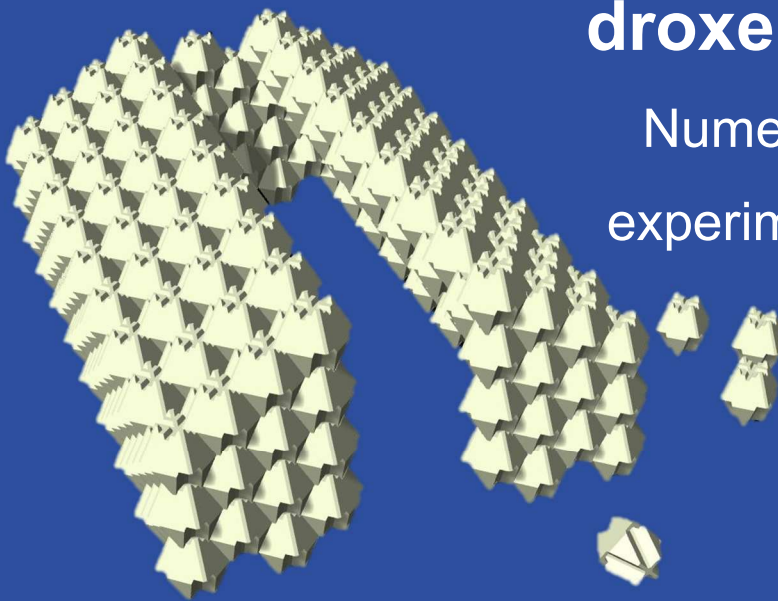


École polytechnique de Louvain

Stability of parametric designed droxel structures

Numerical model with
experimental validation



Author : **Chloé DE KETELE**

Supervisor : **Pierre LATTEUR**

Readers : **Sébastien GOESSENS, David LECOMPTE, Hadrien RATTEZ**

Academic year 2020–2021

Master [120] in Civil Engineering

STABILITY OF PARAMETRIC DESIGNED DROXEL
STRUCTURES:
NUMERICAL MODEL WITH EXPERIMENTAL
VALIDATION

CHLOÉ DE KETELE

*“When you’re curious,
you find lots of interesting things to do”*

— WALT DISNEY

ACKNOWLEDGEMENTS

The first time I was confronted with droxels was in March 2020, the day SÉBASTIEN GOESSENS defended his thesis on drone-compatible stackable elements. I got intrigued by the concept of droxels and I went to talk to him after his defence. After a few emails with PIERRE LATTEUR, I am convinced: my thesis will be on droxels.

These are the first two people I would like to thank. SÉBASTIEN GOESSENS was able to transfer to me his passion for these small blocks, often confused (wrongly) with LEGO®. He has been available several times during this year and has accepted to be part of my jury, for which I also thank him.

Secondly, I would like to thank PIERRE LATTEUR who trusted me to make the right choices and who let me work autonomously. He was always quickly available for a discussion when I had some questions and managed to keep me motivated throughout the year.

I discovered several new programs during this thesis and would not have been able to do so without the supervision of FRÉDÉRIC DUBOIS (*UMontpellier*), JULIEN GENO and JONAS FERON. FRÉDÉRIC DUBOIS freed up a lot of time to remotely train me on the *LMGC90* software. I also worked with *C#* codes, implemented in *Grasshopper*. These two environments were unknown to me at the beginning and I would not have been able to complete this work without the help of JULIEN and JONAS. They helped me a lot for the introduction to the languages and for some specific questions. They were always available despite their busy schedules, I thank them for that.

In addition to mastering these programming languages, I also developed two new algorithms during this thesis. I would like to thank MARTIN STEINMETZ whose insight quickly helped me improve the second algorithm considerably.

In order to achieve all the objectives of the thesis, there was also an experimental aspect: the 3D printing of droxels for which I would like to thank BENOÎT HERMAN. Thanks to his time and experience, we were able to print 250 droxels in a short time lapse. For the experimental aspect, I would also like to thank ALEX BERTHOLET who lent me material to realise the plate in which the droxels are embedded.

I would like to extend my thanks to IGOR BOECKAERT and JONAS FERON for their time during the proofreading of this thesis.

I would already like to thank the members of my jury: SÉBASTIEN GOESSENS, PIERRE LATTEUR, Major DAVID LECOMPTE and HADRIEN RATTEZ for the reading of this thesis and for their presence during its defence.

On a lighter note, I would like to thank all my classmates, especially the Harem, for all the great years spent together. I would also like to thank Alexis for the hours spent at the Vinci working on our respective thesis and of course the kicker team.

Lastly, I would like to thank my family and friends, especially my parents BERNADETTE MIGISHA and GRÉGORY DE KETELE, for their continuous support.

TABLE OF CONTENTS

Introduction.....	1
CHAPTER 1 Stackable elements: state of the art	5
1.1 What is a stackable element?.....	5
1.2 An overview of some existing modules.....	6
1.2.1 Formwork-type blocks.....	6
1.2.2 Blocks developed for developing countries	8
1.2.3 Earthquake resisting blocks.....	9
1.2.4 Blocks used for retaining walls	10
1.2.5 Other blocks	10
1.2.6 Blocks allowing for cantilever structures	11
1.3 Drone development: introduction of dricks.....	11
1.3.1 Conical dricks.....	12
1.3.2 Rectangular dricks	12
1.3.3 Extrudable dricks	12
CHAPTER 2 The droxel: state of the art.....	15
2.1 Origin and development of the droxel	15
2.2 Droxel geometry.....	16
2.2.1 Droxel volume	17
2.3 Droxel parameters and their influence on the final structure	17
2.3.1 Height: H.....	17
2.3.2 Width: L	18
2.3.3 Angle: A.....	19
2.3.4 Cross width: E	21
2.3.5 Big upper hook's height: R.....	22
2.3.6 Height of the small upper hooks: B	22
2.3.7 Chamfer: C_r	23
2.3.8 Upper hook's ratio: F	23
2.3.9 Laying tolerance: L, R, A, H, E, C_r	23
2.3.10 Choice of the parameters: a calculated compromise	24
2.4 Droxel constructions: advantages and disadvantages	24
2.4.1 Advantages	24
2.4.2 Disadvantages.....	27

CHAPTER 3	Droxelised structures: realisation and implementation.....	29
3.1	Realisation	29
3.2	Implementation: Grasshopper, Rhino and C#.....	30
CHAPTER 4	Verification methods of the algorithm.....	33
4.1	Numerical simulation: Discrete Elements Method (DEM)	33
4.1.1	Discrete Elements Method (DEM).....	33
4.1.2	LMGC90	36
4.1.3	Discrete Elements Method applied to Droxels.....	36
4.2	Experimental tests: 3D printed droxels.....	39
4.2.1	3D printing of droxels	39
4.2.2	Realisation of the base	41
CHAPTER 5	Droxelisation of a surface.....	43
5.1	Introduction.....	43
5.2	What is considered a droxelised structure ?.....	44
5.2.1	Droxelised structures.....	44
5.2.2	Notion of stability.....	45
5.3	Influence of several factors on the droxelisation.....	46
5.3.1	Influence of the droxel scale	46
5.3.2	Influence of the wall depth of the structure	47
5.3.3	Influence of the grid orientation	47
5.4	Two droxelisation methods.....	48
5.5	Implementation of the two methods.....	49
5.5.1	Case 1: input of a 3D surface.....	49
5.5.2	Case 2: input of normalised coordinates.....	52
5.5.3	Flowchart representation of both methods.....	53
5.6	Grasshopper program	53
CHAPTER 6	Global stability of a droxelised structure	55
6.1	Introduction.....	55
6.2	Global instabilities due to tilting	56
6.2.1	Stacking modes of droxels.....	57
6.2.2	(Un)hooked droxels	57
6.2.3	Identification of an unstable structure.....	58
6.3	Stability check algorithm	61
6.3.1	For 2D structures	61

6.3.2	Generalisation to 3D cases	63
6.4	Coding choices.....	64
6.5	Limits of the algorithm	66
6.5.1	Friction between the blocks neglected	67
6.5.2	Hooked is binary.....	69
6.6	Examples.....	70
6.6.1	Example 1: stable 2D example	70
6.6.2	Example 2: unstable 2D example	72
6.6.3	Example 3: Limitation of the algorithm : friction	73
6.6.4	Example 4: Limitation of the algorithm: Arch-like structures	75
6.6.5	Example 5: Limitation of the algorithm: hooked is binary	76
CHAPTER 7	Laying sequence of a droxelised structure	79
7.1	Some notions.....	81
7.2	Principles	83
7.3	Laying sequence algorithm : flowchart	85
7.4	Advantages, limits and improvement suggestions of the algorithm	86
7.4.1	Advantages	86
7.4.2	Limits	86
7.4.3	Improvement suggestions.....	87
7.5	Examples.....	89
7.5.1	Example 1: trivial 2D structure – straight wall	89
7.5.2	Example 2: non-trivial 2D structure – arch.....	90
7.5.3	Example 3: non-trivial 3D structure – small shelter	94
7.5.4	Example 4: stable final structure that cannot be build	95
7.5.5	Example 5: showing a limit of the algorithm.....	96
7.5.6	Example 6: Limitation of the algorithm: bridge-like structure	98
CHAPTER 8	Comparison between the 3 methods : examples.....	101
8.1	Example 1: Simple wall.....	101
8.2	Example 2: Arch.....	103
8.3	Example 3: Small shelter	105
8.4	Example 4: Gate (cannot be built).....	107
8.5	Example 5: Limits of the algorithm - example.....	109
Conclusion.....		113
Table of Appendices		161

Table of figures.....	163
Bibliography.....	171

INTRODUCTION

Constructing the buildings of tomorrow with drones ? A few decades ago, this would have been the subject of the latest science fiction movie. However, the droxel was designed with this objective in mind. The droxel, represented below, is a stackable element allowing for the construction of structures without any binder between the elements.



Droixel : perspective view Droixel : bottom view
Figure 1 - Droixel geometry : perspective and bottom view

Over the last decades, not a lot has changes in the construction field. Lately, more and more innovations have however occurred. The automation of construction is an inevitable process which has already started. The droxels, allowing for construction by robots, could be used for this. If their weight is not too important, one could even build a self-made shed out of droxels. In addition to being relevant for the automation of construction, they can also be used for home-made structures.



Figure 2 - Example of possible self-made agricultural shelters out of droxels [15]

The research on the topic of droxels started in 2015 at UCLouvain. Up until now several works have brought us closer to the realisation of structures with droxels.

First of all, the droxel geometry was created and a stability algorithm to determine whether a 2D droxel structure is stable was developed by S. GOESSENS [24]. This algorithm has already been generalised to 3D structures by G. GUILMIN [26] but many limitations, including the important computation time, were identified.

In addition to the development of an a stability algorithm, numerical simulations of structures with droxels using Discrete Element Methods (DEM) also started during the master thesis of M. MULLIER [40]. DEM are methods allowing for the simulation of multibody structures (such as droxels) using certain behavioural laws [16].

This document has three goals :

- 1) Implementing the *stability algorithm* for 3D structures analytically (instead of geometrically as done previously by G. GUILMIN [26]);

- 2) Implementing a *laying sequence algorithm* to find the order in which to build the structure in order to guarantee a stable situation at each step;
- 3) *Comparing* the implemented algorithms with numerical simulations and small-scale experimental tests.

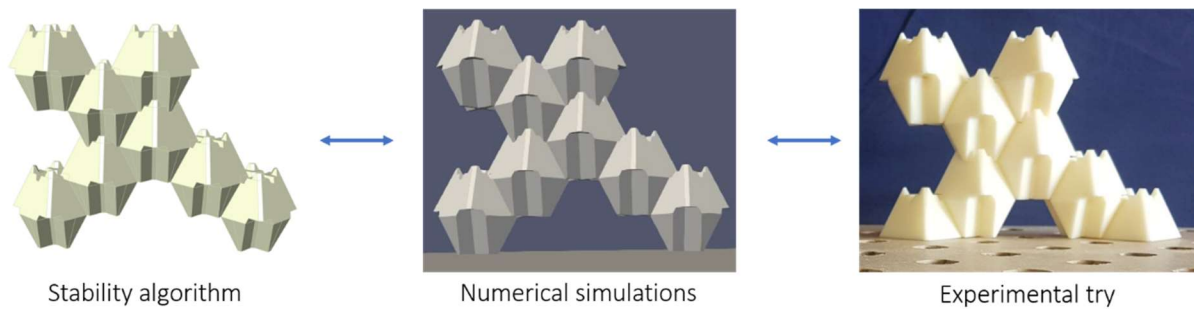


Figure 3 - Third goal : comparing the stability algorithm with the numerical simulations and experimental tests

Before diving into the explanation and implementation of the algorithms, the general context of the development of droxels is explained. CHAPTER 1 is a state of the art on stackable elements in general while CHAPTER 2 goes more into detail on the droxel itself. The development of its geometry, the influence of its parameters and its advantages and disadvantages are discussed.

Then, CHAPTER 3 answers the question : “How can a structure be realised using droxels and what are the design steps ?”. So far, 5 main steps are identified. In order to validate those steps, several verifications are necessary. CHAPTER 3 gives an overview of the steps and introduces how the implemented algorithm can be verified: with numerical simulations and with small-scale tests using 3D-printed droxels.

In order to be able to realise the experimental tests, small 3D droxels were designed and printed. A wooden base in which to fix the droxels was also built. Some details on the experiments are given in CHAPTER 4. The numerical simulations are also detailed.

CHAPTER 5 tackles the issue of how to determine where to place the droxels in order to have a certain aesthetic aspect. This is also called the *droxelisation* of a surface. Two possible methods, both implemented in this thesis, are explained. The first consists of inputting a 3D surface in the program (Figure 4), the second one of inputting the coordinates of the droxels.

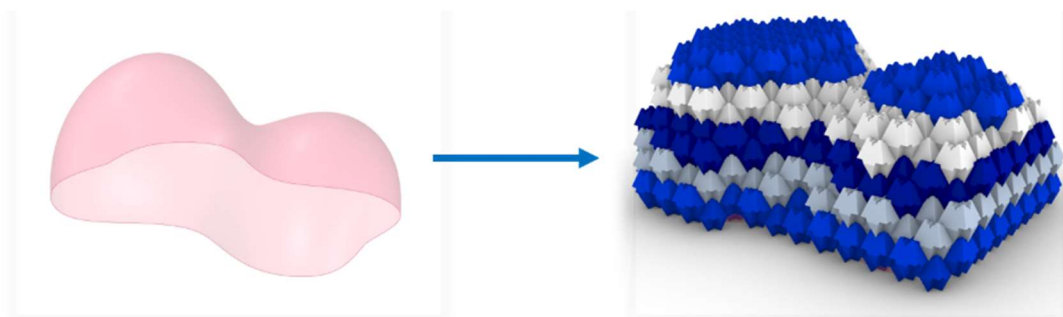


Figure 4 - Example of the droxelisation of a surface

After having found where to put the droxels in order to have a satisfying structure, CHAPTER 6 addresses the stability of a structure. The limitations, mainly due to the neglecting of the sliding between the droxels, are identified and suggestions to tackle these limitations are given.

CHAPTER 7 goes one step further and aims to answer the question of how to determine the order in which the different droxels must be placed to guarantee a stable structure at every step. This laying sequence is not trivial: an algorithm is proposed and proves to be working within the given limits which are also discussed.

The last chapter, CHAPTER 8, tackles the third objective : comparing both implemented algorithms (stability and laying sequence determination) with the numerical simulations and experimental tests. This is done to verify whether the algorithm works and what its limits are. The accuracy and pertinence of the simulations are also checked.



Figure 5 - Concrete or recycled plastic droxels for housing [15]

CHAPTER 1 STACKABLE ELEMENTS: STATE OF THE ART

A droxel can shortly be described as a stackable element that enables to shape an endless number of structural designs sometimes without even using any scaffolding [24]. Before diving into the droxel and its particularities, let us look at the motivations for the developments of stackable elements in general. An overview of already existing stackable elements in construction is also given.

1.1 WHAT IS A STACKABLE ELEMENT?

A stackable element can be described as an element which, when stacked on top of other elements and eventually combined with a binder (e.g. cement), allows for the construction of structures.

The first thing that comes to mind when describing stackable elements is the traditional brick wall: a wall composed of bricks bound together by a cementitious or lime mortar [10].



Figure 1.1 - Traditional wall: bricks and mortar [10]

Even though widely used, this method has some limits and disadvantages which led to development of other stackable elements. First of all, the placement of mortar on the blocks is both time- and material-consuming. Furthermore, mortar needs time to cure before having a certain resistance. This limits the wall height that can be constructed in a day due to the self-weight of the wall.

Development of stackable elements: motivations

The first motivation for the development of stackable elements was to develop systems that could be built quickly without needing very skilled workforce. As explained by S. KINTINGU [34], it was a way to tackle the housing shortage in developing countries: the constructions of houses using stackable elements is easy, fast and cost-effective.

Another aspect of traditional masonry is the necessity to have a correct alignment of the bricks. While this has to be done manually for traditional masonry, interlocking mechanisms can time effectively assure a correct alignment [33].

Next the possibility to create a circular economy or recycle the elements can also be mentioned. This cannot be neglected: the construction industry is one of the industries with the largest carbon-footprint [31]. If it is possible to create structures without the need of any mortar-like link between the blocks, structures could easily be deconstructed, locally repaired in case of a default, or modified.

Lastly, the huge progress made in robotisation over the last decades makes it possible to use robots for the constructions of structures. An example of this is the *Hadrian X*, a brick-laying robot which can place a brick every 20-30 seconds [27].

However, several disadvantages should also be mentioned: the interlocking requirements lead to small production tolerances which increases the production price. The interlocking geometries are often complicated and may need to be reinforced because their resistance to horizontal solicitations is not sufficient. Lastly, the blocks also require complex (and expensive) mouldings.

1.2 AN OVERVIEW OF SOME EXISTING MODULES

This section covers some existing modules developed over the years. This part is meant to be as complete as possible but is certainly not exhaustive as some modules may have been overlooked.

As quick reminder, the first stackable blocks that have been developed had as goal to ensure easy and cost-effective construction methods. The sought-after features were: reasonable weight, easy interlocking mechanism, self-alignment, modular blocks, easy manufacturing of the blocks and the possibility to construct bearing and non-bearing walls.

1.2.1 Formwork-type blocks

To ensure a correct alignment, R. THALLON [45] developed interlocking blocks with a male and female locking mechanism. All the holes are then filled with grouting: this leads to a clear overconsumption of material since the non-bearing walls are also reinforced even though they should not be.

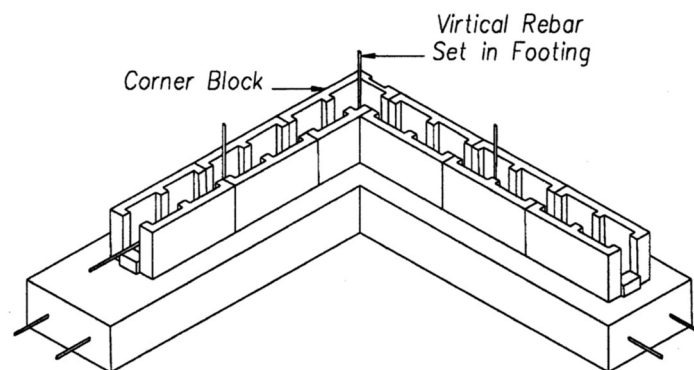


Figure 1.2 - Interlocking blocks developed by R. Thallon [45]

Another system is the *Mecano* block system developed by *Cetholic* [11]. In that system, the blocks do not have an interlocking system but are placed on top of each other. At the end, the entire wall is grouted. The same comment as previously can be made about the overconsumption of grout.

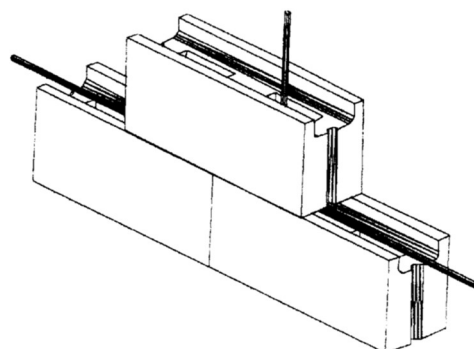


Figure 1.3 - Mecano block system, Cetholic [11]

A last system where all the walls are grouted is the *Azar* block (Figure 1.4) [21].



Figure 1.4 - Azar block

The *Haener* [28] hollow blocks are more or less based on the same principle but in this system only the load bearing walls are reinforced with rebars and grouted at suitable intervals. This is a gain in terms of used quantity of grout but the complexity of the block cannot be overlooked.

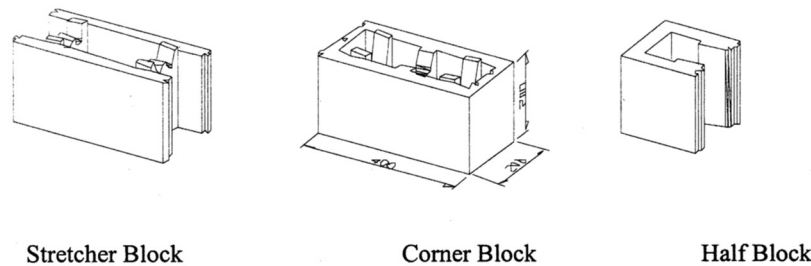


Figure 1.5 - Haener hollow blocks [28]

Researchers of University Putra Malaysia also developed *PUTRA* blocks. For this project, 21 different block models allowing for a fast, easy and accurate building systems were developed. Figure 1.6 shows three of these blocks. Using *PUTRA* blocks, a single-storey house has been built and tested [46].

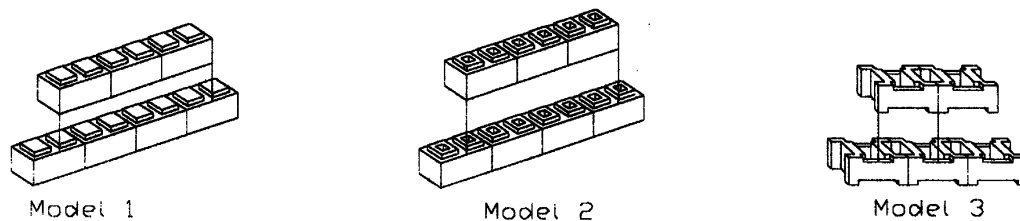


Figure 1.6 - Example of 3 types of PUTRA BLOCKS [46]

Other formwork blocks that can be mentioned are: *IITM SILBLOCK* (solid blocks), *IITM HILBLOCK* (hollow blocks), *FCS* blocks and the *Stepoc* blocks [6][22][44]. They are all filled with grout and reinforcements. The *Stepoc* blocks interlock but the interlockings have no structural function and are just present to assure the correct alignments of the blocks.

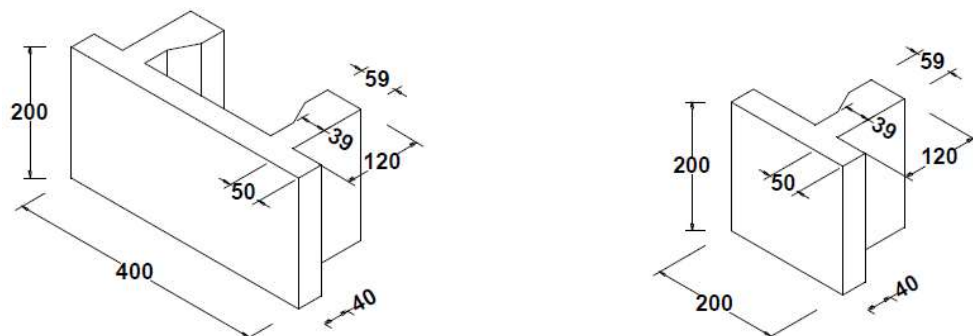


Figure 1.7 - IITM HILBLOCK (dimensions in mm) [6]

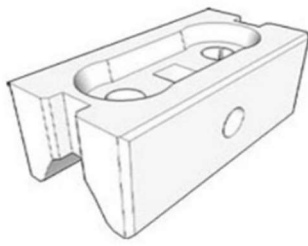


Figure 1.8 - FCS Block, example of a house built with FCS blocks [20][21]



Figure 1.9 - Stepoc blocks [21]

The last examples of formwork blocks are the *Spar-lock* [21] blocks and the *Armo System* [48] which have many similarities.

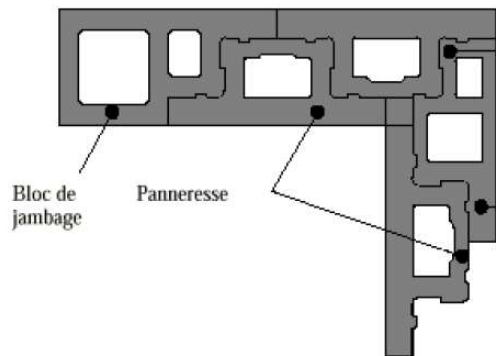


Figure 1.10 - Spar-lock blocks [21]

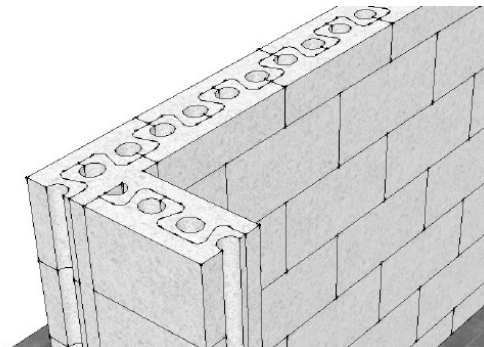


Figure 1.11 - Example of a corner wall with Armo system [48]

1.2.2 Blocks developed for developing countries

As mentioned, the first motivation for the development stackable systems was designing structures that could be built quickly and easily. As a response to the rising housing demand, *Hydraform* blocks were introduced in several African countries such as South Africa, Mozambique and Ethiopia [21]. The use of local materials allowed to decrease the price even more.



Figure 1.12 - Example of a wall built with hydraform blocks [21]

In Tanzania, the *Tanzania Interlocking Brick* is used. The holes allow the addition of vertical reinforcements to improve the resistance of the buildings to earthquakes and vertical forces [28].



Figure 1.13 - Tanzania Interlocking Brick [28]

1.2.3 Earthquake resisting blocks

Quite naturally, the same housing challenges arose in seismic regions. As a response to that, the *CFRC* (*Coconut Fibre Reinforced Concrete*) interlocking blocks were introduced. The inclined faces on the blocks allow them to slide back in the desired position after being submitted to dynamic solicitations [5].

