PERDIX-2L

(**P**urine / pyrimidine **E**ngineering **R**outing **D**esign **I**nverse **X** – DX lattice)

Software Instructions and Demo

Dr. Hyungmin Jun

Laboratory of Computational Biology and Biophysics, MIT

An online version of this software is available at https://github.com/lcbb/PERDIX-2L.

Table of Contents

Welcome to PERDIX-2L!

PERDIX-2L simplifies and enhances the process of designing 2D DNA origami wireframe lattices from the CAD (Computer-aided Design) geometry as an input. With this software, you will be able to render almost any target 2D shape as a scaffolded DNA origami lattice array composed of DX-based edges. By providing a geometry file such as GEO (PERDIX geometry definition file format), IGES / IGS ("Initial Graphics Exchange Specification"), and PLY ("Polygon File Format") files of your design that describes your target 2D geometry, PERDIX-2L can generate the following outputs:

- A CVS file of the list of synthetic staple strand sequences. These staple strands, when combined with your scaffold strand (generated by default by PERDIX-2L or provided by you), will self-assemble into your scaffolded DNA origami nanostructure by following the standard annealing protocol provided in our work.
- A CNDO file of your nanostructures. This output CNDO file (CanDo file format[1]) from PERDIX-2L can be used to predict the flexibility of programmed DNA nanosturctures[2]. Also, it can be used to convert the PDB ("Protein Data Bank") file which gives the coordinates of every atom in your structure as predicted by PERDIX-2L. With software such as PyMOL[3], VMD[4], UCSF Chimera[5], etc., you will be able to visualize and manipulate your atomic model in 3D space.
- Several BILD[6] files. These BILD files are used for visualizing the target geometry, scaffold routing, staple sequence and cylindrical models, which are rendered by lines, polygons, and geometric primitives built in UCSF Chimera (see Fig. 5).
- A JSON file. This JSON file can be imported into caDNAno[7] software that allows to edit the staple paths and sequences.

One of the goals of this software is to broaden the usage of DNA nanotechnology to the larger community. We hope that, even if you are not an expert on DNA origami, you can use PERDIX-2L to begin to explore the capabilities of this powerful molecular design paradigm.

PERDIX-2L features:

- Fully automatic procedure of the sequence design for scaffolded DNA DX-based wireframe lattices
- Importing GEO, IGES/IGS, or PLY file formats
- Exact edge-lengths to design highly asymmetric and irregular shapes

---

[1] https://cando-dna-origami.org/cndo-file-converter/
[2] https://cando-dna-origami.org/atomic-model-generator/
[3] https://www.pymol.org/
[4] http://www.ks.uiuc.edu/Research/vmd/
[5] https://www.cgl.ucsf.edu/chimera/
[6] https://www.cgl.ucsf.edu/chimera/docs/UsersGuide/bild.html
[7] http://cadnano.org/

- JSON output for editing staple path and sequences from caDNAno
- 3D visualization powered by UCSF Chimera
- 24 pre-defined target geometries
- User-friendly TUI (Text-based User Interface)
- Executable files on Windows and Mac OS X systems
- Free and open source (GNU General Public License, version 3.0)

PERDIX variants:

- PERDIX-6P – Designer scaffolded DNA 6HB-based wireframe nanoparticles
- PERDIX-2L – Designer scaffolded DNA DX-based wireframe lattices

Part 1. Preparing geometry

It is tedious and time-consuming to discretize the target geometry with the polygon meshes and requires special software such as MeshLab[8], Gmsh[9] or Autodesk Netfabb[10]. For ease-of-use, PERDIX-2L uses an input target geometry specified by a set of straight lines that are automatically converted to polygon meshes by the external MATLAB and Python scripts (Shapely[11] and DistMesh[12]). Thus, the arbitrary target geometry can be simply defined by a set of points and their connectivity in the text file with the file extension as GEO (Part 1.1), or lines from pre-existing CAD software with GUI environments (Part 1.2).

Part 1.1. Definition of straight lines in the text file

PERDIX-2L is compatible with two drawing methods using straight lines; (1) drawing the border of their target object ("free-from boundary design", Fig. 1a), leaving the internal structure up to the algorithm, or (2) drawing the exact lines that will be converted into DX edges in the final origami ("free-from boundary design and internal mesh design", Fig. 1b).

