Computers & Graphics (2025)

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SynopFrame: Multiscale time-dependent visual abstraction framework for analyzing DNA nanotechnology simulations

Deng Luo[a,*], Alexandre Kouyoumdjian[a], Ondřej Strnad[a], Haichao Miao[b], Ivan Barišić[c], Tobias Isenberg[d], Ivan Viola[a,*]

[a] King Abdullah University of Science and Technology (KAUST), Thuwal, Saudi Arabia
[b] Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, USA
[c] Health and Bioresources, Austrian Institute of Technology, Vienna, Austria
[d] Université Paris-Saclay, CNRS, Inria, LISN, Orsay, France

ARTICLE INFO

Article history:
Received August 28, 2025

Keywords: DNA nanotechnology, temporal data, visual abstraction, abstraction space, illustrative visualization

ABSTRACT

We present an open-source framework, SynopFrame, that allows DNA nanotechnology (DNA-nano) experts to analyze and understand molecular dynamics simulation trajectories of their designs. We use a multiscale multi-dimensional abstraction space, connect the representations to a projected conformational space plot of the structure's temporal sequence, and thus enable experts to analyze the dynamics of their structural designs and, specifically, failure cases of the assembly. In addition, our time-dependent abstraction representation allows the biologists, for the first time in a smooth and structurally clear way, to identify and observe temporal transitions of a DNA-nano design from one configuration to another, and to highlight important periods of the simulation for further analysis. We realize SynopFrame as a dashboard of the different synchronized 3D spatial and 2D schematic visual representations, with a color overlay to show essential properties such as the status of hydrogen bonds. The linking of the spatial, schematic, and abstract views ensures that users can effectively analyze the high-frequency motion. We also categorize the status of the hydrogen bonds into a new format to allow us to color-encode it and overlay it on the representations. To demonstrate the utility of SynopFrame, we describe example usage scenarios and report user feedback.

1. Introduction

Recent years have seen a series of breakthroughs in the DNA nanotechnology (DNA-nano) domain [29, 30], with a technique known as *DNA origami* [25] bringing dramatic success on various designs and use cases. A hallmark of DNA origami designs is their structure, constructed from one long scaffold strand that is folded by many short staple strands. The complexity of these designs, which is often characterized by the structure's huge size, the DNA strands' complicated routing, and the high-frequency motion in their dynamics can lead to difficulties for experts in designing and analyzing them. This challenge becomes even larger when scientists rely on molecular dynamics simulations (MDS) to examine the behavior of DNA-nano structures at nanoscale resolution, where standard visual inspection methods can be overwhelmed by the dynamic complexity and large data size.

DNA-nano design and simulation tools, e. g., caDNAno [8], Adenita [7], CATANA [15], oxView [24], and oxDNA [32], provide semi-automatic workflows to lower these difficulties. Domain experts can author designs in high-level geometries (lines, squares, honeycomb), generate the DNA sequences for

*Corresponding author: Tel.: +966-12-808-0617;
e-mail: deng.luo@kaust.edu.sa (Deng Luo, 罗登), alexandre.kouyoumdjian@kaust.edu.sa (Alexandre Kouyoumdjian), ondrej.strnad@kaust.edu.sa (Ondřej Strnad), miao1@llnl.gov (Haichao Miao), ivan.barisic@ait.ac.at (Ivan Barišić), given_name.family_name@inria.fr (Tobias Isenberg), ivan.viola@kaust.edu.sa (Ivan Viola)

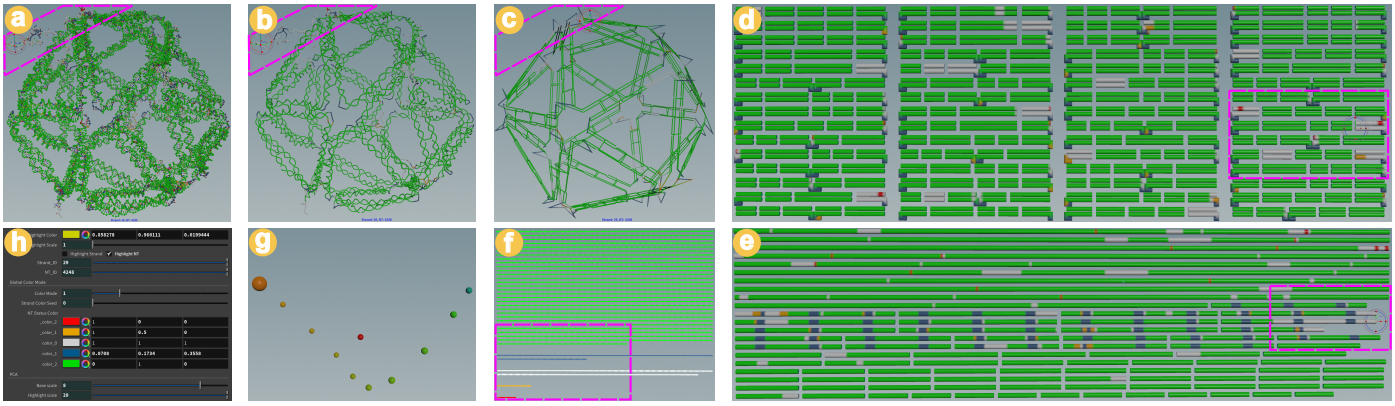


Fig. 1. SynopFrame dashboard shows an icosahedron design (6540 NTs) in various different representations (arranged in a clockwise fashion): (a) All-NT, (b) Snake, (c) Schematic3D, (d) Schematic2D, (e) Heatbar, (f) Progress bar, (g) Principal component analysis plot, and (h) Control panel.

each strand, and then make modifications if necessary, followed by feeding the designs to molecular dynamics simulation (MDS) tools for further analysis.

Yet, experts still have to examine the structure carefully to envision the resulting design and to mentally connect the designed shape, the strands, and the sequences with the findings from the simulations. Visualizing the design can help experts inspect its structure directly. These designs, however, typically contain tens of thousands of nucleotides (NTs) on hundreds of strands. Some of the staple strands are "high-degree" strands that pair with several parts of the scaffold strand to form complicated folding patterns. Conventional visualizations of all the NTs in a design in turn generally produce cluttered and occluded views. In the past, Miao *et al.* [20] had solved some of these problems with abstract views, yet only for static structures rather than for dynamic simulations.

In our work we address the problem of efficiently interpreting and visually analyzing large-scale MDS trajectories for DNA-nano designs. We propose SynopFrame, a multi-viewport, multiscale, multi-dimensional, time-dependent, and comprehensive visual abstraction framework that aims to help experts identify and interpret the dynamic evolution of their DNA-nano structures. Given an MDS trajectory of a DNA-nano design, our solution offers interactive, synchronized viewports for both detailed and abstract representations as we show in Fig. 1. This approach enables users to: (1) navigate and compare the overall structural progress in relation to the designed shape, (2) identify problem regions via color-coded H-bond status, and (3) focus on specific areas for deeper inspection of local pairing events.

Overall, our contributions are threefold:

- we introduce an abstraction space that extends existing representations for DNA-nano structures and bridges design and MDS analysis;
- we provide a new way to categorize and encode H-bond status, enabling the quick identification of design flaws and conformational changes; and
- we develop a synchronized multi-view environment that links different abstraction levels (from detailed 3D shapes to schematic progress bars), helping experts effectively explore and interpret dynamic simulation data.

Our user feedback indicates that these contributions can substantially aid in explaining and troubleshooting unsuccessful self-assembly in DNA-nano designs.

2. Background

DNA-nano structure uses DNA in a novel way, different from its genetic context. By leveraging the H-bond [10] formed according to the Watson-Crick base pairing rule, DNA-nano designs lead to 3D geometric shapes at nanoscale, consisting of DNA strands with carefully prepared NT sequences. Due to the programmable nature of the NT sequences and the versatile applications of tiny shapes, DNA-nano soon became an active research area [30]. To date, three methods have been developed: *DNA origami*, *single-stranded tile*, and *multi-stranded tile* [40]. Among them, DNA origami poses the most visualization challenges due to its long scaffold strand that is folded by many short staple strands. While we thus focus on DNA origami, our approach works for the other two methods as well as we explain below. Here we provide a background of DNA-nano structure design, and discuss simulations in Appendix B.

In Fig. 2 we show, bottom-up, the creation of a DNA origami structure. First, there are four different NTs in the DNA (commonly abbreviated as *A, T, C, G*), which are also used in DNA origami. They consist of a phosphate group, a sugar molecule, and a nitrogen-containing base (Fig. 2a). These NTs are chemically synthesized and actively assembled into specific DNA *strands*, in which they covalently bind in a sequential manner (Fig. 2b) and which can be simplified as polylines (Fig. 2c). Each DNA sequence has a directionality that is defined by its 3' and 5' ends. In DNA-nano, if there is no specific labeling, by convention a sequence starts at the 3' end (Fig. 2b shows a sequence of 3'–ACTCGTG–5'). The Watson-Crick rule specifies that two *H-bonds* can be formed between A and T and three H-bonds between C and G. If the NT sequences of two DNA strands complement each other, meaning that they form Watson-Crick *base pairs* for all the NTs, then a double helix (Fig. 2d) forms and the H-bonds (vertical blue lines) in-between stabilize the helix structure. DNA-nano experts use this feature to design a long *scaffold* strand and many short *staple* strands (Fig. 2e) to form a designed shape (Fig. 2f) once those strands bind with each other via H-bonds. Most staples bind to two distant regions on the

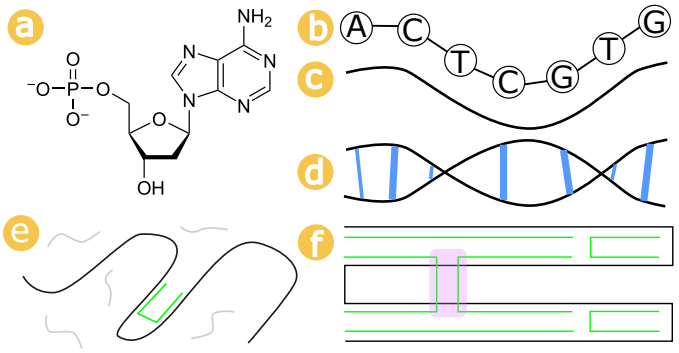


Fig. 2. DNA-nano concept: (a) an NT consists of atoms (image ⓒⓘ); (b) covalently connected NTs form a DNA strand; (c) a DNA strand can further conceptually be simplified into a polyline; (d) complementary strands form a double helix, stabilized by hydrogen bonds (vertical blue lines; as in regular, genetic DNA; thick lines: 3 H-bonds in CG pairs, narrow lines: 2 H-bonds in AT pairs); (e) a long *scaffold* strand (black) is then folded by short *staple* strands (green: bonded, gray: unbonded); (f) the overall 3D structure emerges as the staples bind to the scaffold according to the designed binding pattern (purple-shaded part: *crossover*). Note that staple strands are not created as U-shaped staples (green) but as ordinary strands (gray). As two parts of the short staples bind to different scaffold points as designed, the short strand then folds and effectively "staples" the scaffold.

scaffold, to *fold* it. Points where two staples fold the scaffold at the same place are called *crossover* (purple in Fig. 2f).

The other DNA-nano techniques, single-stranded tile and multi-stranded tile, pose a subset of DNA origami visualization challenges as they essentially split the long scaffold strand in DNA origami into short pieces and then manipulate the strand routing and sequences.

3. Related work

Beyond this background, our work relates to the visualization of molecular dynamics, the modeling and visualization of DNA nanotechnology, and visual abstraction as we discuss next.

3.1. Molecular dynamics visualization

To visualize the dynamic behavior of molecules, it is important to represent the trajectories resulting from the molecular dynamics simulation. Early work on MDS resulted in the VMD tool [12] that simulates and visualizes molecular dynamics for proteins and nucleic acids. Byška *et al.* [6] visualized protein tunnels by representing the path of each amino acid in the tunnel and aggregating the trajectories into profiles. Kolesár *et al.* [14] proposed a three-level system for illustrating the process of polymerization, coupling together an L-system with agent-based simulation and quantitative simulation techniques. Later, Kolesár *et al.* [13] proposed a way to rectify the simulated data to allow for comparative visualization of a cohort. A more general approach to particle-based spatiotemporal data visualization was proposed by Pálenik *et al.* [23], which enables rapid identification of patterns by simultaneously exploring temporal and spatial scales. VIA-MD [31] also focuses on large-scale MDS data visualization and exploration that links the dynamic 3D geometry to statistical analysis of the data to allow users to identify patterns. Recently, Ulbrich *et al.* [36] represented the MDS data as a node-based dataflow, sMolBoxes, to allow experts to analyze it while still being able to explore the 3D structure.

Many of these methods focus on aggregating the trajectory in some way, whereas for DNA-nano MDS it is essential to legibly reveal the simulating target frame by frame. While event analysis that supports frame-by-frame analysis at various granularity levels has been realized in MD simulations in the past (e. g., [31, 36]), for the DNA-nano application we handle phenomena where pervasive small-scale events (H-bond formation) can happen everywhere in the simulation target and can lead to higher-scale events (forming and deforming of the local helix all the way up to the whole assembly)—while also supporting the traditional construction, analysis, and editing of the DNA-nano assemblies in ways that are familiar to the experts. They need to link the dynamics simulation with their envisioned 3D structure and the staple-based folding during construction. In a way our work is thus akin to other applications of MDS in specific domains [3].

3.2. DNA nanotechnology modeling and visualization

Several computer-aided design tools have been developed for DNA nanotechnology. Adenita [7], caDNAno [8], Vivern [16], CATANA [15], and oxView [24] are five examples, sampling the spectrum of available tools. Adenita lets the user design a DNA structure in multiple abstract levels in 3D in a user-friendly and intuitive manner. caDNAno uses parallel nucleic acid helix strands as placeholders and then lets users select active strands and edit the connections between them. The tool's visual interface enforces all editing to be done in 2D, which facilitates easy interactions but hinders the mental understanding of the structure in 3D. Vivern is a VR application designed to enhance the design and analysis of DNA origami nanostructures, offering advanced visualization tools and demonstrating improved capabilities over traditional desktop applications. CATANA and OxView offer design functionality in a web browser. CATANA uses a "novel Unified Nanotechnology Format" and facilitates easy MDS export but does not support MDS trajectory visualization. OxView's interface supports the visualization of dynamic simulations. It can render many nucleotides in 3D in a browser, but its lack of abstract views for different scales hinders users to identify and understand relevant simulation events buried in vast amounts of data, with erratic dynamic behavior and visual occlusion. The handy abstractions used in the designing environment are fully disconnected from the trajectory representations. We aim to bridge this gap by leveraging the advantages of existing abstract views for dynamic scenarios.

