Molecular electronics: controlled manipulation, noise and graphene architecture

Tewari, S.

Appendix

A. Atomic manipulation

A.1 Force derivation

Here the force on an atom resulting from the potential energy (E) of Equation 2.1 is derived.

$$E = -\zeta \sum_{i}^{N} \sqrt{\sum_{j \neq i}^{N} e^{-2q(r_{ij}/r_0 - 1)}} + A \sum_{i}^{N} \sum_{j \neq i}^{N} e^{-p(r_{ij}/r_0 - 1)} \tag{A.1}$$

Now calculating the gradient of this energy will provide the forces acting on atom a,

$$\vec{F}_a = -\vec{\nabla}_a E \tag{A.2}$$

We will have to find

$$\vec{F}_a = -\vec{\nabla}_a \left[-\zeta \sum_{i}^{N} \sqrt{\sum_{j \neq i}^{N} e^{-2q(r_{ij}/r_0 - 1)}} + A \sum_{i}^{N} \sum_{j \neq i}^{N} e^{-p(r_{ij}/r_0 - 1)} \right] \tag{A.3}$$

For the component k of the force ($k = x, y, z$), this becomes

$$
\begin{aligned}
F_{a,k} &= -\frac{\partial E}{\partial k_a} \\
&= -\frac{\partial}{\partial k_a}\left[-\zeta \sum_{i}^{N} \sqrt{\sum_{j\neq i}^{N} e^{-2q(r_{ij}/r_0-1)}}\right] - \frac{\partial}{\partial k_a}\left[A \sum_{i}^{N}\sum_{j\neq i}^{N} e^{-p(r_{ij}/r_0-1)}\right] \\
&= \zeta \frac{\partial}{\partial k_a}\left[\sum_{i\neq a}^{N} \sqrt{\sum_{j\neq i}^{N} e^{-2q(r_{ij}/r_0-1)}} + \sqrt{\sum_{j\neq a}^{N} e^{-2q(r_{aj}/r_0-1)}}\right] \\
&- A \frac{\partial}{\partial k_a}\left[\sum_{i\neq a}^{N}\sum_{j\neq i}^{N} e^{-p(r_{ij}/r_0-1)} + \sum_{j\neq a}^{N} e^{-p(r_{aj}/r_0-1)}\right] \\
&= \zeta \sum_{i\neq a}^{N} \frac{\partial}{\partial r_{ia}} \sqrt{\sum_{j\neq i}^{N} e^{-2q(r_{ij}/r_0-1)}} \frac{\partial r_{ia}}{\partial k_a} + \zeta \frac{\partial}{\partial k_a} \sqrt{\sum_{j\neq a}^{N} e^{-2q(r_{aj}/r_0-1)}} \\
&- A \sum_{i\neq a}^{N} \frac{\partial}{\partial r_{ia}} \sum_{j\neq i}^{N} e^{-p(r_{ij}/r_0-1)} \frac{\partial r_{ia}}{\partial k_a} - A \frac{\partial}{\partial k_a} \sum_{j\neq a}^{N} e^{-p(r_{aj}/r_0-1)} \\
&= \zeta \sum_{i\neq a}^{N} \left[\frac{1}{2\sqrt{\sum_{j\neq i}^{N} e^{-2q(r_{ij}/r_0-1)}}} \left(\frac{-2q}{r_0}\right) e^{-2q(r_{ia}/r_0-1)} \frac{\partial r_{ia}}{\partial k_a}\right] \\
&+ \frac{\zeta}{2\sqrt{\sum_{j\neq a}^{N} e^{-2q(r_{aj}/r_0-1)}}} \left(\frac{-2q}{r_0}\right) \sum_{j\neq a}^{N} \left[e^{-2q(r_{aj}/r_0-1)} \frac{\partial r_{aj}}{\partial k_a}\right] \\
&- A \sum_{i\neq a}^{N} \left[\left(-\frac{p}{r_0}\right) e^{-p(r_{ia}/r_0-1)} \frac{\partial r_{ia}}{\partial k_a}\right] - A \sum_{j\neq a}^{N} \left[\left(-\frac{p}{r_0}\right) e^{-p(r_{aj}/r_0-1)} \frac{\partial r_{aj}}{\partial k_a}\right] \qquad \text{(A.4)}
\end{aligned}
$$

where, when we take k to be x,

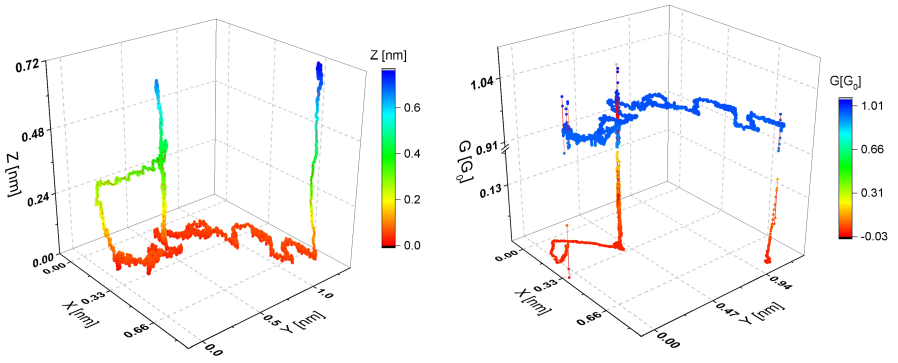
$$
\begin{aligned}
\frac{\partial r_{ij}}{\partial x_a} &= \frac{\partial}{\partial x_a} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \qquad \text{(A.5)} \\
&= \frac{2(x_a - x_j)}{2\sqrt{(x_a - x_j)^2 + (y_a - y_j)^2 + (z_a - z_j)^2}} \delta_{ia} \\
&- \frac{2(x_i - x_a)}{2\sqrt{(x_i - x_a)^2 + (y_i - y_a)^2 + (z_i - z_a)^2}} \delta_{aj} \qquad \text{(A.6)} \\
&= \frac{x_a - x_j}{r_{aj}} \delta_{ia} - \frac{x_i - x_a}{r_{ia}} \delta_{aj} \qquad \text{(A.7)}
\end{aligned}
$$

and similarly for y and z component, with δ_{ij} the Kronecker delta function. Substituting this in the force, we get

$$
\begin{aligned}
F_{a,k} = \zeta \sum_{i \neq a}^{N} & \left[\frac{1}{2\sqrt{\sum_{j \neq i}^{N} e^{-2q(r_{ij}/r_0 - 1)}}} \left(\frac{-2q}{r_0} \right) e^{-2q(r_{ia}/r_0 - 1)} \left(\frac{k_a - k_i}{r_{ia}} \right) \right] \\
& + \frac{\zeta}{2\sqrt{\sum_{j \neq a}^{N} e^{-2q(r_{aj}/r_0 - 1)}}} \left(\frac{-2q}{r_0} \right) \sum_{j \neq a}^{N} \left[e^{-2q(r_{aj}/r_0 - 1)} \left(\frac{k_a - k_j}{r_{aj}} \right) \right] \\
& - A \sum_{i \neq a}^{N} \left[\left(-\frac{p}{r_0} \right) e^{-p(r_{ia}/r_0 - 1)} \left(\frac{k_a - k_i}{r_{ia}} \right) \right] - A \sum_{j \neq a}^{N} \left[\left(-\frac{p}{r_0} \right) e^{-p(r_{aj}/r_0 - 1)} \left(\frac{k_a - k_j}{r_{aj}} \right) \right] \\
& = -\zeta \frac{q}{r_0} \sum_{i \neq a}^{N} \left[\frac{e^{-2q(r_{ia}/r_0 - 1)}}{\sqrt{\sum_{j \neq i}^{N} e^{-2q(r_{ij}/r_0 - 1)}}} \left(\frac{k_a - k_i}{r_{ia}} \right) \right] \\
& - \frac{\zeta (q/r_0)}{\sqrt{\sum_{j \neq a}^{N} e^{-2q(r_{aj}/r_0 - 1)}}} \sum_{j \neq a}^{N} \left[e^{-2q(r_{aj}/r_0 - 1)} \left(\frac{k_a - k_j}{r_{aj}} \right) \right] \\
& + 2A \frac{p}{r_0} \sum_{j \neq a}^{N} e^{-p(r_{aj}/r_0 - 1)} \left(\frac{k_a - k_j}{r_{aj}} \right)
\end{aligned}
\tag{A.8}
$$

