Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

Parallel Generation and Visualization of Bacterial Genome Structures

T. Klein[1] P. Mindek[1] L. Autin[2] D. S. Goodsell[2,3] A. J. Olson[2] E. M. Gröller[1,4] I. Viola[5]

[1] TU Wien, Austria [2] The Scripps Research Institute, USA [3] RCSB Protein Data Bank, Rutgers State University of New Jersey, USA
[4] VRVis Research Center, Austria [5] KAUST, Kingdom of Saudi Arabia

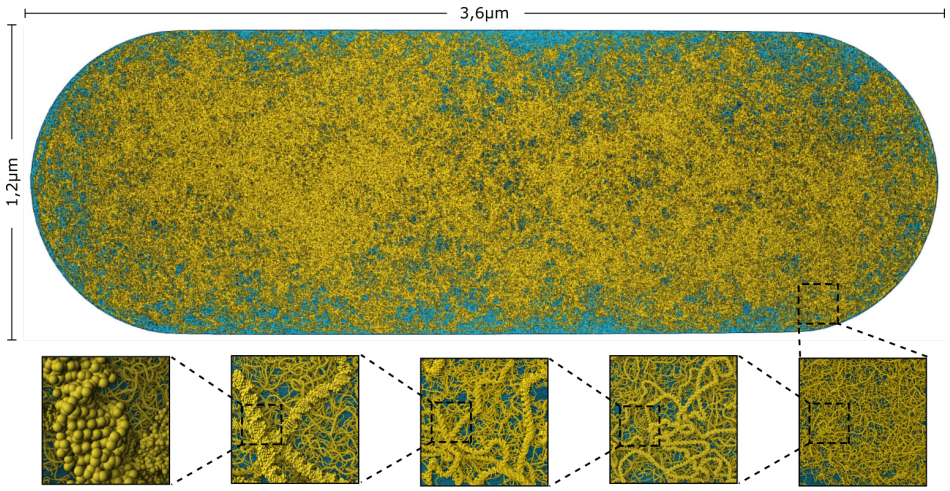


Figure 1: *Atomistic DNA rendering of the Sorangium cellulosum bacterium enclosed in a rod-shaped lipid membrane. The DNA structure is generated instantaneously using our new parallel modeling approach. Sorangium cellulosum currently holds the record for the largest known bacterial genome consisting of about 13 million base pairs.*

Abstract

Visualization of biological mesoscale models provides a glimpse at the inner workings of living cells. One of the most complex components of these models is DNA, which is of fundamental importance for all forms of life. Modeling the 3D structure of genomes has previously only been attempted by sequential approaches. We present the first parallel approach for the instant construction of DNA structures. Traditionally, such structures are generated with algorithms like random walk, which have inherent sequential constraints. These algorithms result in the desired structure, are easy to control, and simple to formulate. Their execution, however, is very time-consuming, as they are not designed to exploit parallelism. We propose an approach to parallelize the process, facilitating an implementation on the GPU.

1. Introduction

Visualization and computer graphics have shown much success in procedural modeling of nature, mostly through simulating images of objects at familiar, everyday scales. Non-biological natural phenomena, such as mountains, rock types, oceans, gases, clouds, or plasma, have been procedurally modeled for several decades [HRRG08]. Organic phenomena like vegetation, large forests [DN04], or dense jungles are modeled by procedural approaches and provide rich detail down to a single tree, or even leaves. The procedural generation of man-made structures is another example. A procedure can direct the generation of single room interior arrangements and can be scaled up to entire urban landscapes [SKK*14a; SKK*14b].

While in computer graphics, the goal is to generate and render visually plausible sceneries, in scientific visualization, the model generation and visualization has to be scientifically accurate, preserving key properties of the studied phenomenon. In biology, the generation and rendering of models of the biological mesoscale (the level between the nanoscale of atoms and the microscale of

cells) turned out to be a particular challenge. Such models are highly complex and heterogenous, and nanoscale objects cannot be directly observed.

The process of modelling mesoscale phenomena starts with understanding the hierarchical structure of living systems. At the finest level, atoms are specifically bonded to form large molecules, such as proteins and nucleic acids. They have characteristic structures and properties, and perform specific tasks. These molecules then interact and associate to form the microscale ultrastructure of the cell, which includes membrane-bounded compartments, and a variety of molecular infrastructure for regulation, support, transport, and communication. Reproducing the distinct shapes and interactions of these components, at scale ranges from nanometers to microns, is essential for creating accurate models of the structure and function of microorganisms.

Deoxyribonucleic acid (DNA) plays a central role in this hierarchy of structure and function. It is one of the most omnipresent structures representing and enabling life. DNA is a long polymer of four types of nucleotides, which carries the genetic information in the specific sequence of the nucleotides. Diverse experimental studies reveal many aspects of the DNA structure. This includes the detailed genetic sequence, topology and packing of the DNA into cells, and its interaction with the other molecular machinery of the cell. These results are making it possible to generate data-driven models of whole bacterial genomes, as shown in Figure 1.

Previous approaches generate such structures sequentially often by concatenating building blocks in many iterations. The main drawback of such procedures is the computation time, requiring minutes to hours to generate models of large-scale bacterial genomes. This limits the applicability for many visualization purposes, where parameters or constraints of the DNA structure are computed on the fly. Furthermore, biological studies that require numerous instances of such structures are limited to coarse-grained models, where the building blocks consist of several thousands of nucleotides. Our approach overcomes these limitations by generating DNA models instantaneously in a parallel fashion, similar to most real-time content generation procedures. Due to the complexity and constraints of the DNA structure, this is a non-trivial task.

Our central idea is to use a divide-and-conquer strategy together with a midpoint displacement algorithm for growing a DNA polymer. Initially, this procedure disregards any potential overlaps of the DNA structure to ignore sequential constraints for the moment. Subsequently, overlaps are efficiently detected and eliminated by a force-based system, similar to the ones used in fluid simulations. Our main contributions include:

- A problem characterization of generating large-scale DNA structures from the perspective of scientifically-accurate visualization
- Novel parallel algorithms for the instant generation of bacterial DNA structures
- An interactive visual environment allowing users to quickly experiment with different DNA generation parameters and constraints to perform *visual* hypothesis generation
- The foundation for *computational* hypothesis generation and verification that requires the generation of a multitude of bacterial genome models

2. Background and Related Work

In this section, we characterize architectural basics of bacterial genomes and review literature related to the generation and visualization of their structures.

Genome Architecture

The atomic details of DNA double helices were revealed in the classic work of Watson and Crick [DH53] and the bacterial nucleoid (the region containing most of the genetic material) was already described more than 60 years ago [Kel58]. However, structures of entire DNA genomes are still subject of intense study and only recently detailed information has become available [Dor13] through various experimental studies [DT16; KFS*14]. Bacterial cells typically have circular genomes, which solves an intrinsic problem with DNA replication: The enzyme DNA polymerase cannot replicate linear DNA to the end, so repeated rounds of replication lead to progressive shortening of linear DNA. Bacterial genomes are also typically highly supercoiled, leading to formation of a compactly-folded and functionally-organized structure [DT16]. Topoisomerase enzymes underwind the two strands of the DNA helix in an energy-dependent process. The underwound superhelical stress then leads to the formation of locally supercoiled loops termed 'plectonemes'. Supercoiling is generally thought to assist with the processes of DNA replication and transcription, which involve unwinding of the double helix. Supercoils have been characterized in plasmids (small circles of DNA) by electron microscopy [BWC90]. Higher-order structure of DNA is often probed using methods such as Hi-C (a technique to study the three-dimensional architecture of genomes). Hi-C uses selective crosslinking to identify regions of a genome that are in proximity. The flexibility of DNA remains a matter of conjecture. It is generally seen to be relatively rigid but there are abundant examples of kinking and bending under the influence of proteins. In general, the flexibility of polymers, like DNA, is quantified with a property called 'persistence length' defining how orientational correlations decay along the polymer chain. Studies indicate that DNA has a persistence length of around 500 Å(ngström) [Hag88]. Ångström is the unit in natural sciences for expressing the size of atoms, where 1 Å is equal to 10^{-10} m [PM10].