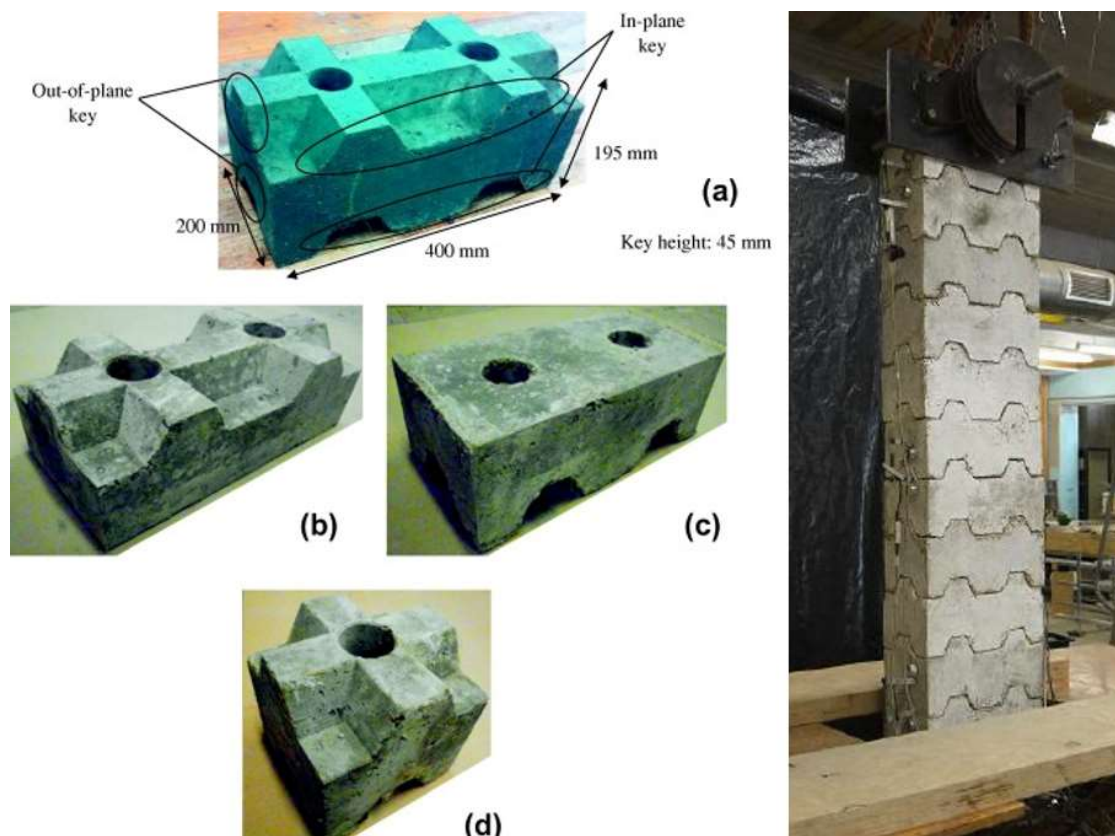


Figure 1.14 - CFRC Interlocking Blocks. On the right the dynamical test of a CFRC column [5]

1.2.4 Blocks used for retaining walls

Plenty other systems of stackable blocks have been developed and are used for retaining walls [41][8][9]. Those blocks are often very heavy: they have to be placed with a lifting equipment and need a laying tolerance to cover the fact that the workers cannot guide the blocks due to their weight.



Figure 1.15 - Muraces: an example of blocks developed for retaining walls

1.2.5 Other blocks

A system developed by ABANG A. ET AL. [2] in Malaysia is the hollow interlocking block system based on the LEGO® system. These blocks can be used for two storey buildings in Malaysia.

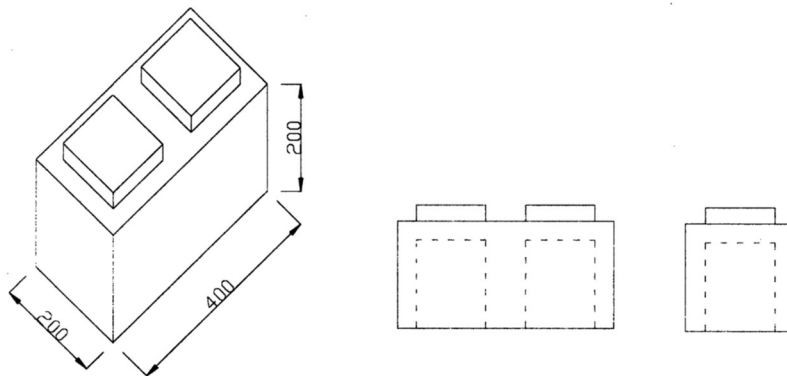


Figure 1.16 - LEGO inspired blocks by Abang A. et al [2]

Similar blocks are the *Interlocking Hollow Block System* of the UNIVERSITI TEKNOLOGI MALAYSIA but these blocks do not allow modular construction and are thus less interesting for real-life applications [41].

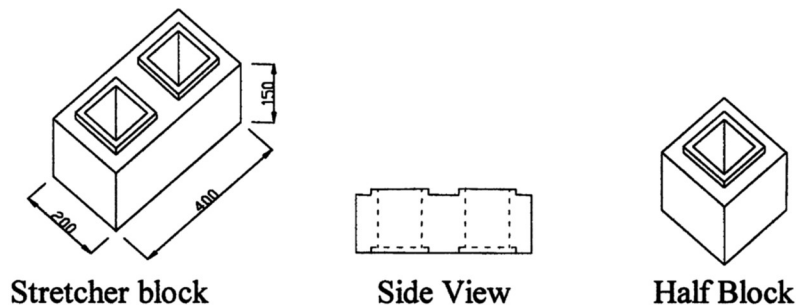


Figure 1.17 - Universiti Teknologi Malaysia [40]

Based on the concept of osteomorphic blocks (Figure 1.18), the *Tetraloc* system was developed [42][47].

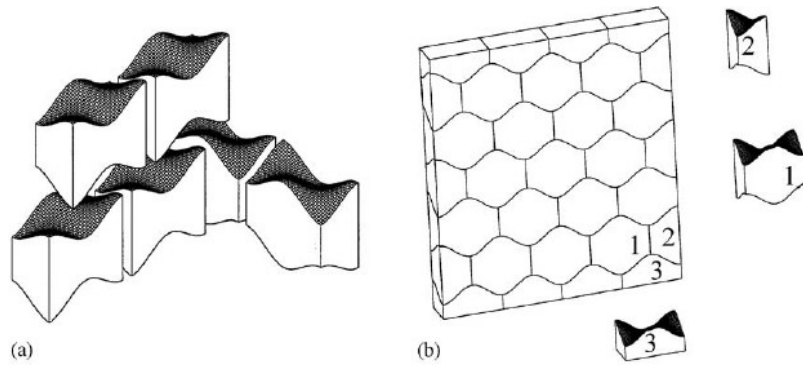


Figure 1.18 - Osteomorphic blocks in different configurations [42]

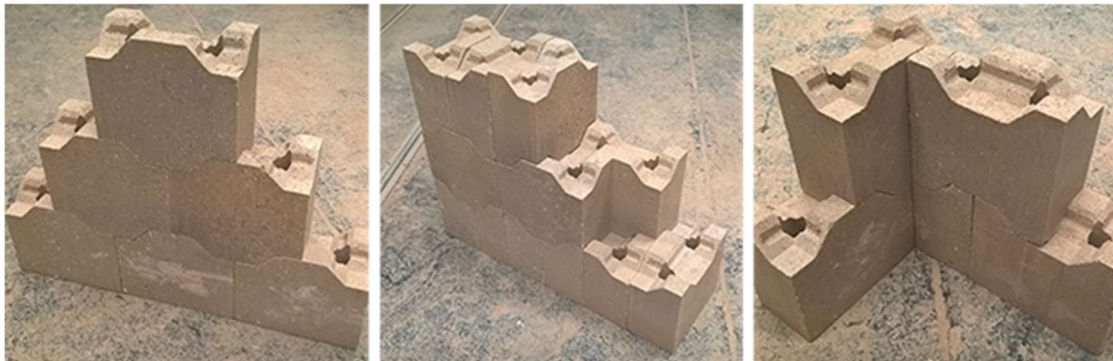


Figure 1.19 - Tetraloc blocks in different configurations [47]

1.2.6 Blocks allowing for cantilever structures

A last type of stackable element that can be mentioned are the interlocking plastic blocks developed by the architect KENGO KUMA [37]. Hollow blocks stacked on each other are used, some of them are filled with water allowing for the creation of cantilever structures. Even though not used for structural reasons, these blocks were an inspiration for the development of droxels.

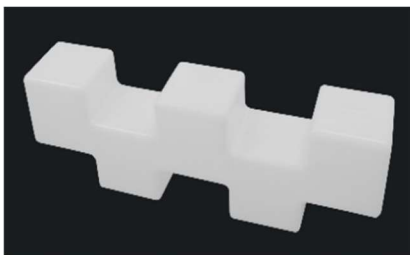


Figure 1.20 - Blocks used by the architect Kengo Kuma [24][37]

1.3 DRONE DEVELOPMENT: INTRODUCTION OF DRICKS

Within his thesis on *drone-compatible additive manufacturing of building-scale structures*, S. GOESSENS introduces several *dricks* (“[...] portmanteau of drone and brick, is used to designate those new drone-compatible (DC) masonry blocks [...]” [24]). The development of those blocks allows for the construction of small structures with piloted drones : the different dricks do not need any mortar between them.

1.3.1 Conical dricks

The first dricks that can be mentioned are the conical ones: their geometry have the advantage of having a large laying tolerance. This is important to make up for the inaccuracy on the positioning of the drone. Figure 1.21 shows a real-life test conducted on the conical dricks.

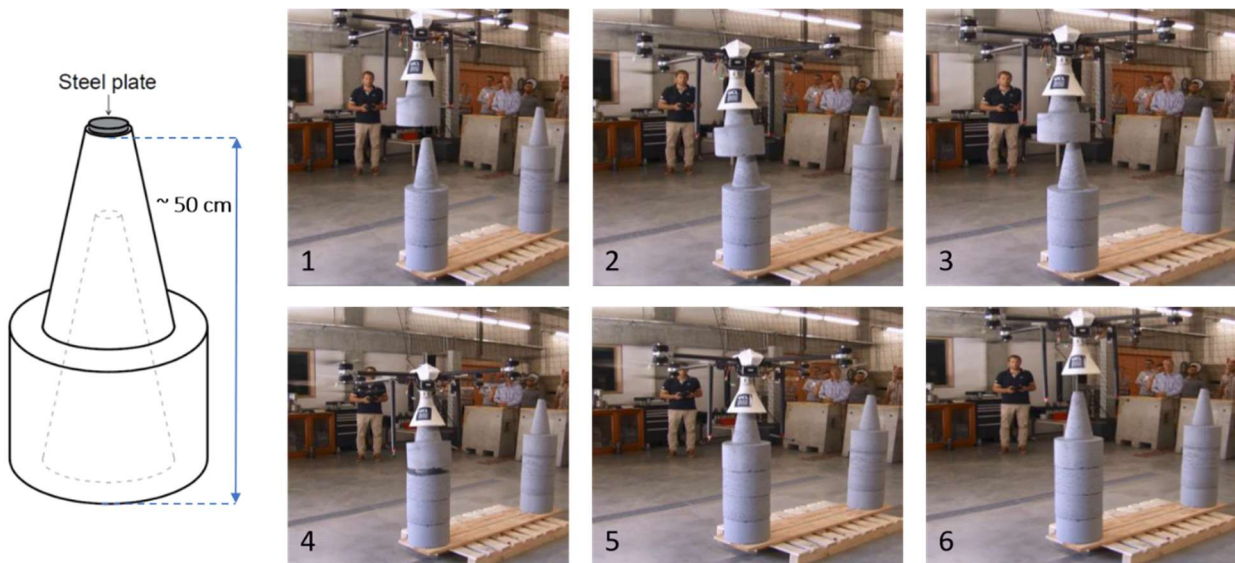


Figure 1.21 - Conical drick with a laying tolerance of 6.5 cm. [24]

1.3.2 Rectangular dricks

Rectangular dricks were also created: their form allows to build enclosed, modular, rectangle-like rooms (see Figure 1.22) [24].

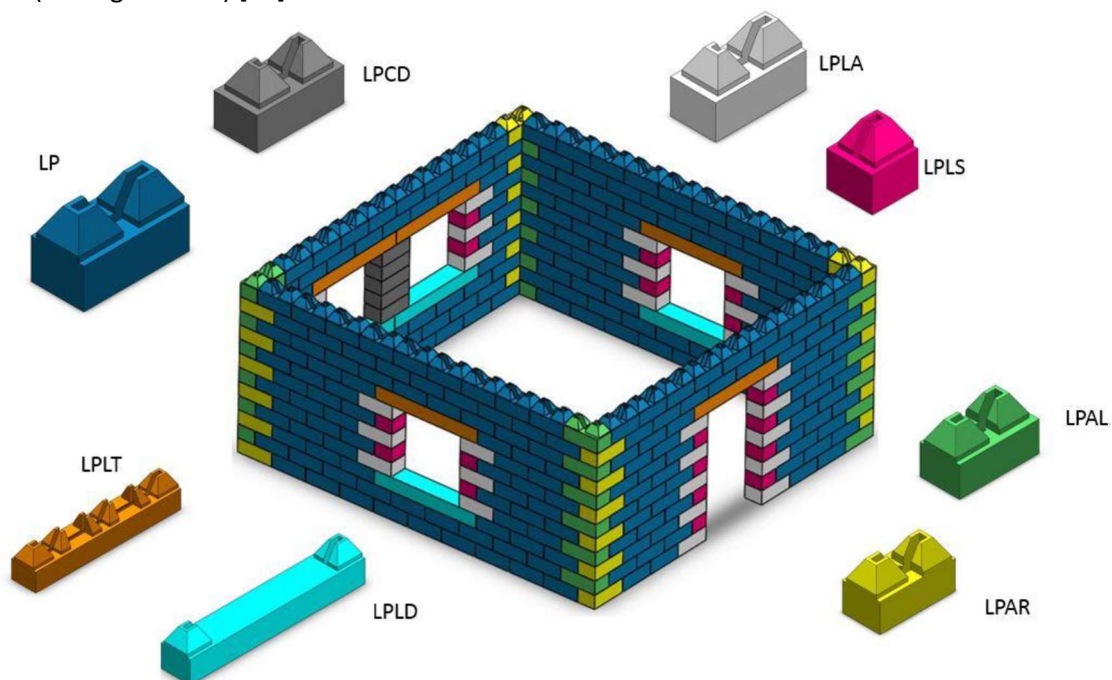


Figure 1.22 - 8 types of rectangular dricks allowing to realise straight wall configurations [24]

1.3.3 Extrudable dricks

A disadvantage of the previous dricks is their complex form and complex moulding form. This is why some thought was given to extrudable dricks: their moulding is quicker and cheaper. The dricks represented in Figure 1.23 are an example. The interlocking capacities of these dricks is however limited and some additional mortar might be necessary.

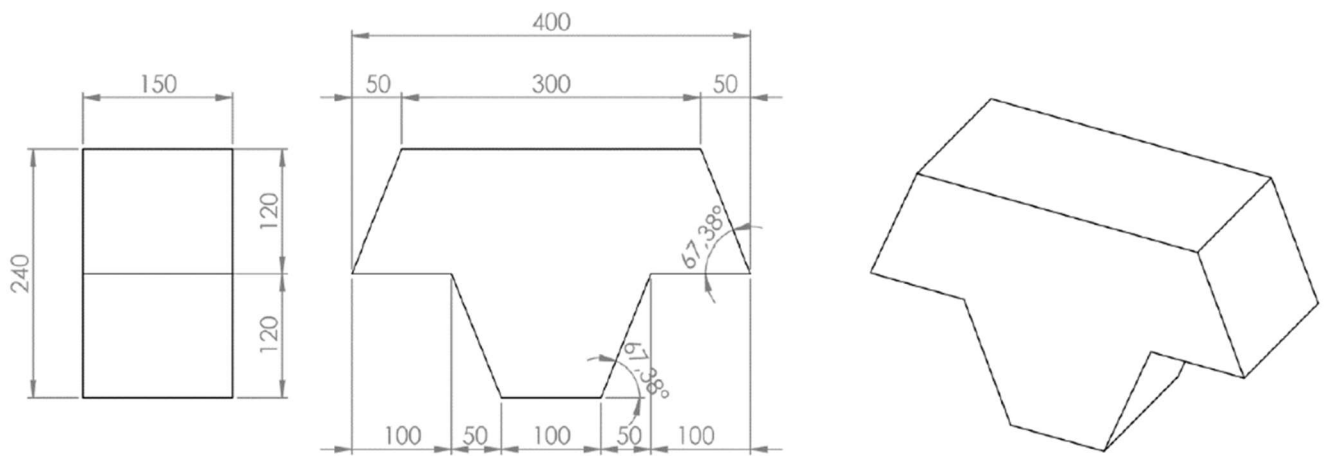


Figure 1.23 - Dimensions of an extrudable brick [24]

CHAPTER 2 THE DROXEL: STATE OF THE ART

The second chapter of this thesis aims to explain the philosophy behind the development of droxels, to give detailed information on the geometry of droxels as well as an overview of the advantages and disadvantages of droxel construction.

2.1 ORIGIN AND DEVELOPMENT OF THE DROXEL

Under the impulse of the development of stackable elements, and with the idea to build a drone-compatible stackable block, the droxel was first developed in 2015 by J.-S. BRETON and J. LEPLAT during their master thesis *“Feasibility study for drone-based manufacturing of architectural structures”*. A synthesis on the evolution of the droxels is given by S. GOESSENS (see Figure 2.1 and Figure 2.2) [24].

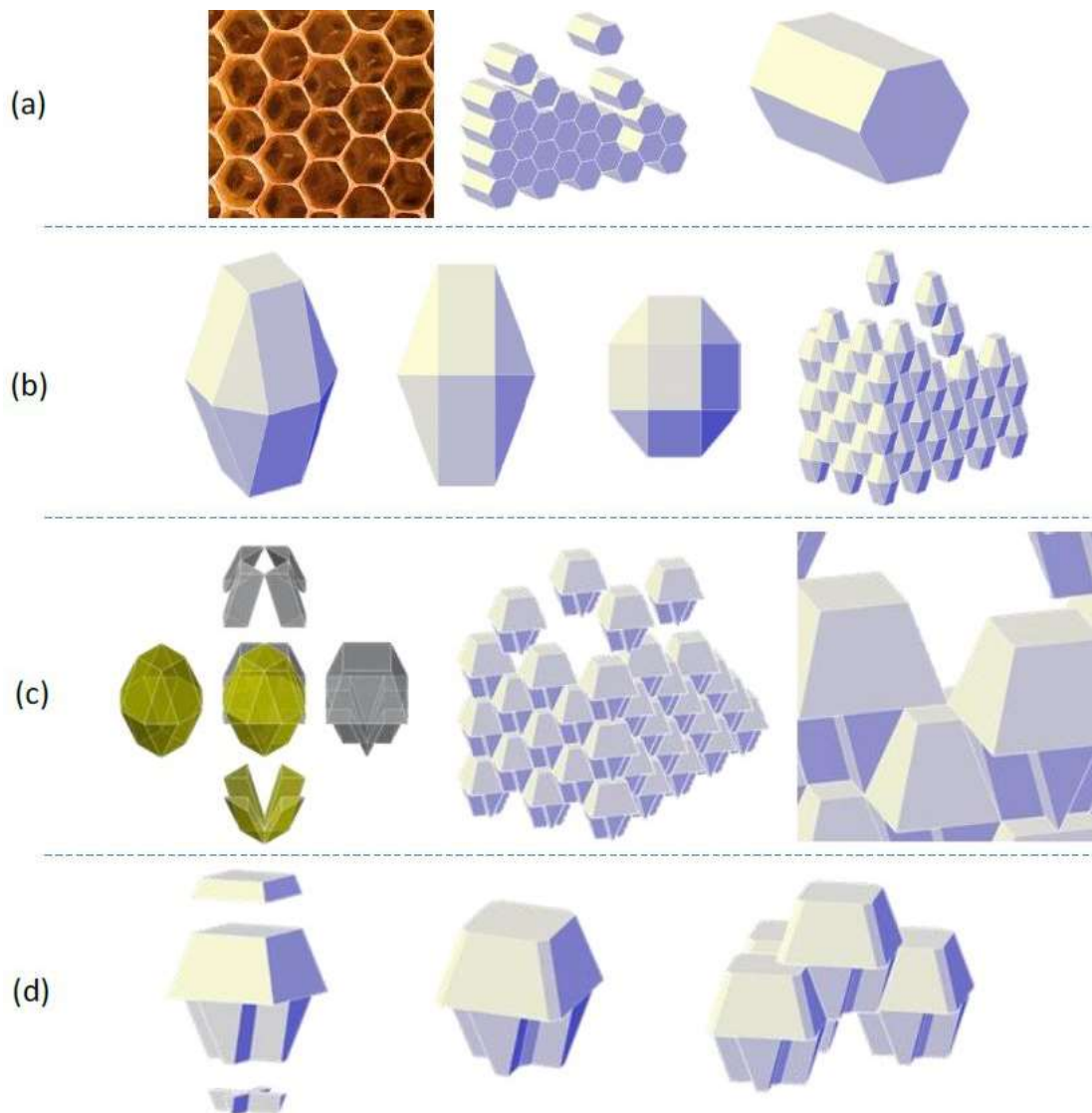


Figure 2.1 - Step-by-step design philosophy of the droxel (part I) [24]

First, a 2D wall was considered: the bees' nests, for which the hexagonal form allow for a large laying tolerance. However, such layout does not allow for the creation of perpendicular walls: this issue was solved by step (b). The laying tolerance is still assured for this configuration considering the blocks are placed in the plane of the wall. In order to have a laying

tolerance out of the plane of the wall a part of the block is removed at the bottom and added at its top (step (c)).

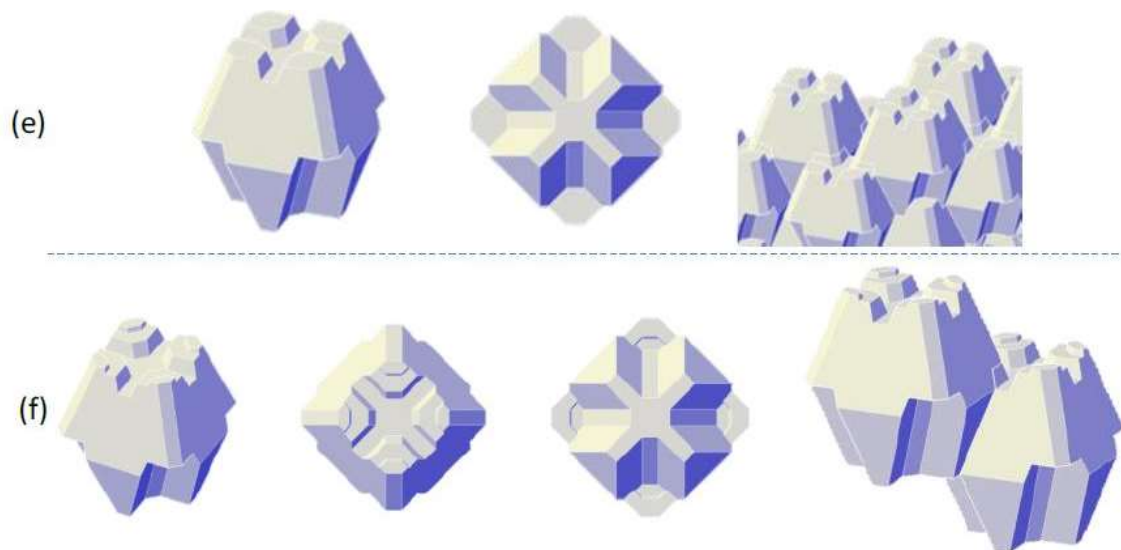


Figure 2.2 - Step-by-step design philosophy of the droxel (part II) [24]

The last two steps are done in order to limit the sharp angles as much as possible in the droxel. This is done to avoid important local stresses. Step (e) assures a better interlocking of the blocks and step (f) allows for the possibility of building cantilever structures. The latter is a great advantage of droxels.

2.2 DROXEL GEOMETRY

Before diving into the design and calculation of structures made out of stacked droxels, which is the final goal of this work, let us first take a closer look at the droxel itself.

The droxel, contraction of the word drone and voxel¹, is a parametric geometrical form. Indeed, the droxel's geometry depends on 8 parameters: H , L , A , E , Cr , R , B and F which are represented in Figure 2.3 :

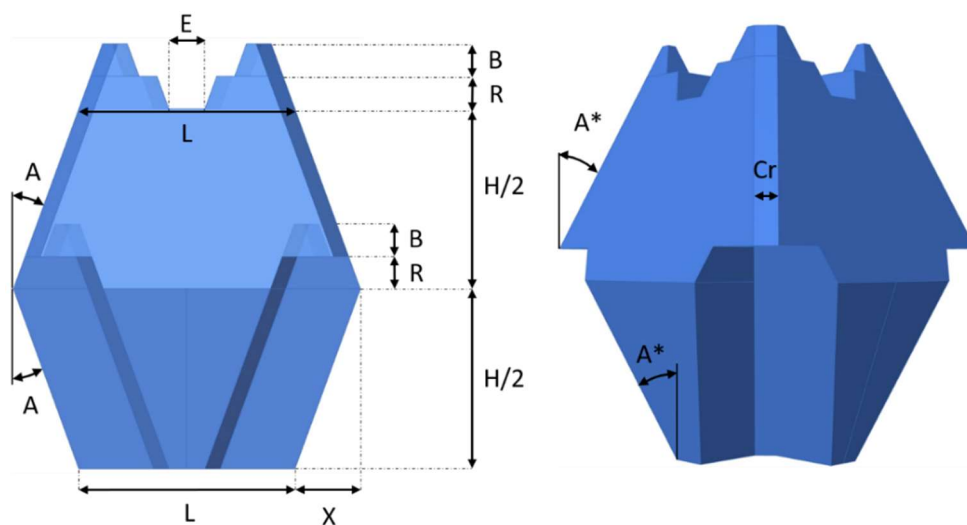


Figure 2.3 - Droxel parameters, X and A^*

The parameters X and A^* are also represented in Figure 2.3 because they are useful values for later.

¹ A voxel is a three-dimensional pixel

Their value can easily be found:

- $X = \tan(A) \cdot H/2$
- $\tan(A^*) = \sqrt{2} \cdot \tan(A)$

The last parameter F is not represented in Figure 2.3. It is a dimensionless number representing the relative length between the base length of the small upper hooks relative to the base length of the big upper hook: $F = B_1/B_2$ (see Figure 2.4).

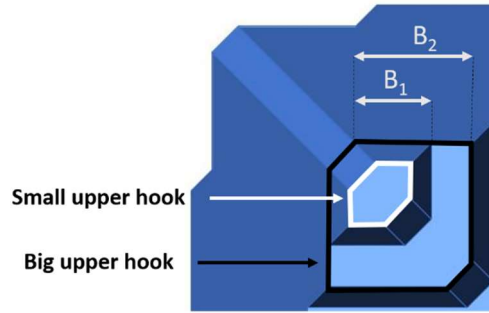


Figure 2.4 - Top view of $\frac{1}{4}^h$ droxel: droxel parameter $F = B_1/B_2$

Changing the 8 parameters changes the aspect of the droxel as shown in Figure 2.5 :

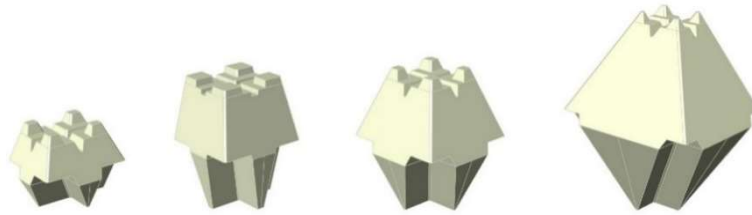


Figure 2.5 - Different possible droxels shapes obtained by varying the 8 parameters

Not only the final aspect of the blocks but the final structure's look, the droxels' laying tolerance and other aspects are influenced by the choice of these parameters.

2.2.1 Droxel volume

The droxel volume, which is used in section 2.3 is given by [35]:

$$V_{droxel} = L^3 \cdot \frac{H}{L} \cdot \left[\frac{\tan(A)^2}{3} \left(\frac{H}{L}\right)^2 + \tan(A) \cdot \left(\frac{H}{L}\right) + \frac{1 + \frac{2E}{L} - \left(\frac{E}{L}\right)^2}{2} \right]$$

2.3 DROXEL PARAMETERS AND THEIR INFLUENCE ON THE FINAL STRUCTURE

In the next part, the influence of every parameter is discussed in order to have a better view of what every parameter has to offer. Some proprieties depend on several parameters, they are also discussed.

