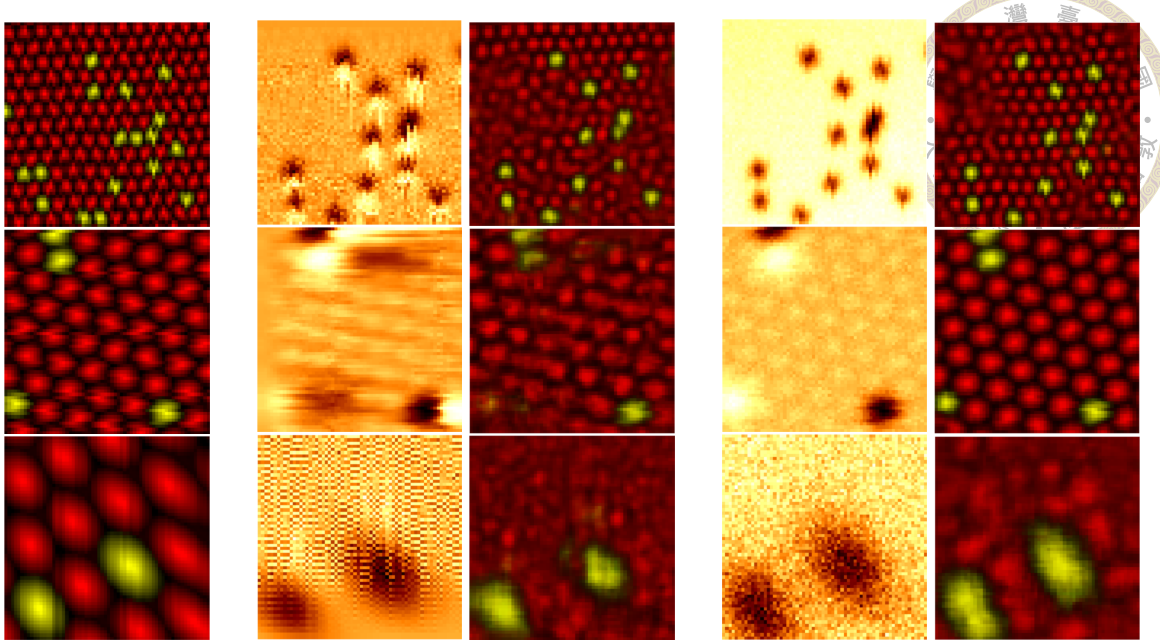
U-Net Training Results

Model	Dataset	MSE	Smooth L1 Loss
Main U-Net	Validation	0.0173	0.00864
	Training	0.0165	0.00824
Control U-Net	Validation	0.0137	0.00686
	Training	0.0131	0.00655

Table 3.2: U-Net training results. Dataset pixel-wise target variance: 0.07096.

Examples of predictions from both U-Nets on validation set images are shown in Figure 3.15.