A.2 Lifting of gold atomic chain

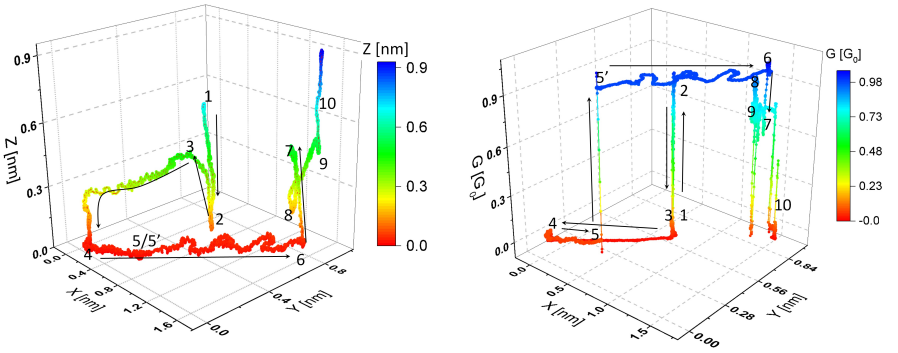
In chapter 2 a controlled formation of free standing atomic chain between STM tip and sample is demonstrated. The total operation is divided into two steps. In first step the ad-atom 'A' (marked in Figure 2.6) is pushed from position 'i' to 'ii' and then a STM image is taken (Figure 2.6(b)). The corresponding x,y,z and x,y,G curves for this operation are given here in Figure A.1. In the second step we start from position 'ii'



(a) (b)

Figure A.1: (a) Complete tip trajectory for the first step starting from position 'i' in the lattice and (b) variation in conductance over the time of operation.

and move the ad-atom 'A' to position 'iii' and then continue with the lifting operation followed by taking the STM image at the end (Figure 2.6(c)). Figure A.2(a) shows the tip trajectory for the second step and Figure A.2(b) shows the corresponding conductance variation over the time.



(a) (b)

Figure A.2: (a) Complete tip trajectory for the second step ending in a lift-off of a mono-atomic gold chain from the Au(111) surface and (b) variation in conductance during the lift-off of the atomic chain.

A.3 Point-contact push manipulation technique

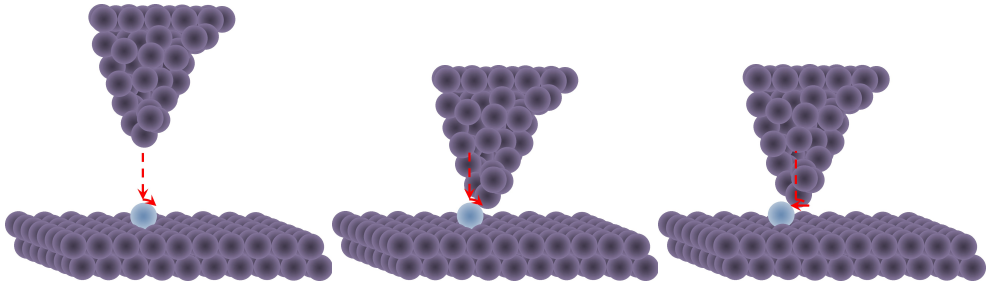


Figure A.3: Point contact push technique: (a) The tip is placed vertically over the ad-atom before a point contact is made by bringing the STM tip vertically downwards, (b) Once the a point contact is made the STM tip is moved in a circular trajectory while keeping into point contact to bring the tip behind the ad-atom, (c) Once the tip is placed behind the ad-atom it moves the later to the next hollow site.

Appendix

B. Molecular manipulation

B.1 GPU programming

What is a GPU (Graphics Processing Unit), how it is different from a CPU (Central Processing Unit) and what makes a GPU useful? Although both CPU and GPU have high computational abilities, the main difference between them is the type of computation they are designed for. CPUs are optimized for doing serial computation (or 'integer calculations' as they are called) for example: writing a text document, browsing the web etc. These are serial tasks because if you have to write a text: "*CPUs are very good in serial computation.*", you have to go one word after the other, you cannot type a text in parallel. GPUs, on the other hand, are good in performing parallel computations (or 'floating-point calculations') for example rendering 3D graphics, processing video compression etc. So, why is rendering 3D graphics a parallel process? In 3D graphics, everything is made of many small triangles (for example see Figure B.1) and rendering the image means moving (or transforming) of the three vertices of these triangles. These transformations (scaling, translation or rotation) are done by matrix multiplications which include a large number of mutually independent computations, that can be executed simultaneously as a parallel computation.

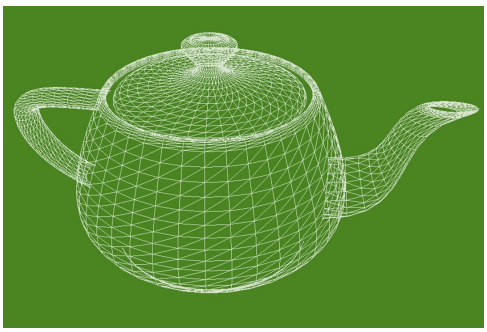


Figure B.1: Image of a kettle with triangular mesh used in 3D graphics.

B.1.1 Programming in the CUDA framework

The CUDA programming interface provides an abstraction to general purpose GPU (GPGPU) programming through the use of a number of models, most importantly the programming model and the memory model[1]. The programming model drives the programmer to follow a certain structure in writing his/her code, which is then mapped to the GPU architecture. This allows for optimizing the code at a low level while retaining the functionality and high flexibility of the C++ language. The memory model obeys this same principle, requiring to some extent manual memory management that is, manual memory allocation and de-allocation on the GPU, and explicit data copying without having to interfere with the details of transfers at the assembly level.

Hardware structure

nVidia GPUs[1] come with three sets of hierarchy: a hardware-based hierarchy, a programming model or task execution based hierarchy and last a memory model based hierarchy. To understand the complete hierarchy with concrete numbers, let us look at the specs of our nVidia GPU only and start with the hardware-based hierarchy. Table 3.1 shows the main device structure related configurations. The highest in the hierarchy sits the GPCs (Graphics Processing Clusters), our nVidia GPU has 2 GPCs. Each GPC further has 4 Streaming multiprocessors (SMM, The extra 'M' coming from 'Maxwell Architecture'). Now, each SMM has four processing blocks, which have their instruction buffer and *warp* scheduler. Inside each processing block, there are 4×8 i.e., 32 CUDA cores, which makes a total of 4×32, i.e., 128 CUDA cores per SMM and a total of 1028 CUDA cores in the GPU card. These CUDA cores are also called SPs (Streaming Processors). More abstract entities like threads, blocks etc. comes under the task execution stream.

Programming model or task execution model

The basic building block of any CUDA application is a *kernel*. This *kernel* is the operation or mutually independent computation that you want to be repeatedly computed many times. For example, if you have to add two matrices then 'adding a pair of elements with same indices in the two matrices' is a mutually independent computation and so is a *kernel*. Such a *kernel* can be called either from the CPU or the GPU. In a GPU many instances of the same instructions or computations defined in *kernels* are executed by what are called *threads*[2]. through a very hierarchical task execution model. In the CUDA programming guide by Cook[1] this hierarchy in which the tasks are performed is seen in analogy with a hierarchy in an army. This analogy helps in appreciating the task divisions, so we will explain it like that. A schematic is shown in Figure B.2. The lowest in the hierarchy are the *threads* or the 'soldiers'. These 'soldiers' are divided into different 'units' called *blocks*. A 'unit' is split into 'squads' of 32 'soldiers' called a *warp*. Different 'units' together constitutes an 'army' called a *grid* in GPU terminology. Though each 'soldier' or *thread* does its individual job, but a task coming from the 'Colonel' or the *kernel* is always allotted per 'squad' and not to an individual 'soldier'. So, a 'squad' or *warp* forms a basic unit

[1] http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html#axzz4r4uG4Vkc

[2] The *thread* is just an abstract entity that represents the execution of the *kernel*.