2.3.1 Height: H

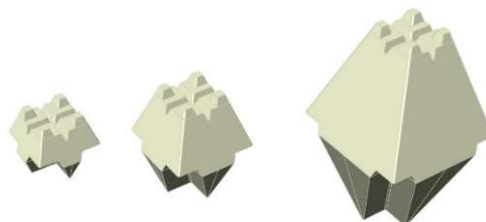


Figure 2.6 - Influence of H on the droxel geometry

The influence of H on a droxels geometry is pretty straightforward: the height of a wall increases with it (see Figure 2.7). In order to reduce the number of used droxels, one might be tempted to take the highest possible droxel height.

However, the total weight of a droxel should also be considered since it increases with H . The total volume of a droxel is proportional to H^3 . For a wall of total height h_{wall} , the number of droxel layers necessary is equal to $\frac{h_{wall}}{H_{droxel}/2}$. The total wall volume is therefore proportional to H^2 . Indeed:

$$V_{tot,wall} \propto \text{number of layers} \cdot \text{volume of a droxel} = \frac{h_{wall}}{H_{droxel}/2} \cdot H_{droxel}^3 = H_{droxel}^2 \cdot h_{wall}$$

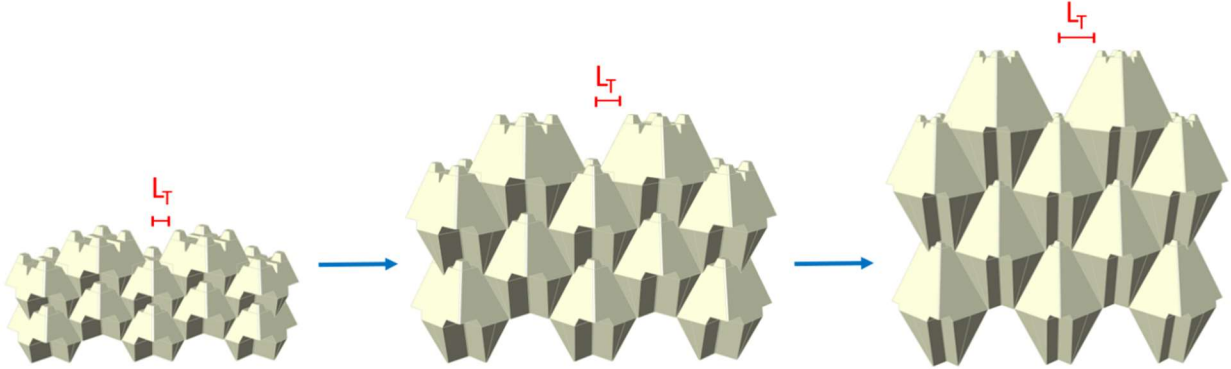


Figure 2.7 - Influence of H on the final geometry of a droxel wall

In Figure 2.7 it can also be seen that the laying tolerance L_T (i.e. zone in which the upper droxel can be placed whilst still ending up in the correct final position) increases with H . More information on the laying tolerance is given in section 2.3.9.

2.3.2 Width: L

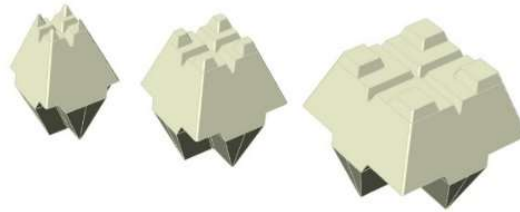


Figure 2.8 - Influence of L on the droxel geometry

Looking at a droxel wall it becomes clear that, the bigger L gets, the longer a wall becomes (Figure 2.9).

The total volume of a droxel is also proportional to L^3 . For a wall of total length l_{wall} , the number of droxel necessary is proportional to $\frac{l_{wall}}{L_{droxel}}$. The total wall volume is therefore also proportional to L^2 :

$$V_{tot,wall} \propto \frac{l_{wall}}{L_{droxel}} \cdot L_{droxel}^3 = L_{droxel}^2 \cdot l_{wall}$$

The L can however not be too small the first reason being that the number of droxels to place has to stay reasonable. The smaller L , the more droxels necessary to build a wall of imposed dimensions.

Naturally, an augmented value of the width leads to a thicker structure and a loss of space (see Figure 2.10), an equilibrium must therefore be found. The possible loss in space indicates us that it might be preferable to increase the H instead of the L if possible.

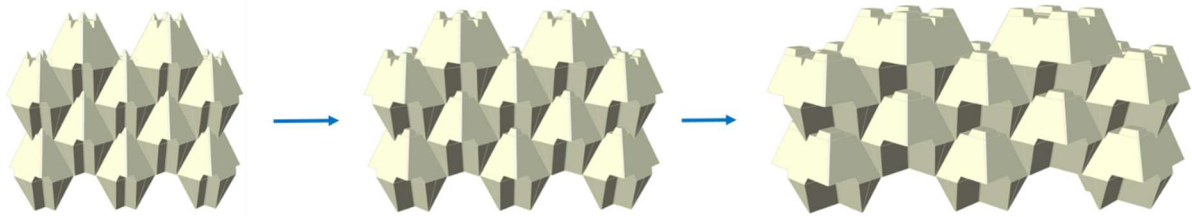


Figure 2.9 - Influence of L on the final geometry of a droxel wall

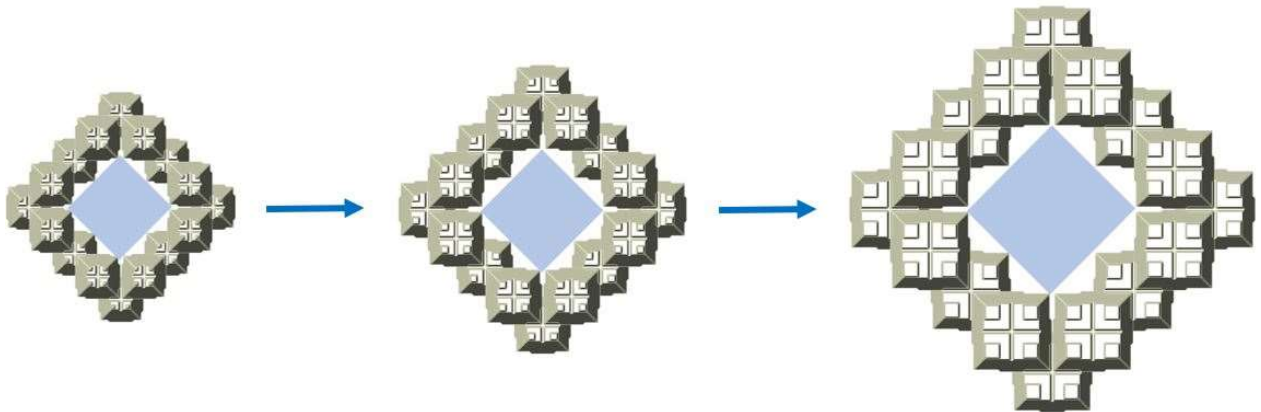


Figure 2.10 - Influence of L on the ground surface of a structure (top view)

2.3.3 Angle: A

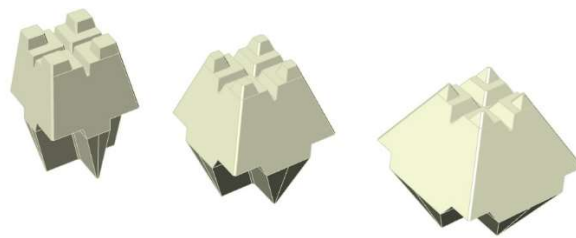


Figure 2.11 - Influence of A on the droxel geometry

The angle A is an important parameter than cannot be neglected. The main motivation for the choice of A is the desired inclination in a structure as well as the maximum number of droxels possible in cantilever mode.

The smaller A , the bigger the inclination and the more droxels can be put in cantilever mode (see Figure 2.12).

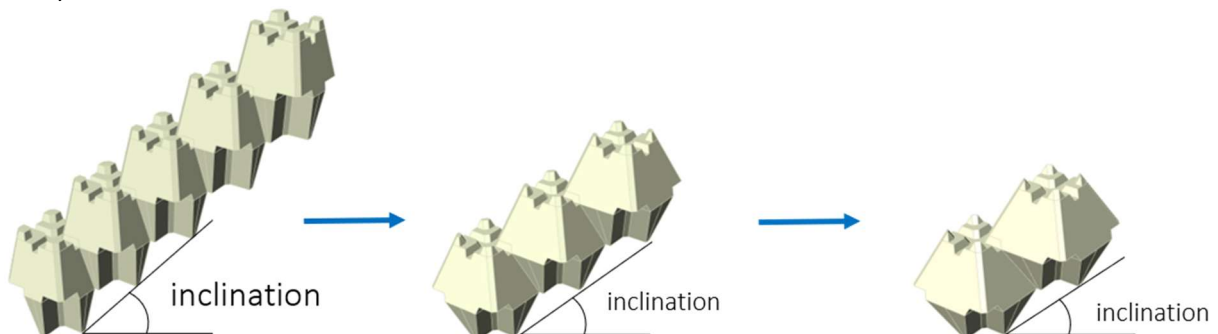


Figure 2.12 - Maximum number of droxels in cantilever mode depends on A (and on the friction coefficient).

Two angles can be obtained in function of where one stands. They are represented in Figure 2.13 :

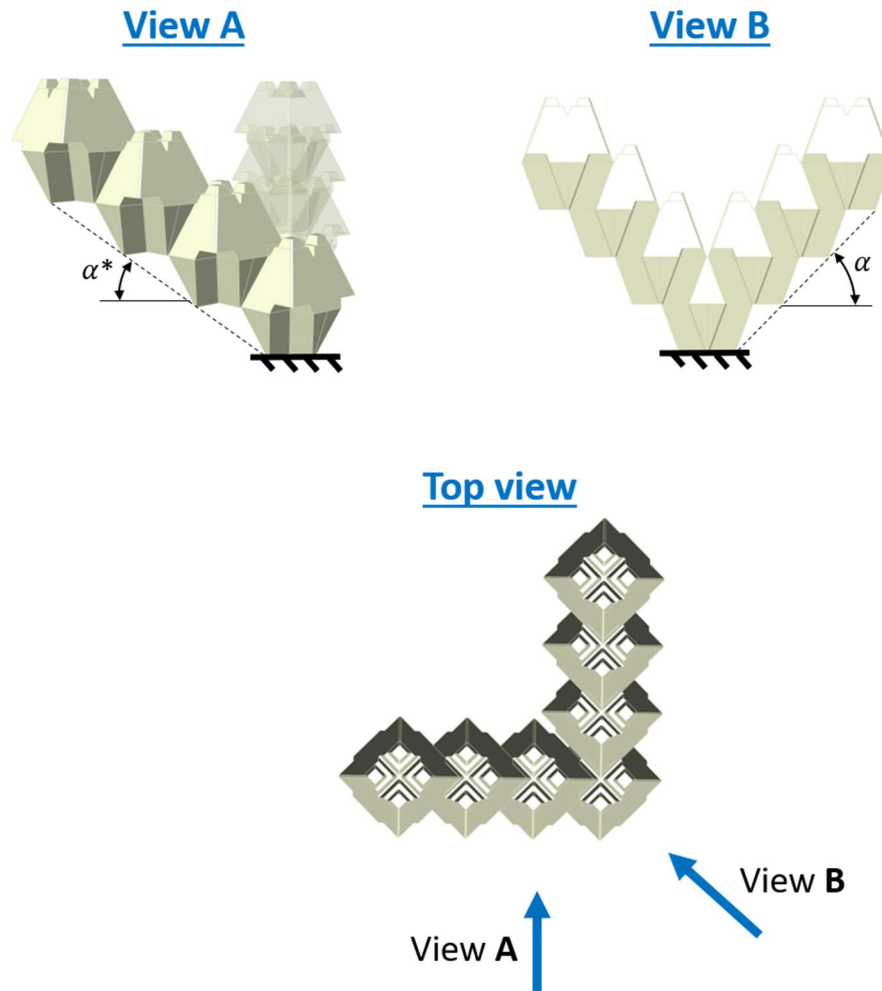


Figure 2.13 - Possible angles in function of the orientation

The angles are given by :

- $\tan(\alpha) = \frac{H/L}{(1+E/L)+\tan(A)\cdot H/L}$
- $\tan(\alpha^*) = \frac{H/L}{\sqrt{2}(1+E/L)+\tan(A^*)\cdot H/L} = \tan(\alpha^*)/\sqrt{2}$

Slope change in droxelised structures

The fact that only 2 angles can be realised with a droxel is rather limiting for the construction of complex structures. This led to a search for possibilities allowing to realise several angles in a structure. A solution, represented on Figure 2.14, was found [35]. This makes construction with droxels even more pertinent: one is no longer limited by an imposed angle.

Two layers of transition droxels are necessary in order to allow for a slope change in the structure.

There are in total 4 droxels: the first (inferior) droxel geometry, two different transition droxels and the second (superior) droxel geometry. Some links must exist between them: R , B , Cr , and F must remain the same for all four. Both transition layers are made of special droxels T_1 , T_2 with 12 parameters instead of 8: L_t , L_b , E_t , E_b , Cr , A_t , A_b , H_t , H_b , R , B and F . The suffix t stands for the top of the droxel. The suffix b for bottom of the droxel.

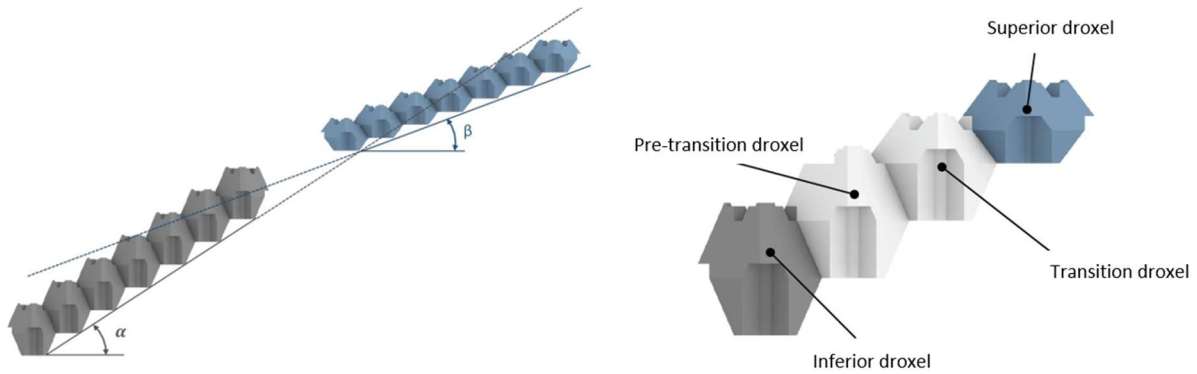


Figure 2.14 - Slope change with transition droxels, as represented by G. GUILMIN [26]

Note: If the angle β is fixed, only three of the parameters H , L , E , A of the superior droxels can be chosen because following relationship is valid: $\tan(\beta) = \frac{(H/L)}{(1+\frac{E}{L})+\tan(A)(H/L)}$.

However, the calculation of structures with different droxel geometries is not the subject of this document. It is from now on assumed that a structure only has one possible droxel geometry.

2.3.4 Cross width: E

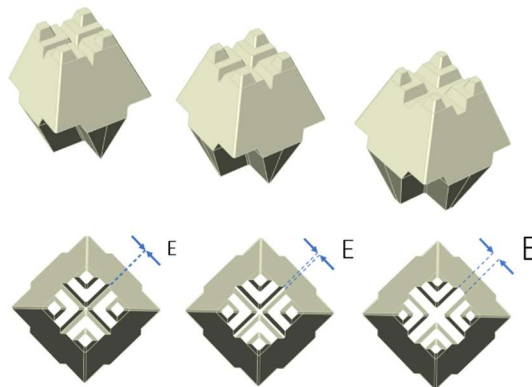


Figure 2.15 - Influence of E on the droxel geometry, perspective and top view

E has to have a minimal value in order to avoid thin edges and locally important stresses on them. Its minimum value depends on the chosen material for the droxels. $L/E = 0.1$ is generally recommended, except for small concrete droxels [35].

Looking at an dome-like structure (Figure 2.16), it can however be seen that that the bigger E , the bigger the “holes” in inclined walls get.

An equilibrium must therefore be found.

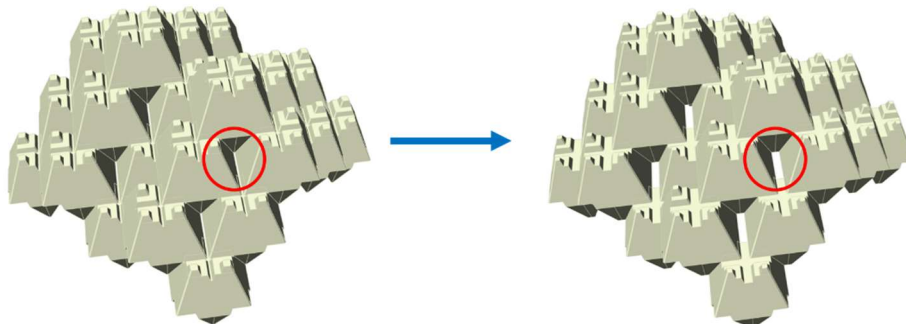


Figure 2.16 - Influence of E on the openings in the structure

2.3.5 Big upper hook's height: R

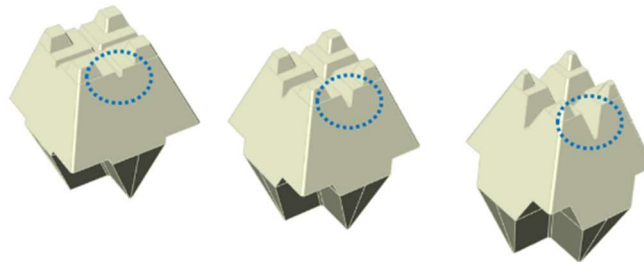


Figure 2.17 - Influence of R on the droxel geometry

A bigger R allows for better interlocking of the droxels because it limits horizontal displacements between placed droxels. It also allows for a bigger laying tolerance.

But with a bigger R , the upper hook's base becomes smaller. The shear resistance and the transmission of vertical forces of the droxel is also reduced for a bigger R .

Taking all those factors into consideration, a rather small value for R ($R/H = 0.05 \dots 0.1$) is recommended [35].

2.3.6 Height of the small upper hooks: B

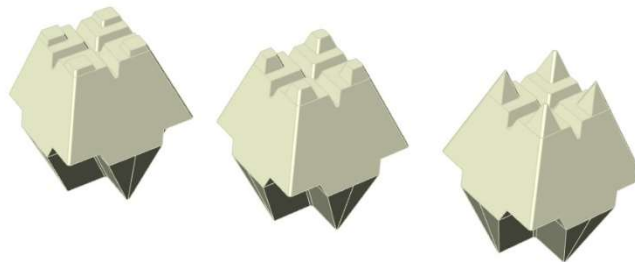


Figure 2.18 - Influence of B on the droxel geometry

The B parameter makes it possible to have cantilever structures (Figure 2.19). The greater B , the better the transmission of the friction forces to the lower droxel. A higher B also results in more droxels possible in cantilever mode.

A greater value of B is however synonym of sharper angles in the droxel and important local stresses.

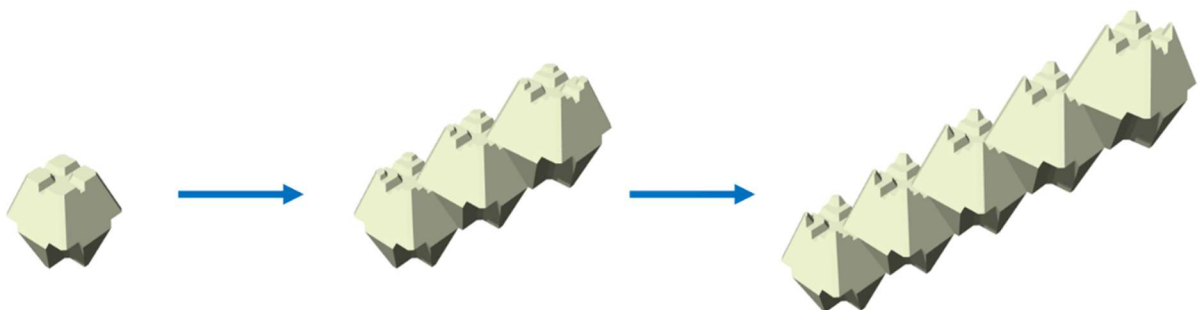


Figure 2.19 - Example of a cantilever structure (possible for $B > 0$)

2.3.7 Chamfer: C_r

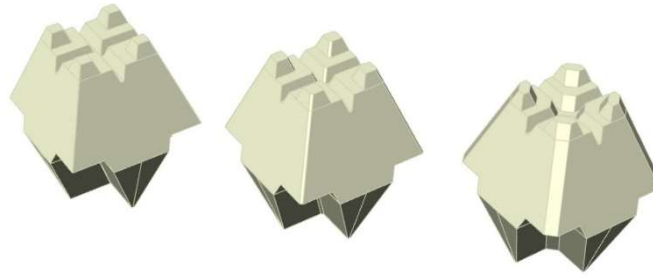


Figure 2.20 - Influence of C_r on the droxel geometry

The chamfer is necessary to avoid sharp angles and local effort concentrations in droxels. The bigger C_r , the bigger the chamfer. However, as C_r increases, the laying tolerance decreases.

C_r is therefore chosen as small as possible in function of the material. The ratio C_r/L should always be limited to 0.05 [35].

For timber, poured materials with no aggregates and for polymers, the minimum recommended values are 3 to 5 [mm]. For concrete, the minimum recommended C_r is 10 [mm].

2.3.8 Upper hook's ratio: F

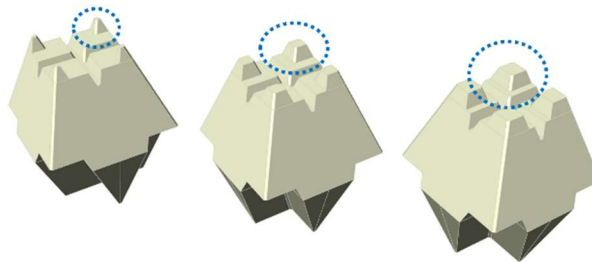


Figure 2.21 - Influence of F on the droxel geometry

The bigger the F -factor, the bigger the shear resistance (i.e., the resistance of a droxel to a horizontal force). A bigger F -value however results in a smaller transition of vertical forces between droxels placed vertically on top of each other.

2.3.9 Laying tolerance: L, R, A, H, E, C_r

As explained before, the laying tolerance depends on several parameters. Those parameters are L, R, A, H, R and C_r . For construction facility, a laying criterion was established [35]:

- $$L_c = E + \tan(A) \cdot (2R + H) + 1,41 \cdot C_r < L$$

If this criterion is respected, the laying zone is a square of size $L_t = (R + H/2) \cdot \tan(A)$ (see Figure 2.22).

In order to maximize the laying tolerance L_t and respect the laying criterion, the lowest possible E and C_r should be chosen.

This criterium is important for the automatization of the construction with droxels. The programmed robot needs to have a laying tolerance of maximum L_t .

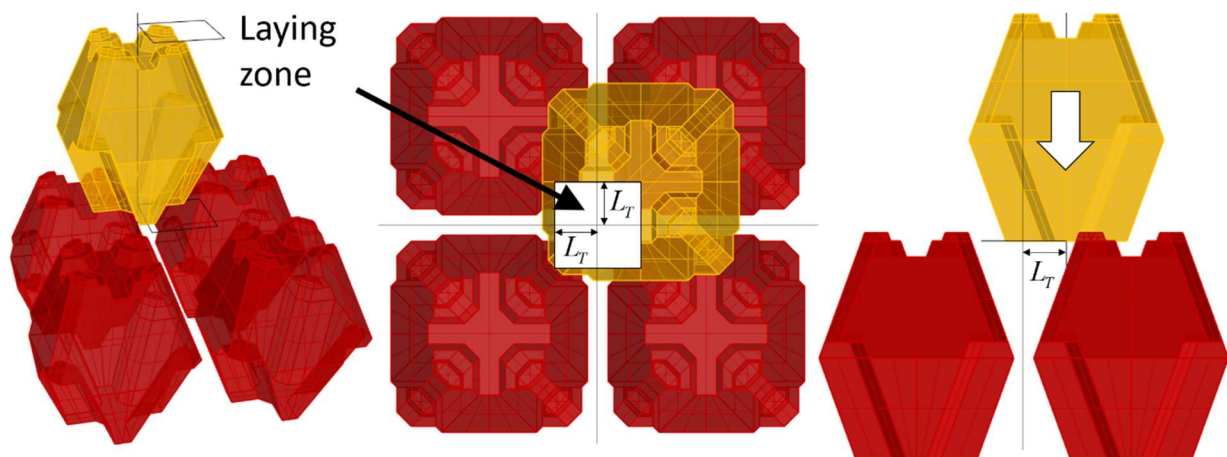


Figure 2.22 - Laying tolerance of a droxel, as represented in [35]

2.3.10 Choice of the parameters: a calculated compromise

The choice of the parameters has an influence on the characteristics of the droxel. The choice of every parameter is a compromise, which could be represented as the following function:

$$Choice(H, L, A, E, R, B, C_r, F) = \begin{cases} \max f(\sum a_i \cdot \text{positive influence}_i) \\ \min f(\sum (1 - a_i) \cdot \text{negative influence}_i) \end{cases}$$

with a_i the weight given to a certain influence_i .

A summary table of the influence of the different parameters is given in [35]:

	Chamfer C_r/L	Base cross E/L	Big upper hook's height R/H	Small upper hook's height B/H	Small upper hook's F factor F	Angle A	Geometrical aspect H/L
High laying tolerance L_T	↓	↓	↑			↑↑	↑↑
Low slope α and α^*		(↑)				↑	↓↓
High cantilever lenght (number of droxels in cantilever mode)				↑		↓↓	
High vertical Load Transfert Ratio VLTR, high value of A_2	(↓)	↓↓	↓↓		↓↓	↓↓	↓↓
High shear resistance of small upper hook (cantilever mode) - high value of A_1	(↓)	↓↓	↓↓		↑↑	↓↓	↓↓

Figure 2.23 - Summary chart with the influences of the different parameters [35]

2.4 DROXEL CONSTRUCTIONS: ADVANTAGES AND DISADVANTAGES

With this development, automated construction with droxels almost seems at the grasp of our hand. But how pertinent are droxels? Next, an overview of their main advantages and disadvantages.

2.4.1 Advantages

Recycling

The first advantage of droxels is the recycling possibilities. This advantage is not neglectable, especially in a time where circular economy and life cycle of elements are important criteria. Droxels are stackable elements that do not require any binding between them. It is therefore possible to build a

structure and to reuse the droxels when the structure is dismantled. This means a droxel could be reused an important number of times before becoming obsolete due to wear, external factors, etc. In case of plastic droxels, the droxels could be made out of recycled plastic. When the droxels get to the end of their life-cycle, the plastic could once again be recycled.

Quick assembly and dismantling

If one wants to quickly build a structure (for example for emergency shelters), droxels are an optimal solution. The dismantling of these structures is also easy since the droxels are simply stacked on top of each other.

Possibility of autonomous use

The interlocking of droxels is easy and their stacking is very intuitive: with a simple manual or with a program that can determine the laying sequence of a structure, it is possible to have several droxels delivered to a desired construction site. The construction itself can then be done by the user or by programmed robots.

