Design Tools for Reinforced 3D DNA Nanostructures

Supervisor: Prof. Pekka Orponen
Advisor: Dr. Eugen Czeizler

Bachelor thesis by:
Oriol Corcoll Andreu

Contents

Abstract

The field of DNA nanotechnology has grown since Nadrian Seeman founded it in 1980 creating a three dimensional structure made of DNA. The field has grown and there are many techniques to build these structures. One of them is DNA origami developed by P. Rothemund in 2006. This technology allows us to build two-dimensional and three-dimensional structures made with a single long strand called *scaffold* and multiple small strands called *staples* made of DNA. These structures can have very long edges, if this happens then the structure could collapse.

This problem can be solved reinforcing the long edges with rigid substructures. In this document we present an algorithm which takes a structure and a path for the scaffold as input and reinforces its long edges with a *bipyramid*-like structure. Also we give a tool to visualise the original structure, its reinforced version and the modified path for the scaffold.

1 Introduction

DNA carries the genetic code of life. This has been the only use of DNA in the past. The intrinsic properties of DNA make it a good candidate to use it as material. Nowadays, it is used to design and build nanostructures. Nadrian Seeman was the first one to build a structure with DNA. After him more people have developed techniques to build structures with DNA. Paul Rothemund is one of them, he developed a technique to fold DNA to create two dimensional shapes. This technique uses a long strand of DNA and many small strands of DNA to build the shape. A structure can be seen as a graph and we can route the long strand of DNA for the structure as a path over its graph representation.

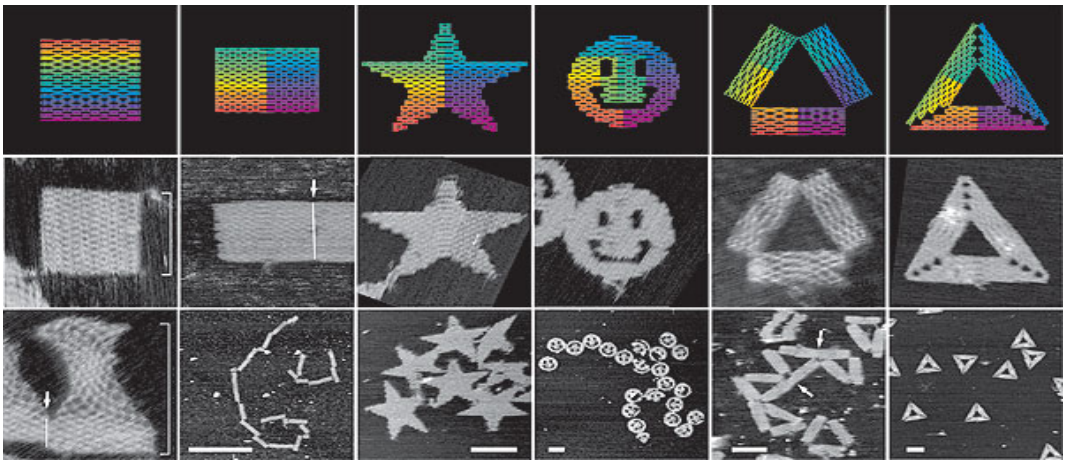

Figure 1.1: Shapes made with DNA origami. Reprinted with permission from Nature [5].

The use of DNA in nanotechnology for the design of shapes and patterns has grown considerably with the popular DNA origami technique. This technique let us fold a single long strand of DNA with the help of small strands to make a rigid two dimensional shape and also a three dimensional structure. These nanostructures can be used to encapsulate drugs, proteins and other nanomaterials. After the encapsulation, drugs can be released towards targeted cells, and proteins can be controllably folded.

1.1 Problem statement

Huge three dimensional structures can be made with the DNA origami technique. These huge structures could have large edges which can affect to the rigidity of the structure. A structure which is not rigid enough can be deformed or even collapse. To avoid this situation these large edges have to become rigid. We can achieve this by replacing the original large edge with a new rigid structure.

We have designed an algorithm that takes a structure and a path over the structure as input and reinforce the large edges in the structure and modifies the original path to go through the reinforced structure. The algorithm outputs the reinforced structure and the new path over it. Also we have developed a visualiser that takes a structure and a path as input and shows a three dimensional representation of the structure where we can navigate to see it. We can see also the path through an animation.

1.2 Structure of the thesis

In Chapter 2, we expose the basic concepts that we will need to understand this document. We talk about the DNA as nanomaterial, how it is useful to build structures and which are the techniques to build those structures. We also give some basic definitions in the graph theory field and we define what is a valid stapled path.

In Chapter 3, we discuss how the scaffold strand is elaborated and how it is used in the DNA origami method. We define the rigidity property and we present a rigid structure that we will use to reinforce a large structure.

In Chapter 4, an algorithm to reinforce large structures is presented. We explain the construction of the new reinforced structure from the original structure and how we modify the original path to go over the new structure.

In Chapter 5, the visualisation tool is exposed with an example. We also explain the technology used to develop it.

In Chapter 6, we explain our conclusions about the problem and our solution.

2 Background

Nanotechnology is a technology that deals with small-sized materials, usually between 1 nm and 100 nm. The materials in this size range have some specific and valuable properties. The concept of arrangement of small components into complex assemblies is called *bottom-up* constructions. DNA nanotechnology makes use of this approach in combination of *self-assembly*, where these components rearrange autonomously, to organise stable structures from molecular components.

Since Watson and Crick defined the structure and composition of DNA [1] different applications for DNA has appear. DNA was first used as a nanomaterial by Nadrian Seeman in 1980 [2]. Seeman and co-workers synthesized the first three dimensional nanoscale object, a cube in 1991 [3]. In 1998, Seeman in collaboration with Winfree published the creation of two dimensional lattices of tiles [4]. This tile-base approach has been used by Winfree and Rothemund in the DNA computing field as Wang tiles to perform computations. DNA nanotechnology has been used in DNA computing, DNA nanorobots and DNA nanostructures.