doi:10.6342/NTU202501885



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

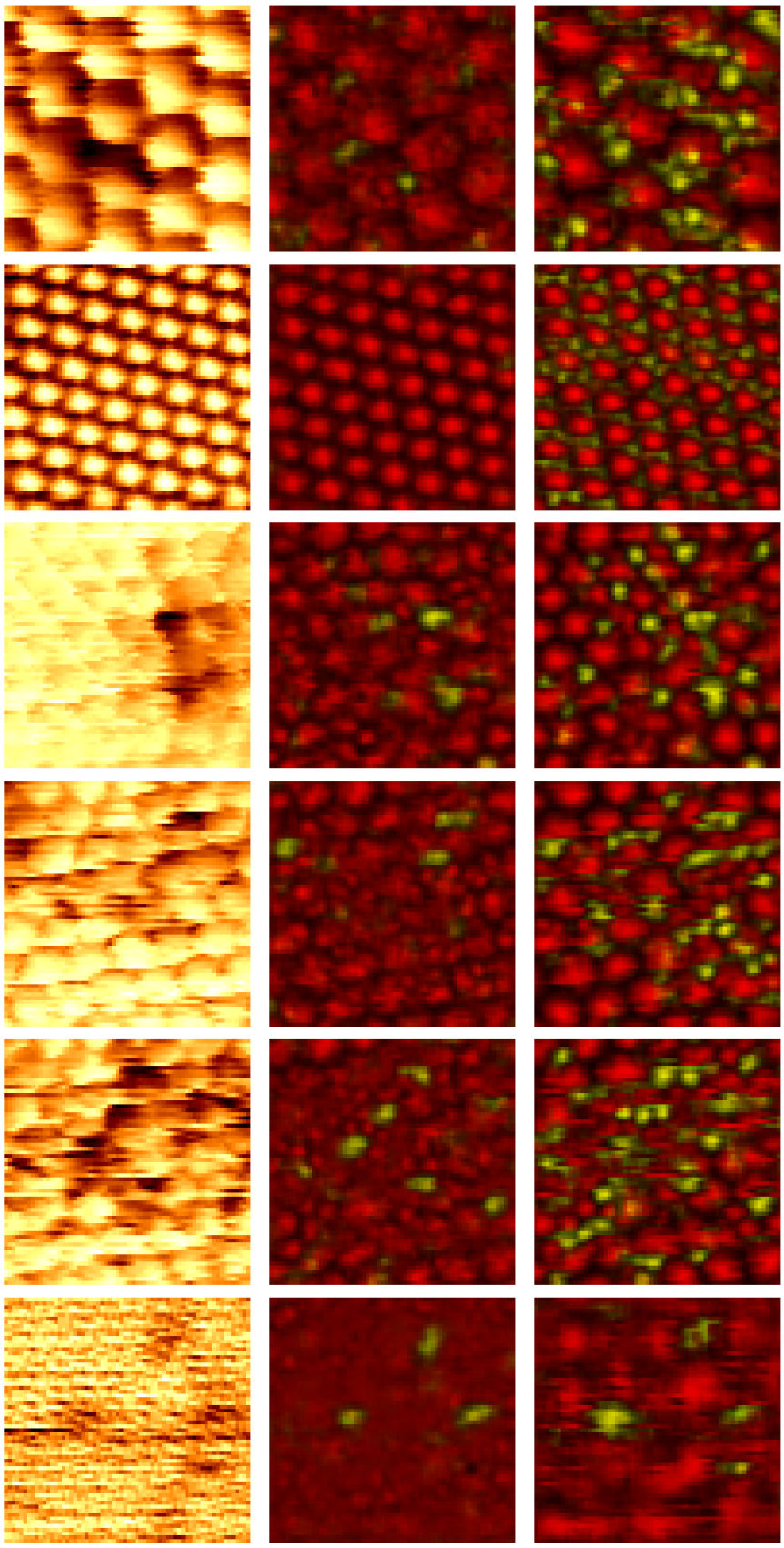
Figure 3.15: Validation of UNet defect localization model on synthetic data. (a) Ground Truth. (b) Main U-Net input. (c) Main U-Net output. (d) Control U-Net input. (e) Control U-Net output.

3.2.4 Evaluation

Because real STM data lacks ground truth labels, model evaluation is performed through qualitative visual assessment on various real WSe_2 samples.

As we see in Figure 3.16, incorporating the full PID simulation significantly improved the U-Net model's ability to generalize to real experimental images.

doi:10.6342/NTU202501885



(a) (b) (c)

Figure 3.16: Testing of U-Net defect localization model on real samples. No ground truth available. (a) Input image. (b) Prediction by U-Net trained using the main dataset including the full PID simulation. (c) Prediction by U-Net trained using the control dataset without PID simulation.

doi:10.6342/NTU202501885

doi:10.6342/NTU202501885

Chapter 4 Conclusion

4.1 Summary of Contributions

In this work, a simulation framework for synthetic STM imaging was developed with a focus on physical fidelity, enabling generation of more realistic training data for machine learning models. The framework integrates tight-binding electronic structure calculations and distortion models such as thermal drift, creep and most notably hysteresis with tip-induced effects.

By training a modified U-Net for selenide defect localization in tungsten diselenide, it was shown that including distortions like tip-induced defects and hysteresis in the simulation pipeline leads to models that generalize better to real experimental images.

These results highlight the importance of integrating detailed physical modeling into synthetic data generation pipelines for machine learning applications in scientific imaging. The developed framework lays the groundwork for more automated, scalable and reliable analysis workflows in scanning tunneling microscopy.

doi:10.6342/NTU202501885

4.2 Limitations and Future Work

Performance The simulation framework currently generates images in sequence using a CPU. Parallelizing image generation on a GPU could significantly accelerate dataset generation, enabling higher-fidelity simulations or larger datasets.

Fidelity The demonstration application omits dull tip apices, structural relaxation, tungsten defects, and approximates selenide dopants using a potential energy term at defective sites. For more complex machine learning tasks such as defect classification, a more accurate dopant representation is needed, this can be achieved either by refining the potential energy approximation or adjusting the tight-binding model's hopping integrals with respect to the dopant site.

Evaluation Furthermore, in order to enable quantitative evaluation of models on real STM images, ground truth data should be obtained through other techniques such as dI/dV analysis [42].