Laying tolerance and robotisation possibilities

The laying tolerance, mentioned in 2.3.9, is another advantage of droxel construction. This makes it possible to use droxels for any kind of robotisation.

Pertinent for vault effect

Structures with droxels are particularly pertinent and efficient when a vault effect can be created. In other words, when a funicular geometry is reproduced (see Figure 2.24).

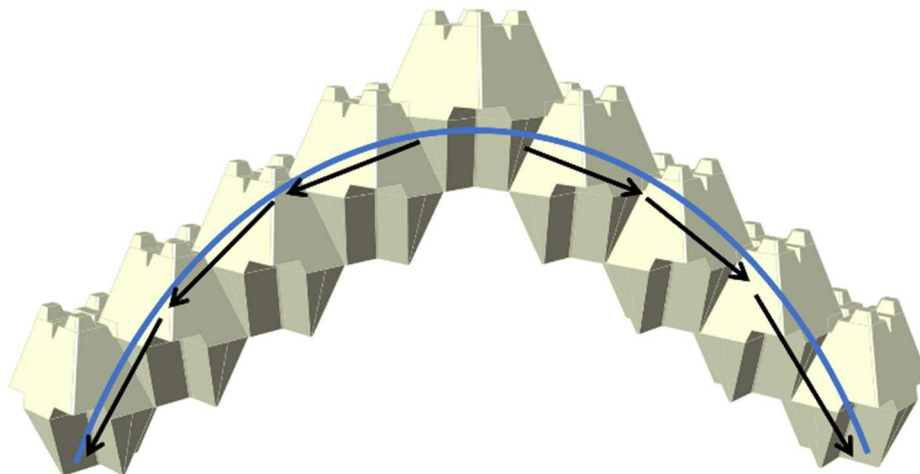


Figure 2.24 - Example of the vault effect in a structure of droxels

Realisation of structures without scaffolding

Droxels allow for the construction of cantilevered structures. This means structures can be built without the need of any scaffolding. An example of a possible laying sequence for the building of an arch is shown in Figure 2.25. The counterbalancing droxels (the pink ones) can be removed once they are no more necessary.

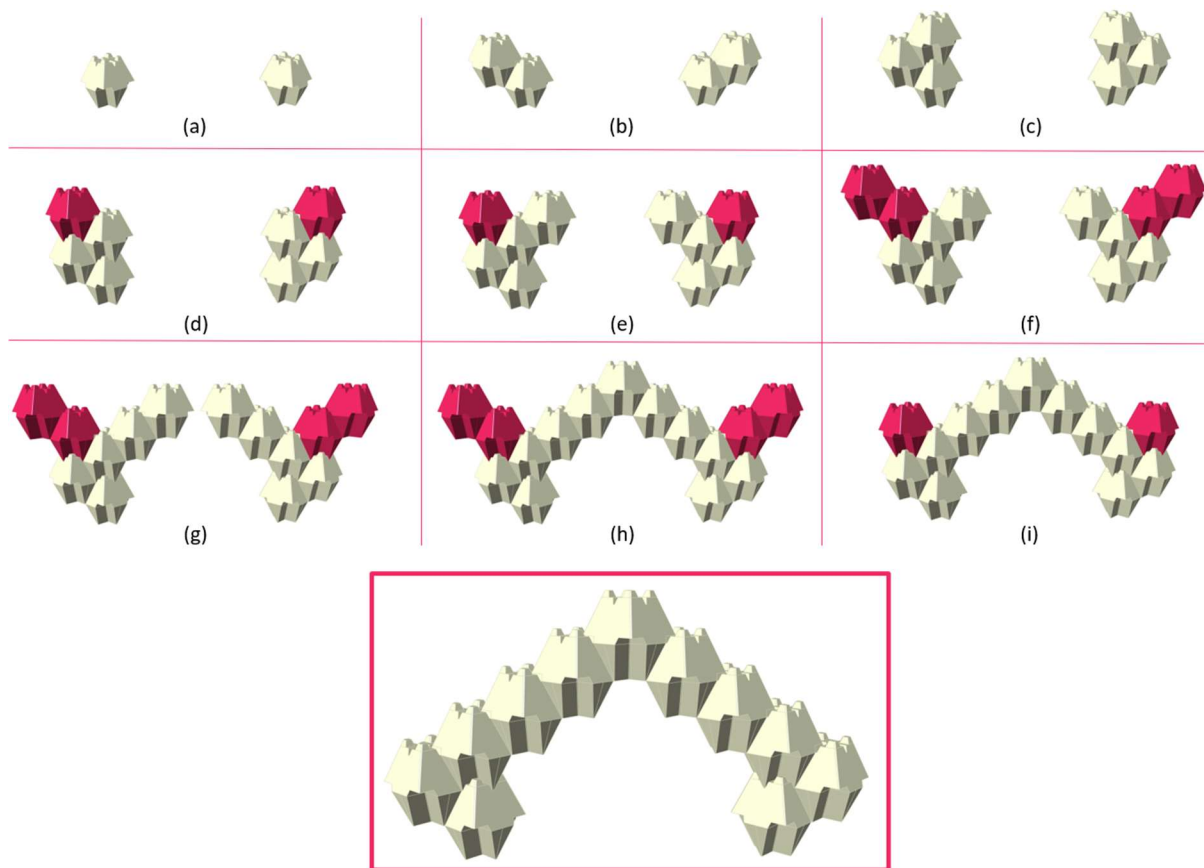


Figure 2.25 - Example of the realisation of an arch in droxels: temporary counterbalancing droxels in pink

Hollow droxels and ballasting

Using hollow droxels drastically reduces the quantity of material needed and thus the total weight of the droxels. This is a great advantage in terms of construction facility. If heavier droxels are preferred (for stability reasons f. e.), it is possible to fill the droxels with some additional weight (sand, water...) once they have been placed correctly.

Going back to the use of temporary droxels in order to realise structures without scaffolding, the quantity of necessary counterbalancing droxels can be reduced if one chooses to fill them with ballast (sand or water e. g.). An example of this is given in Figure 2.26 and Figure 2.27: only two temporary droxels are needed compared to Figure 2.25 where four are necessary.

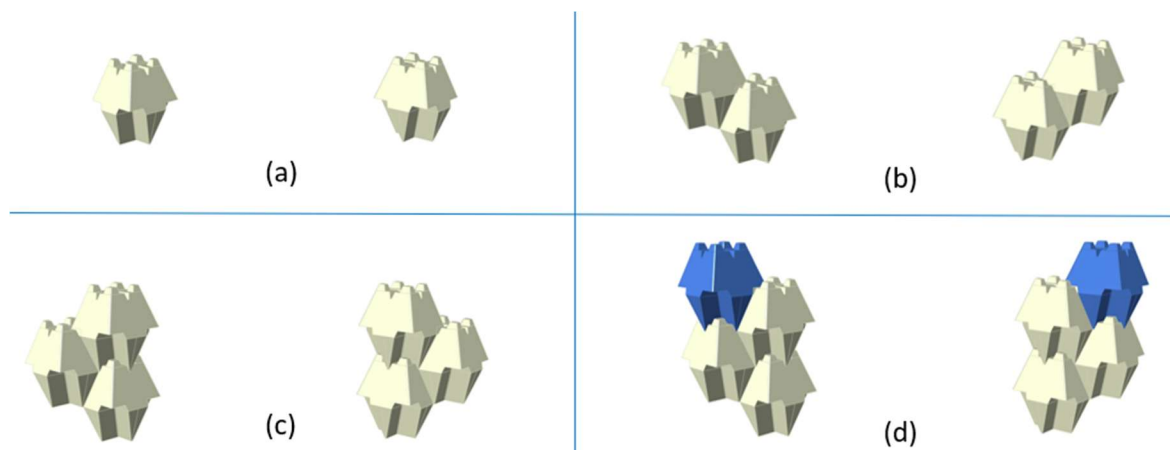


Figure 2.26 - Realisation of an arch in droxels: temporary (ballasted) droxels in blue (part I)

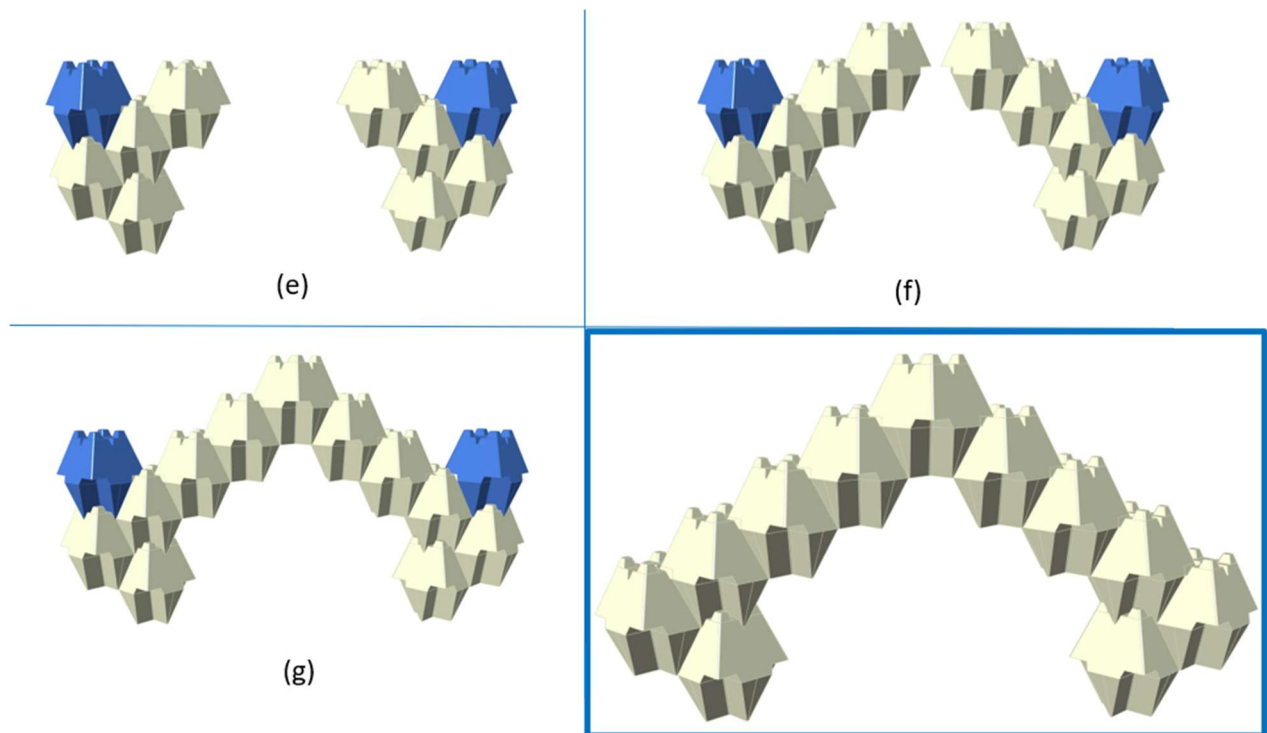


Figure 2.27 - Realisation of an arch in droxels: temporary (ballasted) droxels in blue (part II)

2.4.2 Disadvantages

Calculation

One of the disadvantages of droxel structures is that, at the moment, the calculation of the structures requires time-consuming Discrete Element Methods (DEM). Their calculation must be optimised. This is most certainly one of the next big challenges regarding droxels.

Production

The droxel has a particular and complex geometry. The mass production of droxels requires an adequate and well-thought production method.

Isolation and waterproofing of the structure

The isolation of droxellised structures is another subject that is not tackled yet. The complex form of the droxels might need some specific customized isolation. This is something that must be thought of if one wants to commercialise droxels for isolated structures.

As mentioned, an inclined wall inevitably has some openings (see 2.3.4, p. 21). These openings are problematic when a structure needs to be waterproofed. This makes some structures not (yet) pertinent for droxel construction. It can however easily be imagined that an exterior cover is provided in order to waterproof a certain structure.

Ideally, a slightly different droxel geometry removing the openings will be found.

Change of mentality

The use of droxels, and the further robotisation of the construction industry needs a change of mentality. The idea of a structure being built almost entirely by machines is not something that people can easily conceive yet.

Not pertinent in some cases

In some cases, the use of droxels is not pertinent. For example, when designing straight columns composed of a single droxel. In that case, the interlocking feature of the droxels is not used resulting in a configuration that can easily become unstable when subjected to horizontal forces (see Figure 2.28).

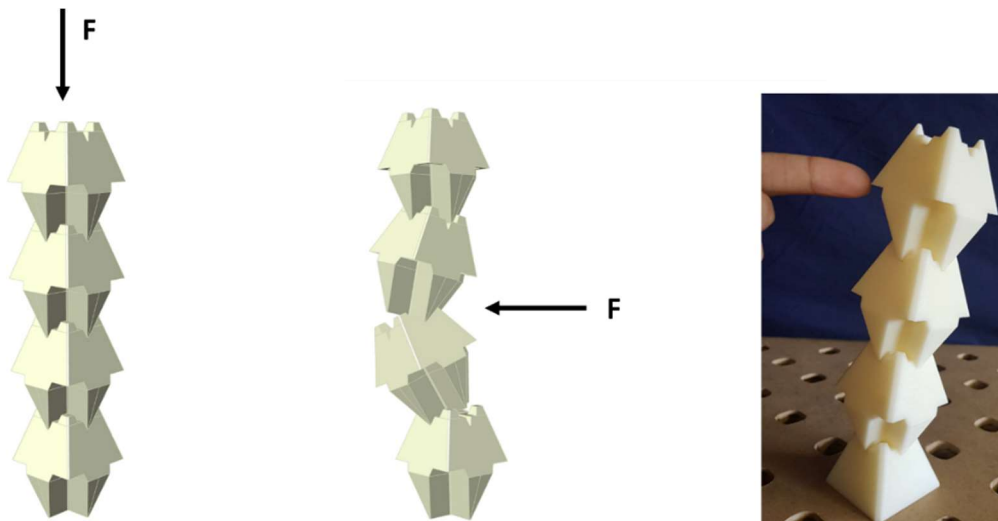


Figure 2.28 - Straight columns composed of one droxel: risk of instability

CONCLUSION

The development of the droxel, as well as its parameters and their influence has been covered. Then, the advantages and disadvantages of droxels were given.

The conclusion is that when used pertinently and with well-thought parameters, construction with droxel can prove to have the upper hand on more traditional methods for some cases.

CHAPTER 3 DROXELISED STRUCTURES: REALISATION AND IMPLEMENTATION

The next chapter has as goal to determine which steps are necessary in order to be able to realise droxelised structures. The implementation and verification of the methods used in this document are also presented.

3.1 REALISATION

In order to be able to construct droxelised structures, several steps are needed. Supposing methods to place the droxels exist (drones, robotic arms, humans...), a proposition of the necessary steps for the realisation of droxelised structures is given in Figure 3.1. The figure shows that this process is not linear: corrections can be made until a satisfying configuration is found.

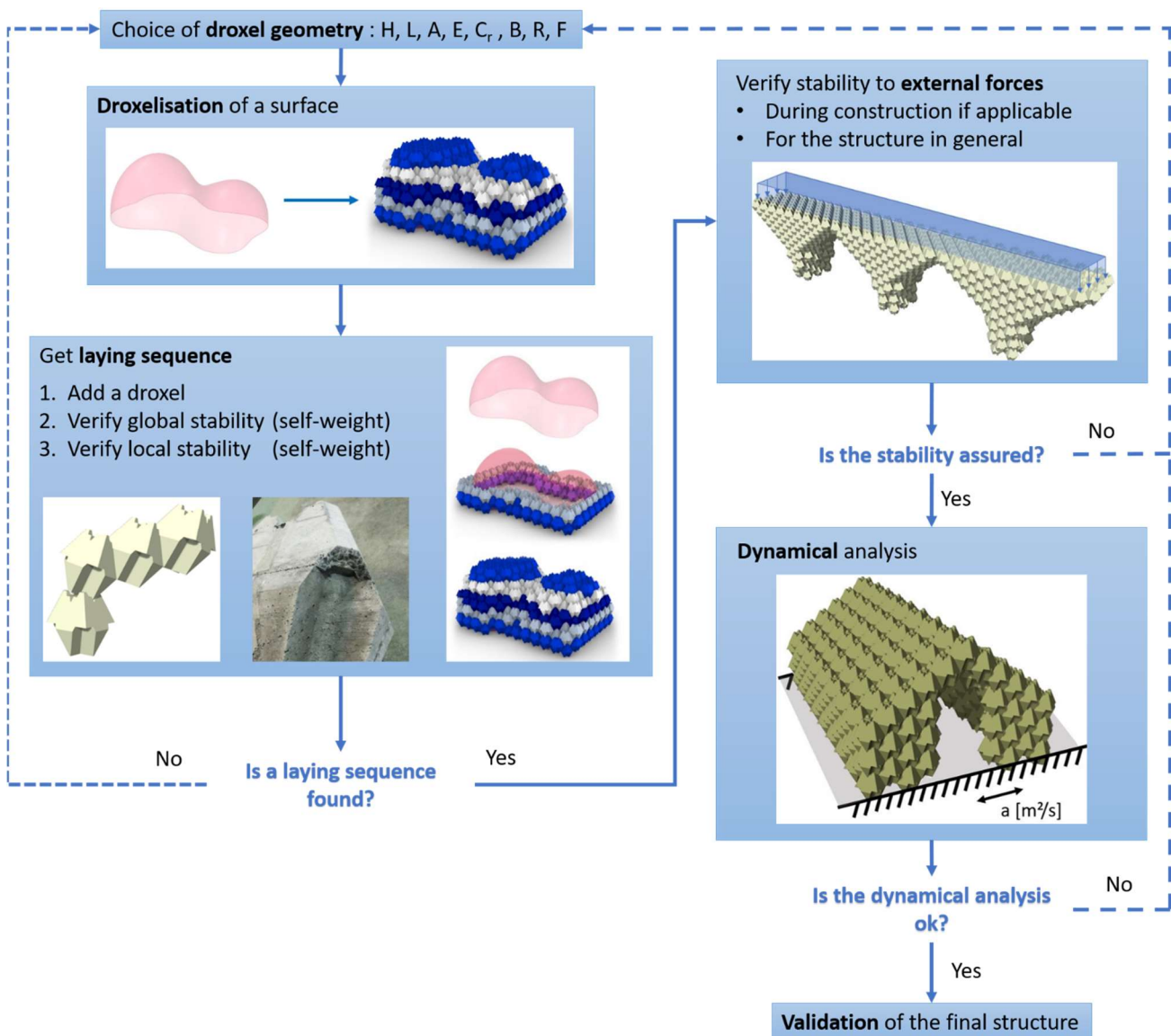


Figure 3.1 - General steps necessary to realise a droxelised structure

Some vocabulary concerning droxels

The **droxelisation** of a surface is the process of determining where to put droxels in order to reproduce a 3D surface (or a desired structure geometry) as accurately as possible.

A **droxelised structure** is a structure composed of droxels in which each droxel that is not resting on the ground surface has at least one lower neighbour (see later CHAPTER 5)

The **laying sequence** is the order in which the droxels must be placed in order to have a stable structure at each step (this is not trivial as is shown in CHAPTER 7).

Steps tackled in this thesis

In this document, two blocks of Figure 3.1 are tackled: *Droxelisation of a surface* and *Get laying sequence*.

There are three important goals in this thesis:

- 1) Implementing an algorithm allowing to determine if a given droxelised structure is stable under its self-weight. At the moment, only the self-weight of the elements is considered;
- 2) Implementing an algorithm to determine an adequate laying sequence for a given droxelised structure. At the moment, only the self-weight of the elements is considered;
- 3) Comparing the two previous algorithms with both numerical and experimental validation.

The stability of a structure is not always guaranteed. Since there is no binding between the blocks, some configurations can create unstable structures (see Figure 3.2).

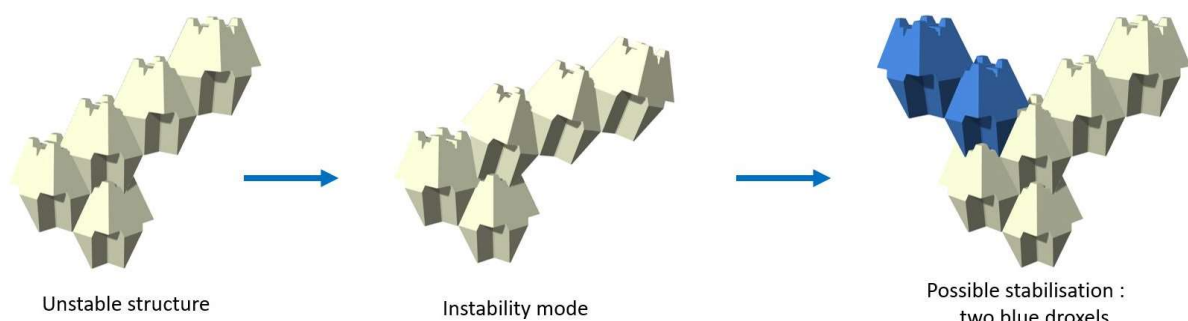


Figure 3.2 - Example of an unstable droxel configuration and a possible stabilisation mode

3.2 IMPLEMENTATION: GRASSHOPPER, RHINO AND C#

The coding the algorithms is done in a *Grasshopper*² interface using *C#*³ codes. The global simplified structure of the coded programs is given below.



Figure 3.3 - Global structure of the Grasshopper code

² "Grasshopper is a graphical algorithm editor tightly integrated with Rhino's 3D-modeling tools." [3]

³ "C# is an object-oriented, component-oriented programming language." [1]

The blocks realise the following :

- (1) Create droxel : creates the droxel geometry
- (2) Droxelisation : computes a droxelised structure
- (3) Stability of a structure : determines whether a given droxelised structure is stable
- (4) Laying sequence of a structure : determines the laying sequence of a droxelised structure

The blocks (2), (3) and (4) are detailed in CHAPTER 5, 6 and 7 respectively.

The first three blocks have already been realised geometrically in previous works [26], however those geometrical constructions require important computation times. This is why all the blocks have been implemented from scratch during this master thesis using *C#* codes, resulting in faster computation times.

Using *Grasshopper* has several advantages when used for droxels:

- 1) *Rhino*, a 3D modelling tool, allows for a live visualisation of the results given by the *C#* codes. This is useful both when debugging the code and in order to see and validate the droxelised structure and/or the final laying sequence.

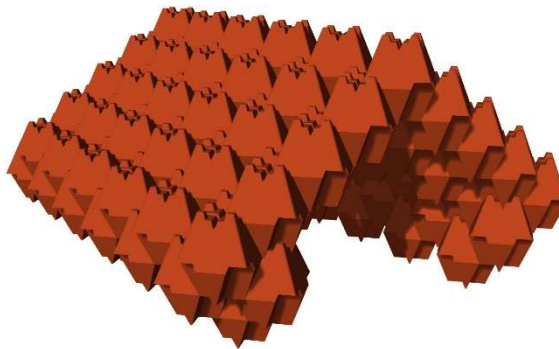


Figure 3.4 - Example of the visualisation of a structure with Rhino

- 2) *Grasshopper* has a very visual interface. It is easy to understand which block realises which part of the algorithm. The green blocks are the one needing human interaction (input of the droxel geometry, of a 3D surface...)

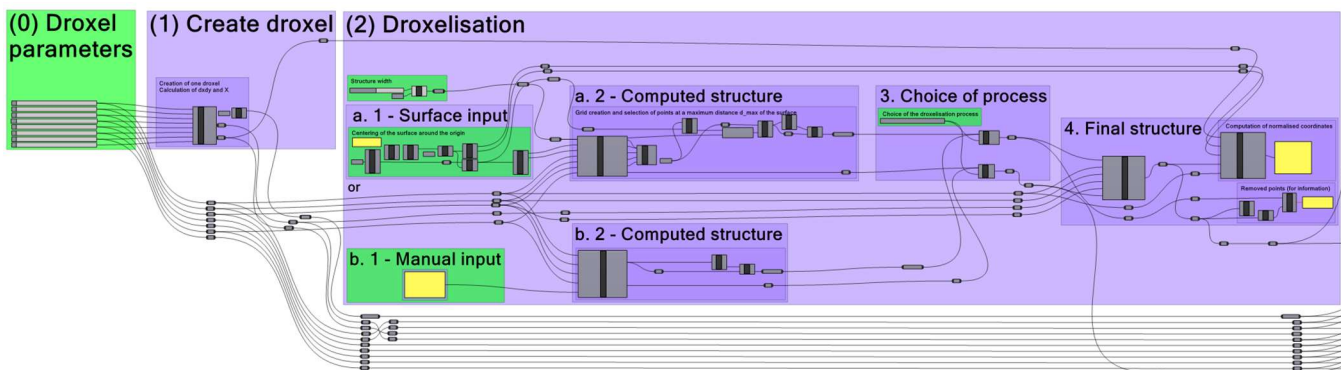


Figure 3.5 - Grasshopper interface (part I)

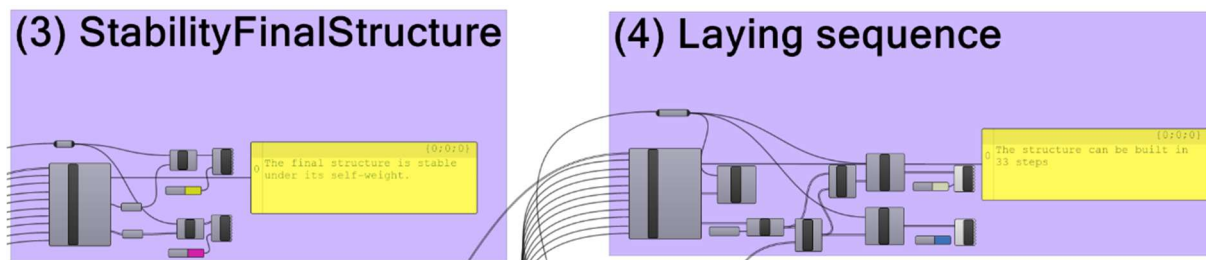


Figure 3.6 - Grasshopper interface (part II)

- 3) The last advantage is that *Grasshopper* also supports *Python*⁴ scripts. This is interesting because the numerical simulations are done with *Python*-code (*Python 2.7*) meaning no translation of the code to *C#* is needed to import the numerical simulation into *Grasshopper*. The codes have been developed in the *LMGC90* library by M. MULLIER [40] under the supervision of F. DUBOIS (*UMontpellier*). The used library allows for the simulation of multibody structures (such as droxelised structures).

The implementation of the numerical simulation in the *Grasshopper* interface is however not the goal of this thesis and is therefore not further discussed.

CONCLUSION

This chapter tackled the necessary steps for the realisation of droxelised structures :

1. Choice of the droxel geometry;
2. Droxelisation of a surface;
3. Determination of the laying sequence
4. Submission to external forces (during construction and final stage)
5. Dynamical analysis

The implementation of the first three steps has been done for this thesis using *C#* scripts implemented in *Grasshopper*. The scripts are discussed in CHAPTER 5, 6 and 7. In order to verify the implementations, numerical simulations and experimental tests are performed. These simulations are addressed in the next chapter.

⁴ Python is an open-source programming language [43].

CHAPTER 4 VERIFICATION METHODS OF THE ALGORITHM

In order to verify the proposed laying sequences, numerical simulations and experimental tests are performed. The numerical simulations are done using Discrete Elements Method (DEM): the first section of this chapter briefly explains the methodology behind DEM, gives some specifications used for the droxel calculation and introduces the software used to implement those DEM: *LMGC90*.

The second section tackles the realisation of small 3D-printed droxels to realise the experiments.

Lastly, some examples are presented in order to compare the three methods (*Grasshopper* algorithm, *LMGC90* and experimental).