[1] Shane Cook. Newnes, 2012.

of execution on the GPU. The 32 number of *threads* in *warp* called *warpSize* is set by nVidia and it reserves the right to change this in the future. How the information flow happens between different parts of this task execution model will be discussed in the next section where we will discuss the memory model based hierarchy. Lets now look at some more quantitative numbers from the Table 3.1. The maximum block size is given as $1024 \times 1024 \times 64$, but this does not mean that we have a number of $1024 \times 1024 \times 64$ threads per block. The maximum number of threads per block is given as 1024 in Table 3.1, this means that the dimensions of the *thread block* i.e. $[\tau_x, \tau_y, \tau_z]$ should be chosen such that $\tau_x \times \tau_y \times \tau_z \leq 1024$. For example: one could have block dimensions $(32, 32, 1)$ or $(1, 1024, 1)$ or $(16, 1, 64)$ etc. *Thread blocks* can also have lower dimensions, though the execution of these *threads* happens in the same way irrespective of whether one uses multiple dimensions or not, doing so can ease the design of algorithms that have to handle multi-dimensional data, such as matrices. The size of each *grid* is normally dictated by the size of the data set to be processed by the *kernel*, while the *block* size is often algorithm-specific. Depending on the algorithm, it may occur that calling a *kernel* with a certain *block* size may result in better performance than other sizes, and the optimal *block* size may depend on many variables[2]; hence, this size is often chosen empirically.

Memory model

As we know, a CPU has different types of memory in it like drive memory, random access memory (RAM), register memory and others. Similar to this a GPU also has a memory architecture which could vary a little from one GPU to other. Data is transferred between CPU and GPU device memory via the PCI-e bus, in our case it is PCIe 3.0 (with x16 link) having a total bandwidth of 32 GB/s. GPU device memory in our case is the GDDR5 graphics RAM and can be divided broadly into global and local memory. Out of the two, the global memory (4096 MB) is what connects to the CPU and can also be accessed by all the threads, through an L2 cache. The memory declared[3] in the global memory stays untill the end of the lifetime of the application or can only be erased by a host command[4] and will then be available for other applications. A schematic view of the CUDA memory model is shown in Figure B.3.

Within each thread block, threads can write to a chunk of so-called shared memory, which may vary in size as defined by the user upon calling a kernel. This shared memory lies on-chip on the streaming multiprocessors (SMM) directly, and so has two orders of magnitude lower latency than global memory. A lot of performance optimisation involves maximising shared memory usage above global memory. One must keep in mind, though, that shared memory is limited (in our case we have around 96 KB of shared memory per SMM), and using too much of it per block will reduce the number of blocks that can run in parallel. By sharing the memory access among different threads, redundant computations can be avoided and thus can reduce the global memory bandwidth. In fact, shared memory is very similar to a memory cache of a CPU with a major advantage in GPU that here it can be

[3] You can declare a part in the global memory by calling *cudamalloc*

[4] *cudaFree* can be used to erase the global memory

[2] Henry Wong et al. In: *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*. IEEE. 2010, p. 235.

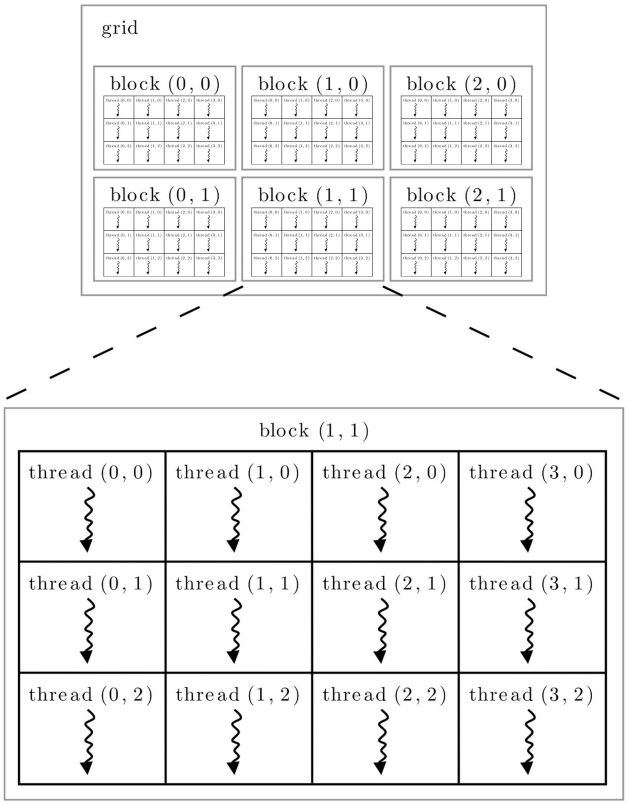
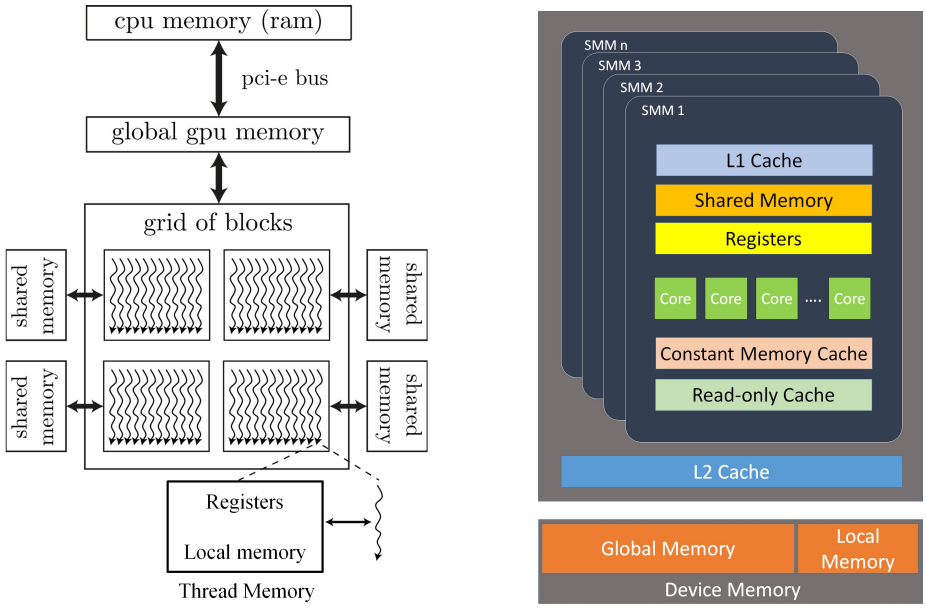


Figure B.2: A two-dimensional grid of blocks, where each *thread block* also has two dimensions. Typical *thread* blocks will in reality contain more than twelve *threads*, since *threads* are dispatched in groups of 32 on the GPU.

completely managed by the programmer. A part of the device memory where you can declare constants directly from the host code[5] is called constant memory. Being in the device memory this readout is slow but each SMM has around 8 to 10 KB of constant memory cache, which means every time you read from constant memory (which is total 64 KB in size) lying in the device memory the data will be cached in the constant memory cache. This helps in reducing the waiting time during one complete task execution.