3.3. Abstraction spaces in visualization

Previous research has shown the usefulness of organizing related visual representations as conceptual spaces, which Viola and Isenberg call *abstraction spaces* [38, 39]. For molecular visualization, Zwan *et al.* [37] and Lueks *et al.* [17] proposed a multidimensional space that organizes the visualization along axes such as *structure*, *illustrativeness*, and *spatial perception*. In contrast, Mohammed *et al.* [21] and Miao *et al.* [19] use their abstraction spaces as interactive panels that allow users to smoothly transition from one representation to another. While the former authors assign an axis for each structure of interest,

the latter propose a more general approach that organizes representations along aspects of interest such as layout and scale. In addition to the mere organization of representations by means of abstraction spaces, the power of animated transitions between different representations—as made possible through the spaces—has been established by Heer and Robertson [11]. We build upon this general foundation by incorporating the MDS data into the concept of abstraction spaces. In particular, we generalize Miao *et al.*'s [19] space by incorporating the concept of *idiom*, which is essential for conceptualizing the design, as well as by designing an interaction framework that incorporates the dynamic character of DNA-nano structures by allowing experts to explore MDS data while they also study the spatial design.

4. Motivation, approach, and prerequisites

As we showed, one major shortcoming of the state of the art of visualizing DNA-nano designs is the lack of representation of the dynamic properties. We thus teamed up with experts who design DNA-nanotechnology structures as a part of their scientific work. Our primary collaborator is a domain expert in DNA-nano design and wet-lab experiments (a co-author of this paper), who has been leading a team that collectively designs components of nano-robots that would be able to destroy cancer cells or neutralize pathogens as a part of next-generation health treatments. To gain understanding of the domain workflows, we met several times a month over the period of six months. We were also in contact with several other experts, including developers of the oxDNA simulation package and researchers, who all are familiar with DNA origami experiments.

4.1. Goals

Through our discussions with these experts, we established a set of high-level goals. The primary objective for DNA-nano designers is to understand the dynamic behavior of their nanoscale structures—specifically, to predict whether a design will self-assemble correctly or exhibit a certain behavior in a wet-lab experiment. MD simulations are a key computational tool for this purpose, allowing scientists to test designs cost-effectively before committing to expensive and time-consuming lab work. The analysis of the resulting large and complex trajectory data, however, is a major bottleneck. Currently, experts analyze MDS properties by compiling statistics such as energy and H-bond occupancy, but these approaches often lose structural information and provide little insight into the structure's dynamic behavior. Structural information is currently conveyed by a fast-forward animation or selection of representative images capturing simulation emergence. Both these methods are of presentational character and cannot be used as analytical tools that would lead to structure-related insight. Therefore, our central goal is to facilitate the efficient visual analysis of DNA-nanotechnology MDS trajectories to help experts interpret simulation outcomes and gain actionable insights for improving their designs.

4.2. Task analysis

To translate our high-level goal into concrete requirements, we worked with our collaborators to identify key analysis scenarios and derive a set of user tasks that a visualization tool must support. First, we found that beginners and lay audiences often view dynamic processes using schematic diagrams to comprehend the process as a whole. Second, experts often work on abstracted spatial views during the designing phase to keep the cognitive load low. However, the DNA simulation is usually performed at a more detailed granularity, such as the nucleotide (NT) and atomistic levels. So the MDS results visualization needs to bridge the gap between the abstracted spatial views and the detailed views. In a third key scenario, the process of structural assembly and disassembly as generated by the DNA simulation models is also of great importance to domain scientists. Of particular interest here is the information about the order of association and dis-association on two conceptual levels of structural detail: on the nucleotide level and on the level of a DNA strand. Finally, experts sometimes do not understand why their designed 3D structure does not self-assemble in laboratory experiments, even with MDS results at hand. So they are looking for ways to examine these structures. From these scenarios, we distilled the following essential user tasks:

T1: Assess the overall quality of the simulation—seeing the current structural assembly progress in comparison to the fully assembled state can help one decide whether more simulation time or more detail is needed.

T2: Identify and interpret patterns throughout the simulation—the high-level observation of an MDS run in the schematic view allows the experts to pinpoint potential problems of a simulation run, interesting simulation periods, and the change of H-bond status over time.

T3: Examine H-bond pairing events for specific, interesting periods—seeing the order and the exact position of individual NT and strand pairing can lead to insights and thus the identification of problems with the design. Thus the proposed solution should allow users to focus on specific periods from the whole simulation and to analyze them in detail, frame by frame.

T4: Inspect how one structural conformation converts to another—the aforementioned tasks should easily interplay with each other to enable the experts to comprehensively and seamlessly understand the overall biological dynamics at both abstract and detailed levels.

T5: Determine why some structures do not form—one of the major challenges in DNA-nano is the low yield in wet-lab assembly experiments, and sometimes (e. g., our case study in Sec. 6) the structure does not form at all. It is thus important to study the dynamic simulation to understand the reasons for the low yield.

T6: Present real data and avoid artifacts such as those caused by structural averaging—the averaged structure of a trajectory often deviates from any specific frame and bears artifacts that might lead to false insights [24]. Such problems are even less visible in more abstracted statistics. It is thus essential to show both the aggregated and the non-aggregated real data from the simulator.

These tasks can be coupled with conventional analysis approaches such as energy, distance, or angle graphs. Such analysis is essential yet beyond the scope of this article: it does not

directly deal with structural abstraction, on which we focus here.

4.3. Challenge analysis

To effectively support the identified tasks, we first had to understand the inherent challenges posed by the data itself. We found that issues arise due to the specific properties of the MDS trajectories, which prevent existing tools from producing visualizations that are effective for drawing actionable conclusions.

Specifically, we inspected a range of various MDS trajectories in oxView. We visualized small and large systems, with short and long simulation durations, for assembly/disassembly and stability simulations. We found that, for extremely small systems (less than 50 NT) and extremely short durations (less than 100 frames), oxView works well. For other configurations we applied the widely recommended structural alignment as well as several smoothing methods but realized that several challenges exist that prevent oxView and similar tools from producing effective visualizations that show both the spatial structure and the dynamic behavior to come to actionable conclusions:

C1: High frequency, large structure, many frames. The high motion frequency is due to each NT's position changing in all consecutive frames. When this characteristic meets a large structure with tens of thousands of NTs and tens of thousands of frames or more, analysts face a huge amount of information for each step. No matter how slowly the trajectory is being presented, the resulting visualization always vastly exceeds an analyst's ability to comprehend and make use of it as long as the actual position of each NT is shown. And even if one can invest the time to digest the changes between two frames, this is an extremely inefficient way of studying the trajectory because MDS often exhibits long periods devoid of any important events.

C2: Clutter and occlusion. Large structures include many NTs, making it difficult to distinguish different strands and causing occlusion in 3D structures. This situation results in an ineffective visualization that can only reveal the surface NTs in the foreground.

C3: Periodic bounding box. OxDNA MDS uses periodic bounding boxes (a technique involving a simulated unit cell that is regularly repeated throughout the space, allowing a finite system to be artificially modeled as infinite but seemingly splitting the molecules over the boundaries [4]) to emulate a boundless environment. It breaks up strands at the borders of the box, however, and thus prevents analysts from understanding the structure correctly.

C4: Lack of well-defined formats for analysis. Most observables from conventional analysis, in particular H-bond occupancy, are only generated on-demand by improvised scripts and lack well-defined formats. Such an approach thus prevents the experts from performing a reliable, standardized, and reproducible analysis.

C5: Conceptual views lost in dynamics. When a disassembly or stability simulation starts, the structure changes immediately and diverges from its "canonical" shape, making it difficult to perceive for analysts. As a consequence, all the design phase helpers that were created to assist users in understanding the topology of the structure are lost in the now-changed structure in the trajectory.

Based on this analysis of goals, tasks, and challenges, we concluded that an effective solution should not rely solely on aggregating data into synthetic statistics, which often obscures critical structural information. Instead, our approach centers on displaying both structural and dynamic information from an MDS trajectory concurrently across multiple, synchronized abstract views. By coupling these views with visual highlighting, we can provide the derived information, usually communicated with statistics plots, directly in the structural context. We believe that seeing both types of information together enables domain experts to find actionable improvements for their designs. In the following section, we describe the design of our framework built on these prerequisites.

5. Design of the SynopFrame

To help users with the mentioned tasks and challenges, we developed our SynopFrame approach. Below, we first explain our efforts in exploring the entire possible visualization space in the context of DNA-nano design that ultimately led to a number of design decisions for our framework. Next, we describe—in an implementation-agnostic way—a sequence of the transformations that realize the representations in use and address the challenges we just described in Sec. 4.3. We also report how we arrange and connect the various representations together, how we color code the NTs by their H-bond status, and how we add the highlighter that links all parts. To ensure reproducibility and extensibility, we discuss in Appendix D the Houdini-specific implementation details and in Appendix E the transitions between different representations.

5.1. SynopSpace: The visualization space

To discuss the entire space of possible data mappings we begin by analyzing well-established visual representations. One of these is the depiction of positions and orientations of each NT output by the simulator (*All-NT* as in Fig. 1a or Fig. 3a).[1] In the simple example in Fig. 3a we show a 3D triangle structure in the spatially final, "relaxed" configuration, while Fig. 3b shows the same structure but in its "designed" configuration (converted from its caDNAno design format to oxDNA format). In this mapping, each NT is drawn as three parts, a sphere at the *backbone* site, an ellipsoid at the *base* site, and a cylinder that connects the sphere and the ellipsoid. The backbone spheres of the NTs from the same strand are then connected by cones. This representation is used in both oxView [24] and Adenita [7] and shows whether an H-bond is formed—through an examination of the distance between a pair of NTs, and their respective types. Structures were traditionally designed in such a configuration because it allowed the experts to focus on the matching between scaffold and staples. Yet, for such a design the NT detail shown in Fig. 3b is not needed and may even be detrimental for some tasks. So the "Single Strands" representation from Miao *et al.*'s work [20] can be used to reduce the detail, while still showing the actual spatial positions of each strand in the dynamic context

[1] In the brackets we give our abbreviations for each representation.

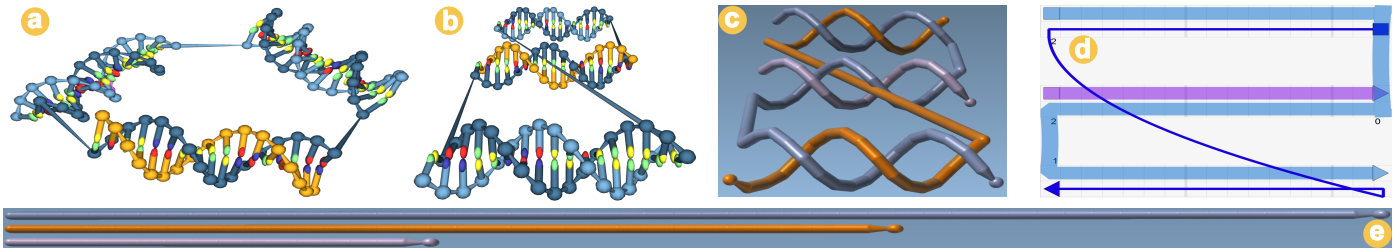


Fig. 3. Inspirations for fundamental representations from existing tools, all for the same triangle structure: (a) All-NT—3D spatial representation from oxView with NT detail, in its relaxed configuration; (b) All-NT in the triangle's designed configuration; (c) Snake—Miao *et al.*'s [20] "Single Strands" 3D spatial representation that only shows each DNA strand as a backbone; (d) Schematic2D—parallel straight lines showing the pairing between the scaffold and staples as well as the "flying lines" (lines that diagonally cross the structures) showing the linking between different parts of the staples; (e) Heatbar—each DNA strand straightened and sorted by length, as done by Miao *et al.* [20].

(*Snake*—once the strands start moving, they look very much like snakes in this representation—as in Fig. 1b and Fig. 3c). A related representation as it is used in caDNAno [8] further simplifies the double helices to parallel straight line segments shown in a 2D abstract representation (*Schematic2D*—as in Fig. 3d). It no longer shows any 3D spatial information but visualizes the pairing between the scaffold and staples as well as the linking between different staple segments. Finally, we may want to compare and see information about the scaffold and the staples by arranging them independently next to each other, sorted by length (*Heatbar*—as in Fig. 3e; by Miao *et al.* [20]).

What is less obvious about these representations is that they can be thought of in terms of three orthogonal aspects or axes. The first of these axes is **Granularity**, which describes the primary intact physical *individual* that we are dealing with. For example, in All-NT (Fig. 3a–b), each NT is depicted individually as sphere-cylinder-ellipsoid so that the primary element is the *NT*. In Snake and Heatbar (Fig. 3c and e), in contrast, all the NTs on the same strand are merged into a line entity and the physical individual now corresponds to the *Strand*. In Schematic2D (Fig. 3d), then, each pair of the parallel straight lines (each individual in focus) represents a continuous double *Helix*. The three levels on the granularity axis we can extract from the discussed representations are thus *NT*, *Helix*, and *Strand*, with a decreasing amount of granularity. In particular, we consider *Helix* to have a higher amount of granularity (finer) than *Strand* because a strand usually spans multiple helices. Even though we did not encounter more granularity levels in the above-discussed representations, we can still reason that there should be another one, *Assembly*, which treats the whole design as an intact individual. This *Assembly* naturally is a coarser level than *Strand*. Similarly, we can see that *Atom* is another level with a higher granularity than *NT*. We show the *Granularity* axis as the horizontal blue coordinate direction in our abstraction space in Fig. 4. In the figure, we label the levels at the bottom to avoid clutter and occlusion. We also do not explicitly indicate *Atom* as a dedicated level because the DNA-nano domain rarely simulates the dynamics of assemblies at that granularity.