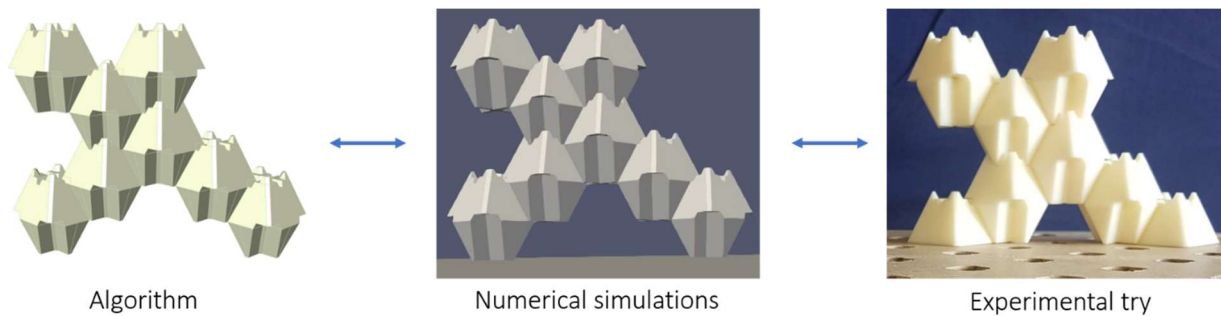


Figure 4.1 - Comparison between the three methods

4.1 NUMERICAL SIMULATION: DISCRETE ELEMENTS METHOD (DEM)

The goal of this section is not to give a complete overview of the Discrete Elements Method but to give some information allowing to understand the basics for the simulation of droxelised structures. A more detailed overview of the DEM is given in APPENDIX B. For even more information, one can refer to M. MULLIER'S master thesis [40] or to specific literature.

4.1.1 Discrete Elements Method (DEM)

Discrete, discontinuous, or multibody structures are structures composed of different interacting bodies (such as droxels f. e.). The interactions between the bodies are modelled as external forces acting on each individual body. DEM can be used to deal with multibody structures and allows for the introduction of two capital features [16]:

- 1) Unilaterality (no interpenetration of the bodies possible);
- 2) Modelling of the friction between the elements.

It is important to note that many methods exist for the simulation of multibody structures, the numerical method DEM is in itself a model, the results must therefore be treated with caution and ideally compared to experiments to validate the model. This is why experimental tests are also realised (see 4.2).

The approach used in DEM is to integrate the equation of motion for each individual body of the structure simultaneously. Since the elements interact with each other, the contact laws between the elements must be correctly determined. In order to reduce the computational cost of these calculations, sorting algorithms can be implemented.

Let us consider a collection of bodies. Their movements can be found solving the equation of motion for each body ($\sum F = m \cdot a$): $\mathcal{M}(q, t)\ddot{V} = F_{ext}(q, V, t) + F_{int}(t) + R$ [17]

With :

- q the degrees of freedom vector;
- V the velocity vector;
- R the resultant of the contact forces;
- F_{ext} the vector of external forces;
- F_{int} the vector of internal forces (zero for rigid bodies) and non-linear inertia terms (centrifugal and gyroscopic);
- $\mathcal{M}$ the vector of external moments;
- $\ddot{V}$ the acceleration of the body.

During every time interval in DEM, 3 steps are performed (see Figure 4.2):

1. Computation of the movement of the bodies;
2. Detection of the bodies in interaction with other bodies;
3. Computation of the contact forces between the bodies.

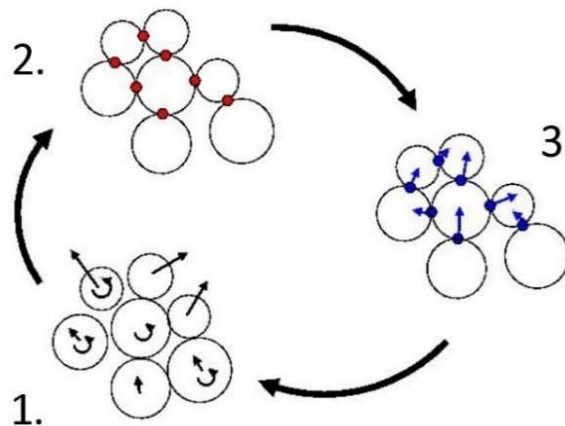


Figure 4.2 - DEM : representation of the tasks executed over a time step [16]

Generally, the iteration stops when a certain imposed t_{tot} has been reached. It is also possible to impose other conditions to stop the simulation (maximum displacement, velocity...). For the simulation of droxels, the iteration stops at a certain time t_{tot} (see 4.1.3).

The hypothesis done for the global stability check (see later CHAPTER 6) is the fact that the droxels are completely rigid. This allows for a faster computation than if the droxels are considered deformable [40]. This hypothesis is justified by the fact that the global stability check verifies a potential tilting in the structure and not an internal rupture.

When checking the local stability, the droxels need to be considered as elastic. This is however not the subject of this work. The modelling of droxels as elastic bodies can also be done using DEM in which case the simulations take the stiffness of the element into account [40].

Contact detection methods

Contact detection is important in order to reduce the computation time for DEM, especially for droxels which are complex forms with a lot of contact surfaces. The contact detection methods are usually divided into 3 phases:

- 1) **Broad phase:** Building of a list with adjacent object within a certain distance (see Figure 4.3).

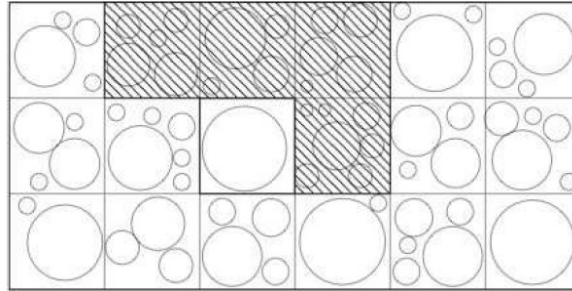


Figure 4.3 - Broad detection: Space division for interacting spheres [17]

- 2) **Narrow phase (optional):** improving the previous phase by removing trivial undesirable items or specifying which part of a body is in contact with another part of a body. An example is given in Figure 4.4: the contact detection is only checked for the pairs *top right* – *down left* and *top left* – *down right*.

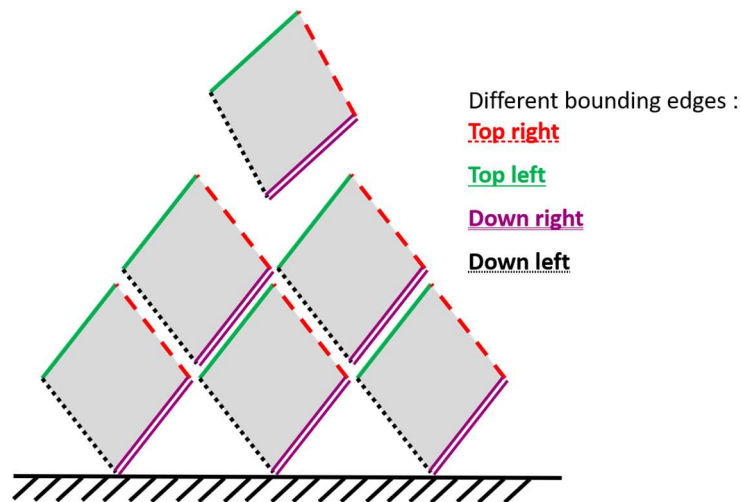


Figure 4.4 - Narrow phase: bounding box hierarchy

- 3) **Contact determination:** Determination of the pairs of touching objects and of the contact loci, relative velocities, distances... Several methods exist for this, an example of the Sweep and Prune method is given in Figure 4.5.

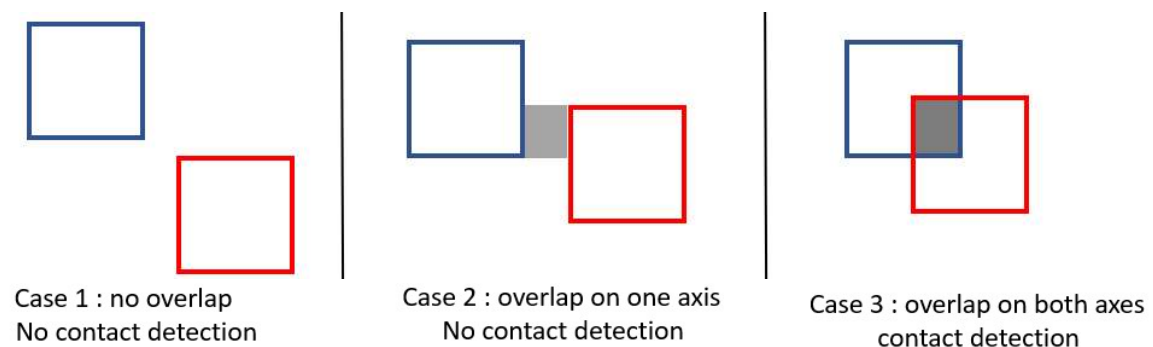


Figure 4.5 - Example of a 2D contact detection method: Sweep and Prune method

4.1.2 LMGC90

The choice of the platform used for the computation of droxelised structures is *LMGC90*, its interface language is *Python*.

As explained by its creators, “*LMGC90* is an open platform dedicated to the modelling of large collections of interacting objects (2D/3D). It aims at modelling objects of any shape with various mechanical behaviour and to take into account interaction laws as complex as necessary.” [18]

LMGC90 being open-source, it has the advantage of allowing the addition of new geometries, behaviours, interaction laws as well as contact detection methods and numerical strategies. The software works both with rigid and elastic bodies.

The structure, implementation and explanation of the code for rigid (and elastic) bodies has already been done by M. MULLIER during his master thesis [40].

4.1.3 Discrete Elements Method applied to Droxels

Used contact law

The used interaction law between the droxels is a SIGNORINI-COULOMB law [17]. Signorini means that no interpenetration of the bodies is possible.

Coulomb refers to the fact that the friction between the elements follows Coulomb’s friction model represented in Figure 4.6. U_N is the normal force between two surfaces, μ their friction coefficient and U_T is the tangential force between them. Two faces start sliding on top of each other when $U_T > \mu \cdot U_N$.



Figure 4.6 - Coulomb's friction model.

Time step

The time steps used in this document are all time steps of 0.15 [sec]. This value is deemed not too small (a much bigger computation time would be necessary for the simulation) nor too big (some contacts might not be spotted resulting in an important interpenetration of the droxels). The impact of the time step Δt is represented in Figure 4.7.

Total simulated time

The total duration of the simulations is always given but to stay coherent it is often equal to 30 or 45 [sec]. This corresponds to 200 or 300 steps of 0.15 [sec]. At the end of each computation, the velocity of the droxels is checked. If a droxel still has an important velocity ($> 1 \text{ mm/sec}$), the simulation is performed with more time steps.

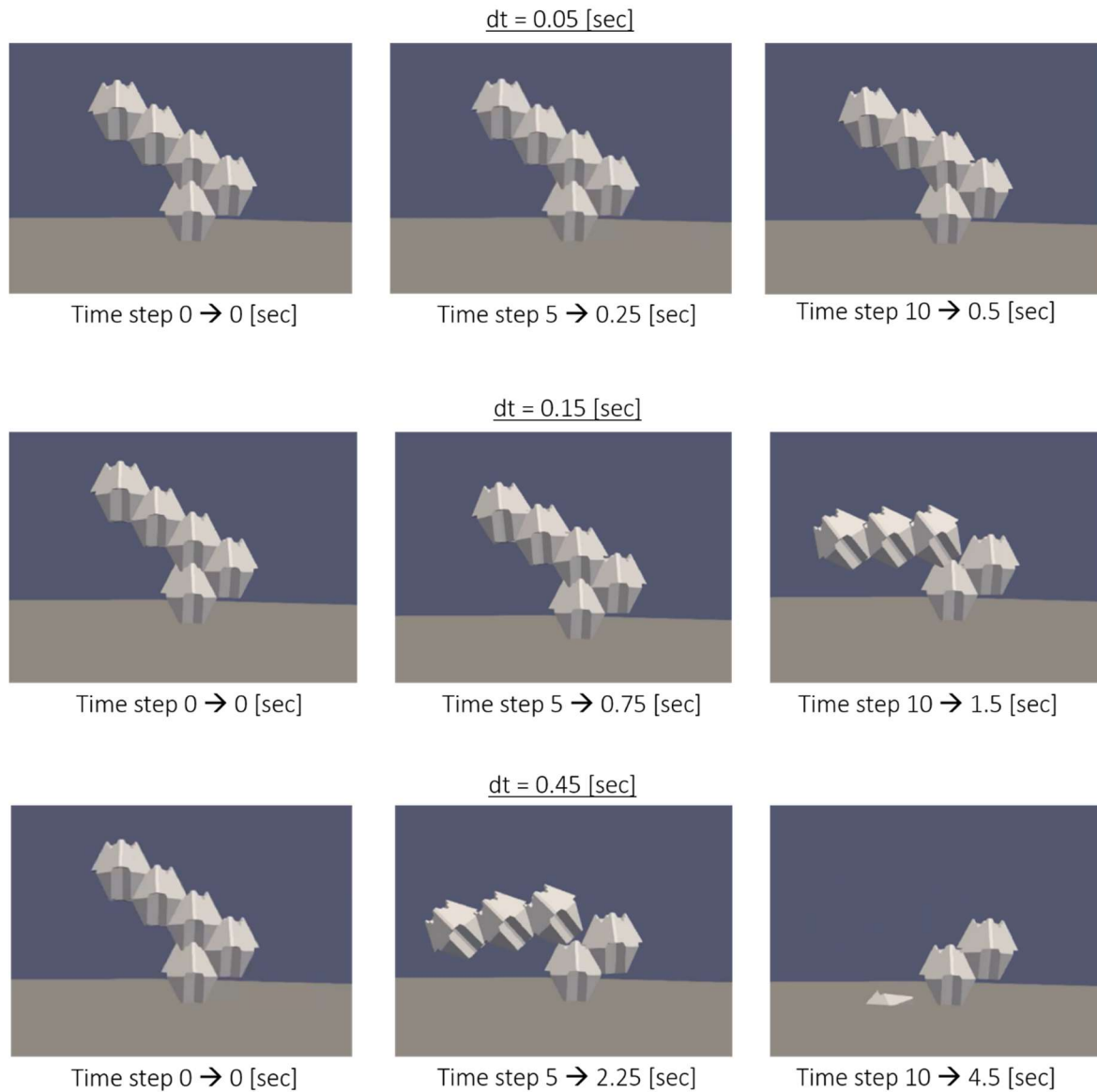


Figure 4.7 - Simulation of a structure with different time steps

Computation time of a simulation

In order to give an idea of the necessary computation times, some examples of (stable) structures and their computation time are given in Figure 4.8. The used computer for this is a *LENOVO™ ideapad 330S-15IKB* with a *i5-8250U* processor. 300 steps of 0.15 [sec] were performed meaning a total of 45 [sec] was simulated.

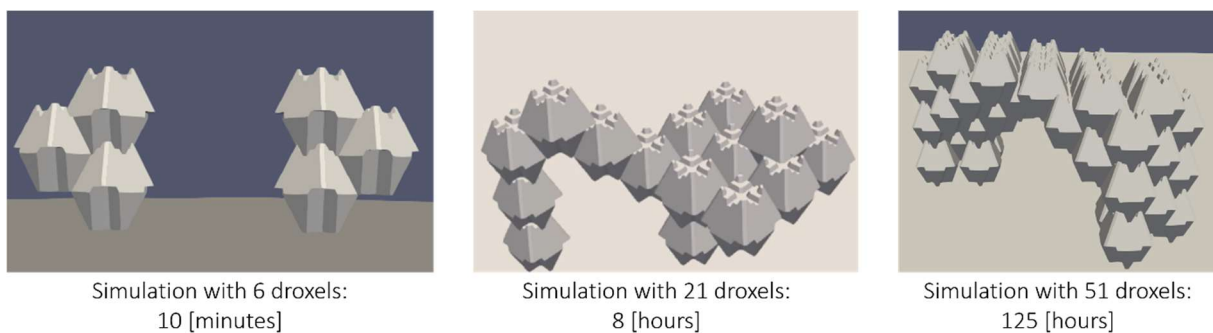


Figure 4.8 - Computation time for different (stable) droxel structures ($dt = 0.15$, 300 steps)

Droxels on the ground floor

The droxels on the ground floor are assumed to be completely embedded into the ground. This is done by imposing a zero displacement of the 6 degrees of freedom of every droxel on the ground floor. They can never move during the whole simulation.

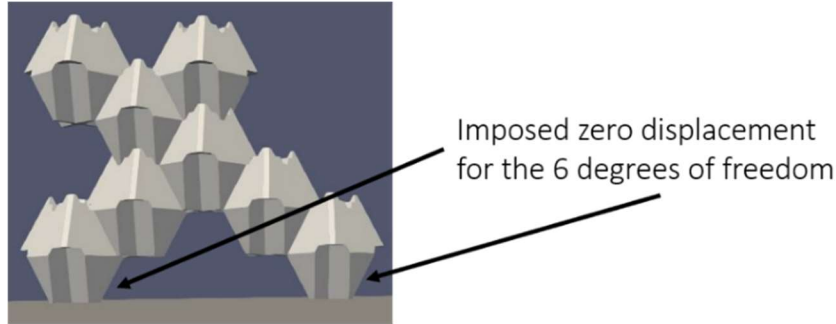


Figure 4.9 - Droxels on the ground floor are fixed during the whole simulation

Geometrical description of a droxel

For the implementation of the droxels in *LMGC90*, a 3D volumic meshing of the droxels is required. The program *GMSH* is used for that: “Gmsh is an open-source 3D finite element mesh generator with a built-in CAD engine and post-processor.” [23]

The realisation and importing of the mesh have also been explained by M. MULLIER. [40] Briefly explained, a *.stp* file containing the geometry of the droxel is imported. Based on that, a 3D mesh (*.msh* file) is created (see Figure 4.10).

The coarseness of the mesh is the one automatically generated by *GMSH*. For the optimisation of the total computation time, changing the coarseness is something that must be looked into.

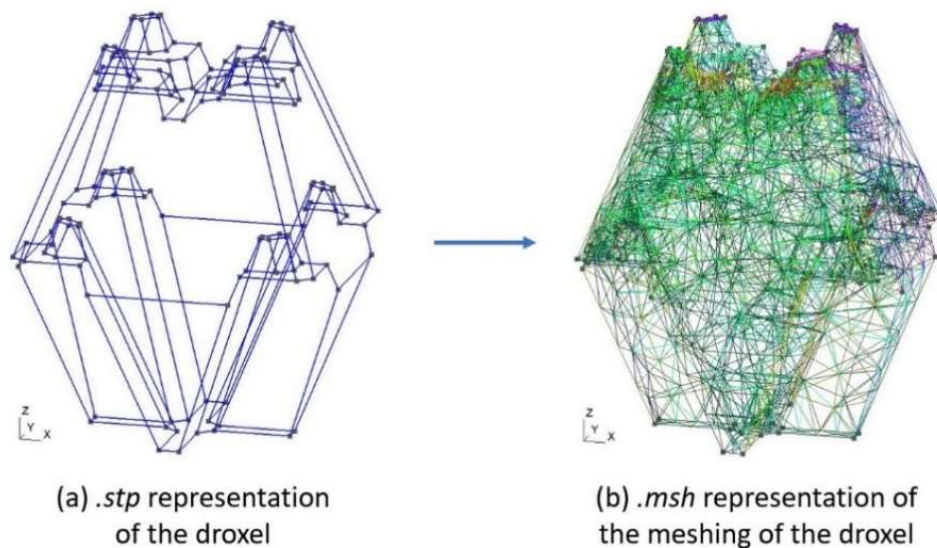


Figure 4.10 - *GMSH* program: from *.stp* volume to *.msh* meshing

The exact dimensions, weight and friction coefficient of the droxels used are given in 4.2.1.

Resultant of forces on the droxel surfaces

The resultant of forces on the droxel surfaces (calculated by *LMGC90*) are a combination of a normal force perpendicular to the surface and a friction force parallel to the surface. This is what is represented in Figure 4.11 for 10 of the droxel surfaces. The same principle apply to the all the other surfaces. The normal force (pointing out of the droxel volume) is drawn in red, the friction forces in green.

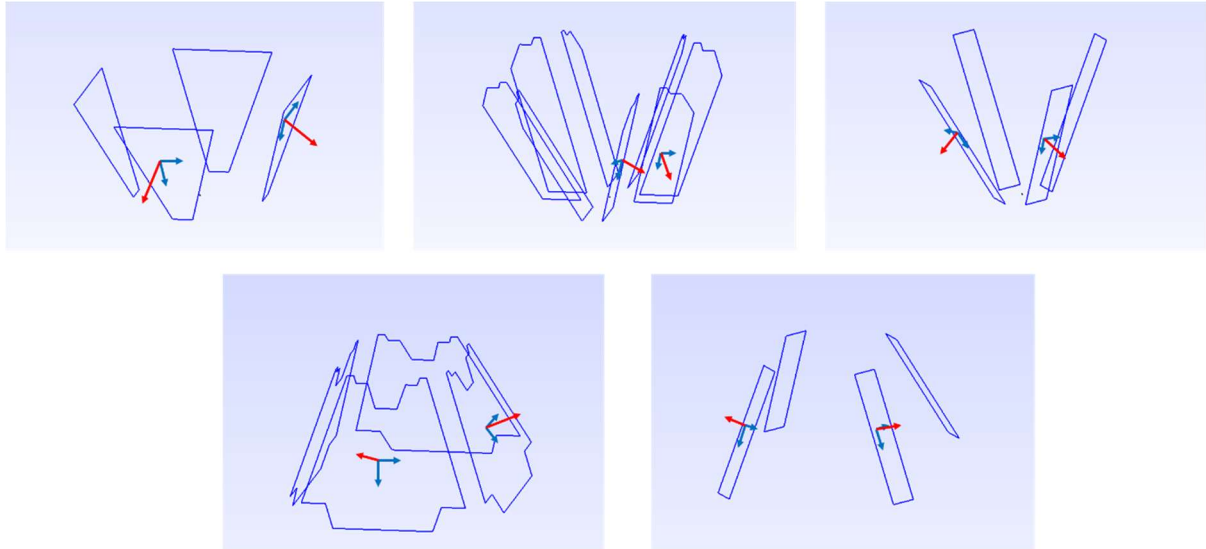


Figure 4.11 - Resultant of forces on droxel surfaces. Normal force in red, friction force in green

4.2 EXPERIMENTAL TESTS: 3D PRINTED DROXELS

After the numerical simulations, the second verification is the experimental tests. This is done using 3D-printed droxels.

4.2.1 3D printing of droxels

Classic droxels

In order to be able to verify the algorithm experimentally, several small droxels have been printed. The evolution of the tried droxel geometry is given in APPENDIX C. Different types of geometries were tested. The final droxel-geometry is shown in Figure 4.12 and has following parameters: $H = 50$ [mm]; $L = 30$ [mm]; $A = 20^\circ$; $E = 3$ [mm]; $Cr = 3$ [mm]; $R = 4$ [mm]; $B = 4.5$ [mm]; $F = 55$ [%]

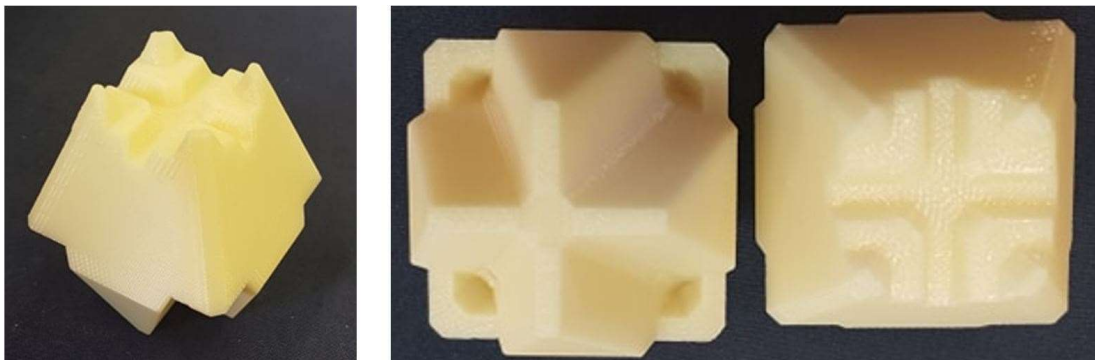


Figure 4.12 - Final droxel geometry

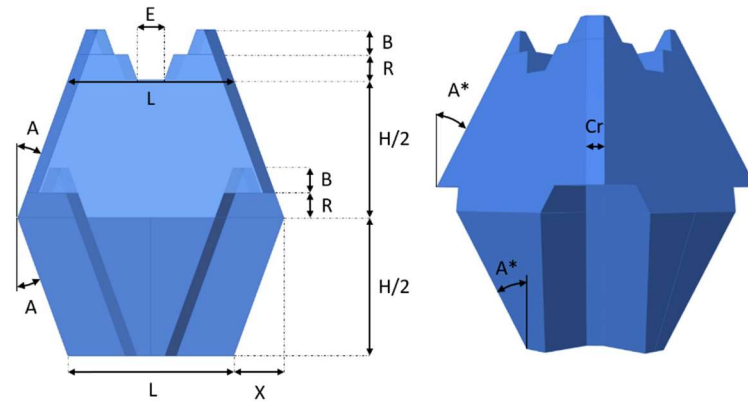


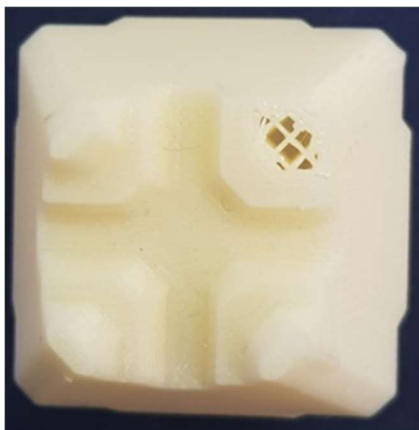
Figure 4.13 - Reminder: geometry of a droxel (F-factor not represented)

Specificities

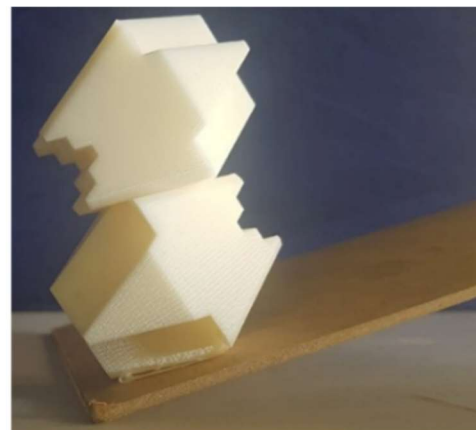
In order to run numerical simulations as accurately as possible, the friction coefficient and the density of the droxels must be given as input to *LMGC90*.

The droxels are not full but filled with a lattice-like structure (see Figure 4.14). The density of the droxel can be found by dividing the mass by the total volume of the droxel. Thanks to *Rhino*, the total volume of the droxels can be obtained. It is equal to $59,83 [cm^3]$. Using a scale, the total weight of a droxel is determined: $22,1 [gr]$. This gives a density of $0,3694 [gr/cm^3]$ or $[369,4 kg/m^3]$.

The friction coefficient is found by determining the angle at which a block starts to slide on top of another block of the same material. The test performed to determine the friction coefficient is shown in Figure 4.14 on the right. The upper droxel starts sliding at an angle of 60° , the angle of the droxel *A* being equal to 20° , this gives a friction coefficient of: $\mu = \tan \left((60 - 20) * \frac{\pi}{180} \right) = 0.6095$.