Lastly, each thread can store its local variables in thread memory, which is implemented as a 32-bit register file. Registers are on-the-chip and thus also the fastest parts of GPU memory with least latency. This memory is private to the thread and so cannot be shared with other threads. In case of insufficient space in the registers, a thread can store its local variable in the local memory which sits off-chip in the device memory and thus is almost two orders of magnitude slower than registers. Both these private thread memories are managed by the compiler directly and cannot

[5] using *cudaMemcpyToSymbol()* command



(a) (b)

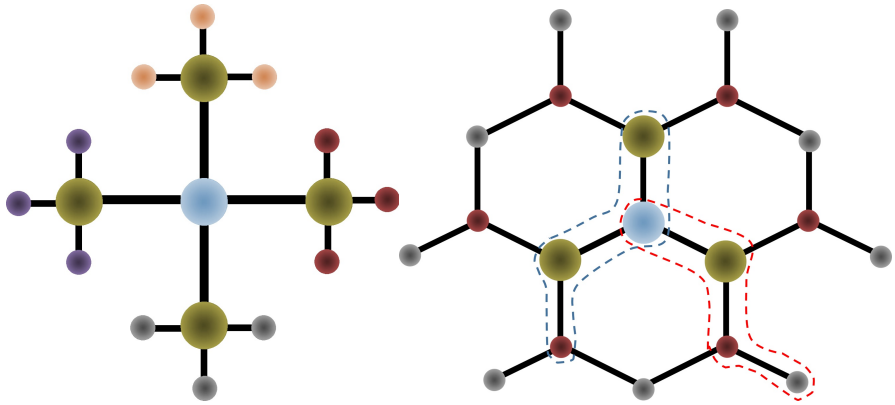
Figure B.3: The Cuda memory arrangement. Each grid of blocks (that is, all threads within a grid of thread blocks) is connected to the global memory, which is bridged to the cpu ram via a pci-e bus. Each thread block has a separate chunk of block memory, and each thread can also use its own thread memory.

be controlled by the programmer. The lifetime of these is equal to the lifetime of the thread itself. So, when a thread has finished executing all the tasks, these thread memories will be automatically erased.

B.2 Intramolecular forces

The intramolecular forces consist of bond forces, angular forces and torsional forces. For a molecule, with a maximum of n bonds per atom (single, double or triple covalent bonds are all counted as one bond), the maximum number of bond forces calculations n_b per atom is n.

Next, we estimate the maximum number of angular forces calculations n_a per atom and the maximum number of torsional forces calculations n_t per atom. For this, consider the illustration in Figure B.4 (a). It shows a simplified geometric construction of a molecule with four bonds per atom. In reality, a molecule with four bonds per atom is more likely to be non-planar, but for the sake of simplicity, we draw it planar here. In the picture, the centre atom (shown in blue colour) has four 1^{st} neighbour atoms (shown in green colour), and each of them has their respective four neighbour atoms including the central blue atom.



(a) (b)

Figure B.4: A geometric construction of a molecule with (a) $n = 4$ bonds per atom and (b) $n = 3$ bonds per atom .

To estimate the *maximum* number of angular forces calculations per atom we calculate them for the central blue atom. The number of ways in which the central blue atom makes an angular bond with only first neighbours (green atoms) is equal to the number of ways in which it can choose any two of the green atoms, which is ${}^4C_2$. In general case for n bonds per atom, it is ${}^nC_2$ which is $n(n-1)/2$. Next, the central blue atom can also make angular bonds with one green atom and one of its other neighbours (i. e. purple, grey, orange or red atoms). The number of ways in which this can be done is equal to the number of ways the central blue atom chooses one of the green atoms (i. e. n ways) times the number of ways the chosen green atom chooses one of its neighbours excluding the blue atom (i. e. $(n-1)$ ways). This makes the total equal to ${}^nC_2 + n(n-1)$. Thus the maximum number of angular forces calculations n_a per atom is $\mathbf{\frac{3}{2}n(n-1)}$.

The estimation of the maximum number of torsional forces calculations n_t per atom is done similarly as the previous calculation. However, the difference is that for torsional forces we have to make a group of four atoms connected in a chain fashion.

For this we look at Figure B.4(b). We need to calculate how many groups of four atoms the central blue atom can make. The central blue atom can make an outer group (shown with the red dashed loop) or an inner group (shown with the blue dashed loop) depending on its position in the group. The number of outer groups that the central atom makes is equal to the number of ways in which the central blue atom chooses one of its first green neighbours times the number of ways the chosen green atom further chooses one of its red neighbours times the number of ways the chosen red atom, finally chooses one of its grey neighbours. This combines into $= n \times (n-1) \times (n-1)$

The number of inner groups (shown with the blue dashed loop in Figure B.4) which the central blue atom makes is equal to the number of ways the central blue atom chooses any two of its first neighbour green atoms times the number of ways one of the two chosen green atoms is selected times the number of ways the selected green atom then chooses one of the red neighbour atoms. This combines into ${}^{n}C_2 \times 2 \times (n-1)$.

This gives the total torsional groups and thus the maximum number of torsional forces calculations n_t per atom is ${}^{n}C_2 2(n-1) + n(n-1)^2 = \mathbf{2n(n-1)^2}$.

B.3 Gold-molecule force calculations

Covalent bonds are usually modeled using harmonic potentials, where the restoring force increases linearly with the bond-length. This simple potential can describe the intermolecular bonds very well and is a good assumption in most situations where the bonds are not stretched enough to break them. However, for studying or simulating the bond formation between the metallic lead (Au) and the anchoring group (N or S) one has to look for more advance potentials like Morse and Lennard-Jones (LJ) potentials. As the Morse potential decays faster than the LJ potential, we use it to simulate the stronger Au-N bond, while LJ potentials are used to simulate Au-C and Au-H bonds.

B.3.1 Morse potential: Au-N bond

The Morse potential is given as:

$$E = \sum_{i}^{N} \sum_{j \neq i}^{N} D_e \left(1 - e^{-\alpha(r_{ij} - r_{eq})}\right)^2 \tag{B.1}$$

Here, $\alpha = \sqrt{\kappa_e / 2D_e}$ and $\kappa_e = \kappa_{Au-N}$ is the force constant at the minimum of the potential where a harmonic approximation is valid. From here the k^{th} component of the force on an atom with index a can be written as

$$
\begin{aligned}
F_{ak} &= -\frac{\partial E}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial k_a} \\
&= -\frac{\partial}{\partial r_{ij}}\sum_{i}^{N}\sum_{j\neq i}^{N} D_{\mathrm{e}}\left(1-e^{-\alpha(r_{ij}-r_{\mathrm{eq}})}\right)^2\frac{\partial r_{ij}}{\partial k_a} \\
&= -\sum_{i}^{N}\sum_{j\neq i}^{N} 2D_{\mathrm{e}}\left(1-e^{-\alpha(r_{ij}-r_{\mathrm{eq}})}\right)\alpha e^{-\alpha(r_{ij}-r_{\mathrm{eq}})}\left[\frac{x_a-x_j}{r_{aj}}\delta_{ia}-\frac{x_i-x_a}{r_{ia}}\delta_{aj}\right] \\
&= -2\alpha D_{\mathrm{e}}\left[\sum_{j\neq a}^{N} e^{-\alpha(r_{aj}-r_{\mathrm{eq}})}\left(1-e^{-\alpha(r_{aj}-r_{\mathrm{eq}})}\right)\frac{x_a-x_j}{r_{aj}}\right. \\
&\quad \left.-\sum_{i\neq a}^{N} e^{-\alpha(r_{ia}-r_{\mathrm{eq}})}\left(1-e^{-\alpha(r_{ia}-r_{\mathrm{eq}})}\right)\frac{x_i-x_a}{r_{ia}}\right] \\
&= -4\alpha D_{\mathrm{e}}\sum_{i\neq a}^{N} e^{-\alpha(r_{ia}-r_{\mathrm{eq}})}\left(1-e^{-\alpha(r_{ia}-r_{\mathrm{eq}})}\right)\frac{x_a-x_i}{r_{ia}}
\end{aligned}
\tag{B.2}
$$