Users can use general purpose text editors (e.g, Notepad++ and Atom) to create the editable input file (*.GEO) in which the number of points, lines, and faces should be sequentially contained at the first row. The following rows are corresponding to the x and y positions of points. Subsequently, following rows contain the connectivity of the lines if the number of lines are not zero, or of the faces if the number of faces are not equal to zero.
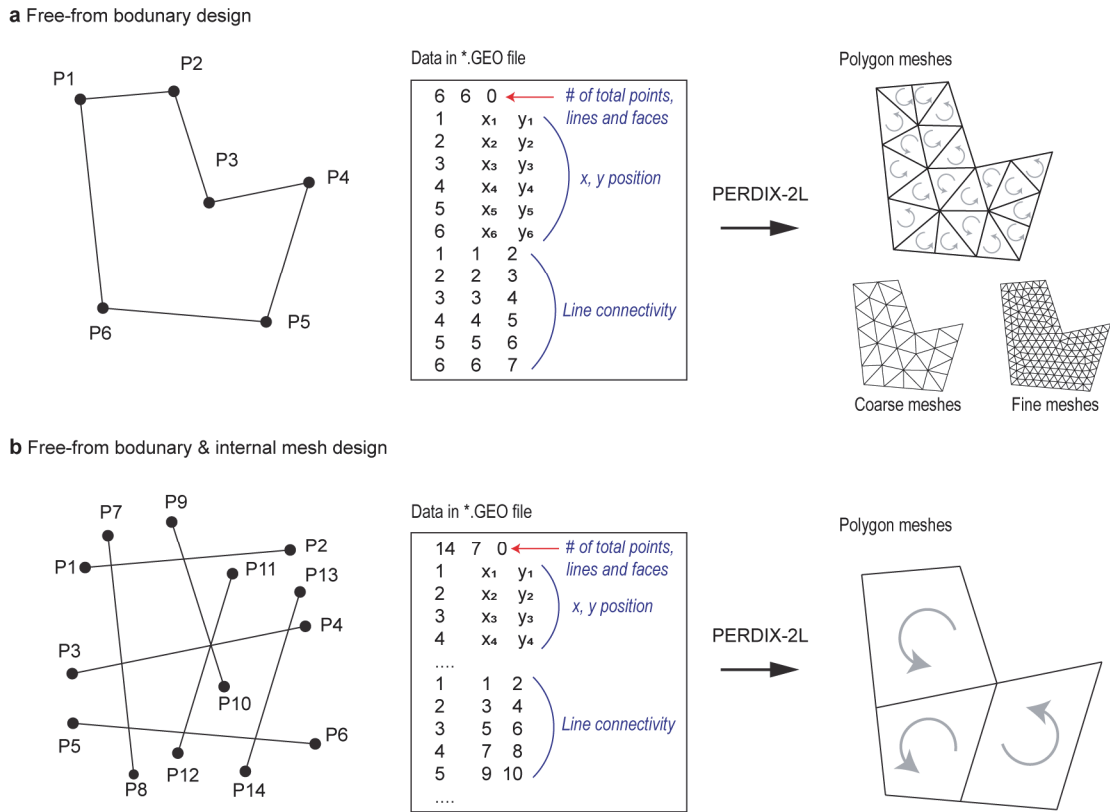
---

[8] http://meshlab.sourceforge.net/
[9] http://gmsh.info/
[10] https://www.autodesk.com/products/netfabb/
[11] https://pypi.python.org/pypi/Shapely
[12] http://persson.berkeley.edu/distmesh/

a Free-from bodunary design

Data in *.GEO file

Polygon meshes

```
6   6   0   ←— # of total points,
1   x₁  y₁      lines and faces
2   x₂  y₂
3   x₃  y₃
4   x₄  y₄      x, y position
5   x₅  y₅
6   x₆  y₆
1   1   2
2   2   3
3   3   4      Line connectivity
4   4   5
5   5   6
6   6   7
```

PERDIX-2L →

Coarse meshes   Fine meshes

b Free-from bodunary & internal mesh design

Data in *.GEO file

Polygon meshes

```
14  7   0   ←— # of total points,
1   x₁  y₁      lines and faces
2   x₂  y₂
3   x₃  y₃      x, y position
4   x₄  y₄
....
1   1   2
2   3   4
3   5   6      Line connectivity
4   7   8
5   9  10
....
```

PERDIX-2L →

Fig. 1 | Target geometry specification using straight lines in the text file, *.GEO. a, drawing the border of their target object, leaving the internal structure up to the algorithm and **b**, drawing the exact lines that will be converted into DX edges in the final origami. When the number of faces are one in case of the "free-form boundary design", triangular meshes are automatically filled in the border of target geometry with the mesh spacing parameters that determine the mesh density.

It is note that, for "Free-from boundary and internal mesh design", the internal polygon mesh is constructed when intersectional points are formed the closed loop and the branching lines that are not involved in the generation of polygon meshes are deleted. Two drawing methods can be easily integrated with pre-existing CAD software such as FreeCAD[13] with GUI environments (see the following section).