Next to the granularity of the model, another way of looking at the abstraction of the depiction is to use different graphical primitives or elements. For example, in Heatbar and Schematic2D, (with the exception of the flying linkage lines in the latter) straight lines are used, to which we refer as *Bars*. If we allow the bars to bend and follow flexible paths, then we call the

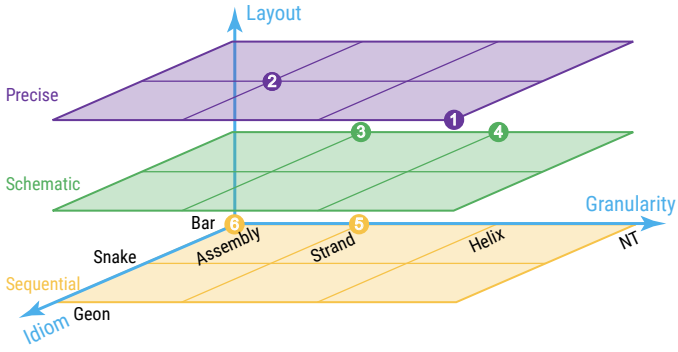


Fig. 4. Schematic view of the SynopSpace with its three axes *granularity*, *idiom*, and *layout*. Among the $4 \times 3 \times 3 = 36$ possible SynopPoints we highlighted those we discuss here with dots and numbers 1–6 in the order of (a)–(f) of Fig. 1 and Fig. 5. We discuss the others at the end of 5.1.

primitives *Snakes*. Finally, the primitives used in the All-NT representations are spheres, ellipsoids, cylinders, and cones. These simple geometric forms are examples of what neuroscientist Irving Biederman called *Geons* [2]—geometrical ions, which is "a modest set of generalized-cone components." We call the resulting abstraction direction the visual **Idiom** axis, in which the shape complexity that an idiom is capable of representing grows from *Bars* to *Snakes*, to *Geons*. Beyond the *Geons*, in fact, we could argue that another level that is able to show even more complex structures is *Surfaces* that are commonly used, for instance, in protein rendering [28]. We show the idiom axis in blue in Fig. 4 and label the levels at the upper-left of the figure. We do not explicitly indicate *Surface* in Fig. 4 because it is not commonly used in the DNA-nano domain.

The third axis in our space is **Layout**: the arrangement of the visual idiom of a granularity in the scene. In All-NT, each NT's sphere, cylinder, and ellipsoid are placed at their *precise* 3D positions as calculated based on the output by the simulator. In Snake, even though the details of the NTs are abstracted out, the snake still passes the *precise* center of mass (CMS) positions of each NT. In Schematic2D, each helix is no longer twisted but straight. So the bars no longer represent the precise positions. As the semantics of the helix and the relative positions between the helices are maintained, we call it a *schematic* layout. In Heatbar, the NTs are *sequentially* arranged along a bar and the bars are again *sequentially* arranged in screen space. So the three levels of the *Layout*, with increasing spatial faithfulness to the

simulation, are *Sequential*, *Schematic*, and *Precise*, which we show as the vertical blue axis in Fig. 4 and label the levels at the right of each layout plane, with the corresponding color.

The three axes are independent from each other, so we can sort them in increasing amounts of detail and arrange them in an abstraction space [38, 39] we call SynopSpace (Fig. 4)—owing to its capability of showing various levels of synopsis of an MDS trajectory—, and points within it SynopPoints. As discussed by Viola et al. [39, 38] and demonstrated in various examples in the past (Sec. 3.3), such abstraction spaces allow us to understand aspects of existing visual mappings of our data. For example, the existing *All-NT* representation is at point *(Precise, NT, Geon)* within SynopSpace (i. e., point (1) in Fig. 4), so we can understand that it is far from the origin of the space and thus the information density it encodes is high and it is likely useful for tasks that require detailed analysis.

But the established representations do not cover all points within our space, and we can also use it to discover alternative representations that may be useful for particular tasks. So let us examine some positions that are not yet covered. A possible representation at the origin of the space at *(Sequential, Assembly, Bar)*, for instance, triggers us to think about how a bar could be used to show the entire structure assembly in a sequential manner. This thought brings to mind the metaphor of a progress bar (such as for showing the progress of copying a file). We could use a progress bar representation, e. g., to show the number of H-bonds that are formed in the entire structure as it dynamically assembles from the scaffold and staples (T1; we discuss this representation further in Sec. 5.2). Another example of reasoning in the space allows us to address the dilemma that, when using a caDNAno representation (Fig. 3d), the detail on helices (T3 and T4) and the information about linkage between segments on the strands (T5) come at the cost of a lot of occlusion and clutter (C2). To improve the situation we examine the caDNAno representation closely in SynopSpace. First, it is easy to tell that its layout is a *schematic*. Second, it mostly uses straight *bars* even though there are also some curves that connect segments across bars, yet those are used to indicate the connections rather than to encode actual NTs. But, third, what is its granularity? The straight bars represent the helices well, while linking curves provide information about the strands. So the granularity is coarser than the helix level but finer than the strand level. So, if we were to assign a point for the caDNAno representation, it would be in-between point (3) and point (4) in Fig. 4. Based on this classification we can now try to address the mentioned issues. For example, we can derive two separate representations, one solely dedicated to helices and the other solely to strands. Creating the first is easy, we can remove the linkages between the segments from the caDNAno representation and thus move it to point (4). The latter is more difficult. We need to remove the linkages while keeping the strand intact. Our solution is to shorten those linkages and place the helices in 3D rather than 2D (as if the curved links are shrunk to drag the helices on both ends together to fold them). We thus changed the original caDNAno representation to now be located at point (3). With the new mappings, which we name *schematic2D* and *schematic3D* and discuss further in Sec. 5.2, we demonstrated that we can use SynopSpace as a mental tool to design proper mappings to tackle the various challenges and fulfill the tasks. Later in Sec. 5.3 we also show that SynopSpace helps us to arrange the different representations in the linked views.

We do not describe all possible SynopPoints, rather focus on few that we found useful in specific situations. For example, *(Precise, Assembly, Bar)* may appear bizarre yet if we need to show multiple different designs in an MDS system then it may be useful to place a *Bar* for each design (*Assembly*) at its *Precise* location. Similarly, the point *(Schematic, Assembly, Bar)* makes sense if we place the *Bars* at predefined locations. A similar scenario can also make use of another set of SynopPoints, *(*, Assembly, Geon)* by using simple geometries to abstract the whole assembly rather than just NT and then showing them either *Sequentially*, *Schematically*, or *Precisely*. We can also potentially extend the space, for instance, by allowing the granularity to have one more level, *Atom*, to allow us to expand our work to all-atom simulators. So the use of new SynopPoints depends on the given application and whether it needs respective representations or not. Once a scenario expands or changes, SynopSpace can be used as a mental tool for designing the proper representations.

5.2. Realizing each representation

Having established the SynopSpace abstraction and identified the six key representations within it, we summarize in Table 1 their SynopPoint index, coordinate tuple, example figures, and the specific tasks and challenges that it addresses, along with a concise description (for more details see Appendix A).

Among the six SynopPoints, Schematic3D is key to bridging the 3D and 2D representations. To realize it, we use the topology and CMS positions (Fig. 6a) of all NTs of one frame (usually the designed configuration) as input, followed by a series of transformations: **Step 1:** We construct polyline primitives differently than for *Snake*, so that only the NTs that form a continuous double helix end up in the same polyline. The singleton NTs that do not form H-bonds are not in any polyline. We use Algorithm 1 (Appendix J) to exhaustively check the conditions that could potentially break a continuous helix and create the primitives for the continuous ones. We show the result in Fig. 6b. **Step 2:** We prepare the required attributes for each polyline for downstream transformations. We run this algorithm for each polyline primitive using the first NT to find its pair and then using the pair to find the pair's polyline ID and save it as an attribute. We also save the following attributes for each polyline: CMS, CMS of its pair, CMS of the whole double helix, and the direction vector from the double helix's CMS to its own CMS, see Fig. 6b for illustrations. **Step 3:** We use a smoothed line to represent each helix (Fig. 6c). We move each NT to the center of itself and its pair. The two strands of each helix thus overlap with each other. **Step 4:** We straighten the helix and shift the two polylines in each helix at a user-controllable distance and evenly space the NTs on each polyline at a user-controllable distance (Fig. 6d). For this purpose, we first perform a linear regression against all the vertices in each polyline. We then use Algorithm 2 (Appendix J) to shift the two lines in each helix and distribute the NTs on them. Now, we can use the same preprocessing and rendering technique as those in the *Snake* representation to create

Table 1. Summary of the six key representations in SynopSpace, with their names, SynopPoints, coordinates in SynopSpace, examples, tasks and/or challenges, and description. They are described in detail in Appendix A.

Name	Point	Coordinate	Examples	Tasks/challenges	Description
All-NT	1	(Precise, NT, Geon)	Fig. 1(a), 5(a)	T3–T4, T6	Displays every nucleotide at precise (T6) position and orientation, enabling detailed inspection of individual H-bond interactions (T3) and subtle conformational shifts (T4)
Snake	2	(Precise, Strand, Snake)	Fig. 1(b), 5(b)	T2–T4, C1–C2	Shows each strand as a smooth 3D curve without nucleotide detail (C1), reducing clutter and occlusion (C2) to contextualize strand-pairing events (T3) in their broader spatial context (T2, T4).
Schematic3D	3	(Schematic, Strand, Bar)	Fig. 1(c), 5(c)	T2–T4, C1–C3, C5	Presents the idealized pre-simulation geometry as straight, parallel bars for each strand (C1), minimizing clutter (C2), avoiding bounding-box artifacts (C3), and leveraging users' mental models (C5), while the user is observing the dynamic H-bond events encoded in color overlaid on the static geometry (T2–T4).
Schematic2D	4	(Schematic, Helix, Bar)	Fig. 1(d), 5(d)	T2–T3, C2	Lays out double helices in a simplified 2D schematic without crossover links, eliminating occlusion (C2) so users can monitor helix pairing and unpairing events (T2–T3) at various zoom levels.
Heatbar	5	(Sequential, Strand, Bar)	Fig. 1(e), 5(e)	T1, C2	Arranges each strand as a colored bar chart, removing 3D occlusion (C2) to provide a quick overview (T1) of strand-level dynamics and highlight strands with unusual behavior.
Progress bar	6	(Sequential, Assembly, Bar)	Fig. 1(f), 5(f)	T1, C1	Abstracts per-nucleotide H-bond status into a linear progress-bar view over time, condensing high-frequency (C1) changes into an overview (T1) that quickly reveals key simulation phases.

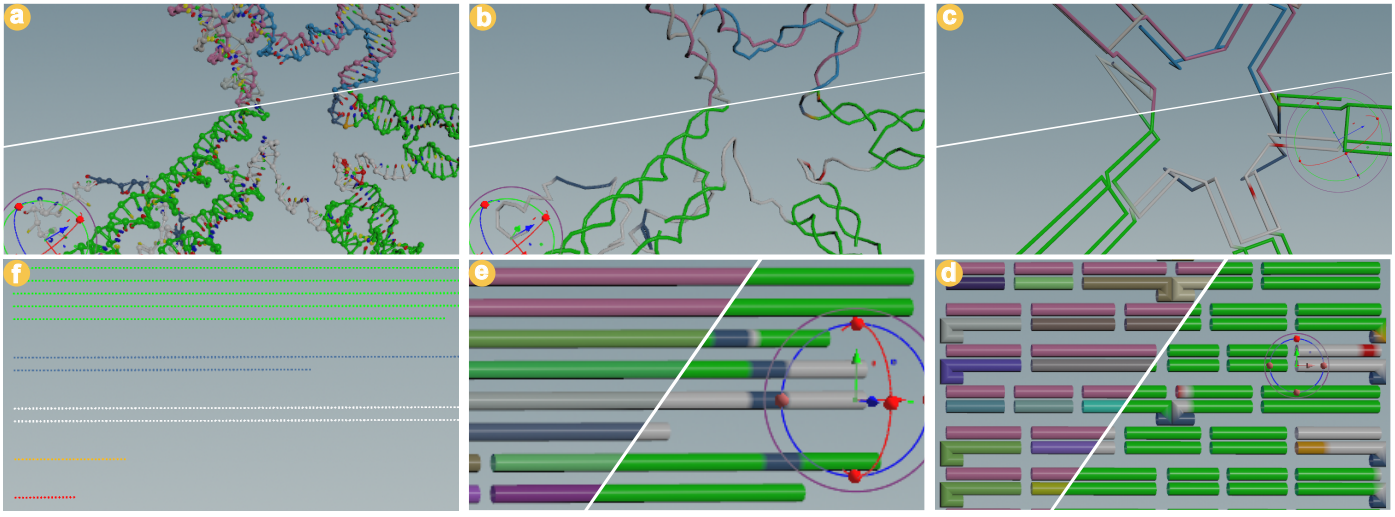


Fig. 5. Zoomed-in views (arranged in a clockwise fashion) for the purple dashed regions in the icosahedron design from Fig. 1. Apart from (f), we show all views in two color schemes—strand identity (by color) in the upper-left and MD progress (five colors) in the bottom-right. In the latter, the not-formed H-bonds are gray and the singleton NTs (i. e., designed not to be in an H-bond) are dark blue. (a) displays the NT details, which shows the NT types of the not-formed H-bonds. (b) removes the backbone and the base details and only shows the strand. (c) shows the designed geometry. Even though the highlighter-indicated NT hangs in the air in (a) and (b), in (c) it becomes clear where this NT should be attached if assembled correctly. (d) shows that this NT is at the end of the helix to which it belongs, while (e) shows it is at the end of a staple strand. (f) shows the progress bar.

the final visual representation. Sometimes (e. g., when a relaxed configuration rather than the designed one is used), however, there are *multiple* double helices that are supposed to be along a straight line but are tilted against each other. We thus use the following steps to refine this issue and to prepare the intermediate data for *Schematic2D*. **Step 5:** We create a new attribute (*schematic2D_row*) for each polyline, and shift those double helices with the same *schematic2D_row* so that they locate precisely along one straight line (Algorithm 3, Appendix J). It works on two user-specified threshold values: one for distance and the other for angle. Those double helices within a certain distance and within a certain tilt against each other will have the same *schematic2D_row* value. For each *schematic2D_row*, if there are multiple double helices, we then extract each helix' CMS into an array and perform a linear regression so that we shift the CMS of all double helices bearing the same *schematic2D_row* to sit on exactly the same straight line (Fig. 6e). **Step 6:** We delete all polyline primitives and construct the *Snake*. Since all NTs are now at straightened positions our final visual representation consists of straight bars (Fig. 6f) instead of curved snakes.