Generation and Visualization of DNA Structures

Early modeling and visualization of DNA structures used models built entirely from scratch. An example is the ground-breaking animated zoom from atoms to chromosomes by Max [Max85]. More recently, there have been several attempts at creating user-friendly tools, such as NAB [SJJ95] and GraphiteLifeExplorer [HF13]. These tools successfully model local details of DNA and its interaction with proteins, but are not scalable to model entire genomes. Modeling of entire genomes has been attempted by several groups. Coarse-grained techniques are often used, where segments of the DNA are represented by single beads. For bacteria these beads may represent single turns of the DNA helix [GAO18] or larger interaction domains [TYM*17]. For larger eukaryotic genomes, these beads often represent locally-compact domains of many thousand base pairs [RZ14]. Modeling of entire genomes at the atomic level has recently been attempted for bacteria, e.g., using a stepwise process with progressively-detailed coarse-graining [HLE17].

Many polymer-modeling methods produce only static models. In contrast, Kolesar et al. [KPV*14] propose a technique for modeling

the polymerization process itself. The work of Klein et al. [KAK*18] presents a novel technique for real-time generation and visualization of biological mesoscale models including fibrous DNA structures. The drawback of this method is its sequential implementation, which constitutes a severe performance bottleneck. Recent work by Lindow et al. [LBLH19] interactively visualizes RNA and DNA structures, but focuses on small-scale models with less than 10,000 nucleotides. The cellVIEW [MAPV15] system implements a novel approach for the assembling of DNA on the GPU. However, it requires an existing curve, approximated by discrete points, which are subsequently interpolated on the GPU.

Random Walk Methods

A random walk is a stochastic process used to model various real-world phenomena. Random walk algorithms often simulate movement and growth, or generate models of linear polymers like nucleic acids. A random walk is a special type of a Markov chain, where each generated point is dependent on its predecessor. We define the result of a random walk similar to Altendorf and Jeulin [AJ11] as a sequence P of points: $P = \{p_0, ..., p_n\}$ with $p_i = (x_i, \omega_i) \in \mathbb{R}^3 \times S^2$. Every point p_i consists of a location x_i and an orientation ω_i.

To model bacterial genomes, a correlated random walk with barriers is typically applied, where each point represents one bead of the genome model. The barrier is defined by a compartment, e.g. the cell nucleoid, that encapsulates the genome. The correlation is defined by the dependency of the orientations among successive steps. The random walk starts with an initial point $p_0 = (x_0, \omega_0)$, located inside of the compartment, and a random orientation. The position x_{i+1} of the next point p_{i+1} is generated through a random walk step, starting from the current position x_i along a new random orientation ω_{i+1}. It is calculated as follows:

$$x_{i+1} = x_i + s\omega_{i+1} \tag{1}$$

where $s \in \mathbb{R}^+$ is the step size of the random walk.

For more detail, we refer to the work of Spitzer [Spi01], which covers the theoretical foundation of random walks. Additionally, Codling et al. [CJB08] provide an extensive overview of random walks for modeling biological processes.

Midpoint Displacement Approaches

The power of parallel processors facilitates the generation of large models of bacterial genomes in real time. However, current genome modeling approaches, like random walks, have intrinsic sequential constraints and thus do not map well to parallel processors. Another approach to approximate various natural processes is midpoint displacement. In contrast to the random walk algorithm, it is well suited for parallelization. The midpoint displacement method has first been introduced by Mandelbrot [B M83] in the context of fractals. It became widely popular after Fournier et al. [FFC82] presented its extension to the diamond-square algorithm. The algorithm generates random heightmaps that are frequently used for terrain models. In the simple 1D case, the midpoint displacement algorithm starts with a line between two points. In the first step, the midpoint of the two initial points is displaced perpendicular to the line segment by a random amount. This process is repeated on the resulting new line segments until the desired level of detail is attained. The displacement magnitude is reduced in each iteration.

Midpoint displacement has been used and extended in various ways. Musgrave et al. [MM89] propose a midpoint displacement method for the generation of locally-controllable fractal terrains. They modify a standard midpoint displacement generation method by simulating erosion features to increase the realism of the generated terrains. Jilesen et al. [JKL12] expand the typical two-dimensional case of midpoint displacement to produce periodic, three-dimensional models of porous media. The method is suitable to create realistic models of rocks, as demonstrated through a comparison with two-dimensional cross-sections of real geological material. While most often used to model geological processes, the application of midpoint displacement in other areas is also well established. For instance, midpoint displacement has been used to simulate Brownian motion, which has various applications in biology [NML00].

3. Overview

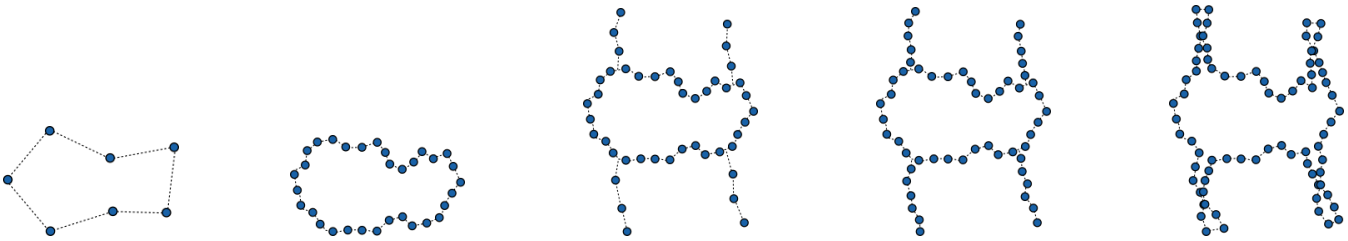
The complex hierarchical structure of DNA, ranging from specific nucleotide sequences at the atomic level to supercoiled topologies at the whole-genome level, poses challenges for modeling. Random walks and similar sequential algorithms are particularly suitable for generating models with shape and flexibility constraints, but are limited in their performance. This is especially problematic for large genome structures, like the genome of Sorangium cellulosum with about 13 million DNA base pairs. We introduce a divide-and-conquer approach, which makes it possible to parallelize the modeling process. Additionally, the real-time GPU implementation opens up the possibility of displaying models that are generally too large to fit into the memory. In the following we describe a typical sequential genome modeling approach and compare it to our parallel one.

Sequential Approach

Coarse-grained models of DNA, where a chain of beads is used to model the linear or circular polymer, may be constructed using a random walk. At each step the random walk chooses a random direction depending on the flexibility and shape of the surrounding compartment. The random walk starts with an initial seed point located inside of the compartment and walks along the first step. If a random walk step chooses a point outside of the given compartment or an occupied area, a new random direction is chosen. This process is repeated until a valid position is found. If no valid position can be found, the random walk retreats to an earlier state and tries again. When the coarse-grained chain of beads is computed, each bead may be exchanged with a more-detailed structural building block to assemble the final structure. Supercoils can be added by starting subsequent random walks from certain points of the generated structure. Other approaches [GAO18] start with a small square of points on a lattice and enlarge and migrate them to build the genome model. However, all of these sequential approaches require minutes up to hours to generate larger genome models.