Part 1.2. Specifying the geometry from GUI drawing software
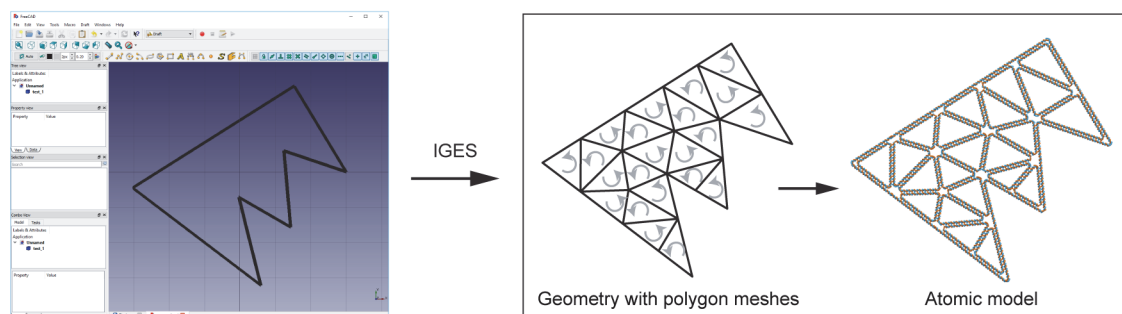
With the unique feature of the use of lines instead of polygon meshes, the pre-existing CAD software to render the target geometry can be easily integrated with PERDIX-2L. FreeCAD[14] software is the best option since it is open-source with the simple GUI interface and supports IGES/IGS format as an export format.

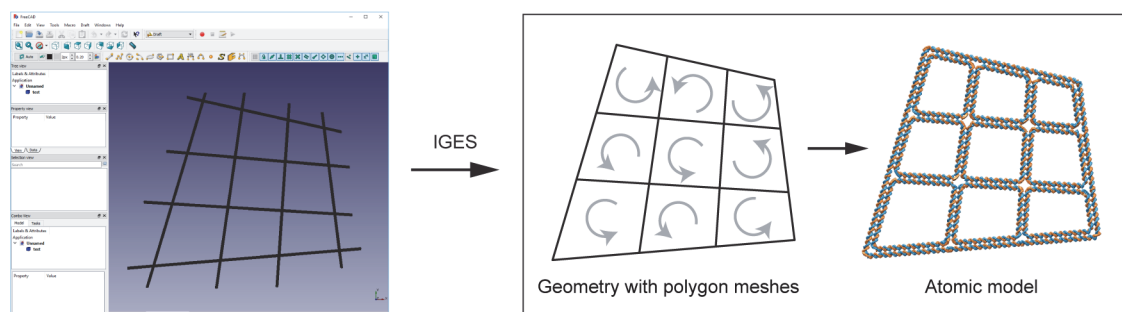[13] https://www.freecadweb.org/
[14] https://www.freecadweb.org/

After installing FreeCAD (on Windows or Mac OS X), open it and create a new empty document (Ctrl + N). To draw 2D geometry, change the workbench to 'Draft' (drop-down list in main toolbar). Select '2-points line' in the drawing tool and draw the target geometry. To save it, select all lines (Ctrl + A) and go the File → Export (Ctrl + E) and save as with IGES format (IGES or IGS). Finally, PERDIX-2L can import the IGES file format as an its input (see Part 2.1).

Fig. 2 | Drawing geometry from FreeCAD software and import the IGES output as an input of PERDIX-2L. a, Free-from boundary design. **b**, Free-form boundary and internal mesh design.

Part 2. Release package

The release package (as an executable file) is available for Microsoft Windows and Mac OS X system. These executable files are the console application for Windows (PERDIX-2L.EXE) and Mac OS X (PERDIX-2L), which accept an input (GEO, IGES/IGS, and PLY) and generate outputs to the console through the command prompt. You can download PERDIX-2L from

https://github.com/lcbb/PERDIX-2L/archive/master.zip

and find the release packages in the folder named as 'release' after extracting zip file. The current version of the release package is compiled with Intel (R) Visual Fortran compiler (Ver. 17.0.1.143) under 64-bit Microsoft Windows 10 with Intel(R) Core(TM) i7-4470 CPU @ 3.40GHz and Intel (R) Visual Fortran compiler (Ver. 18.0.1) under macOS 10.13 High Sierra, respectively. If you are a

user on a Linux system, you will need to download source codes and compile them properly under your OS (see Part 3). In the folder named as 'release', you will have the following subfolders and files:

- File 'Win/PERDIX-2L.EXE': This is an executable file to run PERDIX-2L under Microsoft Windows. The executable file can be run by double-clicking the icon (Part 2.1) or on command shell (Part 2.2).
- File 'Mac/PERDIX-2L-EXE': This is an executable file to run PERDIX-2L under Mac OS X by double-clicking. (Part 2.1).
- File 'Mac/PERDIX-2L': This is an executable file to run PERDIX-2L under Mac OS X. The executable file can be open using Terminal (Part 2.2).
- File 'env.TXT': This text file contains the sequences of the scaffold as input. The sequences can be replaced with the user-defined sequences of the scaffold (see Table 1).
- Folder 'tools': This folder contains the MATLAB script of DistMesh to generate automatic triangular meshes and the Python script to convert a set of lines to polygon meshes using Shapely.
- Folder 'input': The user-defined geometry file (GEO, IGES/IGS, or PLY) must be here.