The graph theory can help us to route the scaffold over a structure trying to minimise the amount of DNA used. In this section we will give some basic concepts about graph theory like: paths, trails, circuits and Eulerian graphs.

2.1 DNA as a nanomaterial

As Watson and Crick showed, DNA has a double helix structure [1]. Each strand of the double helix is composed of nucleotides, adenine (A), cytosine (C), guanine (G) and thymine (T). The strands are connected by pairs of nucleotides, an adenine nucleotide can be paired only with a thymine nucleotide and a cytosine nucleotide can be paired only with a guanine nucleotide. Depending on the sequence of nucleotides the strand is called a *sense strand* or an *antisense strand*. If a strand sequence has the same sequence as the messenger RNA then it is called sense strand, the complementary is called antisense strand. There are three possible conformations of DNA: A-DNA,

B-DNA and Z-DNA. Over all these conformations, B-DNA is the most common form. It is 2.37 *nm* wide and 10 base pairs are 3.4 *nm* and the double helix makes a turn every 10.5 base pairs. The two helices are not equally spaced along the axis, but asymmetrically as a *minor groove* and a *major groove*.

Self-assembly is the process where a collection of simple components, starting in a disorganised state, autonomously combine into a more complex structure. This process is done without external guidance, the components experience only local interactions and typically obey a simple set of rules that describe how they combine. DNA nanotechnology uses strands of DNA as components to self-assembly nanostructures. There are three approaches to design the nanostructures: *tile-based*, *folding* and *dynamic*.

2.2 DNA origami

In 2006, Paul Rothemund [5] at the California Institute of Technology proposed a method to fold a long single strand of DNA called scaffold into almost arbitrary two dimensional shape with the help of short single strands of DNA called staple strands. This method consists in five steps, the first two by hand and the last three aided by computer.

The first step is to design a geometric model of a DNA structure that approximates the desired shape. The model is made with cylinders that represent one turn of the helix and an array of crossovers that represent the places to put the staple strands. The second step proceeds by folding a single scaffold strand over the model designed at the first step. The only constraint on the folding path is that the scaffold can be folded only at the locations where the DNA twist is at a tangent point between helices. The third step is the design of the staple strands corresponding to the complement of the sequence in the scaffold strand. In the fourth step, the twist of the scaffold is calculated and the position is changed to minimise the strain. The last step merges adjacent staple strands to yield fewer and longer strands.

Once the scaffold and the staple strands are designed to create the desired shape, they are mixed in a pot and the mixture is rapidly heated, and then cooled to get the desired shape. In the Rothemund paper, the selected shapes to prove that the method works were squares, rectangles, triangles, stars and smiley faces. He used the genomic DNA of the virus M13mp18 to create those shapes, and observed that the triangle was the most stable shape among all

the shapes created.

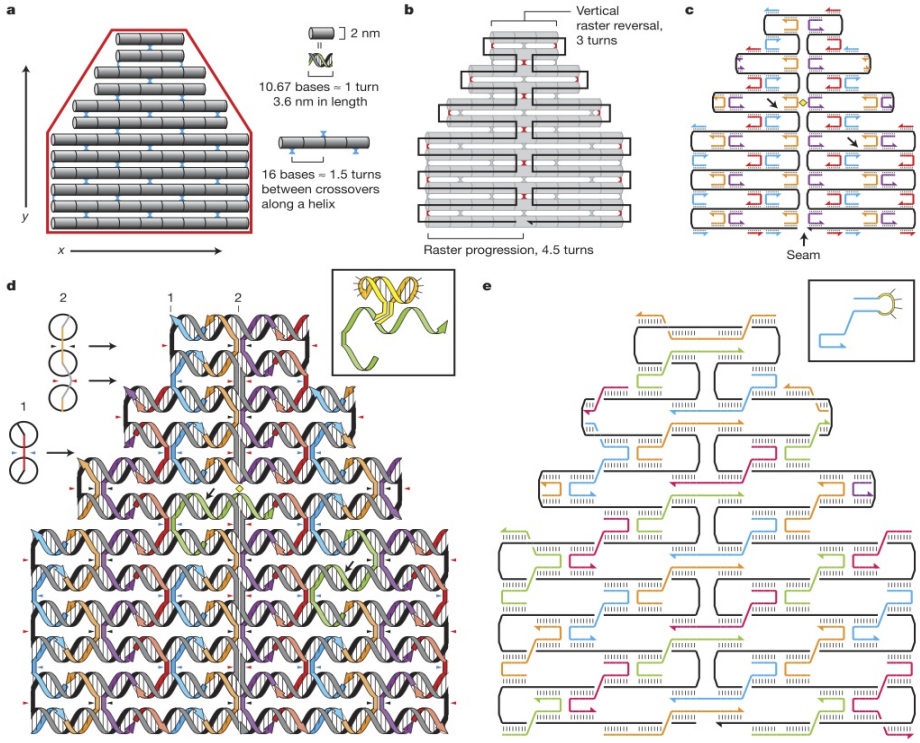

Figure 2.1: Scaffolded DNA origami design steps. Reprinted with permission from Nature [5].

The shapes created by Rothemund were two dimensional but the method has been extended [6] to create also three dimensional shapes. Andersen et al. have constructed a box with the DNA origami technique [7], and Bhatia and co-workers have demonstrated a gold nanoparticle trapped in an icosahedral DNA nanocapsule [8]. He et al. have assembled three different symmetric supramolecular polyhedral frameworks from sticky-ended three-point-star motifs in a hierarchical strategy [9].

Douglas et al. have developed the software CaDNAno, an open source graphical tool that helps with the design of three dimensional structures [10]. CaDNAno simplifies and enhances the process of designing three dimensional DNA origami nanostructures. Through its user-friendly two and three dimensional interfaces it accelerates the creation of arbitrary designs. Another approach is a new origami technique, i.e. the beam-scaffolding technique where the scaffold path forms a polyhedron and the staples are used to

fix the shape of the polyhedron. With this technique the designed structure has to be triangulated in order to be rigid.