5.3. The dashboard: Linked views, highlighter, H-bond status

Each of the six representations reveals the structure at an important level of abstraction, solves particular challenges, and fits certain tasks. To address challenges and tasks we thus link all six via the time axis (Fig. 1 and the supplemental video) such that, once the time slider moves or is moved, all six change together. In addition, we can optionally link *All-NT* and *Snake* in a structural way, i. e., once the camera parameters change in one (rotate or zoom) the other will follow. We purposefully do not

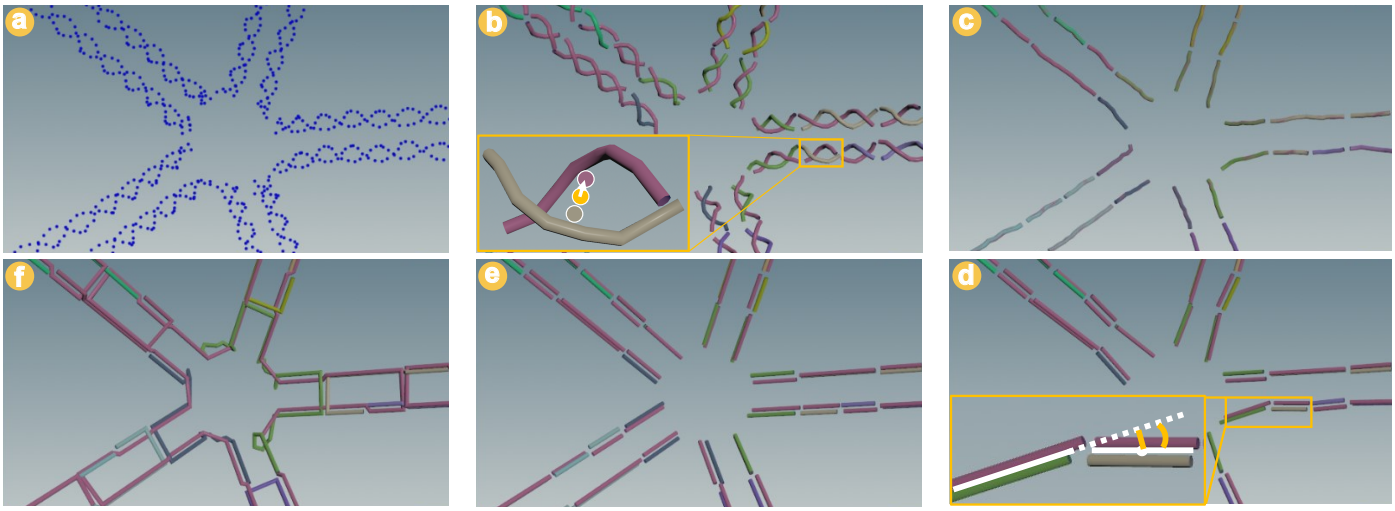


Fig. 6. Transformations in the Schematic3D algorithm (arranged in a clockwise fashion): (a) Input CMS positions. (b) Connected polylines (color showing strand identity). Notice the broken double helices (result of Step 1/Algorithm 1). The enlarged inset shows the attributes CMS (purple dot), CMS of its pair (brown dot), CMS of the whole double helix (orange dot), and direction vector from the double helix's CMS to the purple polyline's own CMS (white arrow). (c) Smoothed double helices (one line per double helix)—result of Step 3. (d) Straightened and shifted double helices (two lines per double helix)—result of Step 4. The enlarged inset shows the distance (straight orange line) and angle (orange curve) that we threshold. (e) Straightened double helices group (those with the same *schematic2D_row* value)—result of Step 5. (f) Connected strands—result of Step 6.

structurally link *Schematic3D* in the same way because, when the structure is no longer fully assembled, the similarity between *Schematic3D* and *Snake* no longer holds. We further support the linking with a synchronized highlighter, similar to the linking and brushing applied by Becker *et al.* [1]. Once, in selection mode, the user clicks on a specific NT in any representation, we highlight the same NT in all rest except in the *Progress bar* as the latter does not convey any structural information. The strand, a highlighted NT belongs to, can optionally be highlighted as well with an increased radius of the circle that we use to sweep along the strand's polyline.

On top of the structural views, MDS analysts often check additional abstract representations. A frequently used one is a PCA projection of the entire trajectory data, where each simulation frame becomes a data point in a 3D space with similar simulation frames being mapped to nearby spatial points. This view can be regarded as a projected conformational space plot. We also generate (similar to Poppleton et al. [24]) and link this PCA plot (Fig. 1g) to further assist the user with navigating the linked views. We link it interactively so that, once the time slider moves, we show the dot for the new frame's configuration with a bigger radius. Alternatively, once the user clicks a certain dot in the PCA plot, we move the time slider (and hence all the linked views) to that frame. We order all seven views clock-wise according to the abstraction level, with *All-NT* first and *PCA* last. The projected conformational space plot essentially shows a vector with three scalars for each frame. In a similar way, SynopFrame can also be coupled with the scalar versus time plots or the scalar versus another scalar plots showing the statistical metrics along the temporal aspect (more detail in Appendix I).

In addition to providing experts with this DNA-nano data dashboard, we also introduce a new way of categorizing H-bond statuses for better comprehensibility. OxDNA's built-in analysis only detects all H-bonds in each frame and then outputs the two NT IDs for each H-bond, regardless of whether that H-bond is in the designed configuration (C4). This reporting does not facilitate an effective analysis: mispairing of NTs may happen and mixing the designed H-bonds with non-designed ones hides the accurate H-bond counts, so potentially important mispairing events may go unnoticed. We thus categorize each NT in each frame into one of five groups (which forms a new format, which we detail in Appendix G), indexed from -2 to 2: (-2) designed to be in an H-bond, but wrongly formed; (-1) designed to be a singleton NT but an H-bond is formed; (0) designed to be in an H-bond but not yet formed; (1) designed to be a singleton NT and is single; and (2) designed to be in an H-bond and correctly formed. We then assign a color to each category (by default green to 2 and red to -2) and use it to color all representations, except for *PCA* which we color by time, by the entire configuration's energy, or other properties. Except for this coloring according to H-bond statuses, we also allow users to color the six representations according to the strand ID to reveal the routing of the staples. Users can also use a custom scalar value for each NT in each frame and use it instead for the color encoding such as the H-bond distance, the forces that NT is bearing, the NT's speed, etc. In the default color scheme for H-bond status we use red for bonds that are designed to be in an H-bond but are wrongly formed; correcting such cases requires first breaking the bond and then forming the correct one. We use orange when a nucleotide is designed to be a singleton but an H-bond is formed; this case is less severe than the former as it only requires breaking the bond. Gray indicates a designed H-bond that is not yet formed, also an uncritical status. Blue shows a correctly single-nucleotide designed to be a singleton, signaling a correct status without need for change. We use green for correctly formed H-bonds as designed, signaling success. For the projected conformation space plot (PCA plot), we use a gradient from red to green based on the timeframe, as simulations typically start with fewer correctly formed H-bonds and progress towards more correctly formed H-bonds

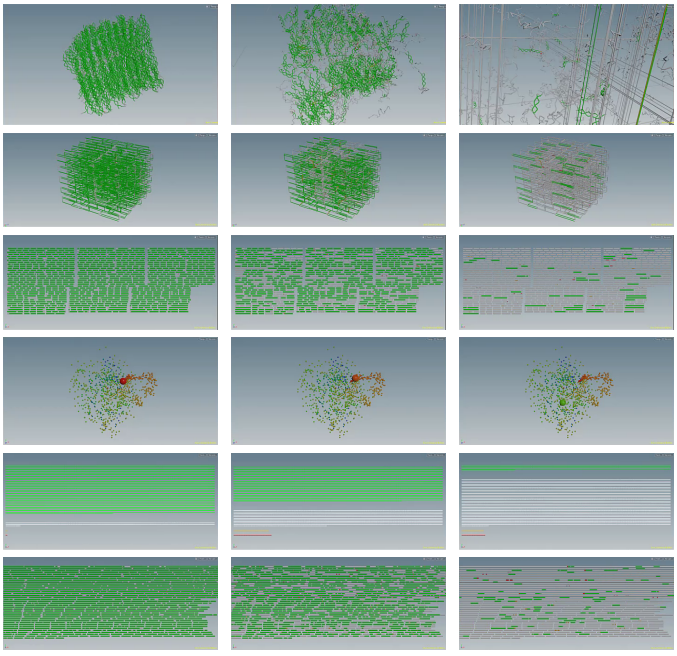


Fig. 7. Three frames from an animation (from left to right) that showcase the break-up of a structure. From the top: snake, schematic3D, schematic2D, PCA plot, progress bar, and heatbar.

over time. We represent the identity of the strands by random colors, due to the large number of staple strands often present, which would otherwise lack meaningful differentiation. For the base ellipsoids, our color scheme follows oxView: blue for A, red for T, green for C, and yellow for G. We also allow users to fully control all colormap assignments and adjust as needed.

5.4. Dynamic data analysis

Previous approaches to coping with the different DNA-nano representations by relying on abstraction spaces [19, 20] focused on showing different visual representations of static data. In contrast to this past work, our approach of embedding our SynopSpace abstraction space into our SynopFrame dynamic framework—for the first time—allows us to not only traverse different representations of the structures at some point in time but to actually use MDS to simulate the *dynamic* behavior of the structures as they assemble or not (T2–T5). We provide such dynamic analysis largely not in the form of summary views of dynamic behavior (the PCA plot is an exception and we describe another in Appendix I) but instead as actual animations (see the accompanying videos) in which the experts can observe whether and how a structure forms or not. Summarizing the dynamic behavior of the designs into a single and dedicated 'dynamics overview' representation is, in fact, not needed—our H-bond status visualization together with SynopFrame's playback possibility allows experts to quickly identify interesting time periods (as we demonstrate in Fig. 7).

The means to show dynamic changes we studied in this paper vary between a progress bar and a full 3D representation with a changing H-bond status. The progress bar is a well known visual encoding of an emergent process completion, which completely abstracts from structure and as such, animates throughout the simulation time how many desired H-bonds have been created and how many are not established. In contrast, on the other end of the spectrum is the animated structure depicted with its highest level of detail, where the bonding is conveyed spatially as well as through color mapping. Our design showcases some of many possible abstractions of animated process visualization that abstract from particular structural detail. Abstracting granularity allows us, in the absence of fine detail and its motion, to abstract these detailed motions, thus lower motion frequencies become visually more prominent. This way our representation allows viewers to follow the animation at faster simulation playback. The Snake representation used for this purpose can be further abstracted into a static 3D structure, where the dynamics remains in the color-coded animation of the bonding state of nucleotides. To facilitate a clear view of every such nucleic-acid substructure, we further abstracted the 3D representation into 2D or even 1D layouts where all details can be observed and additional details, such as strand length, become visually promoted. Finally, due to the linear nature, strands can be hierarchically grouped and merged together. On each level, the animated color-coding conveys the degree of bonding. Finally, at the most abstract level, all 1D strands are grouped into a single linear structure where still the order of the NT sequence is preserved. One final representation that abstracts from this order leads to the animated progress bar. All these visual encodings represent animated visualizations of an emergent process in time. Either only photometric visual channels, i.e., color mapping, are animated, or gradually geometric spatial animation adds the structural detail. Such detail can be varied throughout the simulation, or even within the simulation, different parts of DNA sequence could, in principle, be encoded by varying level of procedural detail. Our work thus explores the visual abstraction continuum where structural analysis of animated DNA is encoded by animated visual metaphors, which can be, if needed, further complemented by visualizations where the time is encoded in a different way than through animation.

In the following section we showcase an example application case and demonstrate the benefit of the analysis of dynamic DNA-nano data with SynopFrame, specifically the identification of a problematic design issue for a given DNA-nano design.

6. DNA-nano MDS exploratory analysis case study

To better illustrate how DNA-nano experts can make use of SynopFrame, we now describe a case study for performing an exploratory analysis for MDS trajectories. This realistic example showcases the overall process that experts can use to understand why it cannot assemble in wet lab experiments (T1, T5).

A cube structure with 16,128 NT (Fig. 8; one scaffold, 238 staples) was designed in caDNAno by a domain expert (a co-author of this paper) and his group. After the in-silico design, the structure appeared to be a robust design that will also self-assemble throughout the experiment. The research team had tried to assemble it in many wet lab experiments, but all attempts had failed, even though an experienced postdoctoral researcher had spent three months and ample resources on the project. We thus performed molecular dynamics simulations to examine the stability of the structure at various temperatures. While animating through the MDS with SynopFrame and looking at

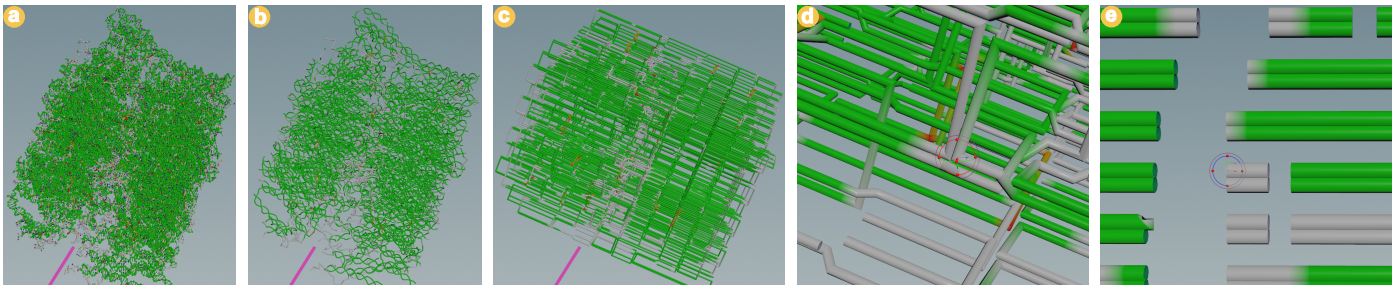


Fig. 8. Cube case study. All-NT (a), Snake (b), and Schematic3D (c) show that the cube's middle disassembles first, as if the cube is cut by a knife (see purple line). A synchronized highlighter attached in Schematic3D (d) highlights the cause: the short double helix visible in Schematic2D (e).

its various views, together with the expert we then noticed an interesting phenomenon that occurred at 78°C.

In the *Schematic3D* view, in which—with the animation—the expert could quickly identify (via fast-dragging the handle on the animation's playbar with the mouse) the time period in which the structure was attempting to assemble, we can immediately perceive a prominent pattern of the H-bond statuses at the beginning of the simulation (Fig. 8c). The pattern shows that the middle of the cube disassembles first, much like being cut by a knife right in the middle Fig. 8a–c. By attaching a highlighter to the disassembled NT (see our supplemental video), we can then easily identify the cause of such a pattern from the combination of *Schematic3D* and *Schematic2D*: there are many short double helices (3 NTs) aligned at that knife-cutting plane Fig. 8d–e. This observation can further be mapped back to the original caDNAno design view (supplemental video).

When observing this behavior of the simulation, the expert commented that it "*is very helpful in understanding why the structures did not form. In caDNAno, these mistakes (the short double helices) are not easily spottable.*" In our video interview with him (see our evaluation below in Sec. 7) he also mentioned that "*it could have saved three months of working time and resources of a postdoc experienced in wet labs but who has limited know-how in biophysics. This tool will give lab practitioners who do not have enough biophysics knowledge to understand the details of an MDS and its analysis a faster way to digest what is happening in the simulation and to remove bad designs. And even people who use traditional statistics such as root-mean-square deviation to analyze an MDS trajectory will face problems in finding insights for the case of large structures with long simulation durations. This tool comes right in place for these cases to help the analyst dive into the details.*" We later learned from our collaborating domain expert that the cube's main designer was surprised by the "knife-cutting" pattern because he had not realized this problem when using caDNAno.