Table 1. The sequences must be defined at third line and the 'para_scaf_seq' must be defined as 1 to use user-defined scaffold sequence.

Field	Value	Descriptions
para_platform	win | mac	Depending on user's operating system
para_scaf_seq	0 | 1 | 2	Scaffold sequence • 0: M13mp18(7249nt) sequence • 1: User-defined sequence • 2: randomly generated sequence

Part 2.1. Opening the PERDIX-2L software by double-clicking

The easiest way to run PERDIX-2L is to double-click 'PERDIX-2L.EXE' in the file manager of the Windows, or 'PERDIX-2L-EXE' in the Finder of the Mac. Note that the text file 'env.TXT' should be in the same folder in order to properly run the software. The user-defined geometry file (GEO, IGES/IGS, PLY) should be in the folder named as the 'input'. By double-clicking the release package, you can see the TUI (Text based User Interface) on the console, which displays the pre-defined target geometries as first input parameters (Fig. 3). There are 24 pre-defined wireframe lattices with the triangular, quadrilateral, and *N*-polygon meshes. If you have your own your geometry file, just type the name of geometry file with its extension (GEO, IGES / IGS, or PLY).

Note: Make sure that PERDIX-2L can only read the PLY file format in ASCII[15] (If you open the PLY file externally, it should be human-readable). Some PLY files obtained from external sources have been found to have errors, like missing vertices or vertices with coordinates that do not

---

[15] http://paulbourke.net/dataformats/ply/

belong to any face. To make your custom PLY file correct, or to convert another structure file format into PLY, you can use some software such as MeshLab[16], Gmsh[17] or Autodesk Netfabb[18].

```
+==============================================================================+
|                                                                              |
| PERDIX-2L by Hyungmin Jun (hyungminjun@outlook.com), MIT, Bathe Lab, 2018     |
|                                                                              |
+==============================================================================+

A. First input - Pre-defined 2D target geometry
================================================

 [ Triangular-mesh objects ]
    1. Square,                2. Honeycomb
    3. Circle,                4. Wheel,                   5. Ellipse

 [ Quadrilateral-mesh objects ]
    6. Rhombic Tiling,        7. Quarter Circle
    8. Cross,                 9. Arrowhead,               10. Annulus

 [ N-polygon-mesh objects ]
    11. Cairo Penta Tiling,  12. Lotus
    13. Hexagonal Tiling,     14. Prismatic Penta Tiling,  15. Hepta Penta Tiling

 [ Variable vertex-number, edge-length, and internal mesh ]
    16. 4-Sided Polygon,     17. 5-Sided Polygon,         18. 6-Sided Polygon
    19. L-Shape [42-bp],     20. L-Shape [63-bp],         21. L-Shape [84-bp]
    22. Curved Arm [Quad],   23. Curved Arm [Tri],        24. Curved Arm [Mixed]

 Select the number or type geometry file (*.geo, *.igs, *.iges, *.ply) [Enter] :
```

Fig. 3 | The first parameter in PERDIX-2L software. The 24 pre-defined target geometries as the first input parameter of PERDIX-2L. Users can use the own their geometry with typing the geometry file name with file extensions (GEO, IGES/IGS, or PLY). The negative value as an input terminates this console application immediately.

The second input shown in Fig. 4 is to select the minimum edge length which is assigned to the shortest edge and the other edges are scaled. User can directly type the any arbitrary edge length that should be larger than 37. Also, user can specify the edge ID to apply the minimum edge length (see Part 2.2). With two inputs, it runs and creates the new folder named as 'output' where PERDIX-2L automatically generates the several outputs (Table 3).

---

[16] http://meshlab.sourceforge.net/
[17] http://gmsh.info/
[18] https://www.netfabb.com/

```
        B. Second input - Pre-defined minimum edge length
        ====================================================

          * 1.   42 bp =  4 turn * 10.5 bp/turn ->  42 bp * 0.34nm/bp = 14.28nm
            2.   52 bp =  5 turn * 10.5 bp/turn ->  52 bp * 0.34nm/bp = 17.85nm
          * 3.   63 bp =  6 turn * 10.5 bp/turn ->  63 bp * 0.34nm/bp = 21.42nm
            4.   73 bp =  7 turn * 10.5 bp/turn ->  73 bp * 0.34nm/bp = 24.99nm
          * 5.   84 bp =  8 turn * 10.5 bp/turn ->  84 bp * 0.34nm/bp = 28.56nm
            6.   94 bp =  9 turn * 10.5 bp/turn ->  94 bp * 0.34nm/bp = 32.13nm
          * 7.  105 bp = 10 turn * 10.5 bp/turn -> 105 bp * 0.34nm/bp = 35.70nm
            8.  115 bp = 11 turn * 10.5 bp/turn -> 115 bp * 0.34nm/bp = 39.27nm
          * 9.  126 bp = 12 turn * 10.5 bp/turn -> 126 bp * 0.34nm/bp = 42.84nm

        Select the number or type the minimum edge length [Enter] :
```

Fig. 4 | The second input to determine minimum edge lengths. The negative value as an input terminates PERDIX-2L immediately.