Parallel Approach

Our parallel approach to generate bacterial genome models is illustrated in Figure 2. In order to map the computation of the model onto several threads of a parallel processor, an initial set of beads is required. For this reason, we first generate a rough backbone of the genome model. In theory, a randomly-placed circular set of beads could be used. However, in order to make this approach applicable to a variety of constraints and enclosing shapes, we choose to use



(a) *coarse-grained backbone* **(b)** *detailed backbone* **(c)** *coarse-grained supercoil axes* **(d)** *detailed supercoil axes* **(e)** *detailed supercoiled hairpins*

Figure 2: *Parallel approach for the generation of bacterial genomes. First, a sequential random walk with a large step size is applied to generated the main backbone (a), a coarse circular structure. The generated beads build the foundation for the subsequent parallel processing. In the next step, the circular structure is enhanced in detail (b) through our parallel midpoint displacement method. Subsequently coarse-grained supercoil axes (c) are generated with random walks branching from the backbone. Again, detail is added to the supercoils (d) with our parallel midpoint displacement method. Finally, superhelical hairpins (e) are generated by splitting the beads of the supercoil axes in two and rotating them resulting in the final model of the bacterial genome.*

a random walk to create the initial backbone (Figure 2a). Subsequently, the backbone is filled with detail (Figure 2b) through a parallel implementation of the midpoint displacement algorithm. For each pair of initial beads, the midpoint displacement algorithm computes a new midpoint and displaces it by a random amount, restricted by the constraints of the model. The midpoint displacement results in a sequence of beads with varying spacing. In order to fit equally-spaced building blocks of the polymer onto the beads, we need to perform a uniform resampling. Uniform resampling of a sequence is a sequential process, potentially slowing down the whole generation process. Therefore, we substitute uniform resampling with an approximation that can be executed in parallel. After the backbone is generated, supercoils are added to the structure. First, the coarse-grained versions of the supercoil axes are computed (Figure 2c), also using the random walk algorithm. Again, detail is added with the parallel midpoint displacement (Figure 2d) and the bead sequence is resampled. In the final step, the supercoiled hairpins are generated by splitting the beads of the supercoil axes into two chains and rotating them about the axes (Figure 2e). This is also computed in a parallel fashion. In order to generate an atomistic model of the bacterial genome, we calculate the orientation of structural units along the chain of beads in parallel and assemble them with an optimized version of the cellVIEW approach [MAPV15].

Applications

We expect that our parallel approach will be useful both in education/outreach settings and in research. For education, the interactive nature of the method is crucial. The ability to construct new models on the fly allows users to gain a more intuitive understanding of the hierarchical relationships between atoms and nucleotides, genes, entire genomes, and cells. The ability to explore different levels of superhelicity will help users to comprehend the role of DNA topology in the packing and function of the genome.

In research, there is a growing interest in the structure of bacterial genomes. It is becoming clear that supercoiling of DNA and subsequent condensation of chromosome interaction domains play a role in the regulation of gene expression [MLS17]. This has lead to the examination of multiple species using techniques such as Hi-C and fluorescence microscopy to quantify the location and interactions of domains within the genomic DNA. These techniques typically provide only a coarse-grained view, with resolutions of just thousands of base pairs. Therefore, modeling is employed to develop hypotheses for structural features at the finer scales.

Given the complexity of the system, most current modeling studies employ one of two simplifications. Coarse-grained modeling techniques support the creation and testing of multiple instances of multiple models, but at the cost of providing reduced-resolution results. Most often, these approaches treat individual chromosome interaction domains as the coarse-grained units, and provide useful information on the overall shape and packing of these domains in the interior space of the cell. More recently, several groups have created full atomic models of bacterial genomes, constrained by available biochemical observations. These models are laborious to produce, and thus typically only a handful of instances are generated. Nevertheless, they open up the ability to calculate detailed properties of the genome that will be important for understanding its interactions with regulatory proteins and the transcription machinery.

The ability to create detailed models in real time reduces the limitations of previous methods. For example, the simulation of Hi-C data requires the evaluation of contact information from many individuals in a population, recapitulating the experimental process of low-probability crosslinking in a culture of many cells. Rapid generation also facilitates the evaluation of multiple hypotheses. For example, the nature of each chromosome interaction domain is currently not known: It could be composed of a single supercoiled plectoneme, several tandem plectonemes, or a complex branched structure. The ability to rapidly build different models exploring multiple hypotheses, and for each, to build multiple stochastic instances of the model, will allow ready comparison to explore the nature of consistent structures. There are also direct connections to medical science that are becoming possible. Several classes of antibiotics, such as quinolones, attack topoisomerases and supercoiling. Studying the cellular consequences of these drugs in multiple bacterial physiological states (rapid growth, starvation, etc.) will help to identify aspects of the process that are amenable for continued antibiotic development.

4. Pipeline for Parallel Genome Generation

In the following, we describe the design of our parallel pipeline for the interactive generation of bacterial-genome models. Each step is designed to incorporate known characteristics and constraints of bacterial genomes.

4.1. Backbone Construction

In the initial step of the pipeline, we construct a rough backbone of the desired polymer structure with a random walk algorithm. The structural rigidity of DNA polymers is typically quantified with the persistence length. It defines the length over which the correlations of the tangents are lost. We use the following well-known equation [KP49] to incorporate the persistence length into the modeling process:

$$\langle \cos\theta \rangle = e^{-L/P} \tag{2}$$

The formula describes that the expectation value of the cosine of the angle θ between two tangents of the polymer chain falls off exponentially with distance L, where P denotes the persistence length. We use Equation 2 to approximate the orientations for the correlated random walk so that it reflects a given persistence length. First an angle θ_{max} is computed that indicates the maximum angle difference between the previous and the new orientation. Then we choose a new orientation using a random angle difference between 0° and θ_{max}; essentially, the computation of a new orientation is sampled from a spherical cap. The angle θ_{max} is defined as $acos(2e^{-L/P} - 1)$.

With the new orientation and Equation 1, the location of the next bead is computed. If a bead lies outside of a barrier, the algorithm retreats and uses a new orientation. This process is repeated until a valid position is found. The random walk algorithm stops when the given number of steps is reached. In order to support closed, circular DNA structures, we introduce an additional bias to the random walk with a direction towards the first initial bead. The bias grows with the length of the generated structure so that is has small to no influence at the beginning of the process.

To save computation time, the backbone is computed with a low resolution, which introduces gaps in between beads. Filling of the gaps increases the number of beads 2^n times, where n is the number of midpoint displacement steps. In practice, we use a maximum of ten midpoint displacement steps, which makes the backbone resolution 1024 times smaller than the resolution of the final sequence. However, using ten midpoint displacement steps is not a theoretical limit of the approach. It corresponds to a performance optimization of the algorithm that uses the shared memory of the GPU, which is limited in size.

4.2. Detail Insertion

Our version of the midpoint displacement approximates a given persistence length by utilizing a greedy approach. In every midpoint displacement step, we insert new beads into the chain and constrain the displacement using the given persistence. We show that the persistence length is efficiently approximated in Section 6.

For every pair of beads p_l and p_r with tangents $\vec{t_l}$ and $\vec{t_r}$, we introduce a new bead p_m using an adapted version of the midpoint displacement algorithm, as illustrated in Figure 3a. The amount of displacement is affected by the tangents and the persistence length. We define the midpoint displacement similar to the random walk algorithm. We start from bead p_l and walk in a direction $\vec{s_r}$ resulting in the displaced midpoint p_m. In order to choose an appropriate direction $\vec{s_r}$, first, a general direction $\vec{s_a}$ is calculated. The direction $\vec{s_a}$ represents a smooth continuation of the bead chain and is determined with the tangents $\vec{t_l}$ and $\vec{t_r}$. Then, the general direction is slightly varied resulting in the random direction $\vec{s_r}$.