An actionable insight from this analysis for the experts is thus that designers need to fix those short helices to prevent the respective parts from disassembling, which our collaborator then incorporated into his future DNA-nano designs. In addition, the caDNAno developers could improve their software by highlighting short strands to easily solve cases like ours. Ultimately, this aspect of the reported case study demonstrates that our H-bond color scheme allows experts to observe dynamic properties of the dataset using "normal" play-back animations of the MDS data, simply by seeing the characteristics of the H-bond status in different parts of a design. This visual representation of the dynamic characteristics relies entirely on the interactive play-back animations of the normally static views that we described (e. g., Fig. 7)—it does not require any dedicated visual summary of the dynamic characteristics of the MDS data to be effective.

We describe another case study of a smaller structure in Appendix H, where we focus on how it converts from one configuration to another (T4) as well as on its potential design flaws.

7. Further feedback

We gathered feedback from six expert oxDNA users and developers through a questionnaire. One of them is our previously mentioned close collaborator, who was our main contact in our user-centered design process and who is also a co-author of this paper. We communicated with him via e-mail and video meetings and he had access to a local SynopFrame installation for independent exploration. We interviewed him for qualitative feedback, and he also filled in the questionnaire with Likert-scale questions. In addition, we contacted eleven active oxDNA users who had raised issues in oxDNA's GitHub repository in the past year and who revealed e-mail addresses on their GitHub profiles. We also personally approached two additional DNA-nano experts. Out of these, the mentioned 5 additional experts answered our questionnaire. Due to their time constraints we did not expect them to install and learn our new interface from the ground up, but instead we provided them with a video description of our system (similar to our supplemental video) and asked them to fill in the same questionnaire as our close collaborator, and we received one anonymous and four signed responses (details about their backgrounds in Appendix K).

In Fig. 9 we report the questionnaire responses (see the full questions and the detailed answers in Appendix K) from all contacted oxDNA users, based on a 5-point Likert scale with 1 meaning *strongly disagree/very useless/very ineffective* and 5 meaning *strongly agree/very useful/very effective*. Generally, the experts found the linked views, the connection with the PCA, the H-bond coloring, and the NT highlighter to be effective for analyzing DNA-nano MDS trajectories. The transitions between different representations, however, play a less important role in the case studies, and may thus have received lower ratings from the experts. In addition, we purposefully separated the transitions from the animation of the molecular dynamics because, otherwise, viewers may confuse both types of animation. The linked views and the highlighter, in contrast, already seem to be sufficient for the experts to understand the relationships between

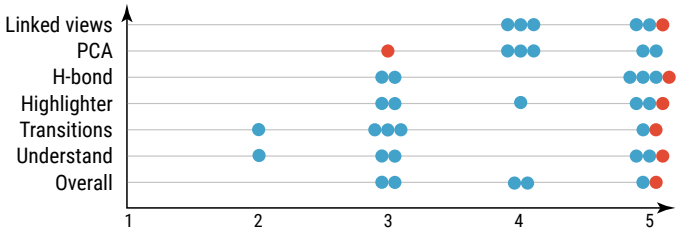


Fig. 9. User feedback for the effectiveness of SynopFrame. Our collaborator's response is colored in red, while the external evaluators' are colored in blue. We provide the full questions (*y*-axis) in Appendix K.

the representations, so the effectiveness of the transitions may be shadowed by the other features in the tested scenario. The 2-rating for "Understand" comes from an oxView developer who prefers much more the conventional statistics-based analysis approach and who tried our tool without training, he may have missed already implemented functionality. We thus conclude that our tool requires more training to be fully effective.

The concerns that the evaluators raised are twofold. First, the accessibility of the tool is reduced if it resides in Houdini and requires careful preparation of the input data in specific formats. So, after our current proof-of-concept, the domain users are eager to get access to a more accessible tool with the functionalities of SynopFrame, e. g., as a web-based tool. Second, even though experts appreciated the structural abstraction, they still would like to see an integration of more conventional statistics. We aim to realize both goals together in our future development.

8. Limitations and future development

While multiple views at different scales and abstractions are necessary, showing all views at once indeed can overwhelm a viewer. A future direction is to adaptively show the appropriate abstraction level based on the MDS phenomena. The implementation in Houdini is more of a prototype for proof-of-concept, which is fast to develop, but installing the whole Houdini software takes unnecessary disk usage, and the huge amount of widgets in Houdini are distractors for the user. A dedicated development with technologies such as WebGPU could be followed after the proof-of-concept to solve these problems. We also think that it would dramatically increase the efficiency of the whole workflow if it was possible to directly modify the design upon identifying the problematic regions, followed by feeding the updated design back to the oxDNA simulation. Features for the comparative analysis of multiple trajectories could also be helpful. In addition, the ability to locally and schematically animate the simplified static geometries to convey certain dynamic properties could further unleash the power of the schematic representations we developed. It is also worth mentioning that our whole design study could be expanded to other domains—e. g., in protein MDS, one could change the granularity axis of our SynopSpace to fit the hierarchical multiscale nature of protein structure and solve similar challenges there.

9. Conclusion

The extensive research on DNA-nano structures for various applications has led to the use of MDS as a cost-effective method for identifying poor designs and understanding the dynamic nature of these structures. Nonetheless, analyzing and comprehending an MDS trajectory continues to present significant challenges. We systematically analyzed the tasks, challenges, and established representations from the domain, and proposed a visualization abstraction space as a foundation, based on which we developed a proof-of-concept visual analysis tool that enables experts to see the connections between the different representations and to better explore and understand MDS trajectories, i. e., the dynamic aspects of DNA origami assemblies. While our tool is not yet integrated into the toolchain of the experts, we still demonstrate that the approach works in principle and with some more development an integration is possible.

In a broader sense, as MDS systems increase in size and approach the mesoscale, such as the whole cell simulation mentioned by Stevens *et al.* [34], a critical need arises for the abstraction of the structure for visualizing the resulting trajectory. With the novel approach we developed, SynopFrame, domain experts are now able to identify design flaws in a DNA cube design, such as the one that troubled our collaborators for months and cost them vast amounts of experiment resources. Users can also identify the transition period for the conformational change in an RNA tile design to understand how the NTs' H-bonds change during that period. SynopFrame goes beyond Miao *et al.*'s work [19] by extending the abstraction space to the *idiom* axis, encoding dynamic spatial data from the simulation in the more detailed views, connecting the static design and its dynamics through the novel Schematic3D view to lower the cognitive load, as well as taking advantage of color-coding the H-bond status in the more abstract views to allow experts to identify problems with their designs through traditional playback or the use of a time slider (for a detailed comparison see Appendix L). As such our approach is best suited for the post-design phase, while Miao *et al.*'s work supports experts in navigating the different design spaces that are important at design time. The organization of various data mappings/representations in a holistic space, reasoning within the space, and linking 3D and abstract views in our work collectively enable a new paradigm of MDS trajectory analysis. Rather than drawing insights only from statistics compiled from the trajectory, this new approach allows experts to gain insights by directly visualizing the trajectory. We thus extend the previous design-only solution to a comprehensive MDS analysis scenario. This analysis is based on all available information in the data, avoiding any bias toward certain statistical approaches that compress information in certain aspects. In this context, SynopFrame facilitates a shift in thinking about the analysis of MDS and the design of new visualization techniques in the emerging mesoscale era.

Acknowledgments

The authors thank all DNA-nano and visualization experts who tested SynopFrame or examined our visuals and gave feedback. This research was supported by King Abdullah University of Science and Technology (KAUST) (BAS/1/1680-01-01).

Supplemental Materials Pointers

We share the appendix of this paper as well as the accompanying video and the evaluation video we used to get feedback from domain experts at `osf.io/qxrzw`. In addition, our software is available on GitHub at `github.com/nanovis/SynopFrame`.

Images and graphs license/copyright

References

[1] Becker, RA, Cleveland, WS. Brushing scatterplots. Technometrics 1987;29(2):127–142. doi:10/bdqp5z.

[2] Biederman, I. Recognition-by-components: A theory of human image understanding. Psychol Rev 1987;94(2):115–147. doi:10/c9dqh2.

[3] Brossier, M, Skånberg, R, Besançon, L, Linares, M, Isenberg, T, Ynnerman, A, et al. Moliverse: Contextually embedding the microcosm into the universe. Comput Graph 2023;112:22–30. doi:10/gr7bbz.

[4] Bruininks, BMH, Wassenaar, TA, Vattulainen, I. Unbreaking assemblies in molecular simulations with periodic boundaries. J Chem Inf Model 2023;63(11):3448–3452. doi:10/mt3x.

[5] Bustamante, C, Marko, JF, Siggia, ED, Smith, S. Entropic elasticity of λ-phage DNA. Science 1994;265(5178):1599–1600. doi:10/dh8fcj.

[6] Byška, J, Le Muzic, M, Gröller, ME, Viola, I, Kozlíková, B. AnimoAminoMiner: Exploration of protein tunnels and their properties in molecular dynamics. IEEE Trans Vis Comput Graph 2016;22(1):747–756. doi:10/gtsjbd.

[7] de Llano, E, Miao, H, Ahmadi, Y, Wilson, AJ, Beeby, M, Viola, I, et al. Adenita: Interactive 3D modelling and visualization of DNA nanostructures. Nucleic Acids Res 2020;48(15):8269–8275. doi:10/gmt49h.

[8] Douglas, SM, Marblestone, AH, Teerapittayanon, S, Vazquez, A, Church, GM, Shih, WM. Rapid prototyping of 3D DNA-origami shapes with caDNAno. Nucleic Acids Res 2009;37(15):5001–5006. doi:10/cb9dg7.

[9] Engel, MC, Smith, DM, Jobst, MA, Sajfutdinow, M, Liedl, T, Romano, F, et al. Force-induced unravelling of DNA origami. ACS Nano 2018;12(7):6734–6747. doi:10/gdqrsr.

[10] Fonseca Guerra, C, Bickelhaupt, FM, Snijders, JG, Baerends, EJ. Hydrogen bonding in DNA base pairs: Reconciliation of theory and experiment. J Am Chem Soc 2000;122(17):4117–4128. doi:10/bm2qkg.

[11] Heer, J, Robertson, G. Animated transitions in statistical data graphics. IEEE Trans Vis Comput Graph 2007;13(6):1240–1247. doi:10/dvqddt.

[12] Humphrey, W, Dalke, A, Schulten, K. VMD: Visual molecular dynamics. J Mol Graph 1996;14(1):33–38. doi:10/b3tgfk.

[13] Kolesár, I, Bruckner, S, Viola, I, Hauser, H. A fractional Cartesian composition model for semi-spatial comparative visualization design. IEEE Trans Vis Comput Graph 2017;23(1):851–860. doi:10/f92dz8.

[14] Kolesár, I, Parulek, J, Viola, I, Bruckner, S, Stavrum, AK, Hauser, H. Interactively illustrating polymerization using three-level model fusion. BMC Bioinf 2014;15:345:1–345:16. doi:10/f6tw85.

[15] Kut'ák, D, Melo, L, Schroeder, F, Jelic-Matošević, Z, Mutter, N, Bertoša, B, et al. CATANA: An online modelling environment for proteins and nucleic acid nanostructures. Nucleic Acids Res 2022;50(W1):W152–W158. doi:10/gs5tck.

[16] Kut'ák, D, Selzer, MN, Byška, J, Ganuza, ML, Barišić, I, Kozlíková, B, et al. Vivern—A virtual environment for multiscale visualization and modeling of DNA nanostructures. IEEE Trans Vis Comput Graph 2022;28(12):4825–4838. doi:10/mpwt.

[17] Lueks, W, Viola, I, van der Zwan, M, Bekker, H, Isenberg, T. Spatially continuous change of abstraction in molecular visualization. In: IEEE BioVis Abstracts. 2011,Url: hal.science/hal-00781520.

[18] Maiti, PK, Pascal, TA, Vaidehi, N, Heo, J, Goddard III, WA. Atomic-level simulations of Seeman DNA nanostructures: The paranemic crossover in salt solution. Biophys J 2006;90(5):1463–1479. doi:10/fm5cb7.

[19] Miao, H, De Llano, E, Isenberg, T, Gröller, ME, Barišić, I, Viola, I. DimSUM: Dimension and scale unifying map for visual abstraction of DNA origami structures. Comput Graph Forum 2018;37(3):403–413. doi:10/gdw4j6.

[20] Miao, H, De Llano, E, Sorger, J, Ahmadi, Y, Kekic, T, Isenberg, T, et al. Multiscale visualization and scale-adaptive modification of DNA nanostructures. IEEE Trans Vis Comput Graph 2018;24(1):1014–1024. doi:10/gcqbtq.

[21] Mohammed, H, Al-Awami, AK, Beyer, J, Cali, C, Magistretti, P, Pfister, H, et al. Abstractocyte: A visual tool for exploring nanoscale astroglial cells. IEEE Trans Vis Comput Graph 2018;24(1):853–861. doi:10/gcqk6d.

[22] Ouldridge, TE, Louis, AA, Doye, JPK. Structural, mechanical, and thermodynamic properties of a coarse-grained DNA model. J Chem Phys 2011;134(8):085101:1–085101:22. doi:10/cbrjcm.

[23] Pálenik, J, Byška, J, Bruckner, S, Hauser, H. Scale-space splatting: Reforming spacetime for cross-scale exploration of integral measures in molecular dynamics. IEEE Trans Vis Comput Graph 2020;26(1):643–653. doi:10/kvh8.

[24] Poppleton, E, Bohlin, J, Matthies, M, Sharma, S, Zhang, F, Šulc, P. Design, optimization and analysis of large DNA and RNA nanostructures through interactive visualization, editing and molecular simulation. Nucleic Acids Res 2020;48(12):e72:1–e72:12. doi:10/gs5xc8.

[25] Rothemund, PW. Folding DNA to create nanoscale shapes and patterns. Nature 2006;440(7082):297–302. doi:10/dzh8m2.

[26] Rovigatti, L, Romano, F, Poppleton, E, Matthies, M, Šulc, P. Documentation – oxDNA. https://lorenzo-rovigatti.github.io/oxDNA/; 2022. Accessed in Apr. 2024.

[27] Rovigatti, L, Šulc, P, Reguly, IZ, Romano, F. A comparison between parallelization approaches in molecular dynamics simulations on GPUs. J Comput Chem 2015;36(1):1–8. doi:10/b5k3.

[28] Sael, L, Kihara, D. Protein surface representation and comparison: New approaches in structural proteomics. In: Biological Data Mining; chap. 5. New York: Chapman and Hall/CRC; 2009, p. 109–130. doi:10/c83vqf.

[29] Seeman, NC. Nucleic acid junctions and lattices. J Theor Biol 1982;99(2):237–247. doi:10/fbkm7q.

[30] Seeman, NC, Sleiman, HF. DNA nanotechnology. Nat Rev Mater 2017;3(1):17068:1–17068:23. doi:10/gghfhj.