Part 2.2. Running PERDIX-2L with command prompt

PERDIX-2L can run through the command shell (command console / Terminal). In Windows, start a command shell with **Start → run → cmd** (enter) or type **cmd** in Search Windows then use the 'cd' command to move to the folder where the PERDIX-2L package exists. To access the Unix command prompt in Mac OS X, open the terminal application. It is located by default the Utilities folder, which in turn is inside the Applications folder. PERDIX-2L can be run with the following 3 arguments (Table 2) from the command shell.

PERDIX-2L.exe argc1 argc2 argc3   for Windows

./PERDIX-2L argc1 argc2 argc3   for Mac OS X

Table 2. Command line arguments for PERDIX-2L.

Arguments		Descriptions
argc1	String	The file name of the target geometry (including the file extension) Ex) square.geo / square.iges / from 0 to 24 (to select the pre-defined geometry)
argc2	Integer	Specific edge ID, user can check the edge ID in the output file named as "_03_sep_line.bild" file. See Fig. 5. The shortest edge in the target is assigned when Opt2 = 0. Ex) 1 – Edge that has ID of 1
argc3	Integer	The minimum edge length, which is any number but greater than 37-bp, to have at least two double-crossover per edge. Ex) 42 – 42-bp as minimum edge length

For example, the square with the 48-edge length at edge ID 2 can be generated by the command as below:

PERDIX-2L.exe square.geo 2 48   for Windows

./PERDIX-2L square.geo 2 48   for Mac OS X

Use can run PERDIX-2L.EXE through the command shell on Mac and Linux environments, after installing Wine[19] which is a free and open-source compatibility layer that aims to allow computer programs developed for Microsoft Windows to run on Unix-like operating systems. We successfully tested PERDIX-2L working with Wine on Mac and Linux systems.

Part 3. Compiling source code

You can download the source code PERDIX-2L in zip format from

https://github.com/lcbb/PERDIX-2L/archive/master.zip

or browse the codes on GitHub,

http://github.com/lcbb/PERDIX-2L

You can also clone the project with Git[20] by running:

$ git clone https://github.com/lcbb/PERDIX-2L.git

The source codes for this project were written in Fortran 90/95. Fortran is a general-purpose, imperative programming language that is especially suited to numeric computation and scientific computing. It is also stable and fast in high performance computing and simulations. In order to compile Fortran source codes, user can install the Fortran compiler such as gFortran, Intel Fortran, PGI (formerly The Portland Group, Inc) Fortran. gFortran is developed under the GNU Fortran project which provides a free Fortran 95/2003/2008 compiler for GCC (GNU Compiler Collection). Intel(R) Fortran Compiler known as IFORT was developed by Intel and available for Linux, Windows and Mac OS X. We have developed this project under Intel(R) Fortran Compiler which is available under a free, non-commercial license for qualified students, educators, academic researchers and open source contributors on Linux, OS X and Windows[21]. Before installing Intel(R) Fortran Compiler, user must have a version of Microsoft Visual Studio installed since the Intel Fortran Compiler integrates into the following versions of Microsoft Visual Studio: Visual Studio 2012 to 2015. Microsoft Visual Studio Community is also free for non-commercial use and it can

---

[19] https://www.winehq.org/
[20] https://git-scm.com/
[21] https://software.intel.com/en-us/qualify-for-free-software

be downloaded from here[22]. Note that if the installer does not find a supported version of Visual Studio (If user do not install Visual Studio), a Fortran-only development environment based on the Microsoft Visual Studio 2013 Shell is provided (thus, PERDIX-2L can only be compiled on command).

Here, under Windows systems and Mac OS X, we will explain how to compile the source codes of PERDIX-2L in two ways as follow:

- Compiling source codes on command (see Part 3.1)
  - o The source codes of PERDIX-2L can easily compiled by the build automation tool under Linux and Mac OS X.
- Compiling sources through Visual Studio IDE (Integrated Development Environment) – Window only (see Part 2.2).
  - o It provides comprehensive facilities to computer programmers for software development such as a source code editor, build automation tools, a debugger, etc. Microsoft Visual Studio is IDE for Fortran compiler, which can run only under Windows operating system. The users for Linux and Mac can find the alternative IDE, Xcode which is support to Intel Fortran.

Part 3.1. Compiling source codes on command

'Makefile' is a simple way to organize or control code compilation. If already installed the Apple developer tools on Mac, user can use 'make' command on the terminal. Windows supports a variation of 'makefiles' with its 'nmake' utility. If we have a version of Microsoft Visual Studio installed, we can use NMAKE in Visual Studio Command Prompt to run 'Makefile'.