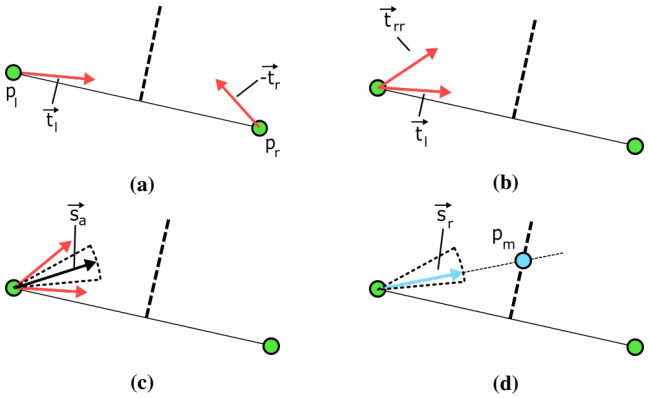


Figure 3: *Adapted midpoint displacement: (a) segment of the generated backbone with beads p_l and p_r and tangents $\vec{t_l}$ and $-\vec{t_r}$. (b) preparation step, where the tangent $-\vec{t_r}$ is reflected at the mid plane (dashed line). (c) depicts the bisector $\vec{s_a}$ of the two vectors $\vec{t_l}$ and $\vec{t_{rr}}$, which is used to find the new midpoint. (d) the midpoint p_m is computed through the intersection of a ray along the new direction $\vec{s_r}$ and the mid plane. The vector $\vec{s_r}$ is chosen randomly from a spherical cap surrounding $\vec{s_a}$.*

For the calculation of $\vec{s_a}$, we originate the direction at bead p_l and adjust the tangents accordingly. This means, we reflect $-\vec{t_r}$ across the mid plane (indicated with dashed lines) resulting in $\vec{t_{rr}}$, as shown in Figure 3b. The mid plane is defined by the center between p_l and p_r, and the normal $\vec{n} = \overrightarrow{p_l p_r} / ||\overrightarrow{p_l p_r}||$. With this the general direction is defined as $\vec{s_a} = (\vec{t_l} + \vec{t_{rr}}) / ||\vec{t_l} + \vec{t_{rr}}||$ (see Figure 3c). Subsequently, we choose a random direction $\vec{s_r}$ from the spherical cap surrounding $\vec{s_a}$, as depicted in Figure 3d.

Finally, the new midpoint p_m is computed through the intersection of a ray, starting at p_l with direction s_r, and the mid plane. In case the location of the new midpoint is outside of the compartment, we repeat the process with a new random direction s_r.

Uniform Resampling

The parallel midpoint displacement greatly improves the computation time, however, it does generate beads with non-uniform distances in between. To be able to fit equally sized building blocks on the beads, the bead chain needs to be uniformly resampled. Resampling is a sequential process, where each sampling step depends on its predecessor. To improve the performance of this process, we approximate uniform resampling so that it is suitable for parallel processing.

We utilize the idea of the divide and conquer design paradigm and first subdivide the sequence of beads into subsequences P_j. We de-

fine the subsequences as $P_j = \{p_{j(s-1)}, ..., p_{(j+1)(s-1)}\}$, where each subsequence contains s beads. Adjacent subsequences P_j, P_{j+1} have common border beads. Even though the uniform resampling within a subsequence is still a sequential process, the separate subsequences can now be processed in parallel.

Through the resampling, each subsequence P_j results in a resampled subsequence R_j. Within each resampled subsequence, the distances between the beads are uniform. However, the distances of adjacent resampled subsequences are still non-uniform. We quantify the offset between the desired uniform distance and the actual distance between adjacent resampled subsequences with the error vector $\vec{e_j}$. The vector $\vec{e_j}$ is defined as the distance between the first bead of a resampled subsequence R_j and the last bead of the non-resampled subsequence P_{j-1}. The error vector of the first subsequence is undefined, if the structure is not circular. If $\vec{r_j}$ is the last bead of the resampled subsequence R_j, then $\vec{e_j} = p_{j(s-1)} - r_{j-1}$. The maximum length of the error vector is always smaller than the sample distance. In order to minimize potential artifacts, we equalize the error among all beads of a subsqeuence. In detail, for a resampled subsequence R_j with indices $i = \{0, ..., n\}$ for its beads, we move each bead by $\frac{n-i}{n+1}\vec{e_j}$. Consequently, the last bead of a subsquence is not moved with the purpose of retaining the distance to the following subsequence.

Genome Length Estimation
In order to generate genome models with specific lengths, we utilize a heuristic approach. Due to the parallel midpoint displacement and the parallel uniform resampling the exact number of generated beads cannot be efficiently determined beforehand. For this reason, a heuristic is computed that estimates the resulting number of beads for a given parameter pair (random walk step size, persistence length). Experiments have shown that after several iterations of the algorithm, the number of generated beads for a specific parameter pair converges to a stable mean value. For example, an experiment with 500 repetitions of different seed values using a persistence length of 500 and 10,000 random walking steps results in a mean of $\mu = 327,863.95$ beads with a minimum of 324,550, maximum of 329,669, and standard deviation of $\sigma = 738.06$. For many visualization purposes the influence of such a variance is insignificant to the overall perception of the generated genome structure proofing the mean value to be a reliable estimator.

However, in some modeling scenarios the exact sequence of the genome is known and should be reflected by the model. Here, we slightly overestimate the length of the model and subsequently apply a trimming, to shorten it to the exact length by removing surplus beads. The removed beads create gaps in the structure, since its neighbors now have a larger distance. These gaps are automatically repaired in the subsequent relaxation step and the proper distances are restored.

4.3. Supercoiling

We model supercoils in three steps, as depicted in Figure 4. In the first step (a), a new sequence of beads (yellow) is generated that branches from the main backbone (blue) representing the supercoil axis. In the second step (b), the beads are replicated and shifted apart from each other. Finally, the beads are rotated around the supercoil axis (c) to generate the twist of the supercoil. The actual values for the twisting and shifting should be chosen corresponding to the characteristic of the desired genome structure.

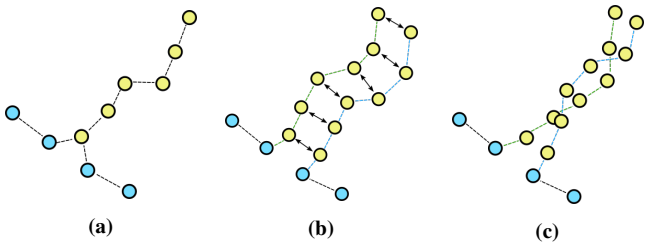


Figure 4: *In the initial step (a) of the supercoil generation, a sequence of beads (yellow) is generated that branches from the backbone (blue). Then (b) the generated beads are replicated and shifted apart from each other. In the last step (c), the beads are rotated to form the supercoil.*

4.4. Overlap Detection and Relaxation

Due to performance reasons, the generation of the backbone and the subsequent detail insertion do not take into account if space is already occupied. Performing space occupancy checks requires synchronization, and would thus introduce sequential constraints in the parallel process. Therefore, we efficiently resolve overlapping through a subsequent relaxation process. The relaxation is based on a force-based system. First, overlaps are detected through a fixed radius nearest neighbor search, as described by Hoetzlein [Hoe14]. This approach is capable of simulating millions of beads in real-time and is often used in fluid simulations. Overlaps are resolved with a combination of a *repulsion* and a *recover force*, based on the work of Altendorf and Jeulin [AJ11]. The *repulsion force* drives overlapping beads away from each other, whereas the *recover force* models a springlike force and recovers structural constraints. The constraints comprise the correct uniform distances between beads and a maximum amount of kinking in the structure.

The application of forces can potentially move parts of the structure outside of its enclosing compartment. For this reason, we apply an additional force that moves parts that now lie outside back into the compartment. The forces are applied until all collisions are resolved or a stop criterion is reached. This criterion can be set by the user and specifies a certain number of acceptable remaining collisions.

4.5. Genome Visualization

Coarse-grained models of bacterial genomes are sufficient for exploring hypotheses about overall packing and topology. However, to explore sequence-specific properties, such as the interaction with proteins, more detailed atomic information is needed, which is provided by finer models. To visualize the atomic structure of the genome, a model is required that shows the individual DNA base pairs and their proper orientations.