[31] Skånberg, R, Linares, M, König, C, Norman, P, Jönsson, D, Hotz, I, et al. VIA-MD: Visual interactive analysis of molecular dynamics. In: Proc. MolVA. Goslar: EG Assoc; 2018, p. 19–27. doi:10/gf9rvn.

[32] Snodin, BE, Randisi, F, Mosayebi, M, Šulc, P, Schreck, JS, Romano, F, et al. Introducing improved structural properties and salt dependence into a coarse-grained model of DNA. J Chem Phys 2015;142(23):234901:1–234901:12. doi:10/f7sbb2.

[33] Sommer, B, Inoue, D, Baaden, M. Design X Bioinformatics: A community-driven initiative to connect bioinformatics and design. J Integr Bioinf 2022;19(2):20220037:1–20220037:8. doi:10/gtsjbc.

[34] Stevens, JA, Grünewald, F, van Tilburg, P, König, M, Gilbert, BR, Brier, TA, et al. Molecular dynamics simulation of an entire cell. Front Chem 2023;11:1106495:1–1106495:9. doi:10/grqd9f.

[35] Šulc, P, Romano, F, Ouldridge, TE, Rovigatti, L, Doye, JP, Louis, AA. Sequence-dependent thermodynamics of a coarse-grained DNA model. J Chem Phys 2012;137(13):135101:1–135101:14. doi:10/gkqqbf.

[36] Ulbrich, P, Waldner, M, Furmanová, K, Marques, SM, Bednář, D, Kozlíková, B, et al. sMolBoxes: Dataflow model for molecular dynamics exploration. IEEE Trans Vis Comput Graph 2023;29(1):581–590. doi:10/kvjv.

[37] van der Zwan, M, Lueks, W, Bekker, H, Isenberg, T. Illustrative molecular visualization with continuous abstraction. Comput Graph Forum 2011;30(3):683–690. doi:10/c893rz.

[38] Viola, I, Chen, M, Isenberg, T. Visual abstraction. In: Foundations of Data Visualization; chap. 2. Cham: Springer; 2020, p. 15–37. doi:10/gk874c.

[39] Viola, I, Isenberg, T. Pondering the concept of abstraction in (illustrative) visualization. IEEE Trans Vis Comput Graph 2018;24(9):2573–2588. doi:10/gd3k7m.

[40] Wang, P, Chatterjee, G, Yan, H, LaBean, TH, Turberfield, AJ, Castro, CE, et al. Practical aspects of structural and dynamic DNA nanotechnology. MRS Bull 2017;42(12):889–896. doi:10/gcqfzw.

Appendix A. Detailed description of SynopPoints

We describe each of the implemented SynopPoints in more detail and discuss how they help us to address the tasks and challenges we outlined before. We use an icosahedron nanostructure (6,540 NTs) as an example DNA-nano design and, for each representation, describe its rendering method and the algorithms used to preprocess the input data when necessary.

All-NT, **SP1 (*Precise*, *NT*, *Geon*)**, (for each representation, we mention its name, SynopPoint index, and its coordinate in SynopSpace) shown in Fig. 1a and 5a, is the traditionally used representation in most DNA-nano-related tools, and is the only one in oxView [24]. It shows the precise positions and orientations (T6) of all NTs output by the simulator and is ideal for scrutinizing an H-bond pairing event in its local environment. Hence it helps with identifying interesting local events and understanding conformation changes (T3, T4). The output data from oxDNA are the center of mass (CMS) position and two orientation vectors for each NT. Following the oxDNA2 model's geometry [26] (see also Appendix C), we calculate the required positions and orientations for the *backbone*, the *base*, the backbone-*backbone connector* and the backbone-*base connector*. We then instantiate the backbone repulsion sites with spheres, base stacking sites with ellipsoids, backbone connector sites with truncated cones, and base connector sites with cylinders. To be consistent with the domain we use the same color scheme as in oxView for the base ellipsoids, i. e., blue for *A*, red for *T*, green for *C*, yellow for *G*. We leave the backbone spheres and connectors to color-encode other properties (Sec. 5.3).

Snake, **SP2 (*Precise*, *Strand*, *Snake*)**, shown in Fig. 1b and 5b, in its dynamic context is inspired by less formal representations used on various occasions such as on-demand drawings, gestures in a conversation, or educational animations (DNA origami folding animation, see `youtu.be/p4C_aFlyhfI`). To create it we remove the detail of the base and backbone of an NT and only show the position of each strand. As such, this representation reduces, to some extent, the high frequencies (C1) as well as clutter and occlusion (C2). It is ideal for examining a strand pairing event in a slightly wider local environment than that of a single H-bond, it thus helps experts to identify events and understand conformation changes (T3, T4) at a coarser level as well as with understanding those events in a larger context (T2). With the CMS data of each NT and the design's topology, we construct a polyline primitive by connecting it from the 3' end NT toward the 5' end for each strand. Translational sweeping of a circle along the curve then forms a snake-like geometry.

Schematic3D, **SP3 (*Schematic*, *Strand*, *Bar*)**, shown in Fig. 1c and 5c, is a caDNAno-like representation that we created by reasoning with the help of SynopSpace. It can be thought of as the 3D version of caDNAno's representation which, instead of using the precise positions of each frame's configuration, relies on a single frame. In practice, we observe that the designed configuration—before any relaxation or simulation (C5)—has the optimal geometry for a user to comprehend the structure, for several reasons: First, it is designed by people who often also analyze its MDS. Second, it is usually the configuration that people mentally construct. And, third, the double helices are still straight, which helps the experts to comprehend the structure. Even though sometimes elongated backbones exist, they help users understand the relationship between the surrounding structures by placing the paired NTs together to keep the clutter low, rather than causing any illusion or confusion. After simplifying double helices into two parallel lines, this representation further decreases the clutter and occlusion (C2), which is most beneficial in large structures (C1). As the geometry is now static, there is no more periodic bounding box issue (C3). After color coding the properties, e. g., H-bond statuses onto the geometry, it also helps the user to understand the simulation schematically (T2) at various levels depending on the zoom level. Furthermore, it serves as a bridge to connect other representations to tackle more tasks and challenges, e. g., the user might observe in the to-be-described *Schematic2D* representation that a helix is unpairing (T3), and then move to *Schematic3D* to understand which strand this helix belongs to, the routing (the crossovers involved) of that strand, as well as the H-bond statuses of the whole strand (T2). Then they may move to *Snake* to examine this strand's precise positions and see how it interacts with other strands in the spatial context (T4). We describe the implementation details in Sec. 5.2.

Schematic2D, **SP4 (*Schematic*, *Helix*, *Bar*)**, shown in Fig. 1d and 5d, is adapted from the caDNAno-like representation for which we removed all the linkages that show the crossover between continuous double helices that can cause a great deal of occlusion. *Schematic2D* is thus capable of showing all the double helices (T3) without occlusion (C2). It is ideal to monitor the pairing and/or unpairing of any number of double helices depending on the zoom level (T2). We use the *schematic2D_row* value from before, but still need *schematic2D_col* to place each segment of a helix and each NT on a segment. For the calculation of *schematic2D_col* for each NT we need to take into account the directionality of the segment: the offset from the 3' end of the segment. We then arrange each helix according to its row and length as well as arranging each NT. We also arrange the singleton NTs next to their closest NT's helices. We use the same rendering method as for *Snake* and expose parameters to allow the user to control the width, height, and row and column spacing of the arrangement of the helices.

Heatbar, **SP5 (*Sequential*, *Strand*, *Bar*)**, shown in Fig. 1e and 5e, is adapted from Adenita [7] where it is used to convey the length of all strands. Each strand is shown as a continuous vertical straight line and the strands are horizontally laid out, resulting in a representation that resembles bar plots. As the scaffold is usually more than ten times longer than the staples, however, in our scenario this representation wastes a lot of screen space and it becomes difficult to observe color changes on the short staples. To address these issues we first define how many NTs each row can harbor, split the scaffold into multiple consecutive rows, and then place the staples horizontally with spacing in-between. We allow users to control the maximum NTs per row, the height of the bar, the width of each NT, and the horizontal and vertical spacing with sliders. This representation is ideal for glancing over the dynamics simulation to understand the overall process (T1) at the strand granularity and identify issues of certain strands if there are any. As the strands are laid out in 2D there is also no occlusion (C2). We realize this view

by assigning a row and column index to each NT according to the length of the strand it belongs to and the offset to the 3' end on that strand. We use the same rendering method as for *Snake*.

Progress bar, **SP6 (*Sequential*, *Assembly*, *Bar*)**, shown in Fig. 1f and 5f, is the linear layout of the NTs based on their H-bond statuses. It is ideal to understand the overall process of H-bond changes (T1) and let the user decide whether a certain simulation period should be further examined. No matter how frequently H-bonds change, how large a structure is, or how many frames a simulation has (C1), at this abstract level the user can quickly perceive any changes and make decisions accordingly. We implemented this view by again assigning a row and column index to each NT, but here only according to the NT's H-bond status. The layout can also be adapted by the user to the viewport size and aspect ratio. We achieve the final visual representation by rendering impostors to form a circle at the given positions. Thus, when zooming out, the representation looks like a progress bar and, when zooming in, each NT is observable as an individual circle. We also do not use numbers to record the H-bond status because changing text does not communicate the dynamic nature of the simulation efficiently. Instead, we use color coding that we explain in Sec. 5.3.

Appendix B. DNA-nano design simulation

After a DNA-nano structure is designed, usually before performing the wet lab experiments, domain experts subject it to simulation to test its ***dynamic*** properties. There are several DNA simulators, ranging from a focus on the atom level [18] to the polymer level [5]. We rely on a commonly used one, oxDNA [22, 27, 32, 35]. OxDNA runs at the NT level and captures the movement of each nucleotide, the binding of Watson-Crick base pairs, and "zipping" events (binding of multiple consecutive NTs) between complementary strands. To prepare a newly designed structures for the actual dynamics simulations, we first need to run relaxation simulations that prepare the structure in a proper configuration that can further be simulated via Monte Carlo or molecular dynamics methods. We focus on the molecular dynamics simulation that, with a given initial configuration, generates the configuration in consecutive time frames via integration on small time steps, according to a selected biophysics model. The resulting configurations can be sampled at a user-specified time interval to form an MD trajectory and then stored. In the case of DNA-nano, a typical trajectory has thousands to millions of frames and each frame has tens of thousands of NTs, which easily leads to gigabytes to terabytes of data.

Appendix C. OxDNA2's model geometry

With the center of mass (CMS) position vector (r) and two orientation vectors (a1, a2) for each NT, we calculate the required positions (P) and normals (N) with the following equations. The *end3* in the equation refers to the 3' end.

$$
\begin{aligned}
P_{backbone} &= r - 0.34 * a1 + 0.3408 * a2 \\
P_{base} &= r + 0.34 * a1 \\
P_{backbone\_connector} &= (P_{backbone\_end3} + P_{backbone\_end5})/2 \\
N_{backbone\_connector} &= normalize(P_{backbone\_end3} - P_{backbone\_end5}) \\
P_{base\_connector} &= (P_{base} + P_{backbone})/2 \\
N_{base\_connector} &= (P_{base} - P_{backbone})/2
\end{aligned}
$$

Appendix D. Houdini-specific implementation

There are significant benefits to using Houdini as our prototyping environment. First, the *Apprentice* version is free and available for all three major platforms (Windows, Mac OS, and Linux). Second, from the user's perspective, after our application is developed, a typical user needs just around a 15-minute onboarding tutorial and a one-page cheat sheet to be comfortable navigating inside the application and then focusing on the analysis of their MDS trajectory. Third and most important, it is extremely friendly from the developer's perspective. Multiple viewports, which are required by the linked views, can be created with a few mouse clicks, followed by specifying the geometry to be rendered. Another valuable feature is Houdini's node-based workflow.[2] Each node harbors a tabular data structure for the points, vertices, primitives, and detail it contains so the developer can assign attributes to them and leverage the handy and fast attribute fetching function via its high-performance expression language VEX.[3] Each node is also a small program that performs certain transformations on the data it receives, much like the concept of a shader. The program can be written in VEX, Python, or C++ via Houdini Developer Kit (HDK).[4] In the case of VEX, the program can be chosen to execute just once, or in parallel for each point/vertex/primitive. Another feature we have used is the data cache node[5] that can cache the processed data into binary for fast loading the next time.

We use C++ via HDK to parse the large trajectory data as well as the newly defined H-bond data to gain the highest performance; we use VEX to perform all the geometry transformations and have leveraged the parallel processing to the best we can; we use Python for various other small tasks as well as the Python Viewer State[6] to achieve most of the user interactions.

We can also adjust the specific coloring of, in particular, the H-bonds to avoid red-green color contrasts for those people with color deficiencies. Fig. D.10 shows an example with a different color map that is safe for people with color deficiencies.

Appendix E. Transitions

Even though all the representations are linked together, it is still important to visualize the transitions (see the supplementary

[2] `sidefx.com/tutorials/intro-to-houdinis-node-based-workflow`
[3] `sidefx.com/docs/houdini/vex`
[4] `sidefx.com/docs/hdk`
[5] `sidefx.com/docs/houdini/nodes/sop/cache.html`
[6] `sidefx.com/docs/houdini/hom/python_states.html`

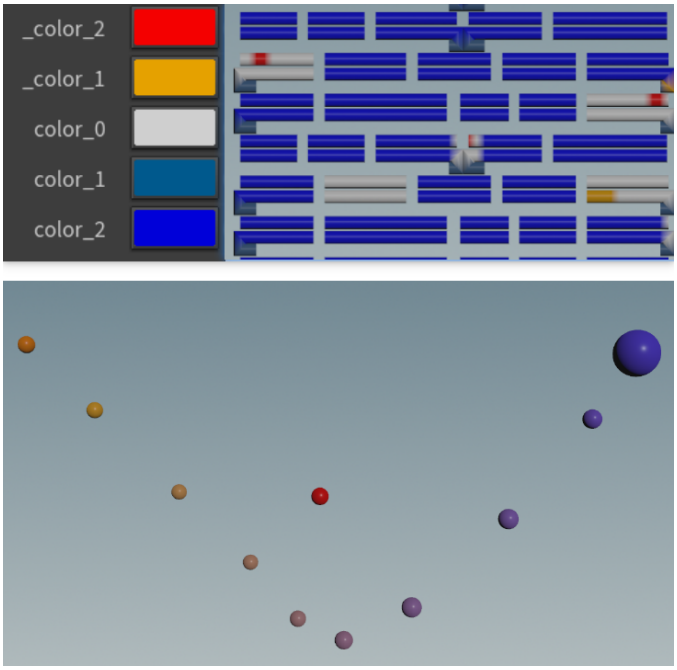


Fig. D.10. Example for the use of a different color map for people with color deficiencies. The top panel shows the new zoomed-in area of Fig. 1d; the bottom panel shows the new Fig. 1g.

video) between some of them. The users can view the transitions at the beginning of the analysis to understand what each representation is conveying for a specific DNA-nano design and how they are related to each other spatially. We implement these consecutive transitions: *Snake* ▹ *Schematic3D* ▹ *Schematic2D* ▹ *Heatbar*. As all of them are transformed from the CMS of all NTs, we interpolate the CMS values between them to achieve the transition. For the case of large structures, the transitions from *Schematic3D* to the more abstracted ones usually give illegible results if all strands transit together. So, we applied staged transitions by moving one strand after another.