(*Alternatives*) The GnuWin32 project provides Win32-version of GNU tools, much of it modified to run on the 32-bit Windows platform. User can download the Window version of MAKE from Gnuwin32 project[23]. The easiest way to use the tools is to add them to your search path using the `PATH' environment variable, usually by prepending the /bin folder to your PATH variable.

We provide 'Makefile' in the folder named as 'make/makefiles'. User should copy the 'Makefile' file to where the source codes exist. The source code is located at the folder named as "src". Follow these steps to invoke the compiler from the command line:

1. For Windows, open the **Start** menu, and under the 'Intel Parallel Studio XE product group', select a compiler command prompt.

    ex) **Start** → **Program** → Intel Parallel Studio XE 2015 / 2016 / 2017 → Compiler 17.0 Update 1 for IA-32 Visual Studio 2015 environment.

    For Mac OS X, open **Terminal**.

2. Use the **cd** command to move in the folder named as 'src'.

---

[22] https://www.visualstudio.com/vs/community/
[23] http://gnuwin32.sourceforge.net/packages/make.htm.

3. Copy the 'MakeFile' in the 'src' folder.
4. Type '**make**' to invoke the compiler using 'Makefile'.
5. After compiling sources, you will have the executable file named as 'PERDIX-2L.EXE' for Windows and 'PERDIX-2L' for Mac OS X.
6. To delete object files generated during the compilation, type '**make clean**'.
7. Make sure that the 'env.TXT' file must be at the same folder where the executable file exist, to run PERDIX-2L.
8. You can open and modify source codes (*.F90) by the general text editor such as Sublime Text, Notepad++, Vim, Atom, Nano, Emacs, and etc.

It is note that, by the same way mentioned above, we successfully compiled the source codes of PERDIX under RedHat Linux using the Linux version of the Intel Fortran Compiler.

Part 3.2. Compiling sources on Microsoft Visual Studio

First, check Microsoft Visual Studio version supported by versions of the Intel compilers. This project was developed under the Intel Parallel Studio XE 2015 / 2016 / 2017 with Microsoft Visual Studio 2015 / 2016 / 2017.

- Launch Microsoft Visual Studio.
- Select **File** > **New** > **Project** to make new project
- In the New Project window, select Empty project under **Intel(R) Visual Fortran**.
- copy all source files and one text file, 'env.TXT' into project folder and added these in project directory
- Select **Build** > **Build Solution (F7)**
- Select **Debug** > **Starting Without Debugging (ctrl + F5)**
- The results of the compilation display in the **Output** window

Part 4. Outputs

Once the sequence design from PERDIX-2L is completed, the output folder is created. The files, 'TXT_PERDIX_2L.TXT' and 'TXT_Sequence.TXT', contain information on all events for sequence design process and the results of the sequence and routing of the scaffold and staples, respectively. The file, '_17_sequence.CSV', contain generated sequences of the staples with the given sequence of the scaffold. Several BILD files are ASCII format that describes lines, polygons, and geometric primitives for the visualization of the geometry, routing, strands, edge-staple and so on. You will be also able to visualize these set of data by UCSF Chimera (Fig. 5 and Table 3).
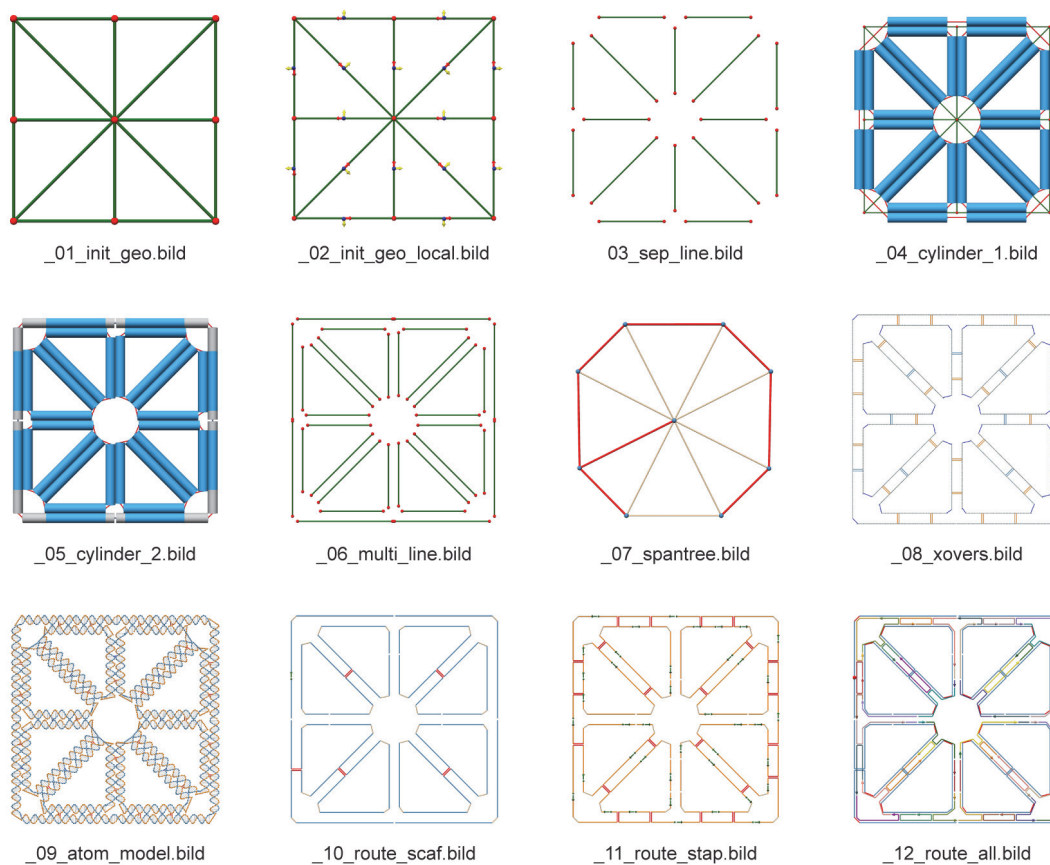
Fig. 5 | 12 rendering from BILD outputs for 42-bp edge length DNA L-shape.