Normal Computation
In order to coherently orient the building blocks along the bead

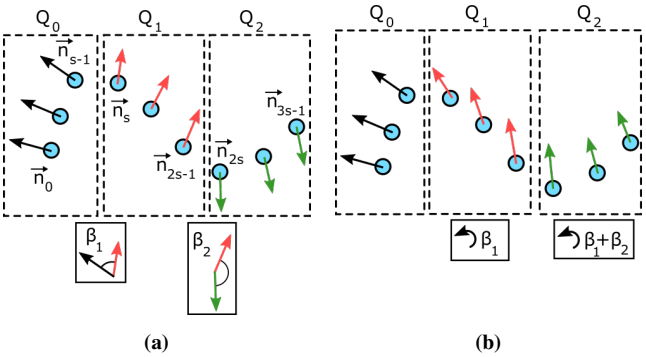


Figure 5: *Overview of the parallel normal computation. In the first step (a), the computation of the normals is parallelized by dividing the computation into several subsequences (indicated with dashed lines) to facilitate parallel processing. This leads to discontinuities at the borders between subsequences, which is quantified with the error angle β_i. Discontinuities are corrected by rotating all normals of a subsequence by the accumulated error angles (b).*

sequence, the orientation of the beads must be computed. This is typically done by computing the orthonormal frame for each bead.

In differential geometry there are many simple methods, which compute a moving frame. However, they often exhibit discontinuities or strong torsion causing visible artifacts. For this reason, we approximate a rotation minimizing frame. The computation of the rotation minimizing frame is sequential in nature since the computation of each frame is dependent on its predecessor. Similar to our parallel resampling approach, we apply the divide-and-conquer scheme and break the computation down into multiple subsequences (Figure 5). We define the subsequences as $Q_j = \{p_{js}, ..., p_{(j+1)s-1}\}$, where each subsequence contains s beads.

For each bead x_i, we compute a corresponding normal n_i. To start the process, we choose an initial random normal for each first bead of every subsequence and compute the remaining normals according to the rules of rotation minimizing frames. Since the normal computation is only continuous within a subsequences, the process leads to discontinuities in between subsequences. We quantify the discontinuity for a subsequence Q_j with the error angle β_j. In detail, the angle β_j is the angle between the last normal of subsequence Q_{j-1} and the first of subsequence Q_j. It is defined as $\beta_j = acos(n_{js} \cdot n_{js-1})$. In order to restore continuity between subsequences, we first apply a prefix sum to accumulate the error angles. Then, we rotate the normals by the accumulated error angles around their corresponding tangents. This means, the continuity is restored by rotating all normals of a subsequence Q_j by the accumulated error angle, which is defined as $\beta_j = \beta_{j-1} + ... + \beta_0$.

Genome Assembling

Once the normals are computed, the final polymer strand can be constructed. In this step, the beads are exchanged with corresponding building blocks and the final image is rendered. In the visualizations of this work, we use the standard model of the B-DNA double helix, which consists of individual base pairs with an angular offset of

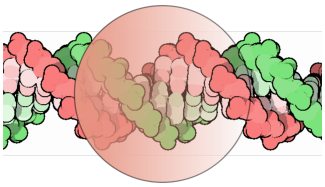


Figure 6: *Structural model of B-DNA overlayed with a bead representing a building block.*

34.3° and a spacing of 3.4 Å between each base, as depicted in Figure 6.

5. Implementation

For the reproducibility of the approach, we provide implementation details of our approach below.

Backbone Construction

In order to uniformly draw samples from a spherical cap we use the following equation: $a = \cos(\theta_{max})$, $z = X(1-a)+a$, $\psi = 2X\pi$, $r = \sqrt[2]{1-z^2}$, $\vec{\omega} = \{r\cos(\psi), r\sin(\psi), z\}$, where $X \sim U([0,1))$ is a random variable uniformly distributed between [0,1). This provides a random orientation ω with the maximal angle difference θ_{max} to the pole.

Detail Insertion

The midpoint displacement algorithm naturally lends itself to parallel processing since each level of repetition is independent from each other. Figure 7 illustrates the parallel implementation of the midpoint displacement algorithm. On the left side, each block represents a thread in the execution. Idle threads are depicted in blue and active ones in red. The right side depicts the beads, where the newly generated ones are shown in red, corresponding to the active threads. Horizontal lines indicate synchronization points between the individual iterations of the parallel midpoint displacement. In the first level, only one thread is active since only one midpoint can be created simultaneously. The number of active threads and generated beads doubles after each execution.

Uniform Resampling

To efficiently implement the parallel resampling, the resampled

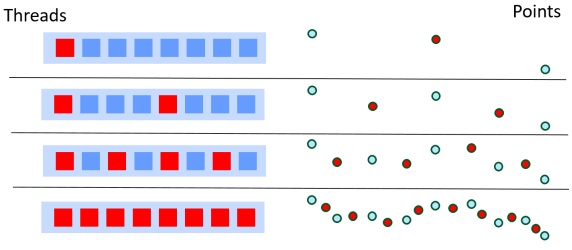


Figure 7: *Illustration of four iterations of our parallel midpoint displacement implementation. Every block (left) represents a thread and is depicted in red, if active, and blue, if inactive. The corresponding generated beads are also shown in red (right). The horizontal lines indicate the synchronization between consecutive iterations.*

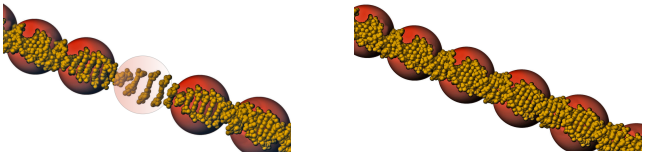


Figure 8: *Illustration of the trimming process. The beads (red) are overlayed with DNA building blocks. (a) shows the DNA structure with a trimmed bead (faded) before the relaxation. (b) shows the repaired DNA structure after the relaxation.*

beads are written in one continuous buffer of GPU memory. For this, we need to know beforehand the memory positions to which each thread has to write to. Therefore, we first compute the number of beads that each thread of the resampling will produce. Subsequently, we apply a prefix sum to the resulting values using the parallel scan algorithm [SHGO11]. With this information, we can resample the bead sequence in parallel.

Length Trimming

In case the exact length of the nucleic acid is known, we use a heuristic that slightly overestimates the resulting number of beads and trim surplus subsequently. To minimize the effect on the structure, we distribute the selection of the surplus beads uniformly along the structure. Figure 8 demonstrates one instance of the trimming process. Figure 8a shows a piece of a DNA structure, where one bead is trimmed. The illustrations show that the DNA structure would be stretched at the place where the bead is missing. In Figure 8b the hole in the structure is repaired due to the relaxation step and proper uniform distances are restored. In order to build a heuristic for various input parameters, we modeled a nonlinear regression using a set of sample results with a persistence length between 100 Å and 5000 Å, and 20 to 5000 random walk steps, whereas the remaining parameters were fixed. The nonlinear regression yields a goodness-of-fit of $R^2 = .998$ indicating a low discrepancy between observed and expected length.

Supercoiling

The supercoiling is parameterized with two values, the distance of the duplicated beads and the twisting factor that represents the level of supercoiling. These values are specified by the user and reflect

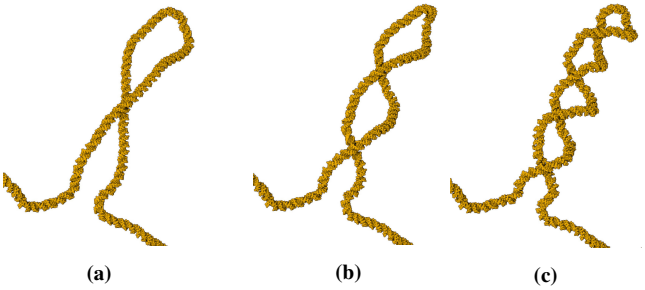


Figure 9: *Comparison of DNA with (a) low, (b) medium, and (c) high levels of supercoiling.*

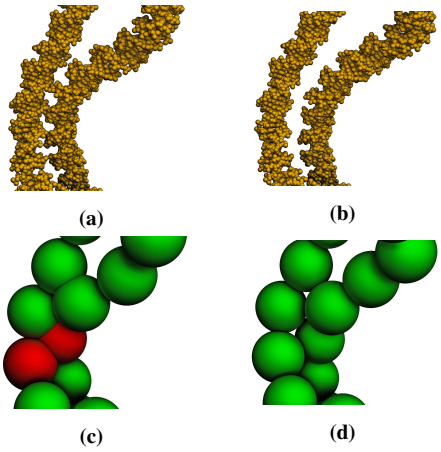


Figure 10: *Illustration of the overlap detection. (a) and (c) depict an occurrence of overlapping in atomistic detail and represented with beads, respectively. Beads highlighted in red indicate overlapping regions. (b) and (d) show the structure after the overlaps are resolved.*

results from experimental studies. In Figure 9, a visual comparison of DNA with different levels of supercoiling is shown.