4.3 Code availability

The simulation and machine learning code developed in this thesis is publicly available at `https://github.com/luzny274/Synthetic-STM-Imaging.git` under the GNU General Public License v3.0. Instructions for setting up the environment, generating the data set, training the model and reproducing key results are provided in the repository's README file.

doi:10.6342/NTU202501885

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, p. 770–778.

[2] F. Joucken, J. L. Davenport, Z. Ge, E. A. Quezada-Lopez, T. Taniguchi, K. Watanabe, J. Velasco, J. Lagoute, and R. A. Kaindl, “Denoising scanning tunneling microscopy images of graphene with supervised machine learning,” Phys. Rev. Mater., vol. 6, no. 12, p. 123802, 2022.

[3] Z. Zhu, S. Yuan, Q. Yang, H. Jiang, F. Zheng, J. Lu, and Q. Sung, “Autonomous scanning tunneling microscopy imaging via deep learning,” J. Am. Chem. Soc., vol. 146, no. 42, p. 29199–29206, 2024.

[4] D. Smalley, S. D. Lough, L. Holtzman, K. Xu, M. Holbrook, M. R. Rosenberger, J. C. Hone, K. Barmak, and M. Ishigami, “Detecting atomic-scale surface defects in STM of TMDs with ensemble deep learning,” MRS Adv., vol. 9, no. 11, p. 890–896, 2024.

[5] M. Rashidi and R. A. Wolkow, “Autonomous scanning probe microscopy in situ tip conditioning through machine learning,” ACS Nano, vol. 12, no. 6, p. 5185–5189, 2018.

doi:10.6342/NTU202501885

[6] S. Wang, J. Zhu, R. Blackwell, and F. R. Fischer, "Automated tip conditioning for scanning tunneling spectroscopy," J. Phys. Chem. A, vol. 125, no. 6, p. 1384–1390, 2021.

[7] J. B. Goetz, Y. Zhang, and M. Lawler, "Detecting nematic order in STM/STS data with artificial intelligence," SciPost Phys., vol. 8, no. 6, p. 087, 2020.

[8] A. G. Okunev, M. Y. Mashukov, A. V. Nartova, and A. V. Matveev, "Nanoparticle recognition on scanning probe microscopy images using computer vision and deep learning," Nanomaterials, vol. 10, no. 7, p. 1285, 2020.

[9] S. C. Cheung, J. Y. Shin, Y. Lau, Z. Chen, J. Sun, Y. Zhang, M. A. Müller, I. M. Eremin, J. N. Wright, and A. N. Pasupathy, "Dictionary learning in Fourier-transform scanning tunneling spectroscopy," Nat. Commun., vol. 11, no. 1, p. 1081, 2020.

[10] L. Kurki, N. Oinonen, and A. S. Foster, "Automated structure discovery for scanning tunneling microscopy," ACS Nano, vol. 18, no. 17, p. 11130–11138, 2024.

[11] O. Krejčí, P. Hapala, M. Ondráček, and P. Jelínek, "Principles and simulations of high-resolution STM imaging with a flexible tip apex," Phys. Rev. B, vol. 95, no. 4, p. 045407, 2017.

[12] C. J. Chen, Introduction to scanning tunneling microscopy. Oxford, United Kingdom: Oxford University Press, 2021, ISBN 978-0-191-88990-5.

[13] J. Tersoff and D. R. Hamann, "Theory of the scanning tunneling microscope," Phys. Rev. B, vol. 31, no. 2, p. 805–813, 1985.

[14] J. Stirling, "Control theory for scanning probe microscopy revisited," Beilstein J. Nanotechnol., vol. 5, p. 337–345, 2014.

doi:10.6342/NTU202501885

[15] R. P. Paganelli, A. Romani, A. Golfarelli, M. Magi, E. Sangiorgi, and M. Tartagni, "Modeling and characterization of piezoelectric transducers by means of scattering parameters. Part I: Theory," Sens. Actuators A, Phys., vol. 160, no. 1–2, p. 9–18, 2010.

[16] R. C. Smith and Z. Ounaies, "A domain wall model for hysteresis in piezoelectric materials," J. Intell. Mater. Syst. Struct., vol. 11, no. 1, p. 62–79, 2000.

[17] Y. K. Lin and G. Q. Cai, "Random vibration of hysteretic systems," in Springer, 1990, p. 189–196.

[18] Y. Zhang, M. Sun, Y. Song, C. Zhang, and M. Zhou, "Hybrid adaptive controller design with hysteresis compensator for a piezo-actuated stage," Appl. Sci., vol. 13, no. 1, p. 402, 2022.

[19] Y. Liu, J. Shan, and N. Qi, "Creep modeling and identification for piezoelectric actuators based on fractional-order system," Mechatronics, vol. 23, no. 7, p. 840–847, 2013.