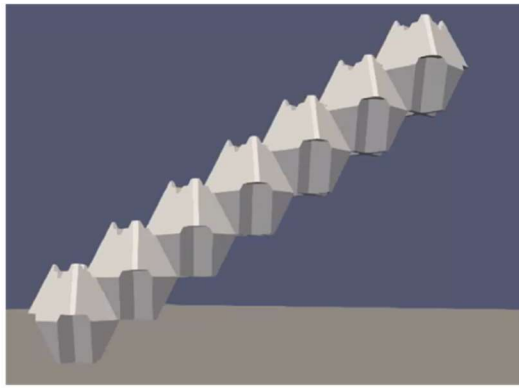
Lattice-like structure in a droxel



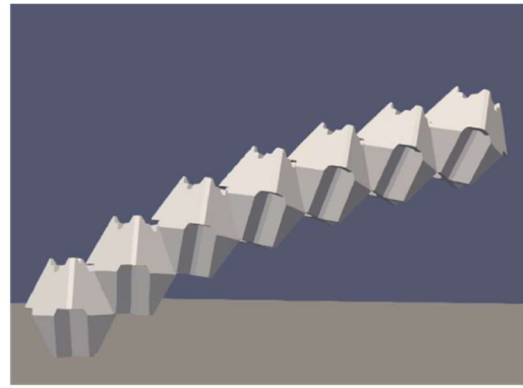
Determination of the friction coefficient of the droxels

Figure 4.14 - Lattice-like structure in a droxel (left), determination of μ (right)

In order to validate this friction coefficient, a second test is performed. An experiment is realised to determine how many droxels can be put in cantilever mode. This number is experimentally fixed at 5. A numerical simulation is then run in order to verify this. As shown by the simulation in Figure 4.15, this coefficient is over-estimated. Indeed, by stacking 6 droxels with a friction coefficient μ of 0.61 in cantilever-mode, the structure, even though extremely deformed, still holds in the simulations.



Simulation with 6 droxels at $t = 0$ [sec]



Simulation with 6 droxels at $t = 75$ [sec]

Figure 4.15 - Numerical simulation with $\mu = 0.61$: no collapsing. Friction coefficient overestimated

The simulation does not match the experimental tests where it is never possible to have 6 droxels in cantilever mode. The coefficient is then gradually reduced in order to find the minimal μ for which a structure with 5 droxels in cantilever mode holds. The found value for the friction coefficient μ is 0.51. This is the value used for all the simulations in this document.

Droxels of the base floor

The base droxels are always considered to be completely fixed in the plane on which they rest. In order to reproduce this as accurately as possible, some specific droxels are printed. Only the upper part of the droxel is reproduced. The lower part corresponds to a chamfered parallelepiped, which is put into a support base.

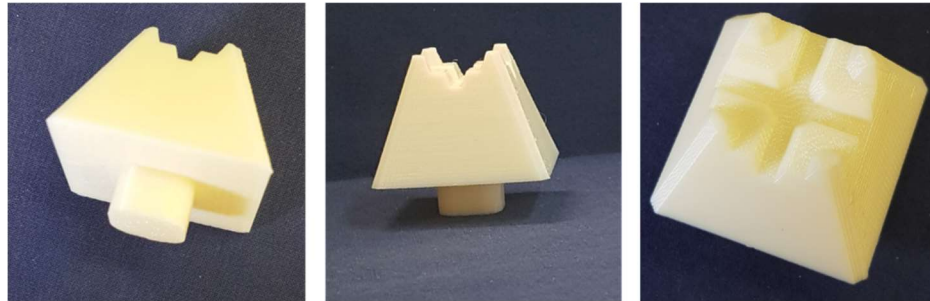


Figure 4.16 - Droxels for the base layers

4.2.2 Realisation of the base

The base plate was realised by gluing several 3mm MDF plates on top of each other. The plates were cut beforehand at the Fablab TRAKK in Namur. The final result is shown in Figure 4.17.



Figure 4.17 - Wooden base plate for the droxels

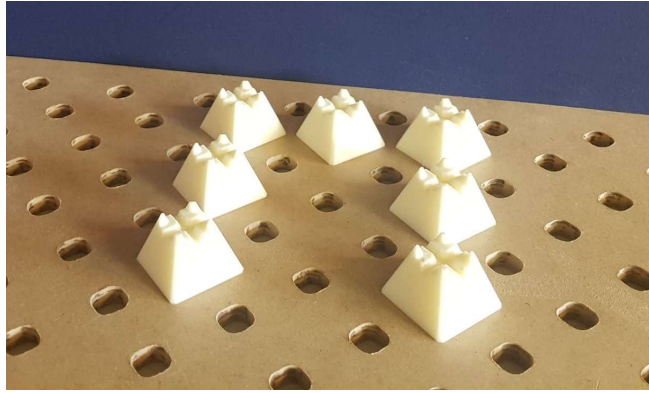


Figure 4.18 - Example of droxels embedded in the wooden base

CONCLUSION

The method behind the numerical simulations, Discrete Elements Method (DEM), has briefly been explained. The realisation of the small-scale droxels as well as their base have also been mentioned. The calibration of the friction coefficient, necessary for the numerical simulations, has been done using the printed droxels.

The comparisons between the coded Grasshopper algorithm, the numerical simulations and the experimental tests are done in CHAPTER 8

CHAPTER 5 DROXELISATION OF A SURFACE

After having set the 8 parameters determining the droxels geometry, the next step is the droxelisation of a structure. This is the process allowing for the representation of a certain 3D surface with several stacked droxels. Two possible droxelisation methods as well as their implementation in *Grasshopper* are presented.

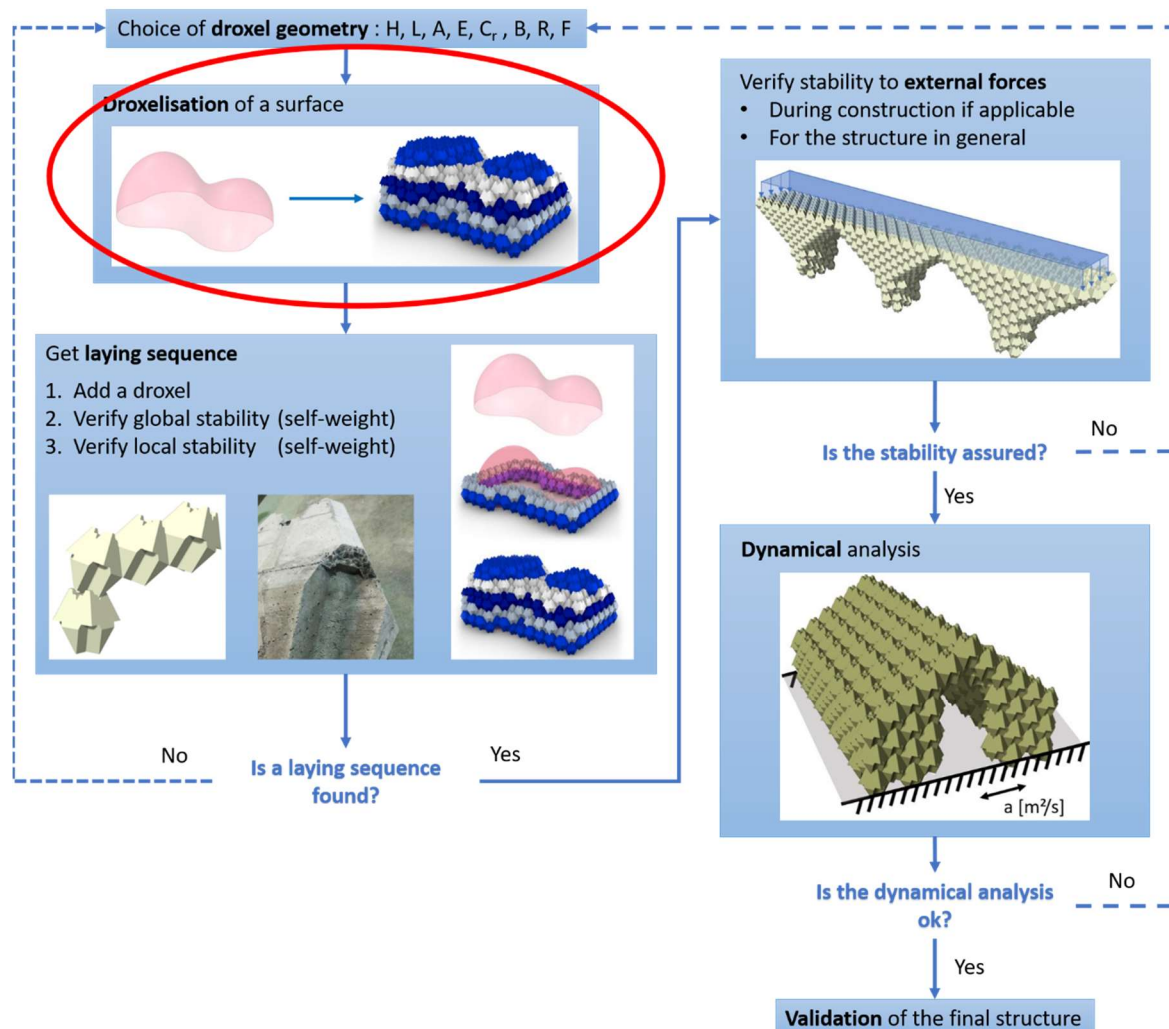


Figure 5.1 - Chapter 5: droxelisation of a surface

5.1 INTRODUCTION

Reminder: The **droxelisation** of a surface is the process of determining where to put droxels in order to reproduce the 3D surface (or a desired structure geometry) as accurately as possible. Some examples of droxelised structures are given in Figure 5.2.

The droxelisation of a surface is an important step since the aesthetic aspect as well as the construction facility of the structure is determined at this point. If the objective is to have a structure that can quickly be assembled and dismantled, the droxelisation with as few droxels as possible should be preferred. If on the other hand the aim is to realise an artistically relevant structure in the landscape, a structure with smaller droxels might be more pertinent.

The droxelisation is probably the step needing the most common sense from the user. It is also a part where there is a lot of human interaction with the program. In order to use droxel construction wisely,

it is important to find the right balance between several factors. Those factors are explained in this chapter.

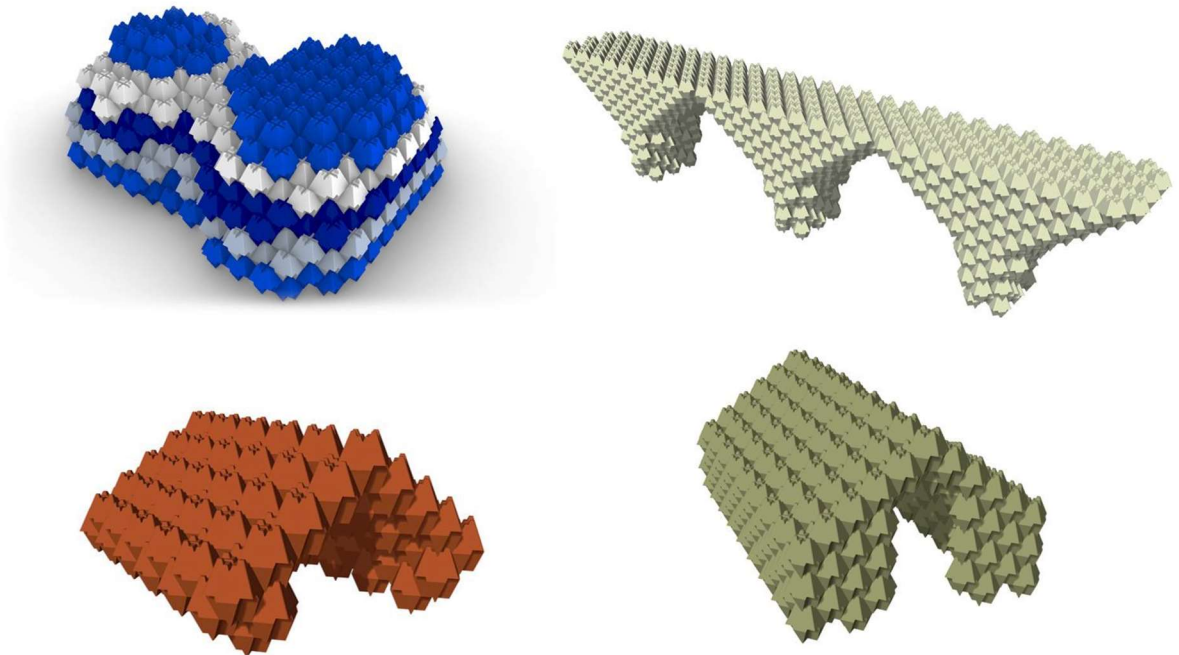


Figure 5.2 - Droxelisation examples: domes, bridges, tunnels, shelters

This chapter is divided into three parts:

1. First the notion of droxelised structure is further explained;
2. Then there is a discussion on which factors should be taken into account when droxelising a surface;
3. Lastly the two droxelisation methods used in the *Grasshopper* program are explained.

5.2 WHAT IS CONSIDERED A DROXELISED STRUCTURE ?

5.2.1 Droxelised structures

A droxelised structure is a structure composed out of droxels in which each droxel that is not resting on the ground has at least one lower neighbour. It is not possible to have a “floating” droxel in the structure.

As example, Figure 5.3 is not a droxelised structure, because the red droxel is floating in the air.

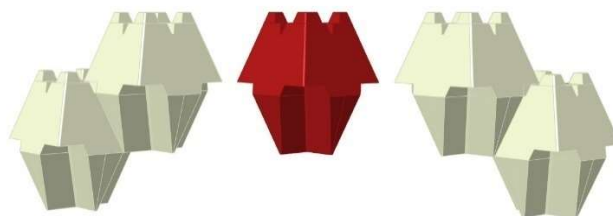


Figure 5.3 - Example of a floating droxel in red

When one wants to droxelise a 3D surface given as input, the droxel geometry may not allow to reproduce the surface accurately (see Figure 5.4). As a solution, one might consider a structure with a bigger width. This sometimes solves the problem (Figure 5.4 (a)). However, this method is not very

efficient since it uses many more droxels everywhere else in the structure too. If the surface geometry can be adapted, the logical solution is to change the desired angle (see Figure 5.4 (b)).

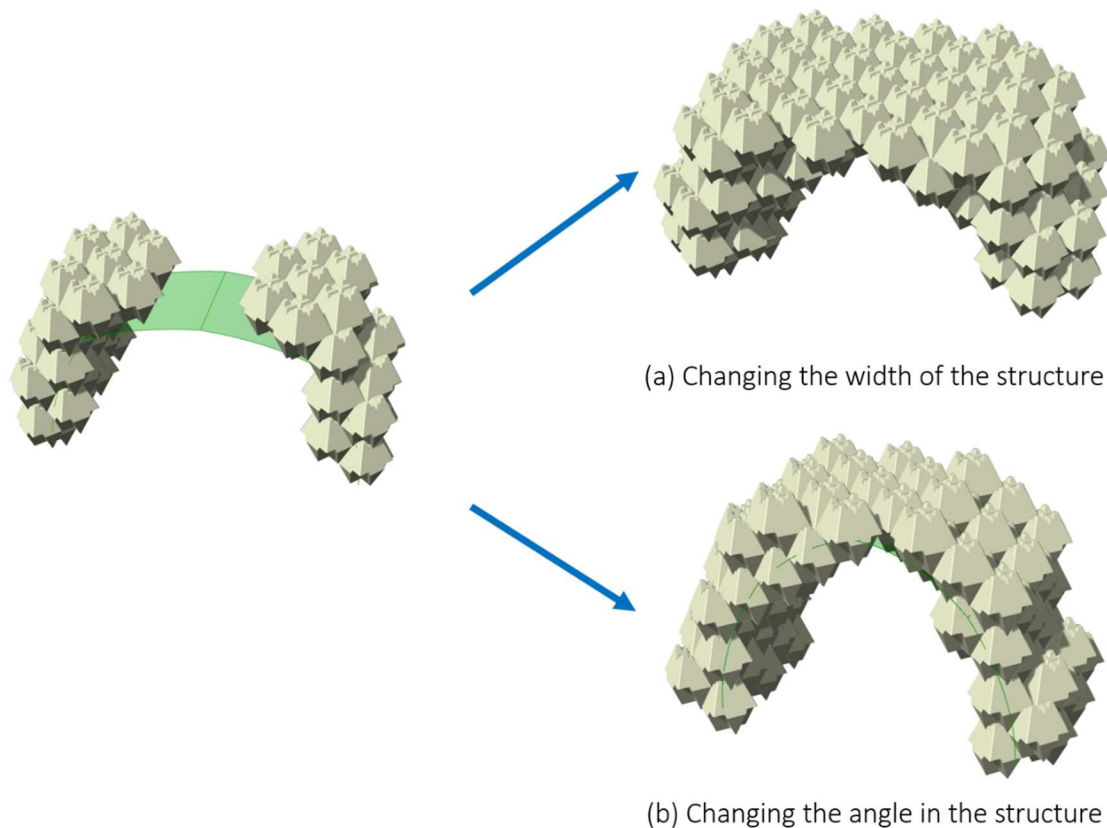


Figure 5.4 - Possible solutions when the angle cannot be realised with the chosen droxel

If the geometry cannot be adapted, droxels with slope change can be considered. These droxels have been briefly explained in section 2.3.3. Since the slope changing droxels are not part of this thesis, this solution is not further detailed.

5.2.2 Notion of stability

It is important to understand that the notion of droxelised structure is independent from the notion of stability of a structure. Figure 5.5 represents a droxelised structure even though it obviously is an unstable one.

Only once the surface is droxelised and the user is satisfied, the stability of the structure can be checked. If the droxelisation does not suit the user, several factors can be adapted: those factors are explained next.

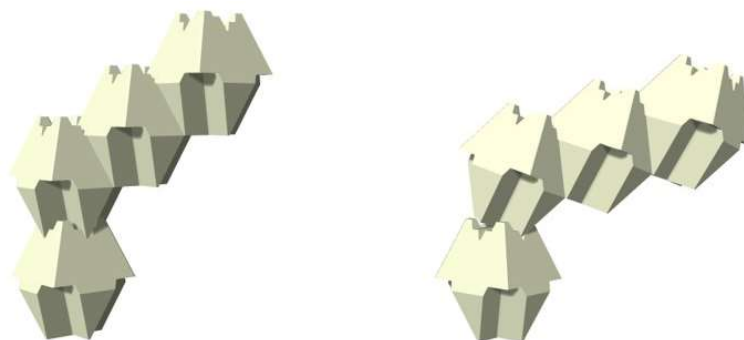


Figure 5.5 - Representation of an unstable droxelised structure and its instability mode

5.3 INFLUENCE OF SEVERAL FACTORS ON THE DROXELISATION

The influence of the droxel parameters on the final form of the structure has already been discussed in the previous chapter. This section aims to tackle more general factors influencing the droxelisation.

5.3.1 Influence of the droxel scale

A concept that has not yet been introduced is the concept of droxel-scaling. Scaling a droxel means that the factors H , L , E , Cr , R and B are multiplied by a given factor. F and A stay the same. This results in a homothetic transformation of a droxel.

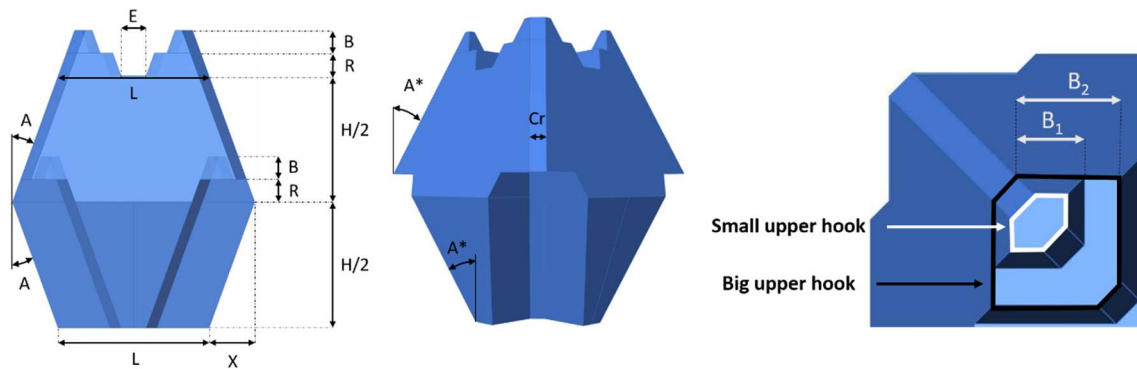


Figure 5.6 - Reminder: Geometry of a droxel, $F = B_1/B_2$

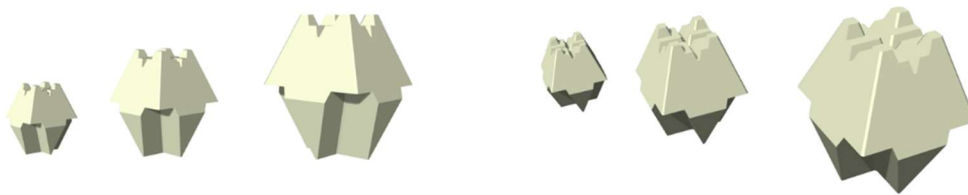


Figure 5.7 - Scaling of a droxel: only the angle A and the parameter F remain the same

The influence of the scaling of the droxels for a given surface is represented in Figure 5.8. Changing the scale of the droxels allows to adjust the “resolution” of a structure. The smaller the droxels, the more accurately the structure matches the original 3D surface. This naturally comes with a higher cost in terms of number of droxels.

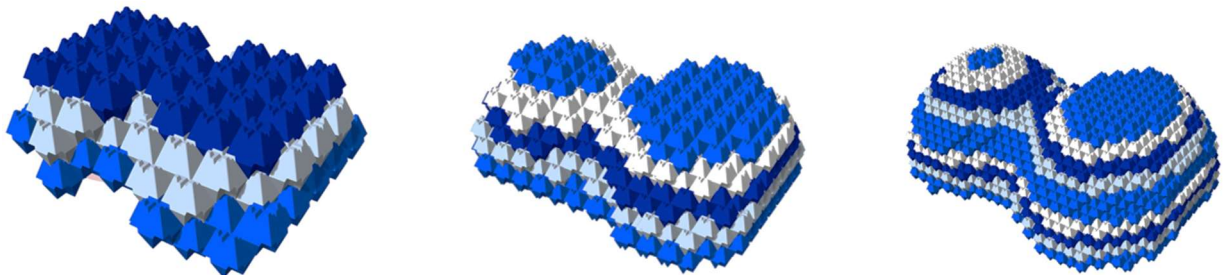


Figure 5.8 - Influence of the droxel size on the resolution of the structure

The example of the droxelisation of the same 3D surface on Figure 5.8 validates this: the first structure has 156 droxels, the second 635 and the third 3 821. An equilibrium must be found between matching the 3D surface and having a reasonable amount of droxels to place.

The droxels of left and right structure in Figure 5.8 have a scale factor of 2 and 0.5 respectively compared to the droxels of the central structure. The volume of the big droxels is therefore about 10 times the volume of the droxels in the central structure. The volume of the smaller droxels is about 10 times smaller. Indeed, as a reminder [35]:

$$V_{droxel} = L^3 \cdot \frac{H}{L} \cdot \left[\frac{\tan(A)^2}{3} \left(\frac{H}{L}\right)^2 + \tan(A) \cdot \left(\frac{H}{L}\right) + \frac{1 + \frac{2E}{L} - \left(\frac{E}{L}\right)^2}{2} \right]$$

If the total volume of the central structure is set to 1, the total volume of the left and right structure is equal to 2,585 and 0,6336. The smaller the droxels, the less volume is used. The counterpart for this is that the number of droxels to place quickly rises.

5.3.2 Influence of the wall depth of the structure

A bigger wall depth comes with a bigger resistance but the inside surface quickly becomes smaller and the number of droxels drastically rises as shown in Figure 5.9: the first structure with a smaller wall depth has a total of 635 droxels, the structure with bigger walls has 1 795 droxels. The challenge is to find a stable structure with the smallest possible wall depth.

Sine the droxels of both structures have the same geometries, the number of droxels, and therefore the total volume almost triples when the wall depth doubles.

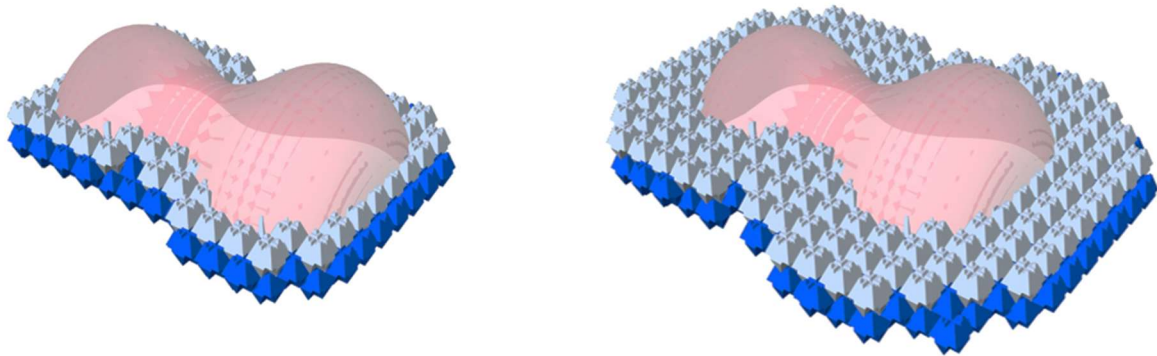


Figure 5.9 - Influence of the width on the droxelised structures

5.3.3 Influence of the grid orientation

As explained in the previous chapter, the angles that can be seen changes in function of where one looks at the structure (see 2.3.3). This is due to a fictive grid rotation of the floor surface: two grid orientations (with a rotation of 0 and 45°) are possible. Figure 5.10 shows the visual impact of a grid rotation for the droxelisation of the same 3D surface.

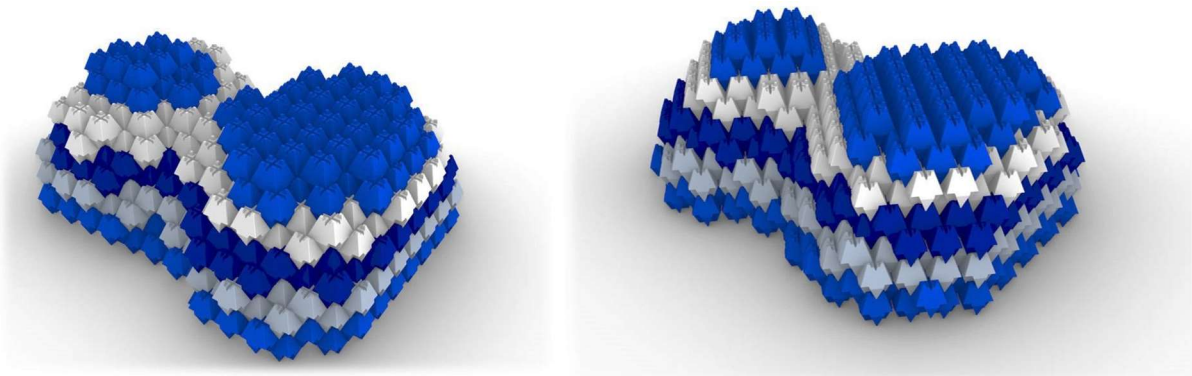


Figure 5.10 - Influence of the rotation of the grid

The total volume of the structures is proportional to the number of droxels (the droxel geometry is identical for both structures).