Overlap Detection and Relaxation

The overlap detection and relaxation is processed on the basis of the beads. Each bead is modelled as a bounding sphere with the radius corresponding to its enclosing building block. The bead slightly overestimates the size of the DNA, as shown in Figure 6. However, the overestimation is a desired approximation from a perceptual point of view, since the elements are easier to distinguish if they are not in direct contact. Figure 10a and Figure 10c illustrate an instance of overlapping, shown with atomistic detail and shown with the corresponding beads, respectively. After the relaxation, shown in Figure 10b and Figure 10d, the overlapping is resolved and the two DNA strands have a distinct but small distance between them.

Normal Computation

Implementing the parallel computation of the normals consists of the following steps. In the first step, the tangents are computed for every bead. Then, the rotation minimizing frames are approximated. In our implementation, we utilize the concept proposed by Wang et al. [WJZL08]. In order to restore the continuity between neighboring subsequences, we first calculate the error angles and then accumulate them. To process the accumulation in parallel, we compute the prefix sum of the error angles using the parallel scan algorithm [SHGO11]. With the result of the prefix sum, we can also restore the continuity of the normals in parallel.

Genome Assembling

In order to assemble the DNA structure, we follow the approach of Le Muzic et al. [MAPV15], where the final assembling in atomistic detail is part of the rendering step. First, the bead sequence is subdivided into smaller beads, until each bead corresponds to a single DNA base pair. Afterwards, the normals for the newly introduced beads are interpolated on basis of the existing normals. In the same

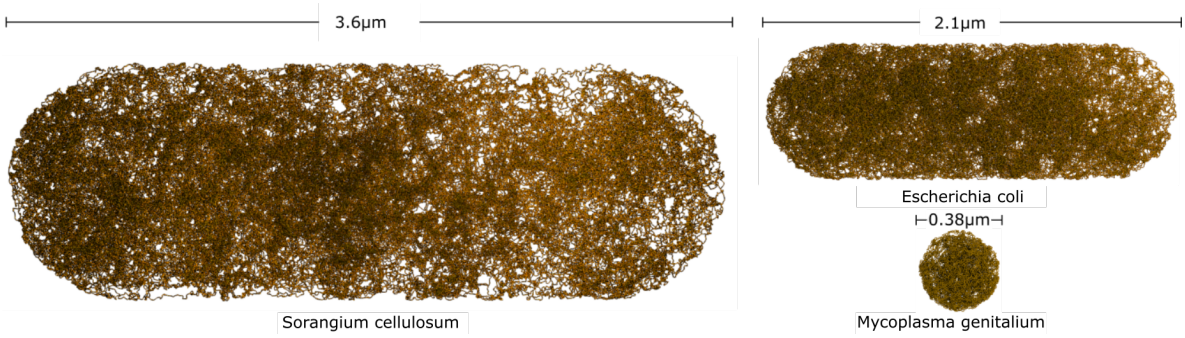


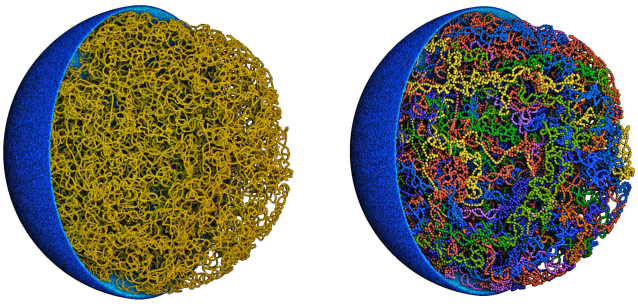
Figure 11: *Structural visualization of bacterial genomes. Left: Genome of Sorangium cellulosum, consisting of more than 13 million base pairs. Top right: Genome of Escherichia coli consisting of more than 4.6 million base pairs. Bottom right: Genome of a Mycoplasma genitalium consisting of more than 580,000 base pairs.*

step, every normal is rotated according to a given angular offset, which corresponds to the winding of the DNA. In the final step, each bead is substituted with the geometric model of the base pair.

6. Results and Evaluation

To evaluate the effectiveness of the method, we have created detailed data-driven models of the genomes of three bacteria. The performance was measured using a computer with an Intel Core i7-6700K CPU 4.00 GHz and a NVIDIA GeForce GTX 1080 graphics card with 8 GB memory. Since the final assembling of the genome model is part of the rendering process that was already presented in previous work [MAPV15], we do not include this step in the performance measurements.

Mycoplasma genitalium
The genome of Mycoplasma genitalium, with **580,000** base pairs (bp), is among the smallest of bacterial genomes. Figure 11 (bottom right) shows the rendering of the genome model, generated in 218 ms. Previous work [KAK*18] required about 100 seconds to generate the genome without including any complex supercoiling structures. Recent work [GAO18] that includes supercoils generate an equivalent model in about 9.7 minutes, thus we outperform the result with the same scientifically-accurate complexity 2600-fold. Figure 12 reveals a closer view of the structure of the genome enclosed in a spherical lipid membrane. The left side (a) shows the atomistic rendering where the right side (b) displays the model with beads. Individual genes are displayed with different colors.



(a) **(b)**

Figure 12: *Model of Mycoplasma genitalium. (a) shows the atomistic rendering of the DNA enclosed in a lipid membrane. (b) shows the individual genes of the model indicated with different colors.*

Escherichia coli
Escherichia coli is one of the best characterized organisms, with

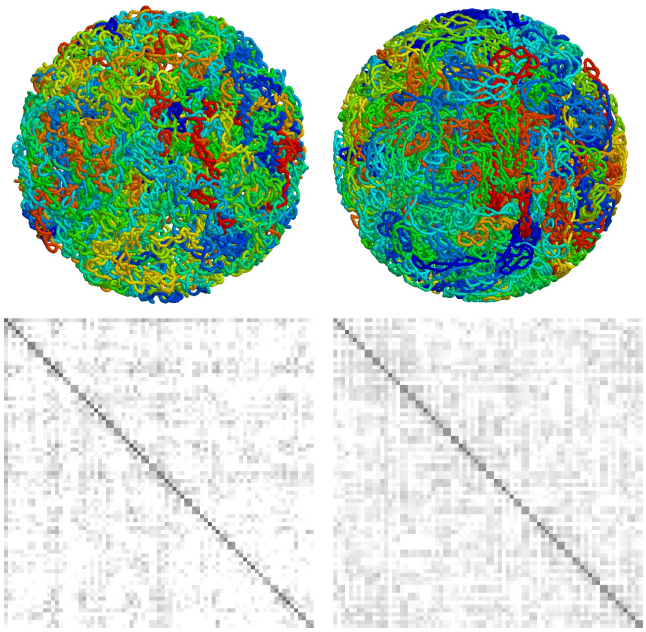


Figure 13: *Models of the Mycoplasma pneumonia genome, generated with (left) our parallel generation approach and (right) the lattice-based method [GAO18]. In the distance maps at the bottom, the entire genome is arrayed on the horizontal and vertical axes with a resolution of 10 kpb, and average distances are shown with a linear ramp from black at 0 nm to white at 200 nm and above. A representative genome model for each approach is shown at the top.*

Table 1: *Performance measurements of the generation of several bacterial genomes, separated into the different steps of the generation process.*

Model	Backbone	Detail	Supercoiling	Resampling	Trimming	Relaxation	Normals	Total
Mycoplasma genitalium	1 ms	11 ms	5 ms	35 ms	6 ms	150 ms	10 ms	218 ms
Escherichia coli	6 ms	39 ms	15 ms	136 ms	36 ms	112 ms	28 ms	372 ms
Sorangium cellulosum	19 ms	86 ms	26 ms	282 ms	88 ms	360 ms	81 ms	942 ms

a genome of about **4.6 million** bp. As in previous work [GAO18], we have separated the genome into 169 units. Each unit consists of a 11,000 bp long plectoneme and an unsupercoiled connecting segment of 16,000 bp. The membrane of the cell is rod-shaped with an approximate diameter of 0.7 micron and length of 2.1 micron, shown in Figure 11 (top right). The model was generated in 372 ms.