2.3 Eulerian circuit

A diagram with lines and dots is a useful tool to describe many real-world situations. We can represent concepts such as people and their relationships or cities and the road connections between them. A mathematical abstraction of this concept is called a *graph*. A graph G is a pair of sets (V, E), where V is the set of *vertices*, also written as $V(G)$, and E is the set of *edges*, also written as $E(G)$. For our purposes we define an edge as a 2-element subset of V, i.e., an edge is an unordered pair of vertices. We will allow multiple edges with same vertices, i.e.$e_i, e_j \in E$ with $e_i = e_j$ and $i \neq j$. Loops will not be allowed, that is, for all $e = \{v, u\} \in E$ it must be the case that $v \neq u$. Graphs with unordered pairs of vertices as edges are called undirected graphs.

Two vertices $v, u \in V$ are *adjacent* if there is an edge $\{v, u\} \in E$. Vertices v and u are *incident* with the edge $e = \{v, u\}$. For a vertex v in V, the *degree* of v is the number of incident edges of v in G and it is denoted by $g(v)$. The minimum degree among all vertices of the graph is denoted by $\delta(G)$ and the maximum by $\Delta(G)$. If the minimum degree equals the maximum degree, i.e. $\delta(G) = \Delta(G)$, then G is called *regular*, or $\delta(G)$*-regular*. As Euler proved in 1736, the sum of all degrees in a graph is twice the number of edges, that is, $\sum_{v \in V(G)} d(v) = 2|E(G)|$. This theorem is also called the *handshaking lemma*. As a consequence of the theorem we have that the number of vertices with odd degree is even. Another important result derived from the handshaking lemma is that if a graph has exactly two vertices with odd degree then there is a path between them [11].

Let $G = (V, E)$ be a graph. A *walk* in G is a sequence v_0,e_1,v_1,e_2,...,e_n,v_n where $e_i = \{v_{i-1}, v_i\}$ is an edge in E; n is the *length* of the sequence. If for all $1 \leq i < j \leq n$ we have $e_i \neq e_j$ then the sequence is called a *trail* in G. If the starting vertex of the trail and the ending vertex are the same then the trail is called a *circuit*. If for all $0 \leq i < j \leq n$ we have $v_i \neq v_j$ then the sequence is called a *path* in G. A graph is called *connected* if there is a path between any pair of its vertices; otherwise, it is called *disconnected*. We will assume that our graphs are connected.

A graph with a trail that goes through all the edges in the graph has an *Eulerian trail* and is called *semi-Eulerian graph*. Similarly, if a graph has a

circuit that visits every edge in the graph then the graph has an *Eulerian circuit* and is called a *Eulerian graph*. As Euler showed in 1736 [11], the necessary condition for the existence of an Eulerian circuit is that every vertex in the graph has even degree. This condition is also sufficient, as was proved, 137 years later, by Carl Hierholzer [12]. The proof was later published by Wiener after the death of Hierholzer. This theorem can be extended to Eulerian trails: there is an Eulerian trail if and only if there are only zero or two odd degree vertices in the graph. More precisely, in the case of zero vertices with odd degree we have an Eulerian circuit and in the other case we have an Eulerian trail with different starting and ending vertices. There are two well known algorithms to find an Eulerian circuit in an Eulerian graph. The first one is Fleury's algorithm and was designed by M. Fleury in 1883, it works in $O(|E|^2)$. The results of Hierholzer gives a better linear time algorithm than Fleury's algorithm. This result is important for us because we have to find a path for the scaffold that goes over every edge in the structure and we would like to use every edge only once.

Eulerian graphs and semi-Eulerian graphs have many applications, e.g. route maps for airplanes, planning a trip around the world or a tour in a museum.

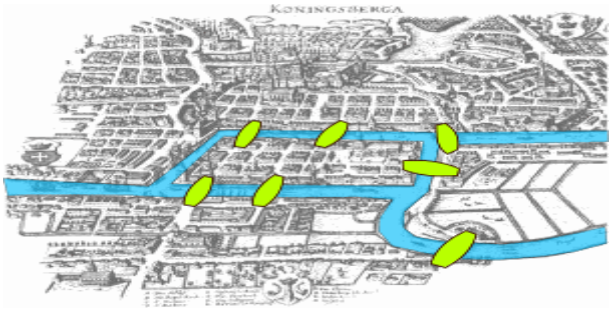

Figure 2.2: Seven Bridges of Königsberg.

Researchers have been sequencing all kinds of DNA, including the human genome. The current methods to sequence DNA is to split the whole sequence in smaller sequences. Assembling the sequences after processing each of them is a complicated process. Pevzner et al. have used Eulerian graphs to assembly DNA fragments [13]. The proposed algorithm splits each sequence in smaller ones, and researchers can then see the sequences as an Eulerian graph with polynomial algorithms to assembly the sequences.

Another application is the circuit layout area minimisation problem, which

is believed to be intractable. The order of the logic gates is a major factor in this problem. Kuntal Roy [14] approaches this problem as an Eulerian graph where one tries to find an order for CMOS logic gates.

2.4 Valid stapled path

A beam structure can be represented by a graph where the joints in the structure are vertices in the graph and the beams are edges. This concept can be used with nanoscale structures made of DNA. We can see the three dimensional nanostructures made with the DNA origami technique as graphs. Similarly, we can see the scaffold path of the DNA nanostructure as an Eulerian trail in the graph, but over all these Eulerian trails only a subset of them are *valid stapled paths*.

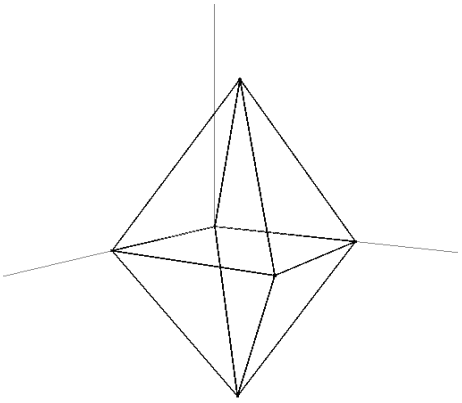

Figure 2.3: Design of a beam structure as a graph, octahedron.