The number of droxels of the left structure is 974, the right structure has 971 droxels. It can therefore be concluded that the grid orientation does not have a significant impact on the total used volume. The choice of the grid orientation is an aesthetical one.

5.4 TWO DROXELISATION METHODS

CONCLUSION

The step of droxelisation is not one to be taken lightly: once the general aspect of the droxels has been determined, their size must be chosen in order to reproduce the desired surface whilst having a reasonable amount of droxels to place. The width of the structure should be as small as possible but a stability study is necessary. If not satisfying, bigger walls might be needed. Lastly, the grid orientation has a significant impact of how the structure looks.

All those aspects should not be overlooked when droxelising a structure and show why the droxelisation of a surface cannot be neglected. It is a critical part of the realisation process of a droxelled structure and must be executed with care for detail.

First method: 3D-surface

There are two possible droxelisation methods implemented in the *Grasshopper* program. The first method allows to droxelise a certain 3D surface given as input. A parameter d_{max} is also given as input. This represents the maximum distance from the surface at which a droxel can be placed. The total width of the structure is equal to $2 \times d_{max}$.

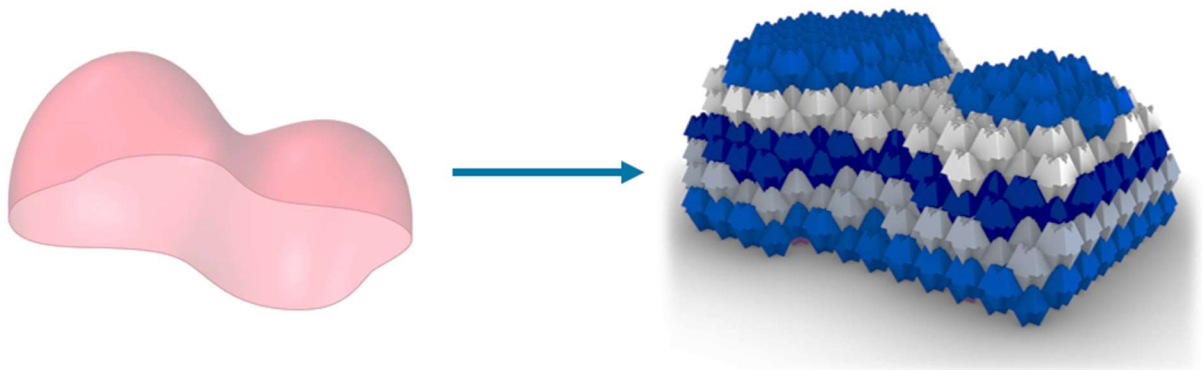


Figure 5.11 - First droxelisation possibility

Second method: normalised coordinates

In order to easily verify smaller structures, it is also convenient to be able to manually input the base point-coordinates (see Figure 5.12) of the droxels. However exact coordinates are often difficult to obtain. In order to avoid rounding errors (and frustrated users), the notion of **normalised coordinates** is introduced. Normalised coordinates are (x, y, z) -coordinates with integers as (x, y, z) -values. The program later calculates the real base point-coordinates. This process is represented in Figure 5.12.

Using normalised coordinates can also have a major impact when studying the stability of a structure. Thinking up ahead, it may later be useful to locally add some droxels to reinforce a structure. In that case it would be tedious to recompute a surface that would only locally have a different form.

With the use of normalised coordinates, it is possible to:

- (1) droxelise a surface → (2) get the normalised coordinates of the structure → (3) manually add droxels to locally reinforce the structure.

The second droxelisation method is thus one that proves to be very useful.

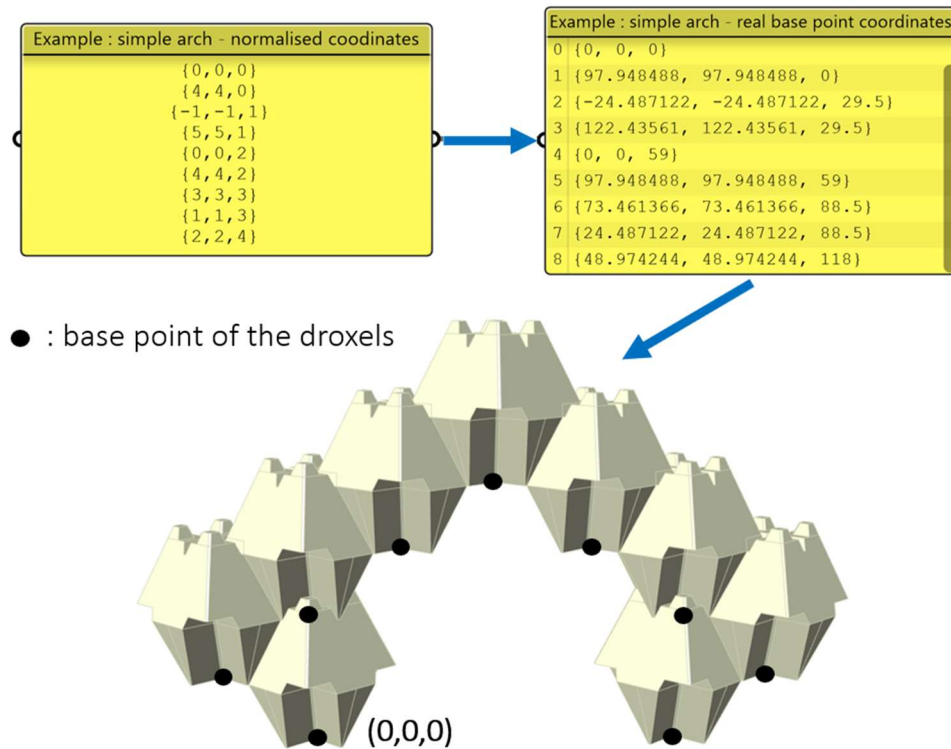


Figure 5.12 - Representation of the second droxelisation method

5.5 IMPLEMENTATION OF THE TWO METHODS

5.5.1 Case 1: input of a 3D surface

The droxelisation for a given 3D surface as input is done according to following steps:

- 1) Horizontal centering of the surface around the origin;
- 2) Computation of all the possible base points around the surface;
- 3) Removing the base points too far away from the surface;
- 4) Removing the floating points;
- 5) Computation of the normalised base point coordinates.

1) Horizontal centering of the surface around the origin

This step, even though not strictly necessary, is done to facilitate the visualisation of the results. Objects visualised in *Rhino* take less time to be computed when closer to the origin, it is therefore convenient to center the surface around the origin. This step is important to reduce the computation time for the visualisation of the results.



Figure 5.13 - Droxelisation example: 3D surface centered around the origin

2) Computation of all the possible base points around the surface

The next step corresponds to the creation of a parallelepiped box around the centered surface. The box dimensions have been augmented by d_{max} in order to be able to have the desired wall width. Inside this box, all the possible base points where a droxel can be placed are computed.

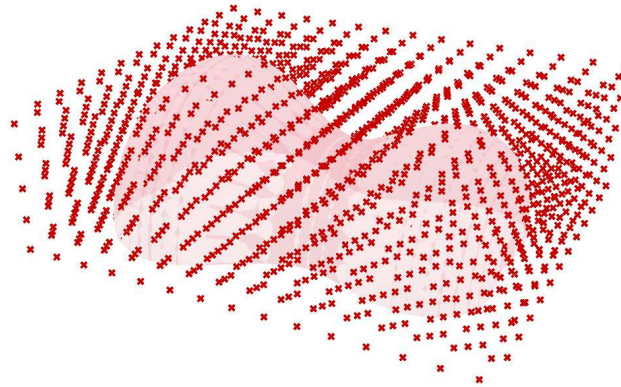


Figure 5.14 - Droxelisation example: computation of all the possible base points around the surface

3) Removing the base points too far away from the structure

Next, only the points at a distance smaller than d_{max} are kept. These points are sorted by ascending z-value for later purposes.

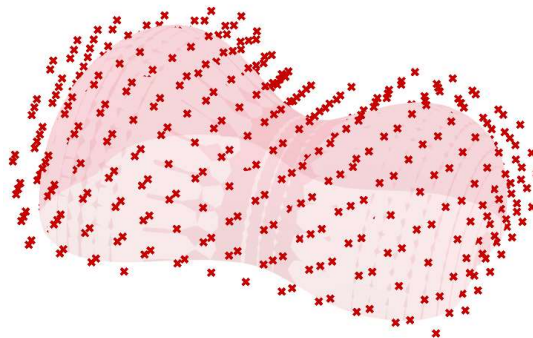


Figure 5.15 - Droxelisation example: keeping points at distance d_{max} from the surface

4) Removing all the floating points

Lastly, all the floating points (i.e.: points without any lower droxel, see 5.2.1) are identified and removed in order to have the final base points coordinates of the droxels.

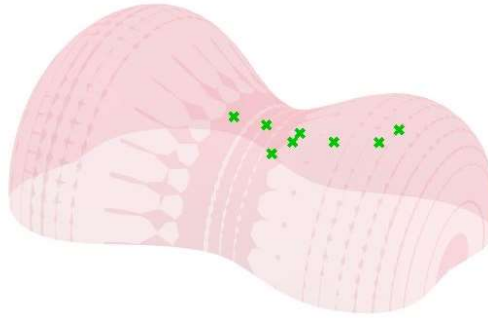


Figure 5.16 - Droxelisation example: identification of floating droxels

The identified floating points identified are represented in Figure 5.16, they are point which had zero lower neighbours.

5) Computation of the normalised base point coordinates

As explained before, it could be useful to locally reinforce a structure, this is why the normalised coordinates of the structure are also calculated. If necessary, they can then be copy-pasted to the manual input of coordinated. Droxels can then locally be added or removed by manual input.

Normalised coordinates	
0	{30, 7, 0}
1	{30, -7, 0}
2	{30, -9, 0}
3	{30, 9, 0}
4	{32, 7, 0}
5	{32, -7, 0}
6	{32, -9, 0}
7	{28, 9, 0}
8	{26, 5, 0}
9	{26, -5, 0}
10	{26, -7, 0}

Figure 5.17 - Droxelisation example: normalised coordinates of the structure

6) Final base points coordinates

Lastly, the final base point coordinates are computed and stored.

Real base point coordinates	
0	{7185.88962, 1721.94791, 0}
1	{7185.88962, -1721.94791, 0}
2	{7185.88962, -2213.933027, 0}
3	{7185.88962, 2213.933027, 0}
4	{7677.874737, 1721.94791, 0}
5	{7677.874737, -1721.94791, 0}
6	{7677.874737, -2213.933027, 0}
7	{6693.904502, 2213.933027, 0}
8	{6201.919385, 1229.962793, 0}

Figure 5.18 - Droxelisation example: computation of the real base point coordinates

5.5.2 Case 2: input of normalised coordinates

As reminder, the second case is when one inputs the normalised coordinated of the droxels manually. This can be done either from scratch or by copying the points calculated in the first case and adding or removing some points manually.

The followed steps are :

- 1) Horizontal centering of the center of gravity of the surface around the origin;
- 2) Multiplication by droxel dimensions;
- 3) Removing the base points too far away from the surface;
- 4) Removing the floating points.

- 1) Horizontal centering of the center of gravity of the surface around the origin

This is done for the exact same reason as before: to accelerate the visualisation of the results.

- 2) Multiplication by droxel dimensions

This part allows to have the real base point coordinates and to visualise and determine the stability and laying sequence of structures.

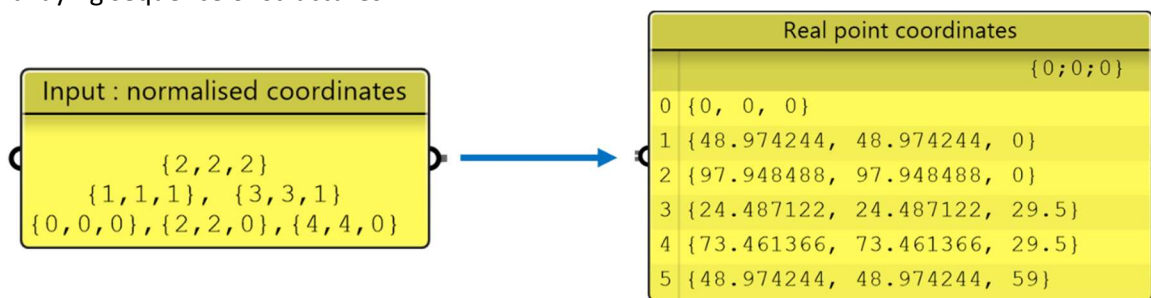


Figure 5.19 - Droxelisation example 2: from normalised to real point coordinates

- 3) Removing the floating points

Same as for the first method, see 4) *Removing all the floating points*, in section 5.5.1. Even though theoretically not needed, the removal of floating points is also done here as an extra security should there be an input mistake.

5.5.3 Flowchart representation of both methods

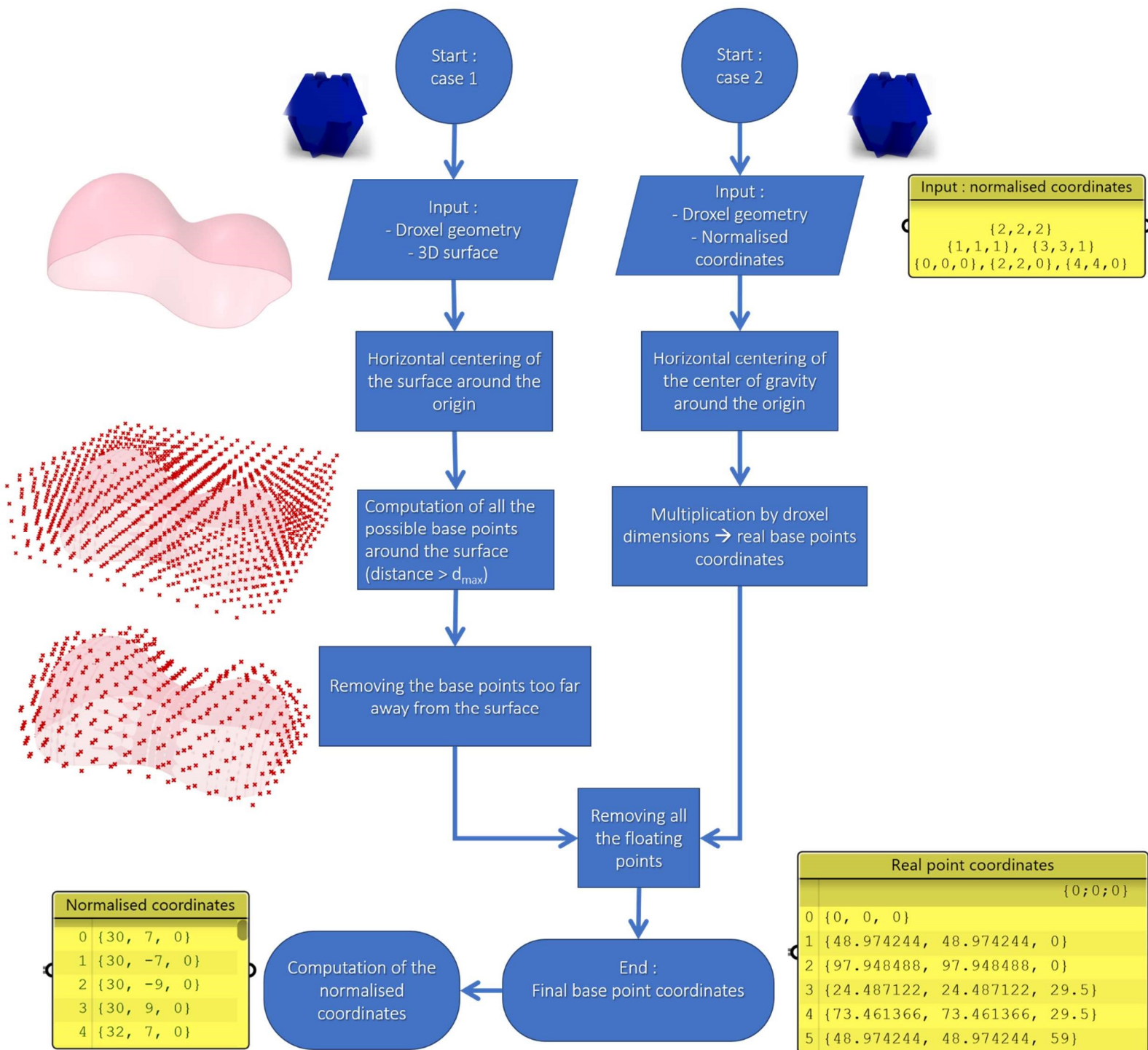


Figure 5.20 - Flowchart of the two possible droxelisation methods

5.6 GRASSHOPPER PROGRAM

The steps represented on the flowchart in Figure 5.20 can easily be found on the *Grasshopper* interface. The *Grasshopper* and *C#* codes can be obtained by contacting Professor PIERRE LATTEUR (pierre.latteur@uclouvain.be).

(2) Droxelisation

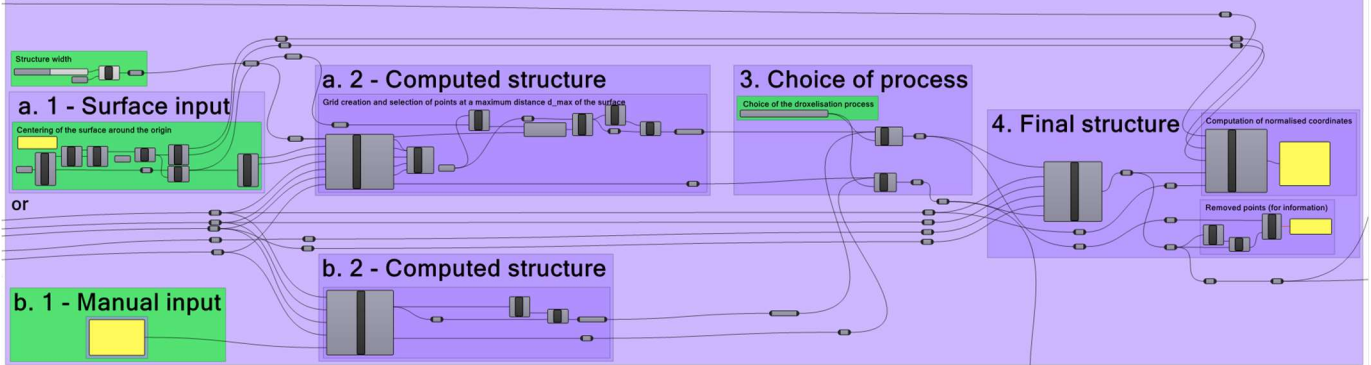


Figure 5.21 - Part of the Grasshopper program corresponding to the droxelisation

CONCLUSION

Thanks to the Grasshopper program, we can now droxelise any structure using one of the three following methods :

- 1) Inputting a certain 3D surface in the program;
- 2) Manually computing the normalised coordinates;
- 3) By combining the two previous methods

Accounting for parameters such as droxel shape, droxel scale, structure depth and grid orientation allows to parametrize the final rendering of the droxelised surface.

CHAPTER 6 GLOBAL STABILITY OF A DROXELISED STRUCTURE

After having droxelised a structure, it is necessary to know whether this structure is stable or not. This is the goal of this sixth chapter: its first part aims to explain the concepts of global and local stability. Next, the implementation of an algorithm capable of determining whether a given structure is globally stable or not is tackled.

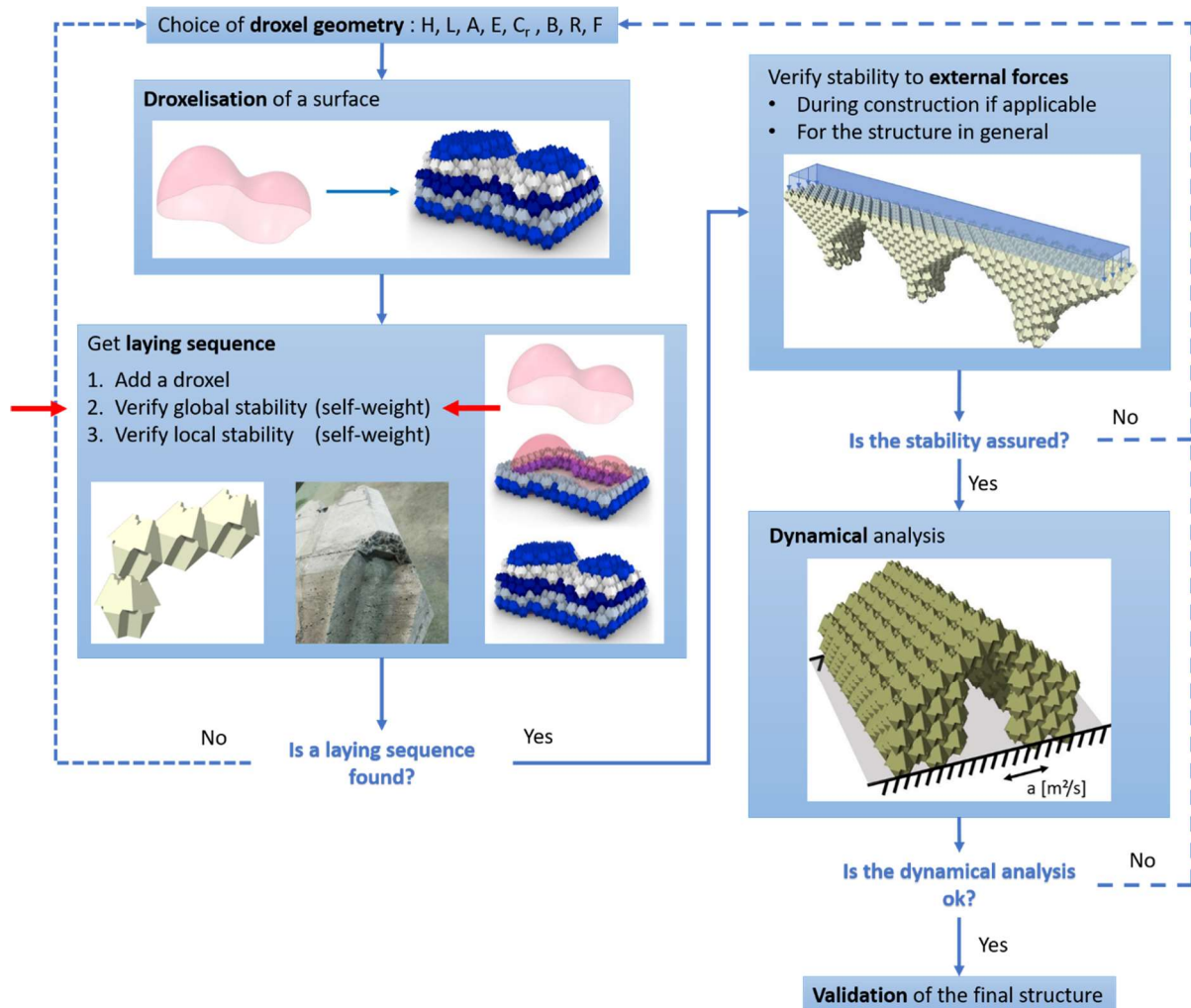


Figure 6.1 - Chapter 6: Global stability of a structure

6.1 INTRODUCTION

Once a structure is droxelised, it is necessary to verify that the structure and its temporary construction stages are stable. In the first instance, only the self-weight of the structures is considered. Later, it will be necessary to be able to consider all the possible loads to which the structure is submitted, both during the construction and in the final stage.

The term *global* implies that the droxels are considered as rigid bodies (deformations are neglected) with infinite compressive strength. Only the equilibrium of the structure is checked. The local ruptures are not verified (it is assumed that the stresses $\sigma \leq \sigma_{max}$ everywhere in the droxel). However, as experienced in the master thesis of M. MULLIER [40] and represented in Figure 6.2, this type of rupture can occur. It must therefore be checked in the local stability part (which is not tackled in this thesis).



Figure 6.2 - Local rupture on a (concrete) droxel [40]

The global instability of a droxelised structure can occur in two different modes. The first mode is when a part of the structure named a group (see later at section 6.2.3) tilts (Figure 6.3 (a.2)). The second instability mode occurs when the friction between two interlocking droxels is not sufficient to prevent the upper droxel to “unhook” itself (Figure 6.3 (b.2)). At this point, and for the rest of the thesis, the second instability mode is not considered. It can be shown that this hypothesis is valid for $A^* \leq \phi$ with ϕ the friction coefficient between the droxels [35].

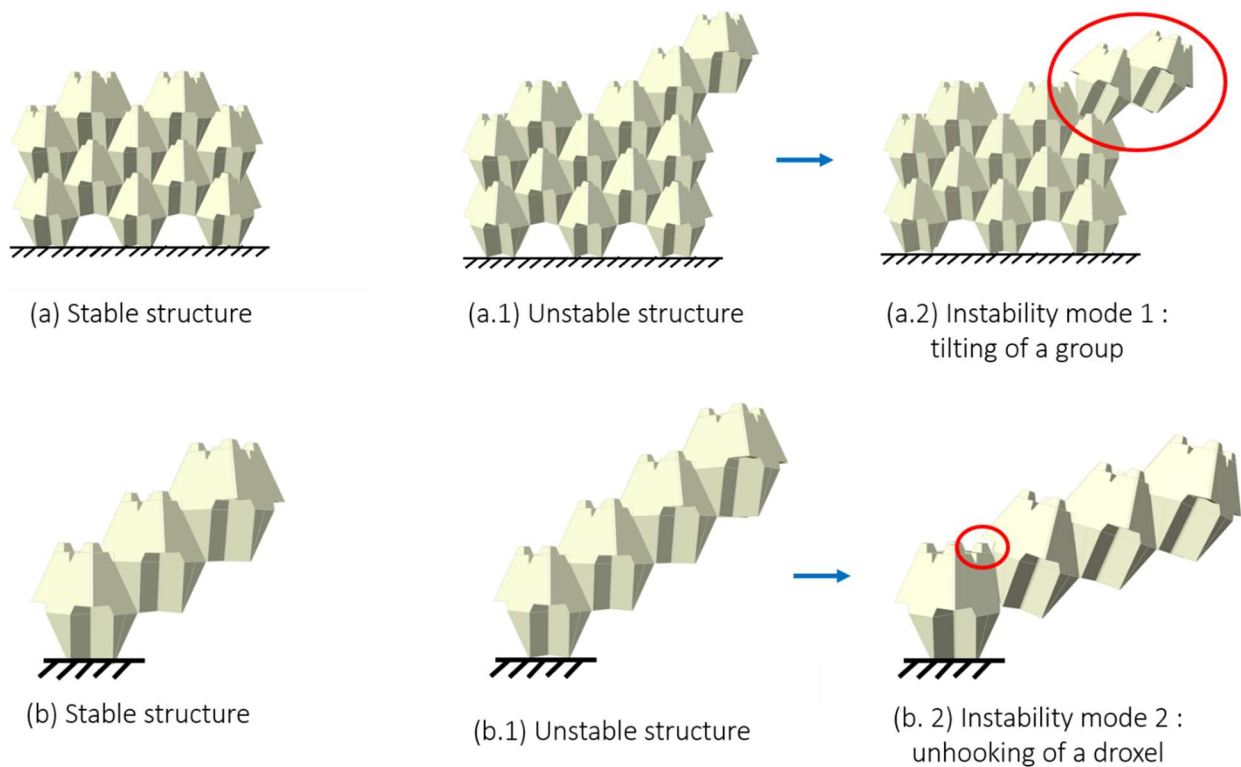


Figure 6.3 - Example of a globally stable and unstable structure, representation of the instability mode

6.2 GLOBAL INSTABILITIES DUE TO TILTING

Due to the (important) hypothesis that the second instability mode never occurs, following principle is used to determine whether a structure is globally stable or not: “An object stays stable if the vertical projection of the center of gravity remains inside the support area of the object” (Figure 6.4).