B.3.2 Lennard-Jones (LJ) potential

The LJ potential has the folowing functional form

$$
E = 4\epsilon\sum_{i}^{N}\sum_{j\neq i}^{N}\left[\left(\frac{\sigma}{r_{ij}}\right)^{12}-\left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{B.3}
$$

Same as before the k^{th} component of the force on an atom with index a can be written as

$$
\begin{aligned}
F_{ak} &= -\frac{\partial E}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial k_a} \\
&= -4\epsilon\frac{\partial}{\partial r_{ij}}\sum_{i}^{N}\sum_{j\neq i}^{N}\left[\left(\frac{\sigma}{r_{ij}}\right)^{12}-\left(\frac{\sigma}{r_{ij}}\right)^{6}\right]\frac{\partial r_{ij}}{\partial k_a} \\
&= -4\epsilon\sum_{i}^{N}\sum_{j\neq i}^{N}\left[-12\frac{\sigma^{12}}{{r_{ij}}^{13}}+6\frac{\sigma^{6}}{{r_{ij}}^{7}}\right]\left[\frac{x_a-x_j}{r_{aj}}\delta_{ia}-\frac{x_i-x_a}{r_{ia}}\delta_{aj}\right] \\
&= -48\epsilon\sum_{i\neq a}^{N}\left[-2\frac{\sigma^{12}}{{r_{ai}}^{14}}+\frac{\sigma^{6}}{{r_{ai}}^{8}}\right](x_a-x_i)
\end{aligned}
\tag{B.4}
$$

The position of the energy minimum and the width and depth of the potential well in both the Morse and the LJ potentials have to be determined using the experiments. For this the deformation in the molecular structure on absorption on a metallic surface, its absorption height and absorption energy can be used to fix the unknown parameters. For a proof of principle demonstration of the simulation, we have used the following parameters in our case:

Table B.1: Parameters used in the simulation

Parameters	Values	Units
κ_{Au-N}*	325	N/m
D_e¶	1	eV
r_{eq}	2.5	Å
σ	2.2	Å
ϵ	0.02	eV
κ_{Au-Au}*	8	N/m

* Value from Rascón-Ramos *et al.*[3].
¶Value similar to Fournier *et al.*[4].

[3] Habid Rascón-Ramos et al. In: *Nature Materials* 14 (2015), p. 517.
[4] N. Fournier et al. In: *Phys. Rev. B* 84 (2011), p. 035435.

Appendix

C. Shot noise

C.1 Excess Noise derivation

We start by writing the total noise power as the sum over noise coming from individual sources times the respective transfer function they see.

$$S_{\text{tot}}(\omega, V) = \sum S_n H_n = S_{\text{m}}/\left|\beta(\omega)\right|^2 \tag{C.1}$$

It is important to note that these transfer functions depend on the position of the source in the system and can be different for each noise source. Expanding this over the known sources we have,

$$S_{\text{m}}/\left|\beta(\omega)\right|^2 = S(V,\omega)\left|H_1\right|^2 + S_{\text{R1}}(V,\omega)\left|H_{\text{R1}}\right|^2 + S_{\text{amps}}(V,\omega)\left|H_{\text{amps}}\right|^2 + S_{\text{x-talk}}(\omega) \tag{C.2}$$

Here, all the transfer functions ($H_1, H_{R1}, H_{\text{amps}}$) depend on $R_s(V)$. Now we can lump all the noise sources together, except the one we are interested in (i.e. $S(V,\omega)$)

$$S_{\text{m}}/\left|\beta(\omega)\right|^2 = S(V,\omega)\left|H_1\right|^2 + \tilde{S}_{\text{x}}(V,\omega) \tag{C.3}$$

Here we define, $\sum S_X(V,\omega)\left|H_X\right|^2 = \tilde{S}_{\text{x}}(V,\omega)$. Next, we make the assumption that the extrinsic noise sources can be taken as voltage independent, and that they depend only on frequency and temperature i.e.,

$$S_{\text{m}}(V) = \left|\beta(\omega)\right|^2\left[S(V,\omega)\left|H_1\right|^2 + \tilde{S}_{\text{x}}(T,\omega)\right] \tag{C.4}$$

The validity of this assumption is supported by the test described in the paper, with a 10 k Ω resistor in the position of the sample. This demonstrates that the voltage dependent noise remains well below our claimed accuracy limit, even up to 1V bias. Note that the small spurious voltage dependent noise observed in this test includes Joule heating of the test resistor, so that the actual voltage dependence of instrumental noise sources is even smaller. The above equation C.1 is the same as the equation for the measured noise power given in Chapter 4 equation 4.4.

Next, we explain how the expression for the shot noise we are interested in can be extracted. We now introduce the notion of excess noise ($S_{\text{e}} = S(V) - S(0)$).

Introducing the quantity of excess noise is used extensively in shot noise theory and experiments alike. Now, using the assumption that $\tilde{S}_x$ is not voltage dependent, we write

$$S_m(0) = \left|\beta(\omega)\right|^2 \left[S(0)\left|H_1\right|^2 + \tilde{S}_x(T,\omega)\right] \tag{C.5}$$

so that,

$$S_m(V) - S_m(0) = \left|\beta(\omega)\right|^2 (S(V) - S(0))\left|H_1\right|^2 = \left|\beta(\omega)\right|^2 S_e \left|H_1\right|^2, \tag{C.6}$$

which gives us the equation 4.5 in Chapter 4.

C.2 Two channel fit

We will include here a second constant transmission channel to our model described in chapter 5 to explain the nonlinear shot noise data shown in Figure 5.3. The transmission value for this second channel is extracted from the experimental dataset. For this we need to solve a set of equations for the low bias noise and differential conductance:

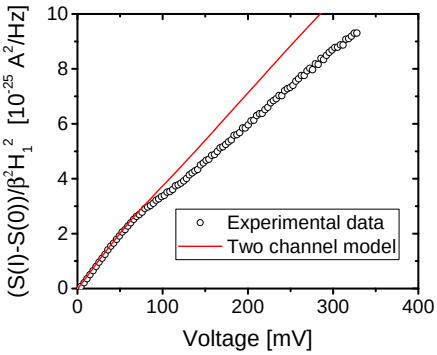
$$S_I = 4\frac{e^3}{h} V \left[T_1(1-T_1) + T_2(1-T_2)\right] \tag{C.7}$$

Next, one can input T_2 in terms of T_1 using Landauer's conductance formula i. e. $T_2 = G_0 - T_1$, here $G_0 = G(0V)$. This gives:

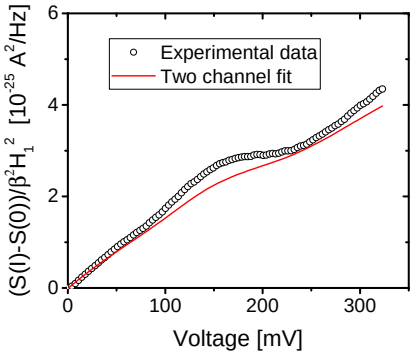
$$\frac{S_I}{V} = 4\frac{e^3}{h} \left[T_1(1-T_1) + (G_0 - T_1)(1 - G_0 + T_1)\right] \tag{C.8}$$

In the above equation left hand side is the slope of measured experimental noise close to zero bias. This can be obtained from experiment. Thus we are left with a quadratic equation in T_1. On solving this we can get the zero bias transmission for the two channels involved in transport i. e. T_1 and T_2.

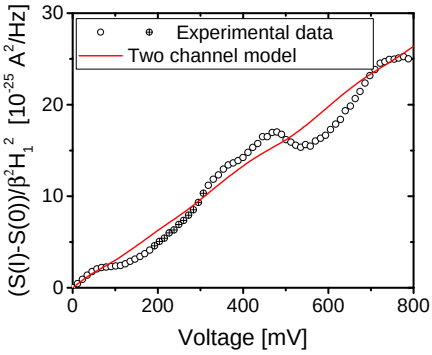
However, we don't know how the transmission of two channels will evolve at higher bias. So, we make an assumption that the second channel transmission remains constant at all bias and the first channel transmission mimics the measured differential conductance shifted down by a value equal to the transmission of second channel. This constant transmission assumption though adds to the noise calculated by the model, but it smooths out the non-linearity as shown in Figure C.1.