The nanostructure can be seen as a solid object, more precisely as a polyhedron. A polyhedron is a solid object made of flat faces. We can define a *face* in our graphs analogously as a face in a polyhedron. As a consequence of this definition of face we have that every edge is *adjacent* to exactly two faces. An Eulerian trail is a *valid stapled path* if for every edge e_i in the trail, the edge e_{i+1} is adjacent to one of the faces adjacent to e_i. The purpose of a path over the structure is to have a path for the scaffold which makes a rigid structure, to achieve this the path has to be a valid stapled path.

3 Scaffold path design

DNA origami allows us to create structures folding a single strand of DNA called scaffold and multiple small strands called staples. The design of the scaffold starts by determining the length of the strand. To get this size we have to design a path over the structure. The path has to go through the whole structure in such a way that the resulting structure is rigid enough and trying to minimise the amount of DNA used.

Once we have a valid path over the structure the next step is to design the sequence of the scaffold and the staples. To achieve this we have to create a scaffold long enough to follow the designed path. Capello and co-workers are able to create synthetic DNA [15], from small strands up to thousands of base pairs with the sequence of nucleotides that we desire. However, this process is prohibitively expensive. Thus, as a second approach biological DNA is employed in DNA origami, i.e. the viral DNA of M13mp18. The bacteria M13mp18 is the most common one used to make long scaffolds, it provides a natural sequence of 7,249 nucleotides and is the one used by Rothemund to create the shapes described in his paper. With the scaffold and the staples designed, they are ready to blend with magnesium and anneal the solution to form the structures.

Bhatia and co-workers [8] constructed an icosahedron to encapsulate gold particles. They constructed the icosahedron by designing three different components. One of the components is common for the top and the bottom of the icosahedron. The two others are the middle parts which join the top and the bottom. The result of this process is a DNA icosahedron which can hold a gold nanoparticle inside.

Ke and co-workers [16] designed a strategy to construct cages with a tetrahedral geometry using the DNA origami technology, where each edge of the tetrahedron is $58nm$ in dimension.

3.1 Scaffold path on DNA origami

A polyhedron is said to be rigid if it can not be deformed. A convex polyhedron is rigid if all its surfaces are rigid, as Alexandrov showed [17]. It is enough for a convex polyhedron to have every face as a triangle in order to be rigid [18]. We assume all our structures are convex polyhedra.

Given a graph representing a convex polyhedron, we have to find a path over it. In order to make the structure rigid the path has to go through every edge of the graph. This restriction is a lower bound for the number of edges that we have to use, that is, the path has to cover at least once every edge. But, in order to minimise the amount of DNA used we have to minimise the number of edges that we use. This restriction makes a desirable upper bound for the number of edges in the path, that is, it is more convenient to use every edge at most once. These two bounds lead us to the problem of finding an Eulerian circuit in the graph.

Since an Eulerian circuit exists if and only if the degrees of all vertices are even. Thus, if the graph is Eulerian then there is a way to place the scaffold over the structure. But, what happens when the graph is not Eulerian? As a consequence of the handshaking lemma we know that the number of odd degree vertices is even, and we can modify the graph by adding a minimum number of double edges such that the degree of all the vertices in the graph becomes even [19]. Thus, we have to ensure that every polyhedron has a path for the scaffold.

Björn Högberg and his team have developed a origami technique for an icosahedral framework [20]. They implemented a icosahedron by subdividing its faces and routing the scaffold over the edges of the icosahedron. The staples were used to fix the vertices on the designed path. Abdulmelik Mohammed at the university of Aalto in Finland has extended Björn's work to arbitrary polyhedral beam-framework [21]. Also, he has investigated the conditions where the scaffold can be routed in an arbitrary polyhedron. Moreover, he statements that the problem of find a path for the scaffold in an arbitrary polyhedron is NP-Complete and presents a backtracking algorithm which takes a polyhedron and finds a path for the scaffold in it, if such a path exists.

3.2 Reinforcement design

A large structure can be made of DNA using the DNA origami technique, but if the edges are long enough, e.g. larger than 70 base pairs, the structure could be deformed or even collapse. In order to avoid this behaviour we propose a structure to reinforce large edges in the original structure.

The *truss* is one of the most common engineering structures that we can see in our cities: bridges or electricity pylons are some examples of trusses. A truss consists of straight members connected at their extremities by joints. A three dimensional truss is called *space frame truss* [22][23]. In a truss two equal forces act along each member in opposite directions, that is, each member experiences axial compression or tension.

Since a polyhedral structure is rigid if all of its faces are triangles. We have designed a space frame truss fulfilling the properties which will replace the long edges in the original structure.

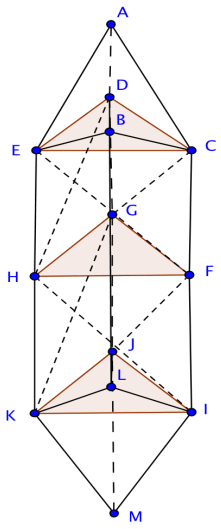

Figure 3.1: Triangulated space frame truss.

We present this space frame truss composed by two tetrahedra as *extremities* and triangular prisms as *levels*. In polyhedral terms, the structure is an elongated triangular bipyramid, and is the Johnson solid number 14 [24]. Each extremity is made of five vertices, four of which are in the base and one in the top of the tetrahedron. The base is a triangle with an internal vertex in the middle forming a complete graph. Only the vertices of the triangle are connected to the top of the tetrahedron. Each level is made of six vertices, three in the top and three in the bottom. The base and the top form a triangle. Every vertex on the base is connected to two vertices in the top. Both of the extremities are connected by the internal vertices in the base.