[20] B. Mokaberi and A. A. G. Requicha, "Towards automatic nanomanipulation: Drift compensation in scanning probe microscopes," in Proc. IEEE Int. Conf. Robot. Autom., 2004, p. 416–421.

[21] C. Rusu, S. Besoiu, and M. O. Tatar, "Design and closed-loop control of a piezoelectric actuator," in IOP Conf. Ser.: Mater. Sci. Eng., vol. 1018, 2021, p. 012002.

[22] M. P. Yothers, A. E. Browder, and L. A. Bumm, "Real-space post-processing correction of thermal drift and piezoelectric actuator nonlinearities in scanning tunneling microscope images," Rev. Sci. Instrum., vol. 88, no. 1, p. 013708, 2017.

doi:10.6342/NTU202501885

[23] N. W. Ashcroft and N. D. Mermin, Solid state physics. Fort Worth, TX, USA: Saunders College, 1988, ISBN 0-03-083993-1.

[24] A. Altland and B. Simons, Condensed matter field theory. Cambridge, United Kingdom: Cambridge University Press, 2006, ISBN 978-0-511-80423-6.

[25] J. C. Slater and G. F. Koster, "Simplified LCAO method for the periodic potential problem," Phys. Rev., vol. 94, no. 6, p. 1498–1524, 1954.

[26] R. Girshick, "Fast R-CNN," 2015, Preprint at arXiv:1504.08083.

[27] I. Goodfellow, Y. Bengio, and A. Courville, Deep learning. Cambridge, MA, USA: MIT Press, 2016, ISBN 978-0-262-03561-3.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, Preprint at arXiv:1412.6980.

[29] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in Proc. Med. Image Comput. Comput.-Assist. Interv. (MICCAI), vol. 9351. Springer, 2015, p. 234–241.

[30] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. 32nd Int. Conf. Mach. Learn. (ICML), vol. 37. JMLR.org, 2015, p. 448–456.

[31] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," 2017, Preprint at arXiv:1712.09913.

[32] J. D. Faires and R. L. Burden, Numerical methods. Brooks Cole, 1998, ISBN 978-0-538-73351-9.

doi:10.6342/NTU202501885

[33] D. Moldovan, M. Anđelković, and F. Peeters, "Pybinding v0.9.5: A Python package for tight-binding calculations," 2020, version v0.9.5. [Online]. Available: https://docs.pybinding.site/

[34] S. Fang, S. Carr, M. A. Cazalilla, and E. Kaxiras, "Electronic structure theory of strained two-dimensional materials with hexagonal symmetry," Phys. Rev. B, vol. 98, no. 7, p. 075106, 2018.

[35] The Materials Project, "Materials data on WSe2," 2020.

[36] R. S. Barbosa, C. A. D. N. Júnior, A. S. Santos, M. J. Piotrowski, C. R. C. Rêgo, D. Guedes-Sobrinho, D. L. Azevedo, and A. C. Dias, "Unveiling the role of electronic, vibrational, and optical features of the 1T' WSe_2 monolayer," ACS Omega, vol. 9, no. 44, p. 44689–44696, 2024.

[37] N. Yang, T. H. Hsu, H. Y. Chen, J. Zhao, H. Zhang, H. Wang, and J. Guo, "Van der Waals heterostructure engineering for ultralow-resistance contact in 2D semiconductor P-type transistors," J. Electron. Mater., vol. 53, no. 4, p. 2150–2161, 2024.

[38] D. A. Papaconstantopoulos, Handbook of the band structure of elemental solids. New York, NY, USA: Springer, 2014, ISBN 978-1-4419-8264-3.

[39] S. K. Radha, "santoshkumarradha/pysktb: Tight-binding electronic structure codes," 2020. [Online]. Available: https://pysktb.readthedocs.io/

[40] The Materials Project, "Materials data on W," 2020.

[41] T. Paschen, M. Förster, M. Krüger, C. Lemell, G. Wachter, F. Libisch, T. Madlener, J. Burgdörfer, and P. Hommelhoff, "High visibility in two-color above-threshold

doi:10.6342/NTU202501885

photoemission from tungsten nanotips in a coherent control scheme," J. Mod. Opt., vol. 64, no. 10–11, p. 1054–1060, 2017.

[42] H. Y. Chen, H. C. Hsu, J. Y. Liang, B. H. Wu, Y. F. Chen, C. C. Huang, M. Y. Li, I. P. Radu, and Y. P. Chiu, "Atomically resolved defect-engineering scattering potential in 2D semiconductors," ACS Nano, vol. 18, no. 27, p. 17622–17629, 2024.

doi:10.6342/NTU202501885