With the JSON file as one of outputs from PERDIX-2L, user can edit the staple crossover positions and sequences using caDNAno (Fig. 6).The file named as '_15_json.guide.BILD' can be loaded in USCF Chimera, which give the information which edges of the target structure is associated with the which cross-sections of caDNAno representation.
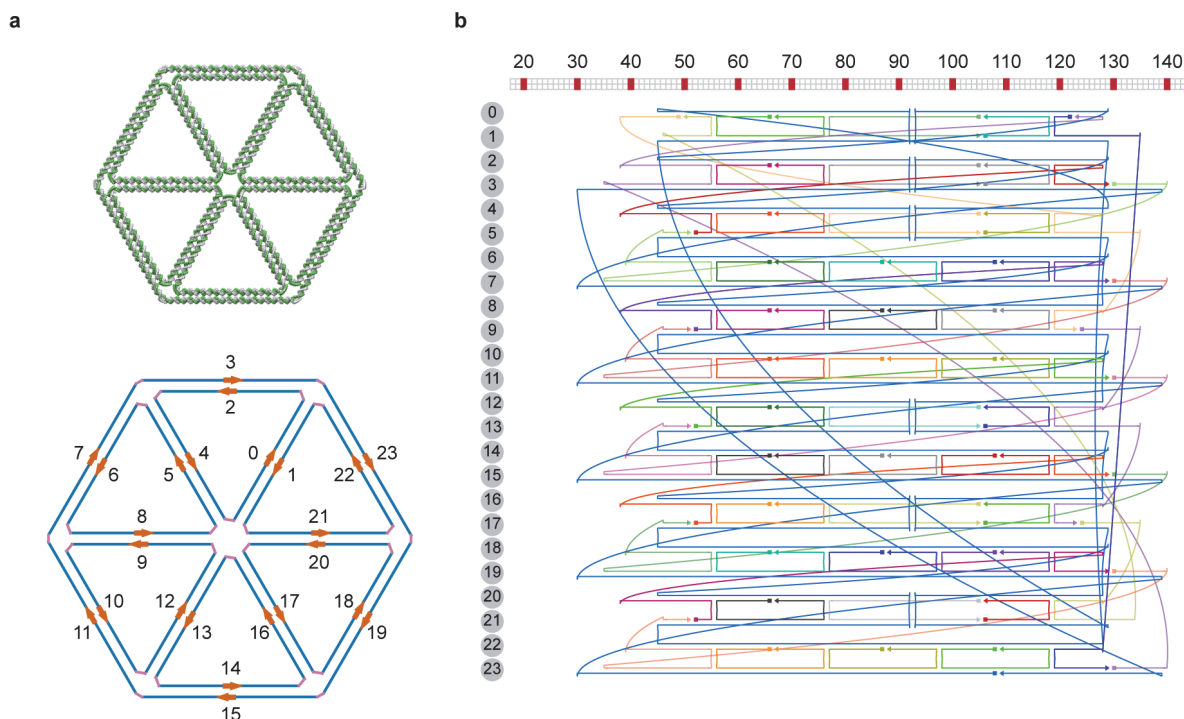
Fig. 6 | JSON caDNAno and guide BILD outputs. a, JSON guide model in which the edge numbers are associated with the cross-section number in caDNAno. **b,** Staple and scaffold organization from caDNAno.
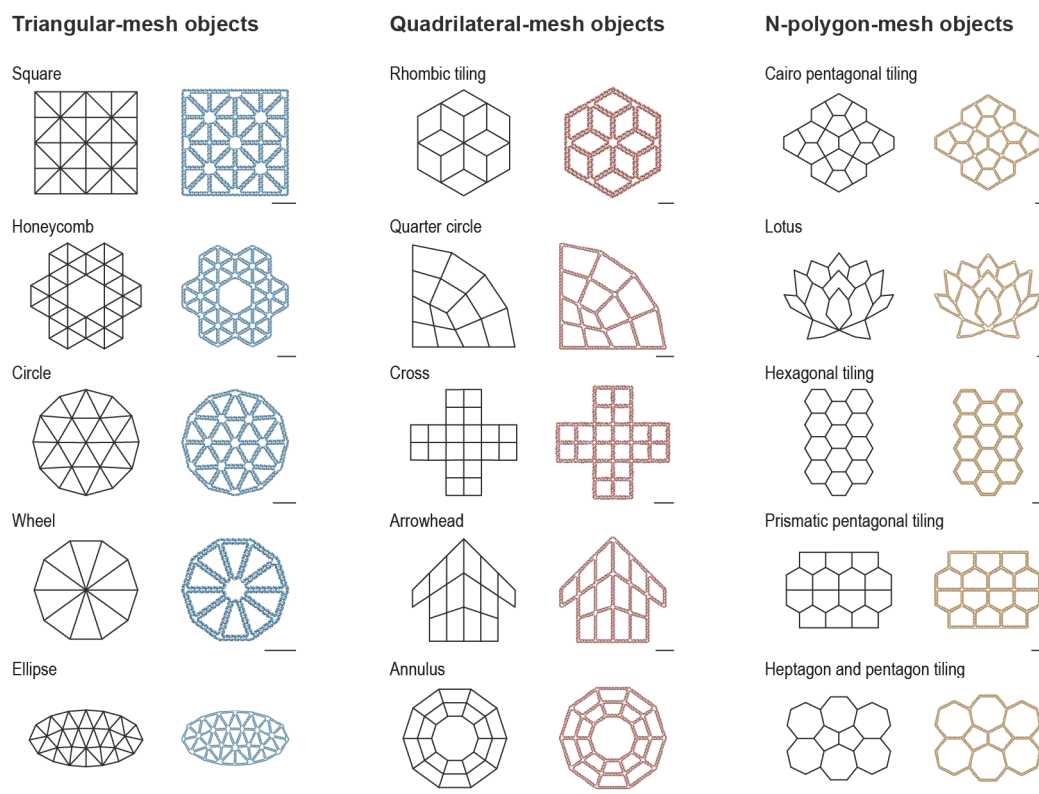
Table 3. Descriptions of 12 BILD outputs generated by PERDIX-2L.