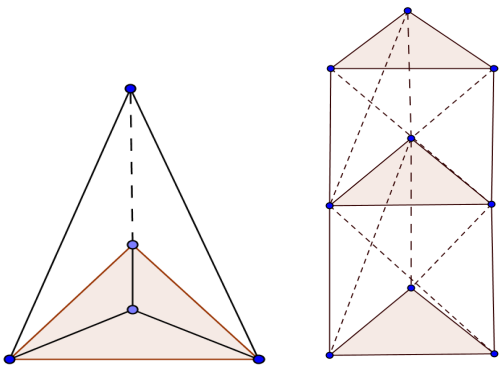

Figure 3.2: Top and single level of the reinforcement structure.

All the vertices constituting a level have degree six and the internal vertices in the extremities have degree four. As we can see, all the vertices have even degree except two that have degree three.

The elongated bipyramid is the smallest polyhedra that we can use to reinforce the original structure. The smallest triangular polyhedron that we can use as levels is a prism, because we need to extend an arbitrary number of levels, that is, the bases of the polyhedra have to be parallel to be able to add more levels and have a vertical structure. We need the extremities to have a central point in the structure to join two of them. With this we avoid to manage the rotation of the structure, as the prisms of the levels can be rotated to any degree around their central point.

3.3 Scaffold path on a reinforced structure

The above structure clearly fulfills the property of rigidity because every face in the structure is a triangle, this is enough for a structure to be rigid. The second property that this structure has is that it is a semi-Eulerian graph, i.e. it has an Eulerian trail. This is true because the structure has only two odd vertices. These vertices are extremities of the bipyramid, i.e., the two vertices from the top of the two tetrahedra.

To reinforce the original structure with the designed bipyramid, we have to go through the edges of the original structure and replace the long edges with our structure. The structure is very convenient for our propose, because we can start the global path at one extremity, go through each edge of the structure, and finish on the other extremity and this is easily archivable because the structure has an Eulerian trail.

We have designed this trail in a constructive way, that is, we give a sequence of vertices starting from an odd degree vertex and covering the extremity. After the extremity we continue covering level by level the structure until we reach and cover the other extremity. The trail can be extended automatically to an arbitrary number of levels. Thus the exact path is described in Section 4.3.

4 Reinforcement algorithm

In chapter 2 we described the basic concepts of DNA and graph theory that are needed to understand the problem and the algorithm. In chapter 3 we presented how the scaffold design works and the rigidity problems when designing three dimensional structures. In this chapter we present a constructive algorithm for the reinforcement of a DNA nanostructure. Given a graph that represents a polyhedron and a path over the graph, we construct the new reinforced graph which represents the reinforced polyhedron. The algorithm outputs a set of vertices and a trail over the new structure. It proceeds as follows:

First, we read the graph and the path which defines the polyhedron. The expected input for the definition of the graph is a file with extension ply:

```
ply          // Start header
...
end_header // End Header
x1 y1 z1     // Float coordinates for vertex 1
...          // Rest of vertices
```

and for the path is a file with extension ntrail:

```
v1 v2 ... vN // Path for the structure
```

After reading the input, we reinforce only the long edges with our reinforcing structure. This reinforcement is done in the same order that the given path. If the given path is an Eulerian circuit over the original structure then the path over the reinforced structure will also be an Eulerian circuit.

The reinforcement of an edge is done only if the length of the edge is greater than a predefined threshold. Every edge is defined by two vertices. We take these two vertices and create the new structure between them, where the two odd vertices of the new structure are the two vertices in the original structure. The next section describes how the structure is created and placed.

4.1 Structure design

We work in Euclidean spaces with points defined as triplets of real numbers. Given two Euclidean points p and q, the distance between them is given by the formula:

$$d(p,q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2 + (p_z - q_z)^2} \tag{4.1}$$

Now, let h be the height of a single level in the reinforcement structure, then the number n of levels in the edge reinforcement is:

$$n = \lfloor \frac{d(p,q)}{h} \rfloor - (2 + extra) \tag{4.2}$$

We keep two levels for the extremities and in some cases one extra level to avoid the overlapping of two structures, as discussed in section 4.2. With formulas 4.1 and 4.2 we can calculate the height of both extremities of the structure as:

$$e = \frac{d(p,q) - nh}{2} \tag{4.3}$$

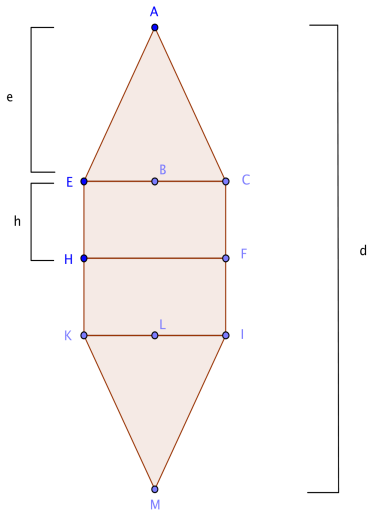

Figure 4.1: Structure with the main lengths and $n = 2$.

The algorithm places the new reinforcement structure for the points p and q in the origin $(0, 0, 0)$ and faced to the positive z-axis. The top vertex of one extremity is placed in the origin and the top vertex of the other extremity is placed in the point $(0, 0, z)$ where $z = d(p, q)$. The structure has $n+1$ equilateral triangles, one of them is shared by an extremity and a level. Each triangle t_i has its centroid in the coordinates $t_{iD} = (0, 0, e + ih)$ with $i \in \{0..n\}$.

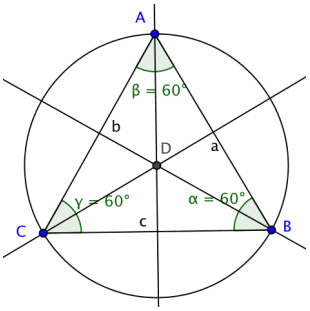

Figure 4.2: Geometry of an equilateral triangle.