(a)



(b)



(c)

Figure C.1: Two channel fit after assuming a constant transmission contribution from the second channel for (a) Ex1, (b) Ex2 and(c) Ex3. The points for which the noise spectra show non-white character are shown with filled circles.

Appendix

D. Robust procedure for creating and characterizing the atomic structure of scanning tunneling microscope tips

The work is done in collaboration with - Koen M. Bastiaans, Milan P. Allan and Jan M. van Ruitenbeek and is published in *Beilstein Journal of Nanotechnology, 8, 2389-2395 (2017)*

Scanning tunneling microscopes (STM) are used extensively for studying and manipulating matter at the atomic scale. In spite of the critical role of the STM tip, procedures for controlling the atomic-scale shape of STM tips have not been rigorously justified. Here, we present a method for preparing tips *in-situ* and for ensuring the crystalline structure and reproducibly preparing tip structure up to the second atomic layer. We demonstrate a controlled evolution of such tips starting from undefined tip shapes.

Many techniques are available for preparing atomically clean sample surfaces, including repeated cycles of sputtering and annealing in case of metals, by fresh cleavage in case of suitable materials, and by high-temperature annealing, as for many semiconductors. However, a well defined tip structure at the atomic scale is still hard to achieve. Mechanical grinding[1], electro-polishing[2] or electrochemical etching[3,4] are standard *ex-situ* methods for preparing microscopically sharp tips. The tip apex can be cleaned *in-situ* using e.g. Ar ion sputtering or electron bombardment[5], but this may disrupt crystalline structure, which cannot be repaired by annealing since this renders a blunt tip. For all these methods, at the atomic scale the tip structure is poorly controlled and could even have multiple local apexes. For many purposes this does not hamper STM operation, since the tunnel current decays exponentially with tunnel gap so that the atom closest to the surface will dominate the imaging signal. However, the reproducible shape of current-voltage spectra depends strongly on the

[1] Gerd Binnig et al. In: *Phys. Rev. Lett.* 49 (1982), p. 57.
[2] Kojima Isao and Kurahashi Masayasu. In: *Hyomen Kagaku* 9 (1988), p. 621.
[3] Michael J. Heben et al. In: *Journal of Microscopy* 152 (1988), p. 651.
[4] Victor A. Valencia et al. In: *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films* 33 (2015), p. 023001.
[5] S. Ernst et al. In: *Science and Technology of Advanced Materials* 8 (2007), p. 347.

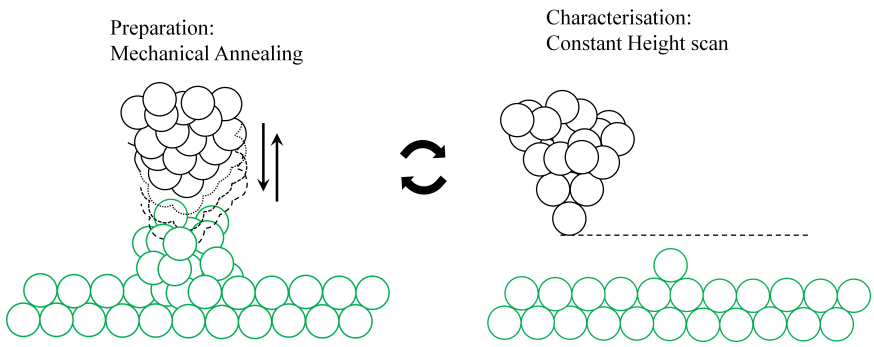


Figure D.1: Schematic representation of the tip preparation process, where mechanical annealing cycles were followed by constant height scans made over a single ad-atom deposited on the surface.

tip shape[6]. Controlled manipulation by STM of ad-atoms and molecules over metal surfaces depends also on the precise knowledge and reproducibility of the atomic tip structure. In the field of molecular electronics, where researchers are now trying to connect single molecules between an STM tip and a flat metal surface[7], knowledge of tip shape is crucial.

We will now present a procedure, illustrated in Fig. D.1, based upon mechanical annealing procedure[8,9] that permits arriving at reproducible crystalline tip shapes starting from any random initial tip shape, and we show how we can verify the evolution of the tip shape. For this the STM tip is indented into the flat metal surface up to a pre-set conductance value and then retracted and this cycle is repeated many times. We then exploit the capabilities of the low-temperature STM setup for imaging the evolution of the tip structure by scanning over an isolated ad-atom on the surface. The obtained STM images are convolutions of the topography and electronic states of sample and tip. Lang[10] has shown theoretically using two planar metal electrodes with a single ad-atom on each of them, that upon scanning one with the other symmetric convolutions of the topography and electronic states are expected, giving circularly symmetric images. Any asymmetry in the STM images of an isolated ad-atom will reflect the asymmetry of the atomic structure behind the front atom.

We start the experiments by depositing a single ad-atom from the Au covered PtIr tip in the center of the FCC sector of the Au(111) herringbone reconstruction, following the procedure summarized by Hla[11]. The result is imaged in the usual topographic mode of STM shown as an inset in Fig. D.2 (a), which demonstrates that the ad-atom does not have circular symmetry. In order to verify that this asymmetry is associated with the tip, we deposit a second ad-atom. The image of the second

[6] B. Ludoph and J M van Ruitenbeek. In: *Physical Review B* 61 (2000), p. 2273.
[7] Christian Wagner and Ruslan Temirov. In: *Progress in Surface Science* 90 (2015), p. 194.
[8] I. Guillamon et al. In: *Physica C: Superconductivity and its Applications* 468 (2008), p. 537.
[9] C. Sabater et al. In: *Phys. Rev. Lett.* 108 (2012), p. 205502.
[10] N. D. Lang. In: *Phys. Rev. Lett.* 56 (1986), p. 1164.
[11] Saw Wai Hla. In: *Reports on Progress in Physics* 77 (2014), p. 056502.

atom is a replica of the first, in shape and orientation, confirming that the asymmetry is associated with the tip shape.

In the next steps we use an individual ad-atom for imaging the structure of the tip apex, employing constant-height mode scans with a box size of 2 nm centered on the ad-atom. The goal is to pick up tunneling current signal from the second row of atoms above the apex atom. For FCC packing the distance to the second row is 2.5 Å larger than that to the apex atom, from which we estimate the current level to the second-row atoms at 100 pA for a current of 30 nA at the apex atom. As the tunnel current varies exponentially with distance, even small deviations from surface-parallel FCC packing of the second-layer tip atoms will give a detectable contribution. In order to scan at such high tunnel current we first switch off the current feedback and bring the tip closer to the flat part of the surface to a fixed tunnel current value. This value is chosen to ensure that the current is as high as 30 nA, when the tip is over the ad-atom during scan. Then we take the constant-height scan at a tip speed of 2 nm/s. Figure D.2(a) shows an example of the resulting image at the initial stage, before starting the tip preparation procedure.

After imaging the tip apex we move the tip to an edge of the scan range, about 700 nm away from the ad-atom, and perform a series of 20 mechanical annealing cycles. For this we use the same MATLAB controlled procedure as described above, except that we indent the tip farther until it reaches a preset conductance of 4 G_0. After 20 of these mechanical annealing cycles, we return to the original ad-atom and image it again using the same procedure as above. We repeat these steps of mechanical annealing and imaging until we obtain a symmetric tip image, as illustrated in Figs. D.2(a-f).