Sorangium cellulosum
Sorangium cellulosum currently holds the record for the largest known bacterial genome at about **13 million** bp [SPK*07], shown in Figure 11 (left). The genome is split into 482 units similar to Escherichia coli. Each unit consists of a 10,000 bp plectoneme and a 16,000 bp unsupercoiled connecting segment. It is enclosed by a rod-shaped membrane, with a diameter of 1.2 micron and 3.6 microns in length. The model was generated in 942 ms.

A more detailed investigation of the runtime, shown in Table 1, displays the individual timings for each generation step. It is interesting to note that the relaxation step for the genome model of the Mycoplasma genitalium requires more time than the one for the larger Escherichia coli. This is due to the fact that the Mycoplasma genitalium model is more crowded, thus making the relaxation more complex. The bottleneck of the performance resides in the relaxation and resampling steps. In order to resolve overlapping beads, the relaxation requires several iterations. For instance, resolving the overlaps for Mycoplasma genitalium models takes around 70 iterations, for Escherichia coli models around 32 iterations, and for Sorangium cellulosum models around 21 iterations. However, it is likely that a thorough parameter fine tuning of the force-based system potentially leads to even faster resolving timings. Our implementation also offers the possibility to run the relaxation progressively during the visualization to mitigate its performance impact. The other bottleneck is the resampling. We have reduced the sequential constraints of the resampling through a divide-and-conquer scheme, where we divide the sequence of beads into subsequences. However, the computation of each subsequence is still processed sequentially, making this the slowest step of the pipeline.

A direct comparison of our parallel generation approach with the previous lattice-based method [GAO18] is shown in Figure 13. The comparison is based on a genome model of Mycoplasma pneumonia. Ten instances of a single circular genome with 44 repeating units of a 17,000 bp plectoneme and a 1000 bp connecting region where modeled, within a spherical space with diameter 380 nm. Distance maps were computed based on the average distance between 10 kbp segments. For both, our parallel approach and the lattice-based method, these maps show the characteristic diamond-shaped features along the diagonal that are observed in Hi-C experiments [TYM*17]. They are a consequence of the proximity of chains within superhelical plectonemes. The weak signal in the off-diagonal area shows that our parallel approach produces models, where the DNA is distributed randomly through space, rather than in local regions.

In order to evaluate the stiffness of our fiber generation approach we compare it to the idealistic model given by Equation 2. We have generated and measured several genome structures of different persistence lengths. Figure 14 shows the corresponding curves for the expected correlation (orange) and the measured correlation (blue).

The constantly changing domain knowledge that influences the generation process of bacterial genomes renders it difficult to compare our parallel approach more explicitly to existing sequential approaches. For this reason, we conducted a simplified comparison of the sequential generation of polymers with our parallel approach, as shown in Figure 15. In the comparison, we generated simple linear polymer structures without supercoils, which were only constrained by the persistence length and the shape of an enclosing compartment. Our parallel approach shows a significant performance improvement starting with 100,000 beads rendering it useful for many applications of instant polymer modeling.

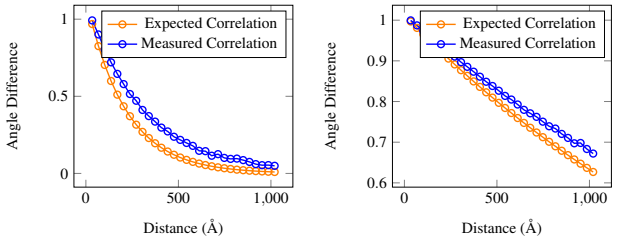


Figure 14: *Comparison of the expected stiffness (orange) and the measured stiffness (blue) of the genome generation approach. The plot on the left shows the curve for a persistence length of* $1,000$ *Å, whereas the one on the right side uses* $10,000$ *Å.*

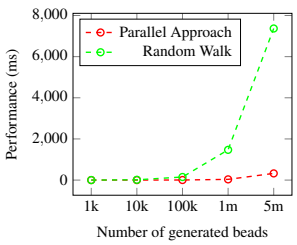


Figure 15: *Performance measurements for random walk and parallel midpoint displacement. Starting at 100,000 beads, the parallel approach indicates a significant performance speedup.*

7. Conclusion and Future Work

We have presented a new method for interactively constructing models of entire bacterial genomes, and validated its application on three bacteria that span the range of natural genome complexity. Looking to the future, there are many enhancements that will need to be approached. Modeling of transcription complexes (RNA polymerase, RNA, co-transcriptional ribosomes) will allow to study in detail the consequences of the genome structure on gene expression. As we move to more complex bacteria and to eukaryotes, methods for including DNA-binding proteins will be essential, since they play a central role in compacting, organizing, and regulating the genetic information. Detailed atomic structures and physicochemical properties are known for most of these additional molecules, so we are optimistic about future enhancement of the method.

ACKNOWLEDGMENTS

This work was funded under the ILLVISATION grant by WWTF (VRG11-010). It is based upon work supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research (OSR) under Award No. OSR-2019-CPF-4108 and BAS/1/1680-01-01. The Scripps Research Institute researchers acknowledge support from the National Institutes of Health under the grant R01-GM120604. This paper was partly written in collaboration with the VRVis Competence Center. VRVis is funded by BMVIT, BMWFW, Styria, SFG and Vienna Business Agency in the scope of COMET - Competence Centers for Excellent Technologies (854174), which is managed by FFG. The authors would like to thank Nanographics GmbH (nanographics.at) for providing the Marion Software Framework.

References

[AJ11] ALTENDORF, HELLEN and JEULIN, DOMINIQUE. "Random-walk-based stochastic modeling of three-dimensional fiber systems". *Phys. Rev. E* 83 (4 2011), 041804 3, 6.

[B M83] B. MANDELBROT, BENOIT. "The Fractal Geometry of Nature". Vol. 51. 1983, 468 p. ISBN: 0716711869 3.

[BWC90] BOLES, T. CHRISTIAN, WHITE, JAMES H., and COZZARELLI, NICHOLAS R. "Structure of plectonemically supercoiled DNA". *Journal of Molecular Biology* 213.4 (1990), 931–951. ISSN: 0022-2836 2.

[CJB08] CODLING, EDWARD, J PLANK, MICHAEL, and BENHAMOU, SIMON. "Random walks in biology". *Journal of the Royal Society, Interface / the Royal Society* 5 (2008), 813–34 3.

[DH53] D WATSON, JAMES and H CRICK, FRANCIS. "Molecular Structure of Nucleic Acids: A Structure for Deoxyribose Nucleic Acid". *The American journal of psychiatry* 160 (1953), 623–4 2.

[DN04] DECAUDIN, PHILIPPE and NEYRET, FABRICE. "Rendering Forest Scenes in Real-time". *Proceedings of the Fifteenth Eurographics Conference on Rendering Techniques*. EGSR'04. Eurographics Association, 2004, 93–102. ISBN: 3-905673-12-6 1.

[Dor13] DORMAN, CHARLES. "Genome architecture and global gene regulation in bacteria: Making progress towards a unified model?": *Nature reviews. Microbiology* 11 (2013) 2.

[DT16] DAME, REMUS T and TARK-DAME, MARILIIS. "Bacterial chromatin: converging views at different scales". *Current Opinion in Cell Biology* 40 (2016), 60–65. ISSN: 0955-0674 2.