In Figure 4.2 we can see the geometry of an equilateral triangle with side length $L = a = b = c$. By the law of sines we can derive the length of the segments AD, BC and distance from D to the intersection of the line D with the segment BC. Let $L' = sin(30°)\frac{L}{sin(120°)}$ be the length of the segment AD. We define the three coordinates of the vertices of each triangle t_i as follows:

$$t_{iA} = (0, L', 0) + t_{iD} \tag{4.4}$$

$$t_{iB} = (sin(60°)\frac{L'}{sin(90°)}, -sin(30°)\frac{L'}{sin(90°)}, 0) + t_{iD} \tag{4.5}$$

$$t_{iC} = (-sin(60°)\frac{L'}{sin(90°)}, -sin(30°)\frac{L'}{sin(90°)}, 0) + t_{iD} \tag{4.6}$$

The algorithm generates the vertices $p, q, t_{iA}, t_{iB}, t_{iC}$ for all $i \in \{0, \ldots, n\}$ and additionally t_{0D} and t_{nD} for the reinforcement structure. We also output the edges of each structure in the following order. First, we generate the edges of the top extremity, then the edges of each level and finally the edges of the bottom extremity.

Once we have the structure for the points p and q, we have to place it in the correct position. To achieve this we need to translate the vertices of the structure and rotate the whole structure. As the points p and q define a line in space and the rotation of the structure over the z-axis does not affect it, we do not need to rotate the structure by the z-axis. But to have the structure in the same position as the line we have to rotate over the x-axis and y-axis. Also to have the structure in the same place as the segment pq, we have to translate it to the point p.

The angle between two vectors is the smallest angle that a vector can be rotated to be aligned with the other. The geometric *dot product* of two vectors is defined as $A \cdot B = ||A||||B|| \cos(\theta) = A_x B_x + A_y B_y + A_z B_z$. This function gives us a way to calculate the angle between two vectors:

$$\theta = \arccos(\frac{A_x B_x + A_y B_y + A_z B_z}{||A||||B||}) \tag{4.7}$$

where $||A||$ is the magnitude or length of vector A. The *cross product* function takes two vectors and gives us a new vector which is perpendicular to both vectors. Cross product is defined as follows:

$$A \times B = (A_y B_z - A_z B_y, A_z B_x - A_x B_z, A_x B_y - A_y B_x) \tag{4.8}$$

With the dot product we can calculate the smallest angle between two vectors which goes from $0°$ to $180°$. With the cross product we can find out if the angle is positive or negative. We can use this to find the rotation angles over the x-axis and y-axis.

Let us define the vector v as $(q_x - p_x, q_y - p_y, q_z - p_z)$. We calculate the rotation angle α over the y-axis with the vectors $z = (0, 0, 1)$ and the projection of the vector v in the plane XZ. The angle α can be found using formula (4.7) with $A = z$ and $B = (v_x, 0, v_z)$. If the y component in the cross product of z and $(v_x, 0, v_z)$ is positive then the turn is positive, otherwise it is negative. To calculate the rotation angle β over the x-axis we can use the same method but this time with the vectors z and $(0, v_y, \sqrt{v_x^2 + v_z^2})$ where the component z of the vector is the distance between the components x and z of v.

4.2 Avoiding overlapping

If in the original structure we have two edges forming a small angle, this might generate an overlap in between the reinforcement structures replacing these edges. To avoid this undesired situation we compute the amount of levels we have to remove from the reinforcement structure in order to avoid this overlapping. We are going to assume that the minimum angle in between two edges is 45° which is a reasonable constraint for DNA nanostructures.

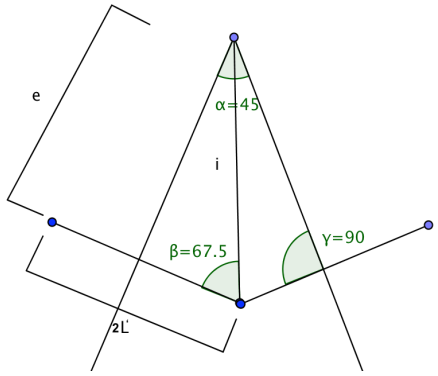

Figure 4.3: Overlapping of two edges.

As in the previous section, L' is the radius of the circumcircle of the triangle. As the first triangle comes after an extremity, we have to find how long e has to be to avoid the overlapping. We do it for the worst case scenario, which is case when the two edges form a 45° angle. The minimum length e to avoid overlapping is given by the formula:

$$e > \sin(\beta)\frac{i}{\sin(90°)} \tag{4.9}$$

where i is derived by the law of sines:

$$i = \frac{L'}{\sin(\frac{\alpha}{2})}\sin(90°) \tag{4.10}$$

If we take, for example, 3 as the length L then replacing α by 45° and $\beta = 180° - 90° - \frac{\alpha}{2} = 67.5°$, e has to be greater than 4.19. In the implementation of our algorithm we have set the height of each level h to 3. If e, calculated with formula (4.3), is less than 4.19 then we have to remove one extra level. Which means that the *extra* variable in formula (4.2) is equal to 1.

4.3 Path over the structure

The designed path has to fulfill two important properties. It has to be an Eulerian trail going through all the edges in the reinforcing structure and the path has to be a valid stapled path. We propose a path which fulfils both properties.

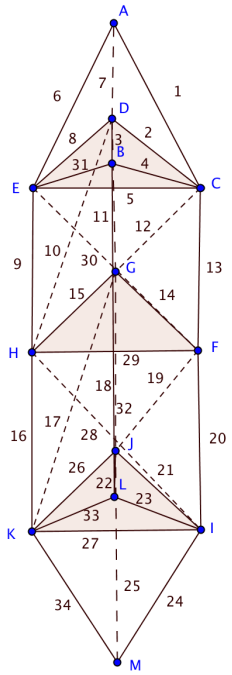

Figure 4.4: Path over the reinforcement structure.