For analyzing the results of this procedure more quantitatively we would like to define a parameter which could capture the convergence of the tip structure to this symmetric state. For comparing the constant height images we select a contour at a fixed tunnel current level (which for our case is 13.3 nA) and fit an ellipse to this contour. From the fit we extract the ratio of major to minor axis of the ellipse, a/b, as a measure of the deviation from circular symmetry. Figure D.3 shows a plot of the evolution of the a/b ratio with the number of annealing cycles. The blue and the green data points show two independent runs, for different tips and different samples. The red dots show the start and end points for a third set of data applying 1000 mechanical annealing cycles. Surprisingly, the evolution of the tip is very regular and reproducible. Starting from uncontrolled and asymmetric tip apex configurations, we find that the a/b ratio decays following a common pattern, and arrives at similar minimum values. The deviation of about 4 % from 1 for the a/b ratio is probably limited by thermal drift and electrical noise of our system. The curve shows that the data are closely described by an exponential dependence.

To conclude, we have demonstrated a method for shaping a metallic tip apex in STM. By placing an ad-atom over a smooth Au surface the structure of the tip apex can be imaged, and we find that the shape of the STM tip evolves surprisingly smoothly and reproducibly towards an atomically sharp and symmetric structure of the second layer from the tip apex atom, starting from any random and poorly defined tip shapes.

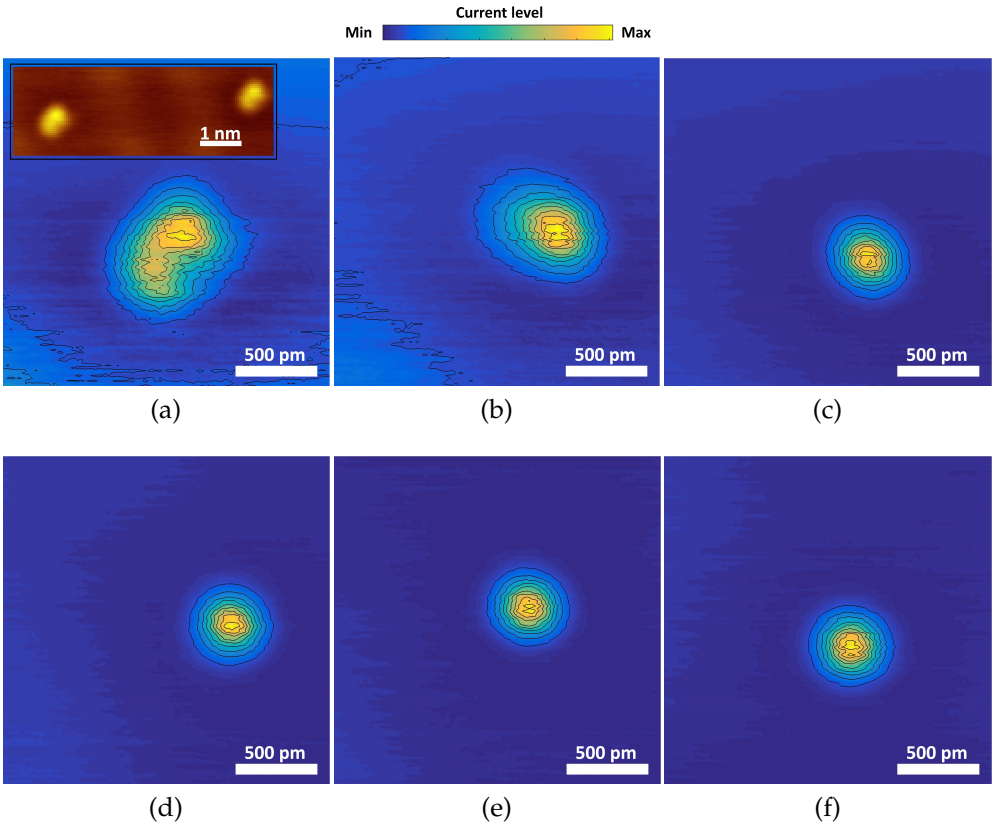


Figure D.2: The six panels show constant-height images of a single ad-atom. Here (a) shows a non-circular image of an ad-atom due to the random tip structure at the start. The inset of (a) shows constant-current image of two separate ad-atoms prepared on the surface to confirm that the asymmetric structure is due to the tip. The evolution of tip apex is shown in the next panels, leading towards a symmetric and reproducible structure (b-f). Between each of the images we apply 20 mechanical annealing cycles. The contours shown are linearly spaced in current. For ease of comparison the current levels in the images have been normalized to the maximum current level, which for the panels (a)-(f) are 66 nA, 31 nA, 26 nA, 26 nA , 25 nA and 22 nA, respectively.

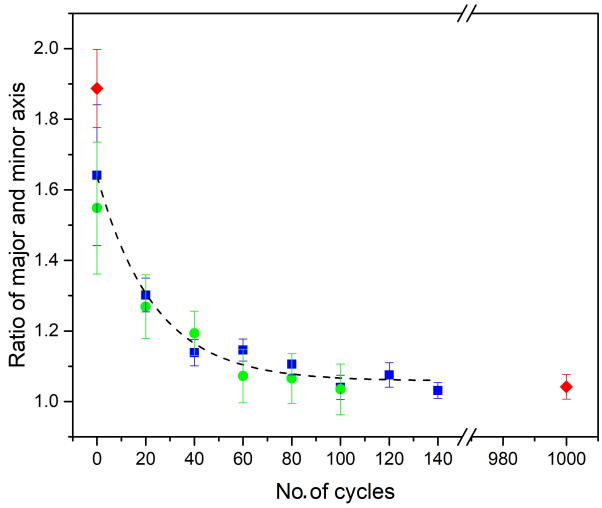


Figure D.3: Ratio of the major to minor axis of the ellipses fit showing convergence of the tip apex structure to a circularly symmetric shape. Three independent runs are shown by blue, green and red symbols. The two points shown as red diamonds represent initial and final images of an ad-atom for 1000 mechanical annealing cycles. The black dashed curve is a guide to the eye showing exponentially smooth transition to a circularly symmetric state.

Appendix

E. Inhomogeneous broadening of the conductance histograms for molecular junctions

The work is done in collaboration with - Julian M. Bopp, Carlos Sabater and Jan M. van Ruitenbeek and is published in *Low Temperature Physics, 43, 8, 905-909, (2017)*

Scanning tunnelling microscopy based break junction[1] and lithographic mechanically controlled break junction systems (MCBJ)[2] are commonly used tools for studying single-molecule electronic transport. We demonstrate here using notched-wire MCBJ technique an electronic transport study on an oligo phenylene-ethynylene (OPE-3) molecule (shown in Fig. E.1). OPE3 is a conjugated rod-like molecule with a conductance of $1 - 2 \cdot 10^{-4}\, G_0$,[3–5] where $G_0 = 2e^2/h$ is the conductance quantum. For the deprotection of the anchor groups we follow the procedure by Frisenda *et al.*[4].
Pure OPE3 is a powder, that we dissolve in dichloromethane at a concentration of 1 mmol/l. For splitting off the protecting acetyl groups we use a hydrate of tetrabutylammonium hydroxide (TBAH) solution of 10 mmol/l . The solutions were prepared by sonication for one hour to fully dissolve the compounds. Immediately following sonication of the solutions, the deprotection was done by adding 42 μL of the TBAH solution to 2 mL of the OPE3 solution. The color of the OPE3 solution changes from a pale yellow to a slightly brighter yellow.

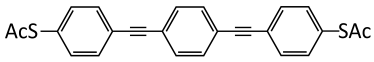


Figure E.1: Chemical structure of the OPE3 molecules used in this study.

The procedure for deposition of molecules on the junction proceeds according to the

[1] Bingqian Xu and Nongjian J. Tao. In: *Science* 301 (2003), p. 1221.
[2] J. Reichert et al. In: *Phys. Rev. Lett.* 88 (2002), p. 176804.
[3] R. Frisenda and H. S. J. van der Zant. In: *Phys. Rev. Lett.* 117 (2016), p. 126804.
[4] R. Frisenda et al. In: *physica status solidi (b)* 250 (2013), p. 2431.
[5] Veerabhadrarao Kaliginedi et al. In: *Journal of the American Chemical Society* 134 (2012), p. 5262.