[FFC82] FOURNIER, ALAIN, FUSSELL, DON, and CARPENTER, LOREN. "Computer Rendering of Stochastic Models". *Commun. ACM* 25.6 (1982), 371–384. ISSN: 0001-0782 3.

[GAO18] GOODSELL, DAVID S., AUTIN, LUDOVIC, and OLSON, ARTHUR J. "Lattice Models of Bacterial Nucleoids". *The Journal of Physical Chemistry B* 122.21 (2018), 5441–5447 2, 3, 9, 10.

[Hag88] HAGERMAN, PAUL J. "Flexibility of DNA". *Annual Review of Biophysics and Biophysical Chemistry* 17.1 (1988), 265–286 2.

[HF13] HORNUS, SAMUEL and FOURMENTIN, ERIC. "Easy DNA Modeling and More with GraphiteLifeExplorer, in "PLoS ONE". *doi : 10.1371/JOURNAL.PONE.0053609], http://hal. inria.fr/hal-00924190*. 2013 2.

[HLE17] HACKER, WILLIAM C., LI, SHUXIANG, and ELCOCK, ADRIAN H. "Features of genomic organization in a nucleotide-resolution molecular model of the Escherichia coli chromosome". *Nucleic Acids Research* 45.13 (2017), 7541–7554 2.

[Hoe14] HOETZLEIN, RAMA. "Fast Fixed-Radius Nearest Neighbors: Interactive Million-Particle Fluids". *GPU Technology Conference (GTC) 2014*. 2014 6.

[HRRG08] HAN, CHARLES, RISSER, ERIC, RAMAMOORTHI, RAVI, and GRINSPUN, EITAN. "Multiscale Texture Synthesis". *ACM Trans. Graph.* 27.3 (2008), 51:1–51:8. ISSN: 0730-0301 1.

[JKL12] JILESEN, JONATHAN, KUO, JIM, and LIEN, FUE-SANG. "Three-dimensional midpoint displacement algorithm for the generation of fractal porous media". *Computers & Geosciences* 46 (2012), 164–173. ISSN: 0098-3004 3.

[KAK*18] KLEIN, TOBIAS, AUTIN, LUDOVIC, KOZLÍKOVÁ, BARBORA, et al. "Instant Construction and Visualization of Crowded Biological Environments". *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), 862–872. ISSN: 1077-2626 3, 9.

[Kel58] KELLENBERGER, EDOUARD. "Electron Microscope Study of DNA-Containing Plasms: II. Vegetative and Mature Phage DNA as Compared with Normal Bacterial Nucleoids in Different Physiological States". *The Journal of Cell Biology* 4 (1958), 671–678 2.

[KFS*14] KLECKNER, NANCY, FISHER, JAY K., STOUF, MATHIEU, et al. "The Bacterial Nucleoid: Nature, Dynamics and Sister Segregation". *Current Opinion in Microbiology* 22 (2014) 2.

[KP49] KRATKY, OTTO and POROD, GÜNTHER. "Röntgenuntersuchung gelöster Fadenmoleküle". *Recueil des Travaux Chimiques des Pays-Bas* 68.12 (1949), 1106–1122. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/recl.19490681203 5.

[KPV*14] KOLESAR, IVAN, PARULEK, JULIUS, VIOLA, IVAN, et al. "Interactively illustrating polymerization using three-level model fusion". *BMC Bioinformatics* 15.1 (2014), 345. ISSN: 1471-2105 2.

[LBLH19] LINDOW, NORBERT, BAUM, DANIEL, LEBORGNE, MORGAN, and HEGE, HANS-CHRISTIAN. "Interactive Visualization of RNA and DNA Structures". *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2019), 967–976. ISSN: 1077-2626 3.

[MAPV15] MUZIC, MATHIEU LE, AUTIN, LUDOVIC, PARULEK, JULIUS, and VIOLA, IVAN. "cellVIEW: a Tool for Illustrative and Multi-Scale Rendering of Large Biomolecular Datasets". *Eurographics Workshop on Visual Computing for Biology and Medicine*. EG Digital Library. The Eurographics Association, 2015, 61–70. ISBN: 978-3-905674-82-8 3, 4, 8, 9.

[Max85] MAX, NELSON. "DNA Animation from atom to chromosome". *Journal of Molecular Graphics* 3.2 (1985), 69–71. ISSN: 0263-7855 2.

[MLS17] MIRAVET-VERDE, SAMUEL, LLORÉNS-RICO, VERÓNICA, and SERRANO, LUIS. "Alternative transcriptional regulation in genome-reduced bacteria". *Current Opinion in Microbiology* 39 (2017), 89–95. ISSN: 1369-5274 4.

[MM89] MUSGRAVE Forest K.and Kolb, CRAIG E. and MACE, ROBERT S. "The Synthesis and Rendering of Eroded Fractal Terrains". *SIGGRAPH Comput. Graph.* 23.3 (1989), 41–50. ISSN: 0097-8930 3.

[NML00] NORROS, ILKKA, MANNERSALO, PETTERI, and L. WANG, JONATHAN. "Simulation of Fractional Brownian Motion with Conditionalized Random Midpoint Displacement". *Advances in Performance Analysis* 2 (2000) 3.

[PM10] PETERS, JUSTIN P. and MAHER, L. JAMES. "DNA curvature and flexibility in vitro and in vivo". *Quarterly reviews of biophysics* 43 (2010), 23–63 2.

[RZ14] ROSA, ANGELO and ZIMMER, CHRISTOPHE. "Chapter Nine - Computational Models of Large-Scale Genome Architecture". *New Models of the Cell Nucleus: Crowding, Entropic Forces, Phase Separation, and Fractals*. Ed. by HANCOCK, RONALD and JEON, KWANG W. Vol. 307. International Review of Cell and Molecular Biology. Academic Press, 2014, 275–349 2.

[SHGO11] SENGUPTA, SHUBHABRATA, HARRIS, MARK, GARLAND, MICHAEL, and OWENS, JOHN. "Efficient Parallel Scan Algorithms for GPUs". 2011, 413–442. DOI: 10.1201/b10376-29 8.

[SJJ95] S. DUNCAN, BRUCE, J. MACKE, TOM, and J. OLSON, ARTHUR. "Biomolecular visualization using AVS". *Journal of molecular graphics* 13 (1995), 271–82, 299 2.

[SKK*14a] STEINBERGER, MARKUS, KENZEL, MICHAEL, KAINZ, BERNHARD, et al. "On-the-fly Generation and Rendering of Infinite Cities on the GPU". *Computer Graphics Forum* 33.2 (2014), 105–114. ISSN: 0167-7055 1.

[SKK*14b] STEINBERGER, MARKUS, KENZEL, MICHAEL, KAINZ, BERNHARD, et al. "Parallel Generation of Architecture on the GPU". *Comput. Graph. Forum* 33.2 (2014), 73–82. ISSN: 0167-7055 1.

[Spi01] SPITZER, FRANK. *Principles of Random Walk*. Graduate texts in mathematics. Springer, 2001. ISBN: 9780387951546 3.

[SPK*07] SCHNEIKER, SUSANNE, PERLOVA, OLENA, KAISER, OLAF, et al. "Complete genome sequence of the myxobacterium Sorangium cellulosum". *Nature biotechnology* 25 (2007), 1281–9 10.

[TYM*17] TRUSSART, MARIE, YUS, EVA, MARTINEZ, SIBOLESRA, et al. "Defined chromosome structure in the genome-reduced bacterium Mycoplasma pneumoniae". English (US). *Nature Communications* 8 (2017). ISSN: 2041-1723 2, 10.

[WJZL08] WANG, WENPING, JÜTTLER, BERT, ZHENG, DAYUE, and LIU, YANG. "Computation of Rotation Minimizing Frames". *ACM Trans. Graph.* 27.1 (2008), 2:1–2:18. ISSN: 0730-0301 8.