Vertex B is adjacent to C, D, E and L, and similarly vertex L is adjacent to I, J, K and B. The path in Figure 4.4 is:

Top extremity: $A - C - D - B - C - E - A - D - E-$
Levels: $-H - D - G - C - F - G - H - K - G - J - F-$
Bottom extremity: $-I - J - L - I - M - J - K-$
From top to bottom: $-I - H - F - E - B - L - K - M$

The path starts at vertex A and ends at vertex M forming an Eulerian trail. It is easy to see that we can extend the path to an arbitrary number of levels following the same strategy.

5 Visualisation

In addition to the algorithm, we have developed a *visualisation tool* which takes a polygon definition as input and shows its three-dimensional representation. It allows us to navigate over the polygon in the three axes and change the angle of view through the scene. The tool is implemented using the OpenGL library for rendering three dimensional graphics and the *GLUT* library which helps us to draw geometric primitives.

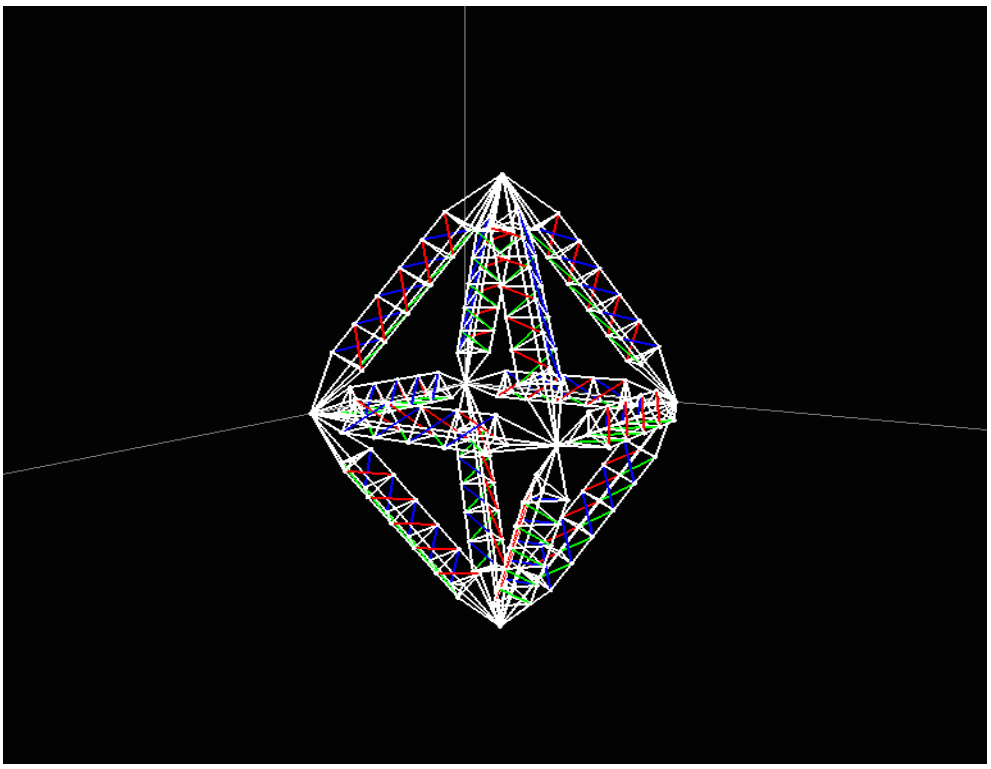

Figure 5.1: Visualisation tool.

The visualiser accepts, besides the polygon, the definition of the path for the scaffold. The tool shows an animation of the path over the polygon. The input polygon is defined using the *Polygon File Format* (PLY) where we can specify the position of the vertices in the space and the edges of the polygon. Additionally, we have added the path definition for the polygon.

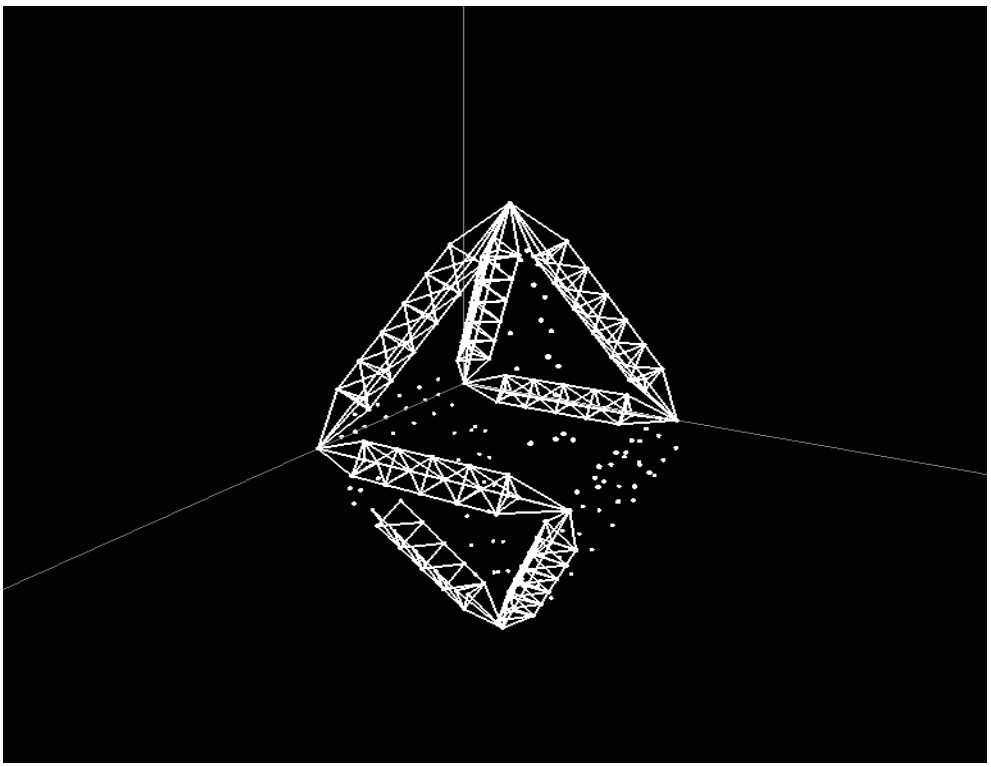

Figure 5.2: Path animation.