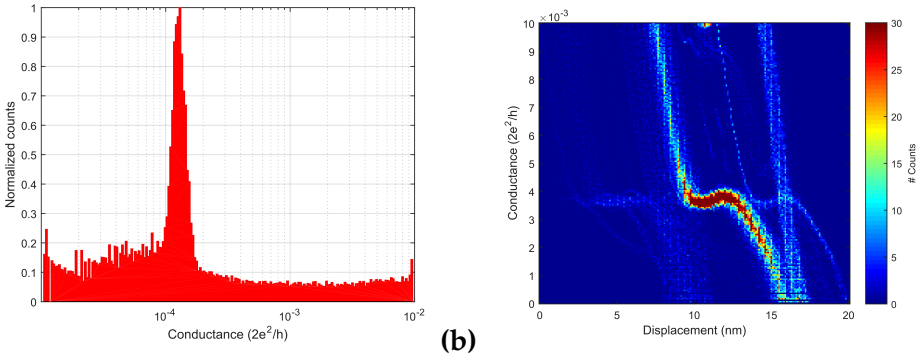


Figure E.2: Conductance histograms for OPE3 molecular junctions at 4.2 K. (a) Logarithmic one-dimensional conductance histogram showing the counts of conductance data collected for the first 900 breaking cycles of the NW-MCBJ junction. (b) The same data represented as a two-dimensional color coded histogram, with the electrode displacement plotted along the horizontal axis.

following steps. Under continuous flow of dry nitrogen the gold junction is broken by manual control and reformed to verify that the mechanism is working properly. A drop of the solution is then deposited, immediately after deprotection. The contact is then broken again to allow the OPE3 to completely cover the electrodes. The vacuum can is then closed, evacuated and the system is cooled down by immersion of the dip-stick sample holder into liquid helium.
After deposition of OPE3, already at room temperature, a broad asymmetric peak becomes visible at about $2 \cdot 10^{-4}\ G_0$ (not shown here), in agreement with the typical conductance values for OPE3 reported in the literature.[3,4,6,7] Approximately 10 % of the recorded traces contained signatures of OPE3 here. While this already demonstrates that the NW-MCBJ technique can be employed for the investigation of thiol-coupled molecules, the most remarkable observation was made when cooling down to liquid helium temperatures. After reaching helium temperatures it required several attempts at deep indentation of the wire ends for signatures of OPE3 molecules to manifest themselves in the conductance traces. But as soon as they appeared they persisted in a large number of subsequent traces. Fig. E.2a shows a logarithmic histogram for 900 breaking traces. The histogram has a sharp peak at $1.3 \cdot 10^{-4}\ G_0$, and has an unprecedently narrow width of only 40 % at FWHM. The typical width of the conductance histogram for such molecular junctions is one decade.
We gain more insight into the nature of the sharp conductance peak by making a rupture trace density plot. The same data as for the histogram we re-analyzed by aligning the points of metal-metal contact breaking in the horizontal axis for all curves. In practice we identify this point as the point with the steepest slope, and it occurs when the conductance suddenly drops below the gold atomic contact value of 1 G_0. Fig. E.2b shows a color coded plot of the density of points obtained in the plane spanned by displacement (x-axis) and conductance (y-axis). The conclusion

[6] Songmei Wu et al. In: *Nature nanotechnology* 3 (2008), p. 569.
[7] Roman Huber et al. In: *Journal of the American Chemical Society* 130 (2008), p. 1080.

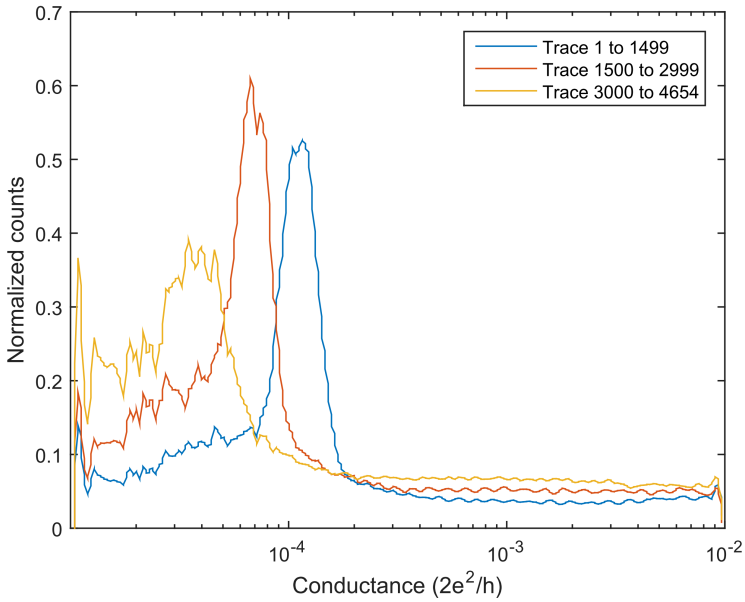


Figure E.3: Logarithmic conductance histogram of OPE3 at 4.2 K built from 4654 rupture traces, including the first 900 of Fig. E.2. The histogram shows a broad peak which can be decomposed into several distinct peaks indicating different configurations of OPE3 anchoring to the gold electrodes.

we draw from this graph is that the far majority of breaking traces follows the same pattern. The finite width of the peak in the conductance histogram is not the result of statistical variation in the molecular junction properties, but is an intrinsic property of the dependence of the molecular junction on the state of stretching of the junction. The reproducibility of the conductance traces exists despite the fact that the range of displacement is about 8 nm, much larger than the size of the molecule of less than 2 nm. During the breaking cycles the indentation of the metal-metal contacts is at least 3 nm, equivalent to a conductance of several times the conductance quantum. It seems plausible that a single OPE3 molecule is being repeatedly caught in the same junction configuration.

The data shown in Fig. E.2 represent the first 900 breaking cycles. Continuing the measurements beyond this point and taking into account the full set of rupture traces yields a much broader peak in the conductance histogram, as shown in Fig. E.3. This peak has an appearance and a width that is much better comparable with typical results presented in the literature.[3,4,6] However, in our case the wide peak appears as if being a superposition of several sharper peaks with slightly shifted conductances. This is confirmed by splitting the data set into three subsets of consecutively recorded traces, as depicted in Fig. E.3. This observation suggests that at least three different contact configurations have been probed. Each of the three is highly reproducible, but spontaneous switches from one configuration to the next occur that make the previous configuration inaccessible.

To conclude, as a first remarkable observation we find exceptionally sharp conductance histogram peaks for about 1000 conductance breaking cycles. We attribute the sharp peaks to precisely recurring molecular configurations. At the same time we find that the regularly found broad histogram peaks reported in the literature can be attributed to inhomogeneous broadening resulting from the contribution of many different molecular configurations. The reproducibility of molecular configurations is possibly related to a similar phenomenon observed for atomic contacts.[8] After training metallic contacts, i.e. applying many cycles of contact making and breaking, the conductance is found to repeatedly retrace the same evolution during breaking. This has been attributed to the effect of mechanical annealing, which removes defects from the junction area and induces ordered stacking of the atoms near the contact. We speculate that for such annealed contacts a molecule present in the junction may also retrace the same conductance many times. The fact that this has not been reported before may be associated with the fact that L-MCBJ are usually operated by mechanical motor control, which is much less precisely repeatable than the piezo control, allowed for NW-MCBJ.[9]
The observations indicate that only a single molecule takes part in the process. Although the method of deposition leads to a high degree of coverage of molecules over the metal surface, we suspect that hard closing of the junction during cool down expels a large part of the molecules. This explains why several deep-indentation attempts are required for molecular signatures to re-appear at low temperatures.

[8] C. Sabater et al. In: *Phys. Rev. Lett.* 108 (2012), p. 205502.
[9] Christian A Martin et al. In: *New Journal of Physics* 10 (2008), p. 065008.