BILD file	Colored object	Description
_01_init_geo	Red circle	Point of the target geometry
	Green edge	Edge of the target geometry
_02_init_geo_local	Red circle	Point of the target geometry
	Green edge	Edge of the target geometry
	Red arrow	Local vector, t_1
	Yellow arrow	Local vector, t_3
	Blue arrow	Local vector, t_2
_03_sep_line	Red circle	Point separated from the vertex
	Green line	Line connecting two points
_04_cylinder_1 / _05_cylinder_2	Blue cylinder	Double helix DNA strand
	Grey cylinder	Extended part to fill the gap
	Red line	Scaffold strand crossing the vertex
_06_multi_line	Red circle	End point of the double helix
	Green line	Double helix DNA strand
_07_spantree	Black circle	Node of the dual graph
	Blue line	Non-member of the spanning tree

	Red line	Member of the spanning tree
_08_xovers	Yellow circle	Base pair
	Blue line	Scaffold crossover
	Orange line	Staple crossover
	Dark blue line	Scaffold crossing the vertex
_09_atom_model	Blue line	Scaffold strand
	Orange line	Staple
_10_route_scaf	Blue line	Scaffold strand
	Red line	Scaffold crossover
_11_route_stap	Orange line	Staple
	Red line	Staple crossover
_12_route_all	Blue line	Scaffold strand
	Multiple colored line	Staple strand

With the CNDO (The CanDo file format) file, which was designed to describe DNA nanostructures, contains sufficient information to generate the all-atom models of these DNA nanostructures. The atomic model generator[24] uses the CNDO file as its input and creates the PDB file consisting of two phosphates, two deoxyriboses, and two paired bases (Fig. 7). This atomic model generator is written by a MATLAB script which produces a *.PDB file, which can be similarly visualized using UCSF Chimera.

---

[24] https://cando-dna-origami.org/atomic-model-generator/

Triangular-mesh objects

Square

Honeycomb

Circle

Wheel

Ellipse

Quadrilateral-mesh objects

Rhombic tiling

Quarter circle

Cross

Arrowhead

Annulus

N-polygon-mesh objects

Cairo pentagonal tiling

Lotus

Hexagonal tiling

Prismatic pentagonal tiling

Heptagon and pentagon tiling

Fig. 7 | Atomic model of 15 diverse wireframe lattices generated by PERDIX-2L. Scale bar for atomic structures is 20 nm.