Appendix F. SynopFrame performance

We record the performance-related number on a Windows 10 desktop (Intel(R) Xeon(R) Gold 6242 CPU @ 2.80GHz (2 processors), 256GB RAM) with an Nvidia RTX 3090 graphics card. To load and cache the cube structure (in the first case study) with 16,128 NTs, 1,000 frames, 3,076 MB data, SynopFrame takes 63 seconds. After caching into Houdini native binary data, the file size reduces to 494 MB, corresponding to 494 KB per frame. Reloading from the cache takes 6.2 seconds. The frame rate to show all seven views together is 1.65 fps; to show All-NT only is 6.27 fps; to show Schematic3D only is 24.97 fps; to show Progress bar only is 19.75 fps. As analysts very often stop at certain frames and scrutinize the structure, such frame rate is still considered to be interactive.

Appendix G. The `SynopSpace.hb` format

For a whole trajectory with many frames, we record the H-bond status code (mentioned in Sec. 5.3) of each NT in each frame as follows. We first record the frame number and then sequentially record the status code for each NT in a new line, with `StatusCode` $\in [-2, 2]$ as an integer to classify the H-bond pairing status. Below we first give a *generic format description* (`*.synopspace.hb`)[7] and then a specific *example* for its use.

```
t = <FrameNumber>
<StatusCode> [Pair ID if StatusCode == 2] # for NT0
<StatusCode> [Pair ID if StatusCode == 2] # for NT1
<StatusCode> [Pair ID if StatusCode == 2] # for NT2
...
t = <FrameNumber>
...
```

An example that shows in Frame number 1000, the first 5 NTs' StatusCode, with the 5th's pair (NT ID 7923) recorded:

```
t = 1000
-2
-1
0
1
2 7923
...
t = 2000
...
```

Appendix H. Case Study 2: An RNA tile design

As oxDNA can simulate RNA designs, we also performed a case study for an RNA tile design to demonstrate that SynopFrame can also work with RNA designs. In an experiment with an RNA tile with 132 NT (one strand) from [24] two different conformations were known to occur. From a simulation at 45 °C, the two groups of conformations are well manifested in the *PCA* plot Fig. D.11a. Further examination reveals that H-bond statuses are not always correlated with PCA dimension reduction results. For example, frame 259's conformation and frame 7765's are very close to each other in the *PCA* plot. But their H-bond statuses show a big difference in one critical crossover Fig. D.11b. This observation raises a caveat that although in general, *PCA* followed by clustering performs well in categorizing the conformations, it cannot reliably capture the subtle H-bond changes that will cause disproportionate effects on the conformation.

The transition period between the RNA tile's two conformations can be easily identified, see the supplementary video. The biggest difference between the two conformations is the loss of the two crossovers at around 1/3 of the structure. So we can attach a highlighter to this helix and then focus on *Schematic2D*, which is much easier to absorb the H-bond status changes. But the *Schematic2D* view first shows the opening of another helix from frame 601 to 607. So it forces the analyst to go back to the 3D structure (*Schematic3D, Snake, All-NT*) and think about the relation between these two helices. Only after this helix

[7] This format alone triggered a discussion among domain users and has since been integrated by some users into their own analysis approaches: `github.com/lorenzo-rovigatti/oxDNA/issues/45`

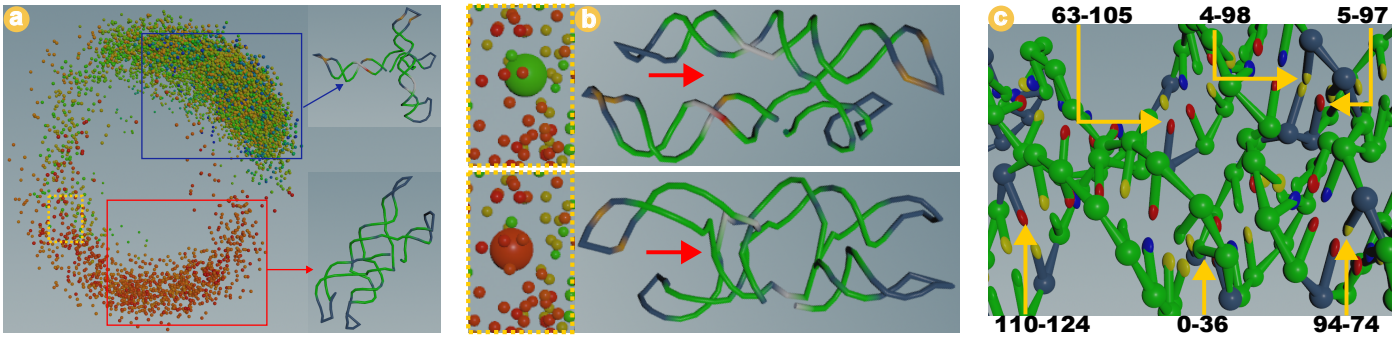


Fig. D.11. Case study for an RNA tile design. (a) PCA plot shows two groups of configurations. The yellow dashed rectangle is further zoomed in (b). (b) Two different configurations are located closely in the PCA plot. The difference is indicated by the red arrows. (c) The non-Watson-Crick base pairs in the design are easily identified as they are colored dark blue (indicated by the yellow arrows).

Algorithm 1: Create polylines for continuous double helices

Data: CMS of all NTs, topology of the design
Result: Polyline primitives for continuous double helices

totN = total number of NTs
N = 0
primList = [*N*]
while *N* < *totN* **do**
  end5 = the 5' end ID of *N*
  if *end5* == −1 **then**
    `CreatePolyline(`*primList, N*`)`
  else
    pair = the pair ID of *N*
    strand = the strand ID of *N*
    if *pair* == −1 **then**
      `CreatePolyline(`*primList, N*`)`
    else
      end5Pair = the pair ID of *end5*
      if *end5Pair* == −1 **then**
        `CreatePolyline(`*primList, N*`)`
      else
        pairStrand = the strand ID of *pair*
        end5PairStrand = the strand ID of *end5Pair*
        if *pairStrand*! = *end5PairStrand* **then**
          `CreatePolyline(`*primList, N*`)`
        else
          pairEnd3 = the 3' end ID of *pair*
          if *pairEnd3*! = *end5Pair* **then**
            `CreatePolyline(`*primList, N*`)`
          else
            push(*primList*, *end5*)
          end
        end
      end
    end
  end
end
Function `CreatePolyline(`*primList, N*`):`
  `/* When this function is called, it means the continuity of the helix is broken. */`
  Add a polyline primitive that has all the NTs in the *primList*
  primList = [++*N*]
return *None*

Algorithm 2: Shift the two polylines in each helix and evenly space NTs on each polyline

Data: User define: halfPairDistance, spacing
Result: Two parallel polylines with NTs evenly distributed for each continuous double helix

`/* This algorithm is supposed to be run in parallel for each polyline */`
dir = the white arrow vector in Fig. 6b
primDisplacement = *halfPairDistance* ∗ *normalize*(*dir*)
unitBackboneDir = *spacing*∗ *(the direction along the polyline)*
pts[] = all the points (NTs) ID on this polyline
avgPtnum = *average*(*pts*)
`/* The IDs on the same polyline are guaranteed to be consecutive so that we can take the average value */`
for *ptnum* ***in*** *pts*[] **do**
  offset = *ptnum* − *avgPtnum*
  newP = *polylineCMS* + *unitBackboneDir* ∗ *offset* + *primDisplacement*
  set the position for *ptnum* as *newP*
end

opened, the initially attended helix is then opened at frame 1181 to 1184. The analyst may then proceed to crop out these frames to perform further-detailed analysis, such as distance, angle, and energy distribution.

Looking at the bizarre *Schematic2D* view, one might wonder what those NTs are that are not forming helices. A close examination from the linked view with help of the highlighter reveals that apart from the non-pairing loop on the side of the structure, there are six base pairs that are within the H-bond distance threshold, but their sequences do not obey the Watson–Crick rule Fig. D.11c. The designer might then proceed with changing some of the sequences on those base pairs and see how the design behaves.

Appendix I. Statistical scalar plots

Although SynopFrame mainly explores the abstract views to visualize the MDS trajectory frame-by-frame, it can also be coupled with traditional data analysis plots focusing on the aggregated statistical metrics along the temporal aspect, i. e., generate a scalar to represent each frame and then plot that scalar. Fig. I.12 shows two such plots, with the top panel showing the energy versus time and the bottom showing the "force-extension curve" [9] previously used in the domain. The force-extension curve encodes each frame by two scalars, one for the force between two NTs, showing on the Y axis, and the other for the extension, or the distance, between these two NTs.

Algorithm 3: Assign *schematic2D_row* for each polyline)

Data: User define: distanceThreshold, angleThreshold (Fig. 6d)
Result: *schematic2D_row* assigned for each polyline

Function `PushPrimToDict(`*prim, primPair, nextRow ,rowDict, key*`):`
  helixInfo = a dictionary to hold the *prim*'s CMS and direction
  helixArray[] = fetch from *rowDict* by *key* or initialize an empty array
  push(*helixArray*, *helixInfo*)
  dictKey = *key* if *key* exists, otherwise *nextRow*
  rowDict[*dictKey*] = *helixArray*
  set *schematic2D_row* for *prim* and *primPair* as *int*(*dictKey*)
return *None*
numprim = total number of polyline primitives
schematic2D_row = {}; *nextRow* = 0
for *prim* = 0; *prim* < *numprim*; *prim* + + **do**
  primPair = the pair ID of *prim*
  if *prim* < *primPair* **then**
    `/* process only once per helix */`
    if *len*(*schematic2D_row*) > 0 **then**
      `/* initialize thresholding terms to large numbers */`
      minDist = 100.0, *minAngle* = 90.0
      overlapKey = " − 1"
      rowKeys[] = *keys*(*schematic2D_row*)
      for *key* ***in*** *rowKeys* **do**
        dict *helixArray*[] = *schematic2D_row*[*key*]
        Loop through the CMS and direction of each helix in *helixArray*
        Calculate the *distance* and *angle* value as shown in Fig. 6d
        if *distance* < *minDist* **then**
          minDist = *distance*
          minAngle = *angle*
          overlapKey = *key*
        end
      end
      if *minDist* < *distanceThreshold AND minAngle* < *angleThreshold* **then**
        `/* This means the current helix is overlapping with a row group, so we push it using the overlapKey */`
        PushPrimToDict(*prim*, *primPair*, −1, *schematic2D_row*, *overlapKey*)
      else
        `/* This means the current helix is not overlapping with any row group, so we push it using nextRow++ as a new key */`
        PushPrimToDict(*prim*, *primPair*, *nextRow* + +, *schematic2D_row*, '')
      end
    else
      `/* Same as the last push */`
      PushPrimToDict(*prim*, *primPair*, *nextRow* + +, *schematic2D_row*, '')
    end
  end
end

Appendix J. Algorithms

Here we detail the algorithms used to perform the transformations needed for the *Schematic3D* representation (Sec. 5.2). With the CMS of all NTs and the topology of the design as input, Algorithm 1 exhaustively checks the conditions that could potentially break a continuous helix and create the primitives for the continuous ones. With the CMS data of a straightened polyline, its pair, and the whole double helix it belongs, Algorithm 2 shifts the two polylines in each helix for a user-defined distance and then evenly space the NTs on each polyline. With the user-specified distance and angle thresholds, Algorithm 3 assigns a *schematic2D_row* value for each polyline so that those with the same value can be then aligned along one straight line.

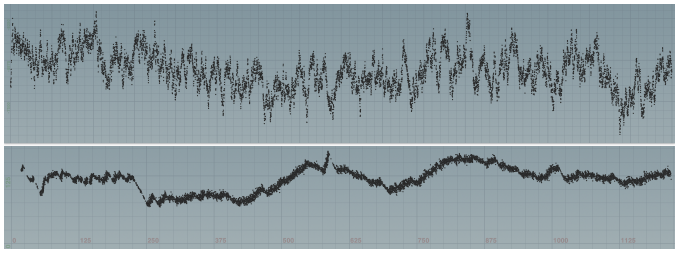


Fig. I.12. Statistical scalar plots. The top panel shows an energy versus time plot. The bottom panel shows a force-extension curve.

Appendix K. User feedback details

As explained in the main paper, we received feedback from our co-author collaborator, from one anonymous respondent, and four signed responses. These latter four were a senior researcher in computational chemistry who focuses on molecular dynamics and bioinformatics, a PhD student in the OxDNA developers group, a PhD student who focuses on wet-lab DNA-nano experiments, and a professional bioinformatician who specializes in wet-lab experiments for DNA origami. Please note that, for completeness, we also provide the video (`Video_for_User_Feedback.mp4`) that we sent to the invited experts as an explanation of our approach, which served as the basis of the following questionnaire, as additional material.

At the end of the appendix we provide the filled-in questionnaire responses from all participants that provided us with feedback. Note that we also provide the video we provided to participants as the basis of their evaluation as additional material, in addition to the actual paper video itself. In total we received 1 anonymous response and 5 signed responses (the latter including our closely collaborating expert who is a co-author). We provide these answers as screenshots from the online questionnaire tool to show both the stimuli and the specific questions we asked about it, as well as the answer possibilities and the detailed comments that all our participants provided.

We also note that SynopFrame visuals won 2nd place in the Design X Bioinformatics [33] Student Competition. Committee members from the Scripps Research Institute (USA), STUDIO ABOVE&BELOW (UK), the Royal College of Art (UK), and the Tokyo Institute of Technology (Japan) praised our "*highly useful and computationally interesting system to display DNA data in multiple ways*."