6 Conclusions

In this document, we discuss the DNA origami technique and how it works with two-dimensional shapes and three dimensional structures. We have exposed the problematic of building large structures with DNA. We propose a method to avoid the collapse reinforcing those large structures.

We presented a bipyramid-like structure which the vertices on the top of each pyramid are the only ones with degree odd. This very convenient for our purposes because the rotation over its vertical axis does not affect the resulting global structure. Additionally, the structure fulfils the rigidity requirements, since it is triangulated. Moreover, we have defined a valid stapled path in section 3.3 and we showed in section 4.3 the constructive design of a valid stapled path over the structure.

We propose a constructive algorithm which takes a polyhedron definition with a path, then it reinforces the original structure with our bipyramid design. The algorithm works following the given path for the original structure and for every edge, it creates a new reinforcement structure. Then translates and rotates the new structure to be aligned with the vertices of the edge, i.e. the vertices of the edge are the top vertices of the two pyramids. Once the structure is in place, the algorithm modifies the original path to go through the new structure.

Also, we have developed a visualisation tool which takes a polyhedron and a path definition. It shows the polyhedron representation and an animation where we can see the designed path for the polyhedron. We can navigate over the polyhedron and look at it in different angles.

Bibliography

[1] Watson J.D. and Crick F.H.C. A structure for deoxyribose nucleic acid. Nature 171, (1953), 737-738.

[2] Seeman, N. Nucleic acid junctions and lattices. Journal of Theoretical Biology 99, (1982), 237-47.

[3] Chen, J. and Seeman, N. Synthesis from DNA of a molecule with the connectivity of a cube. Nature 350, (1991), 631-3.

[4] Winfree, E., Liu, F., Wenzler, L. and Seeman, N. Design and self-assembly of two-dimensional DNA crystals. Nature 394, (1998), 529–544.

[5] Rothemund, P. W. Folding DNA to Create Nanoscale Shapes and Patterns. Nature 440, 7082 (2006), 297–302.

[6] Douglas, S., Dietz, H., Liedl, T., Högberg, B., Graf, F. and Shih, W. Self-assembly of DNA into nanoscale three-dimensional shapes. Nature 459, (2009), 414-418.

[7] Andersen, E., Dong, E., Nielsen, M., Jahn, K., Subramani, R., Mamdouh, W., Golas, M., Sander, B., Stark, H., Oliveira, C., Pedersen, J., Birkedal, V., Besenbacher, F., Gothelf, K. and Kjems, J. Self-assembly of a nanoscale DNA box with a controllable lid. Nature 459, (2009), 73-76.

[8] Bhatia, D., Mehtab, S., Krishnan, R., Indi, S. S., Basu, A., and Krishnan, Y. Icosahedral DNA Nanocapsules by Modular Assembly. Angewandte Chemie International Edition 48, 23 (2009), 4134-4137.

[9] He, Y., Ye, T., Su, M., Zhang, C., Ribbe, A. E., Jiang, W., and Mao, C. Hierarchical Self-assembly of DNA into Symmetric Supramolecular Polyhedra. Nature 452, 7184 (2008), 198-201.

[10] Douglas, S. M., Marblestone, A. H., Teerapittayanon, S., Vazquez, A., Church, G. M., and Shih, W. M. Rapid Prototyping of 3D DNA-origami Shapes with caDNAno. Nucleic Acids Research 37, 15 (2009), 5001-5006.

[11] Euler, L. Solutio problematis ad geometriam situs pertinentis. Commentarii academiae scientiarum Petropolitanae, Vol. 8 (1741), 128-140. Reprinted in Opera Omnia Series Prima, Vol. 7. pp. 1-10, (1766).

[12] Hierholzer, C., Chr, W. Ueber die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Mathematische Annalen (in German), (1873), 30–32.

[13] Pevzner, P., Tang, H. and Waterman, M. An Eulerian path approach to DNA fragment assembly. PNAS 98, 17 (2001), 9748–9753.

[14] Roy, K. Optimum Gate Ordering of CMOS Logic Gates Using Euler Path Approach: Some Insights and Explanations. Journal of Computing and Information Technology, 1 (2007), 85-92.

[15] Capello, J. and Causey, S.C. and Chambers, J. and Crissman, J.W. and Ferrari, F.A. and Pollock, T.J. and Richardson, C. Construction of synthetic DNA and its use in large polypeptide synthesis. Google Patents, (1993).

[16] Ke, Y. Sharma, J. Liu, M. Jahn, K. Liu, Y. and Yan, H. Scaffolded DNA Origami of a DNA Tetrahedron Molecular Container. Nano Letters 6 (2009), 2445-2447.

[17] Dairbekov, N., Alexandrov, A., Kutateladze, S., and Sossinsky, A. Convex Polyhedra. Springer, Berlin, Germany, (2005).

[18] Connelly, R. Rigidity of Certain Cabled Frameworks and the Second-Order Rigidity of Arbitrarily Triangulated Convex Surfaces. Adv. Math. 37 (1980).

[19] Cromwell, P. R. Polyhedra. Cambridge University Press, Shaftesbury Rd, Cambridge, United Kingdom, (1999).

[20] Areddy, P. K. Computer-Aided Design of Polyhedral DNA Nanostructures. Master?s thesis, KTH, 2012.

[21] Mohammed, A. Combinatorial Algorithms for the Design of Nanoscale Systems. Master?s thesis, Aalto University, 2014.

[22] Hibbeler, R. Engineering Mechanics-Statics. New York: Macmillan Publishing Co., Inc.

[23] Greene, C. Graphical analysis of roof trusses; for the use of engineers, architects and builders. New York, J. Wiley & sons (1885).

[24] Johnson, N. W. Convex Solids with Regular Faces. Canadian Journal of Mathematics, 18 (1966), 169-200.

[25] Joyner, D. Van Nguyen, M. and Philips, D. Algorithmic Graph Theory and Sage. 2013.