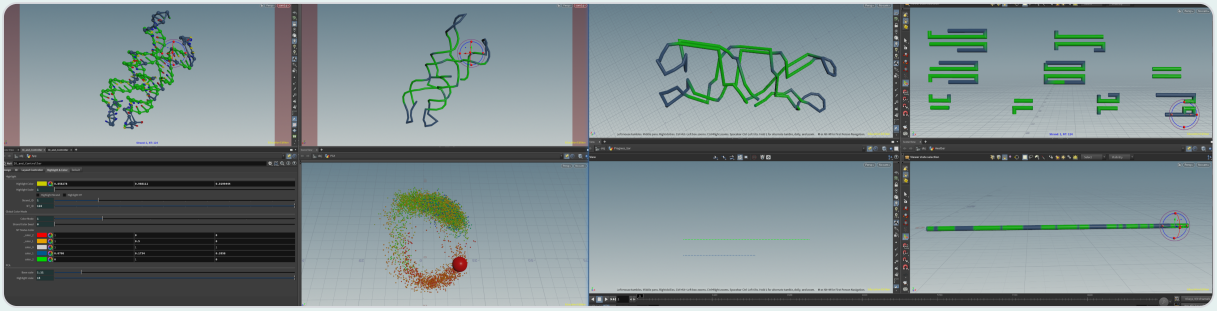
Table L.2. Comparison to Miao et al.'s [19] DNA origami abstraction space.

criterion	Miao et al.'s DimSUM [19]	our own work
focus of the work	seamless animation between 1D, 2D, and 3D layouts and multiple 3D semantic representations for different levels of detail	comprehensive MDS analysis scenario to identify issues for the wet-lab assembly of previously designed DNA-nano structures
target application	DNA-nano ***design phase***	DNA-nano ***post-design phase***
represent. of layout	1D, 2D, 3D layouts; i. e., a ***geometric*** view of possible layouts	sequential, schematic, precise; i. e., a ***domain-centered*** view of layouts
represent. of scale	***10 named scales***, organized based on a perceived level of "concreteness" or visual abstraction	***two separate and independent components*** for a greater flexibility: • the four levels of *granularity*: nucleotide, helix, strand, assembly; also organized based on a perceived level of "concreteness" or visual abstraction • the three–four levels of *idiom*: bar, snake, geon, and surface; to characterize different visual encodings of a given data component
unique views	• DimSUM abstraction view	• Schematic3D (Fig. 1c, Fig. 5c) • progress bar (Fig. 1f, Fig. 5f) • PCA plot (Fig. 1g)
represent. of dynamic data character	***n/a*** (only one data view is shown at any given time)	MDS-based ***animation of assembly*** that relies on a visual encoding of the ***H-bond status***, observable in ***multiple points*** of the abstraction space that are ***shown in parallel***

Appendix L. Comparison to Miao et al.'s DNA origami abstraction space DimSUM

To better illustrate our conceptual extension of past work, in particular the work by Miao et al. [19, 20], we list the fundamental differences between their and our abstraction spaces in Table L.2 (we do not include Miao et al.'s [20] first approach because it only focused on a single visual representations sequence).

Images license/copyright



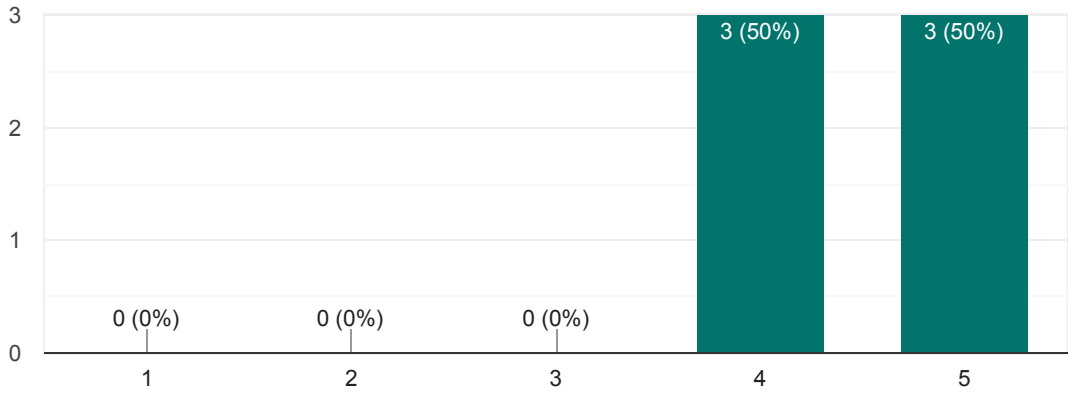
SynopFrame Feedback - 7 Questions (5 min maximum)

6 responses

Feature 1: the linked 3D and 2D views. Useful?

Copy

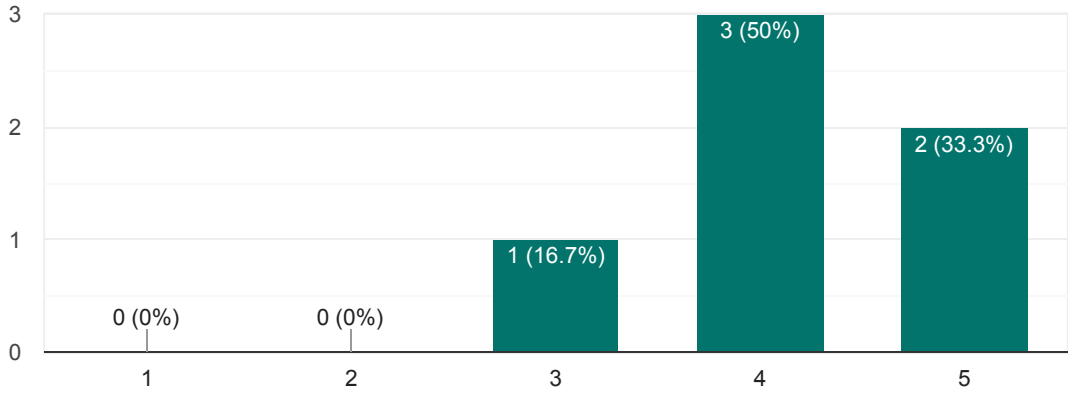
6 responses



Feature 2: the connection between PCA plot and structural representations. Useful?

Copy

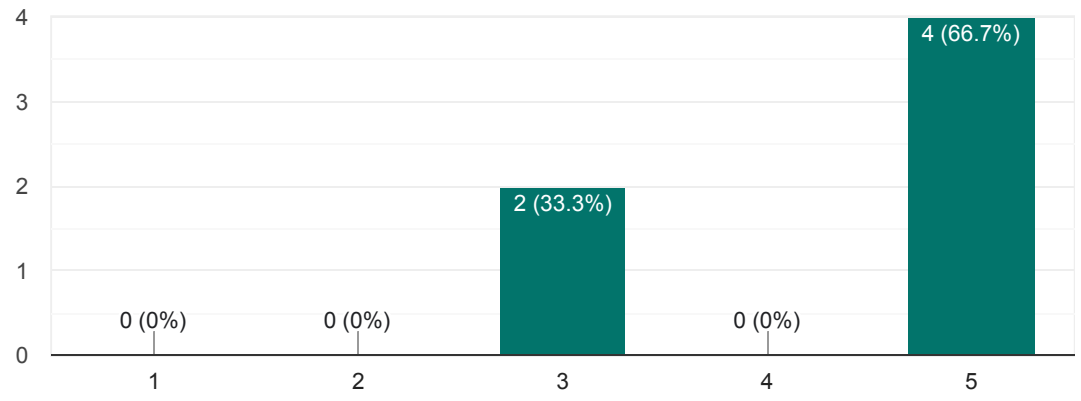
6 responses



Feature 3: the H-bond status coloring. Useful?

Copy

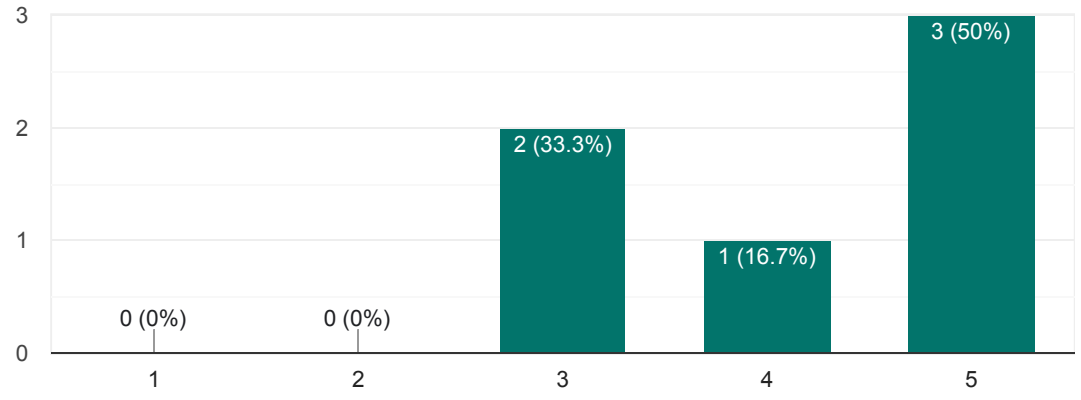
6 responses



Feature 4: the synchronized highlighter across views. Useful?

Copy

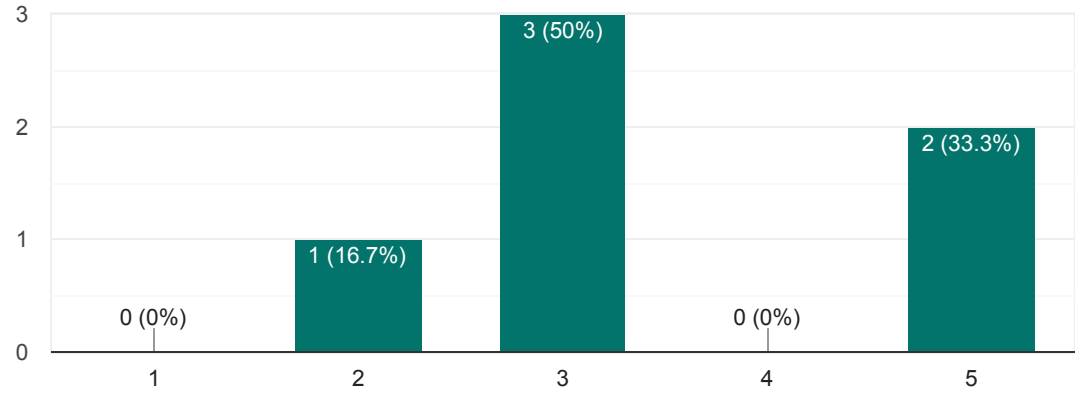
6 responses



Feature 5: the transitions between different views. Useful?

Copy

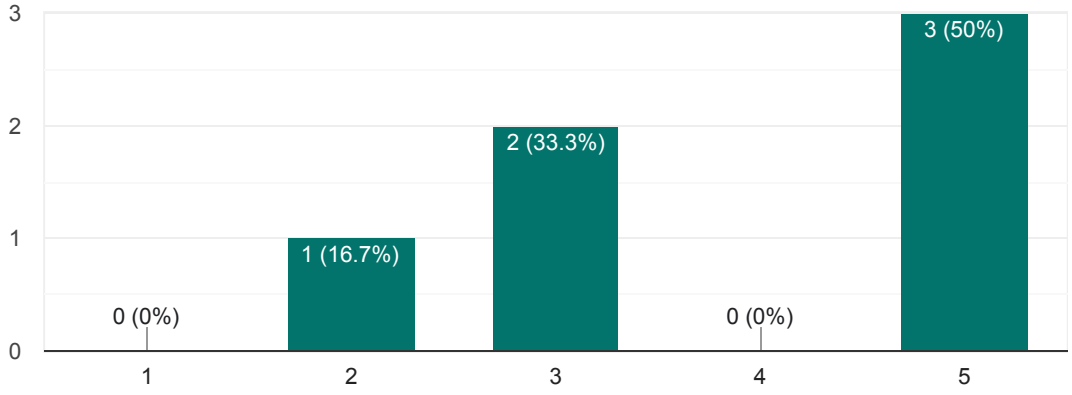
6 responses



Rating 1: SynopFrame will help you understand, communicate, and improve your designs

Copy

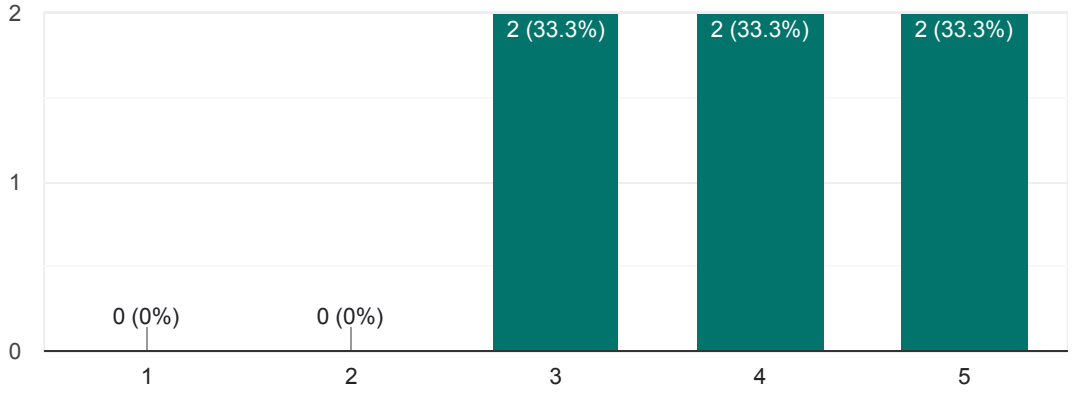
6 responses



Rating 2: overall, how would you rate SynopFrame?

Copy

6 responses



General feedback

5 responses

Very effective is to be determined after using the tool.

SynopFrame explores some interesting aspects of oxDNA trajectory visualizations. However it's really impractical to download a huge commercial tool, which is tailored towards video effects production just to have a look at a simulation.

practical

The video is not easy to understand out of context and without additional explanations.

I appreciate that SynopFrame could help me with my designs

Suggestions for improvement

4 responses

- Most parameters which need to be calculated are project specific so adding an interface to easily add new order parameter visualizations might help.
- Along these lines most of the order parameters are 1D
So having 2D / 3D plots of them might show relationships. (similar to the PCA view) but have 3 / 4 (if you count the energy) parameters of the users choice plotted out.
- Exploring the PCA view i was expecting interactive clicking through the coordinates, but somehow this was not implemented , browsing through the PCA space using the trajectory slider makes little sense.

As a user new to the software, a user-friendly interface or tool to import data into the program instead of requiring users to run a few oat analysis would be helpful. The tool could allow users to select and upload their dataset (just oxdna.dat and oxdna.top), and then automatically generate the necessary input files and folder structure required by SynopFrame. This would make it easier for users to get started with the program and increase its accessibility.

Describe the features before demonstrating them. For instance I haven't understood the hydrogen-bond color coding feature and couldn't follow in the video what I was supposed to see.

I think making the packaging/setup for SynopFrame easier might help make it more widely usable - since grad students are mostly the ones using this tool, saving time for them would always be appreciated. I think also being able to easily choose the sequences that require redesign and export them would also help.

Name

4 responses

anonymized

anonymized

anonymized

anonymized

Email

4 responses

anonymized

anonymized

anonymized

anonymized