In order to explain when a tilting occurs for droxelised structures, the notion of stacking mode and hooked droxel must first be explained.

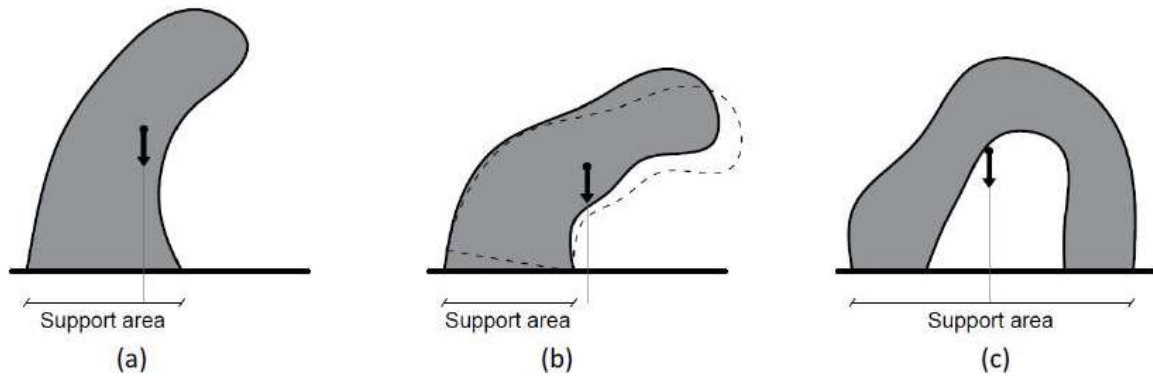


Figure 6.4 - Stability of a structure: principle [24]

6.2.1 Stacking modes of droxels

There are 11 different ways for a droxel to be supported by one or more lower droxel(s). They are named as represented in Figure 6.5 (stacking modes corresponding to the beige droxel):

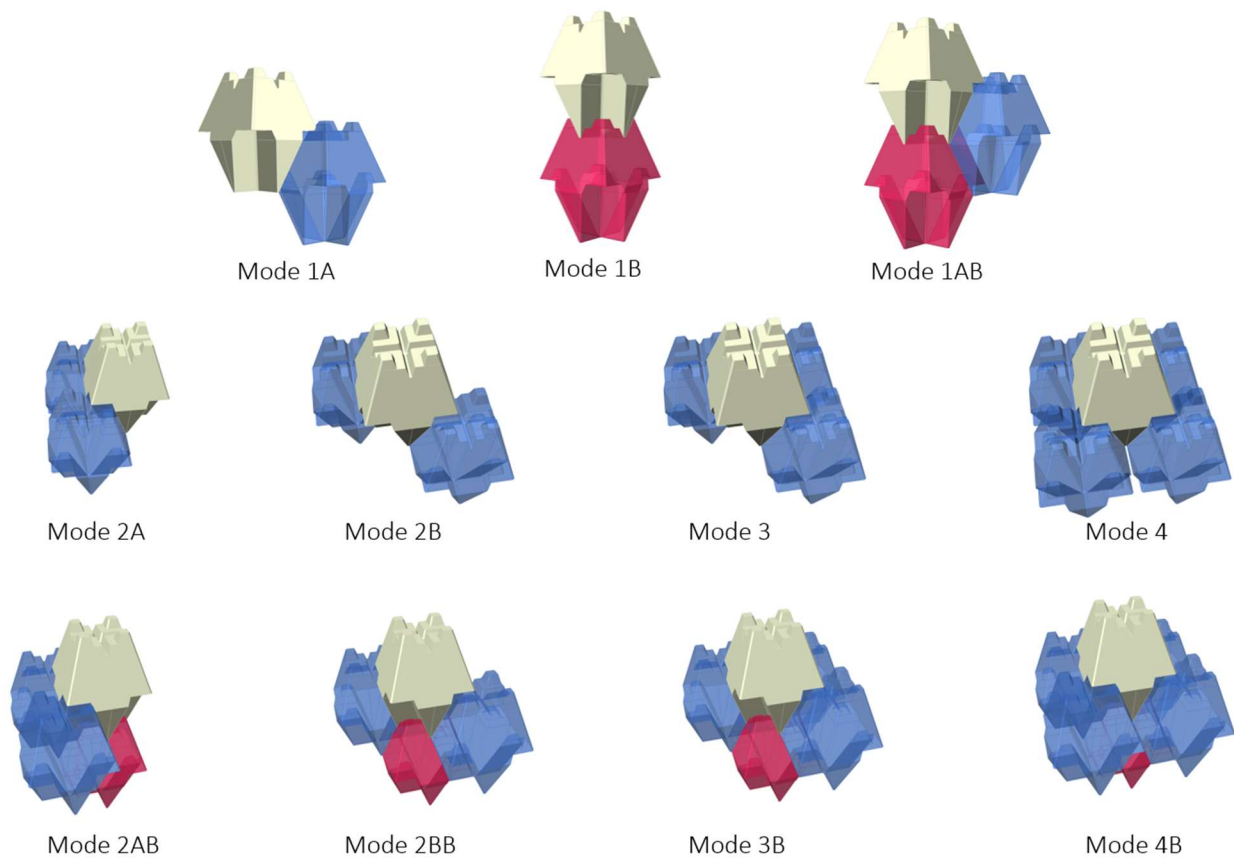


Figure 6.5 - Different possible stacking modes of droxels

6.2.2 (Un)hooked droxels

When a droxel is hooked, it means that it remains stable no matter where a next droxel is placed (i.e., no possible tilting of the hooked droxel at its base). A droxel in mode 1A, 2A, 2B, 2BB, 3, 3B, 4 or 4B is hooked.

The droxels on the ground floor are also always considered to be hooked.

Modes 1B, 1AB and 2AB correspond to unhooked droxels, a possible tilting of the droxels in those modes is given in Figure 6.6.

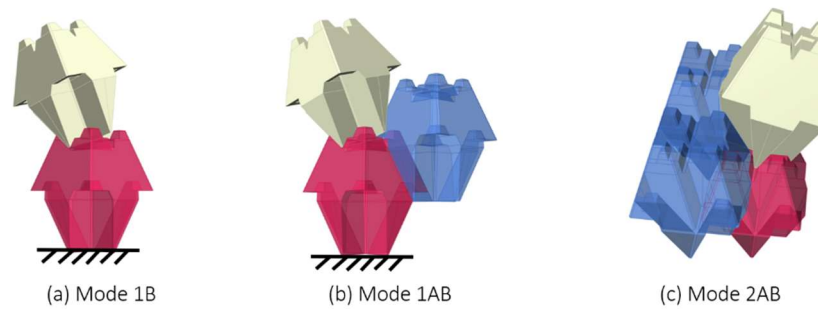


Figure 6.6 - Possible unhooking of the droxels in modes 1B, 1AB and 2AB

Unhooked droxels: creation of local hinges

The notion of unhooked droxel allows to determine which droxels are prone to a instability due to tilting. If a droxel is hooked, there is no risk of tilting and no global check must be performed. When a droxel is unhooked, a local hinge might appear at the base point of the droxel and the global check must be performed.

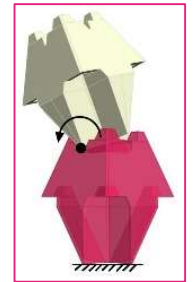


Figure 6.7 - Creation of a local hinge at the base of the unhooked (beige) droxel

6.2.3 Identification of an unstable structure

Notion of group of droxels

A group corresponding to a certain droxel is a collection of several droxels composed of:

- 1) The droxel itself;
- 2) Its lower droxels;
- 3) Its upper droxels;
- 4) The upper droxels of all the droxels present in 1), 2), 3) and 4)

An example of the determination a group is given below:

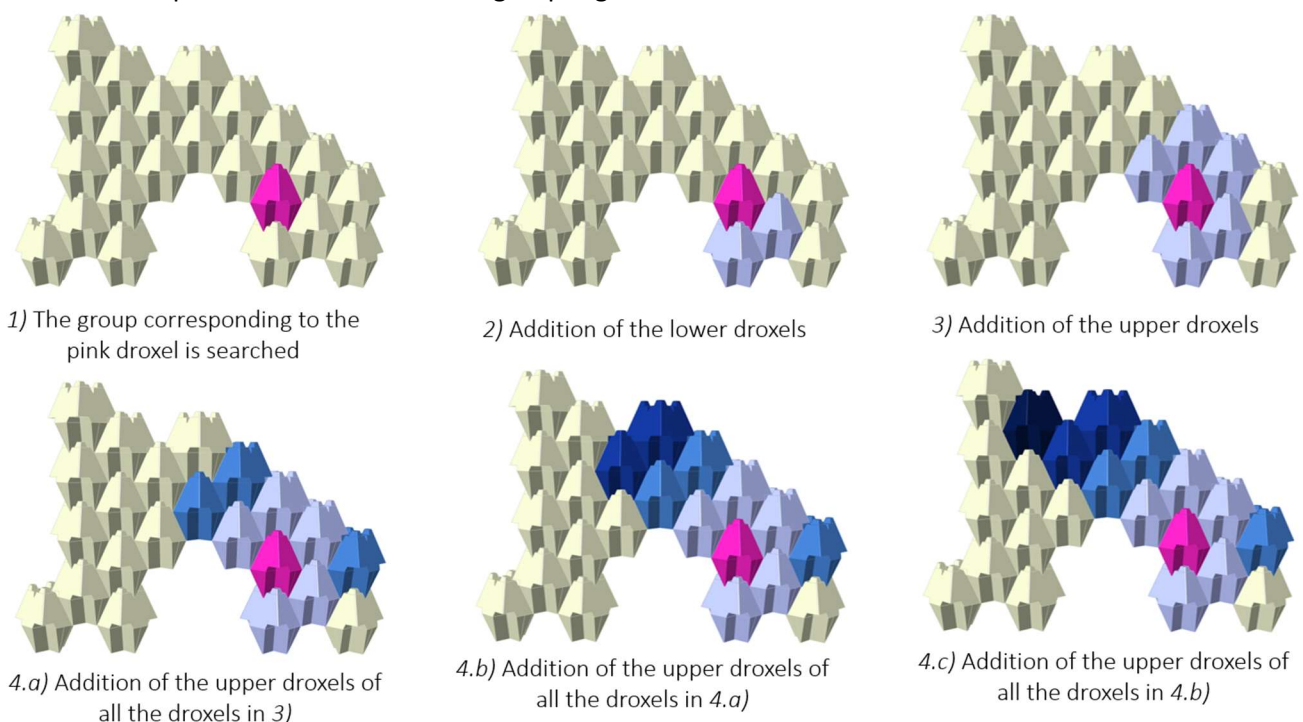


Figure 6.7 - Determination of a group of droxels (part I)

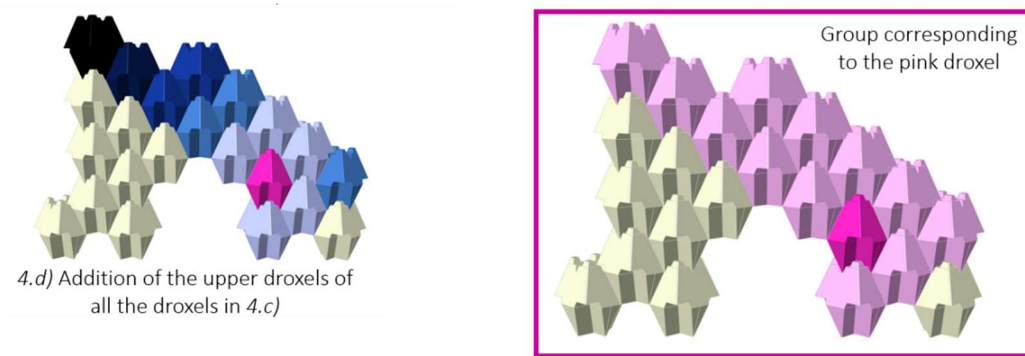


Figure 6.8- Determination of a group of droxels (part II)

Hooked group

The notion of hooked droxel has been explained in section 6.2.2. A group of droxels is said to be hooked if (at least) one of its lowest droxel is hooked (see Figure 6.9).

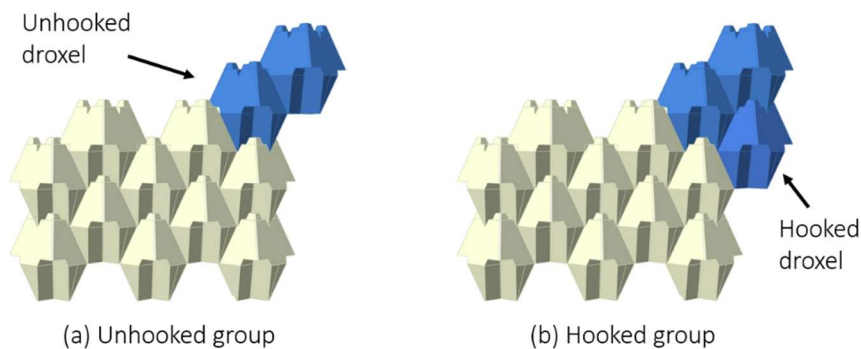


Figure 6.9 - Example of an unhooked and hooked group

Support area of a group

The support area of a group is found by creating the **complex hull** of the support point of all the droxels. The complex hull of a set of points is : *“The line completely enclosing a set of points in a plane so that there are no concavities in the line. More formally, it can be described as the smallest convex polygon which encloses a set of points such that each point in the set lies within the polygon or on its perimeter.”* [39].

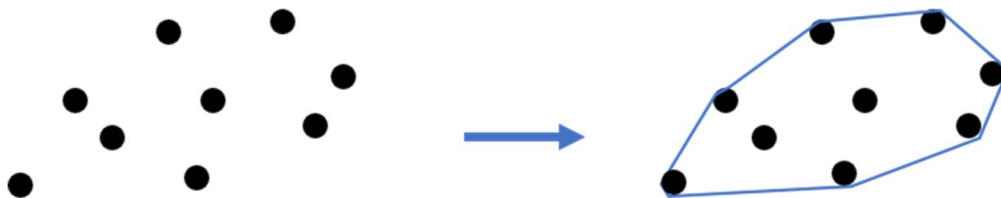


Figure 6.10 - Convex hull of a set of points : example

For droxels, the set of points consist of the corner points of the contact surfaces between a droxel and its supporting droxel(s) (see Figure 6.11).

The complex hull is computed in the program by implementing the Graham Scan [13] algorithm into C# [30].

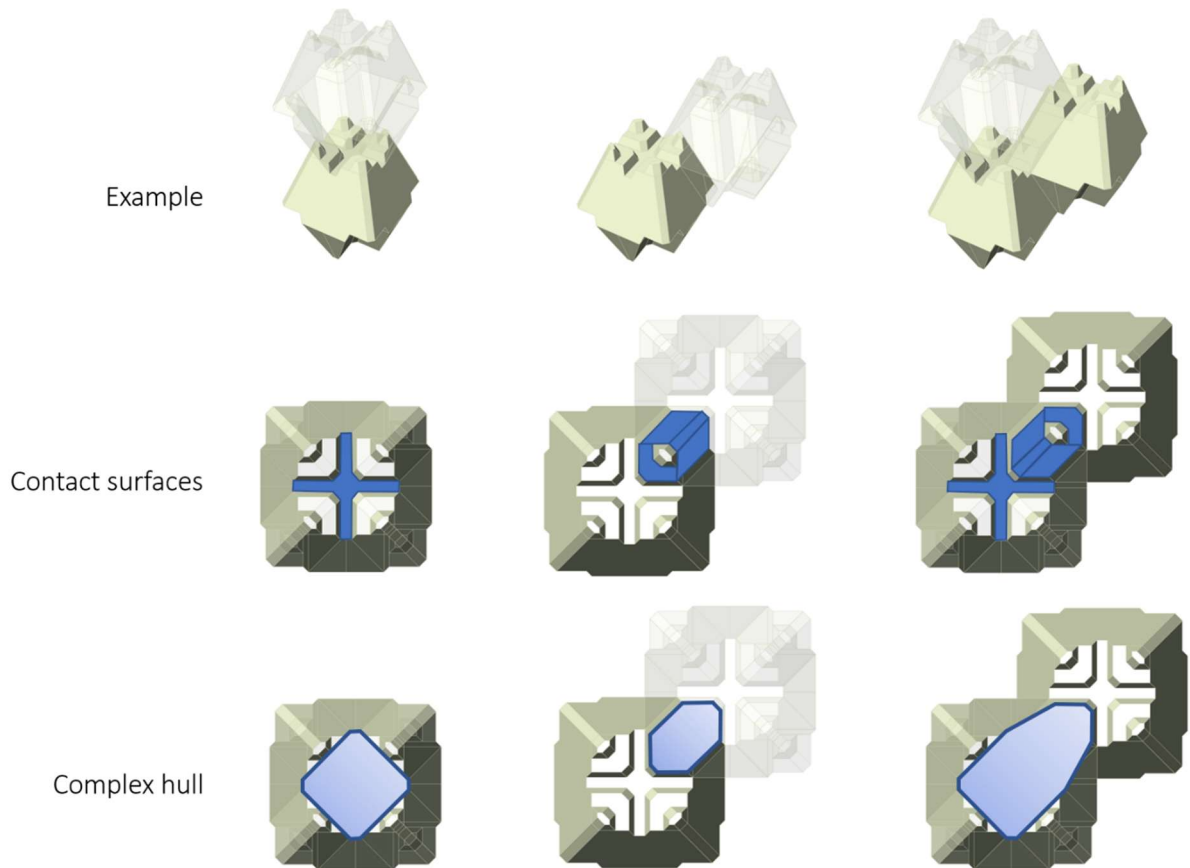


Figure 6.11 - Determination of the complex hull for droxels

Gravity test of a group

A group is always stable when hooked. When unhooked, a group is stable only if the projection of its center of gravity is inside the support area of the group. This last condition has been called the **gravity test** of a group.

Figure 6.12 represents the (unhooked) group corresponding to the pink droxel. This is an example of a group which passes the gravity test because the projection of its center of gravity is inside its support area. The structure is therefore stable.

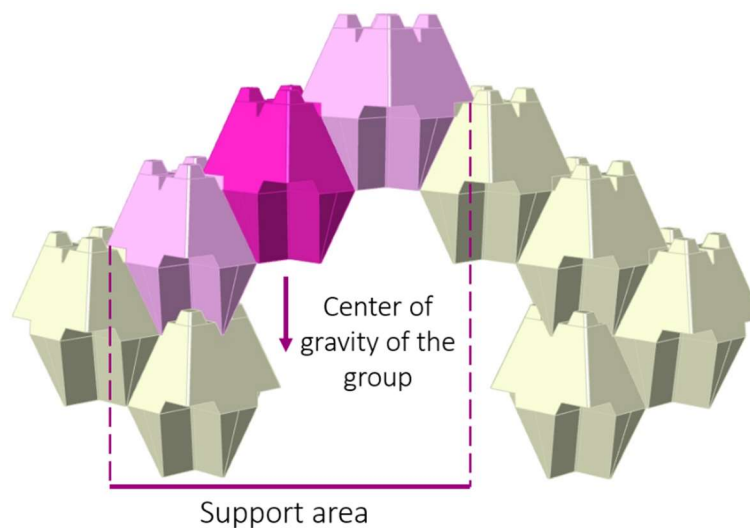


Figure 6.12 - Example of the gravity test for the light pink group. The test is passed

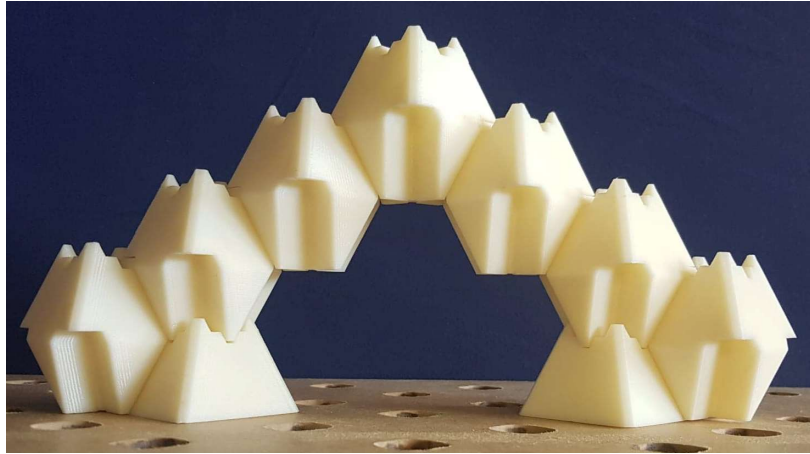


Figure 6.13 - Experimental validation with 3D printed droxels: the structure is stable

Stability of a structure

The stability of a structure is assured if for every droxel of the structure (1) the gravity test is passed or (2) if the group associated to the droxel is hooked. This is illustrated in Figure 6.14:

- For the structure (a) the projection of the center of gravity is inside the support area, the structure is stable.
- For the structure (b) the lowest droxel of the group is hooked, the structure is stable.
- For the structure (c), the lowest droxel of the group (the blue one) is unhooked and the projection of the center of gravity of the group is not in the support area, the structure is unstable.

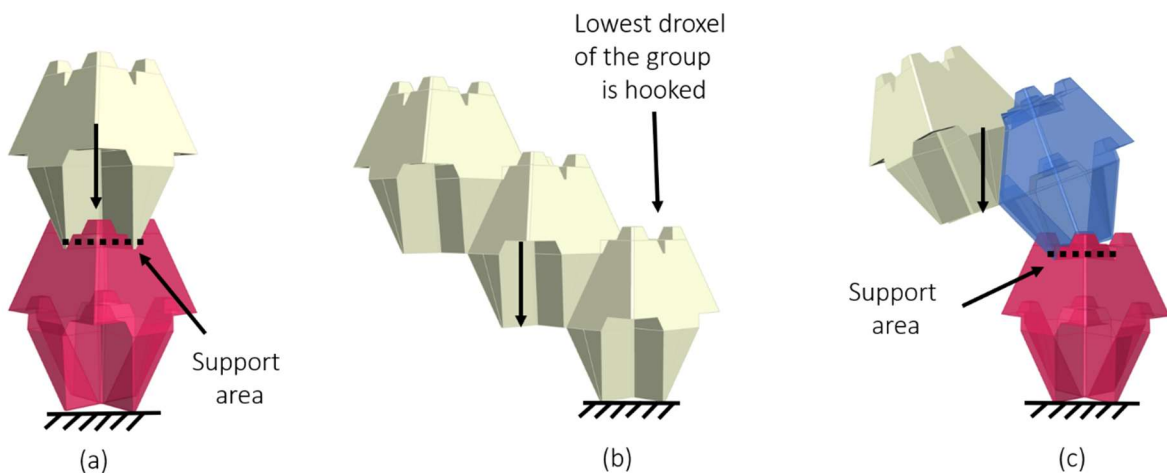


Figure 6.14 - Stability determination

6.3 STABILITY CHECK ALGORITHM

6.3.1 For 2D structures

Flowchart representation of the algorithm

A stability check algorithm has already been proposed for 2D structures by S. GOESSENS [24]. This algorithm allows to verify whether a certain structure remains stable after the addition of a droxel or not. A flowchart representation of this algorithm is given in Figure 6.15.

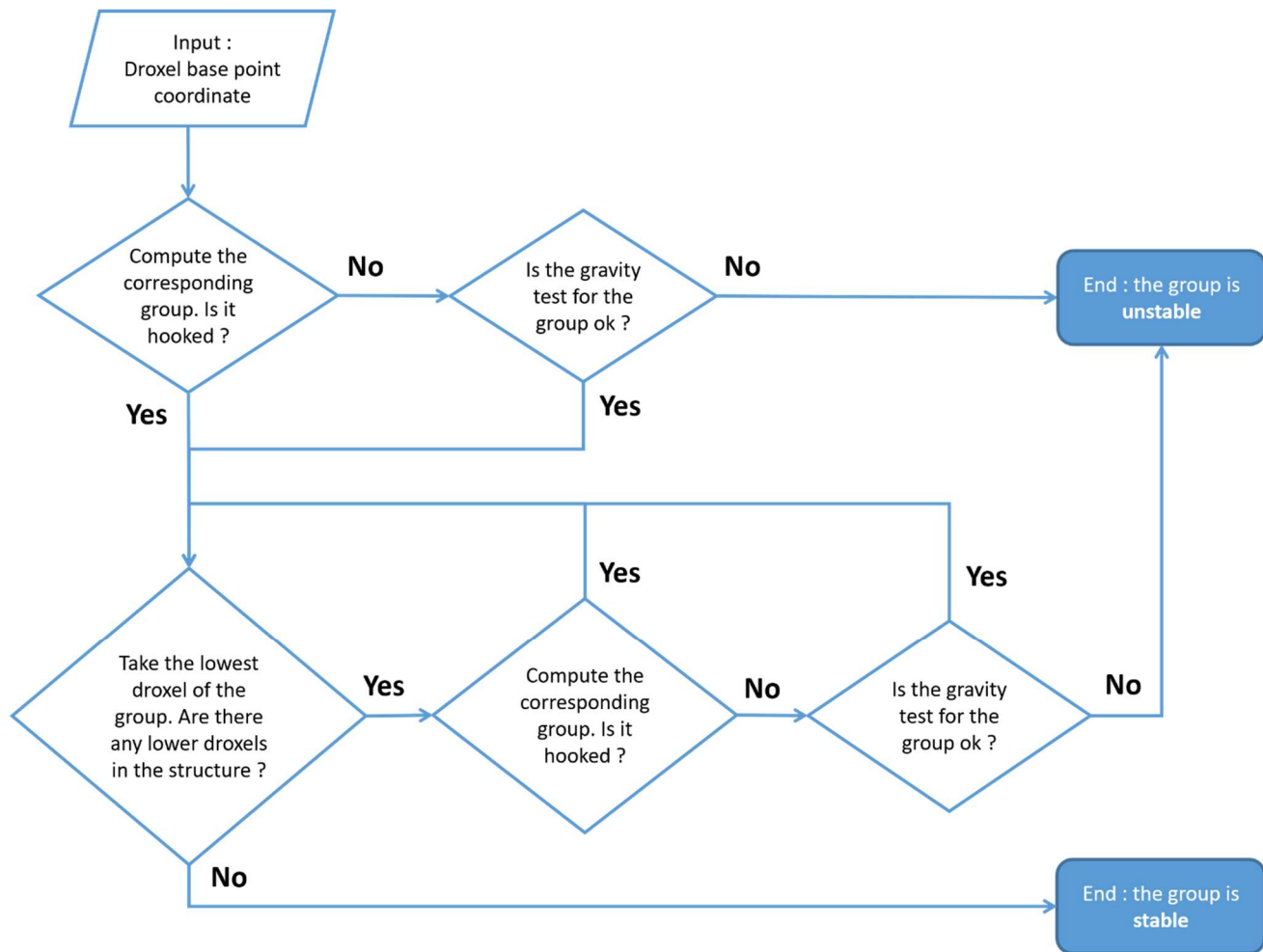


Figure 6.15 - Flowchart : stability algorithm check of a group

Stability check of the whole structure

In order to verify the stability of a final structure, the two conditions (gravity test or being hooked) are checked for all the groups associated to every droxel of the structure. If one of the two conditions is met, a given group is stable. If it is the case for all droxels, the structure is stable. An example of this is given in section 6.6.

This last method is however time consuming. In reality, the global stability of a structure does not need to be checked beforehand: if it is possible to find a laying sequence (see next chapter) that is stable during every construction stage, the final structure is stable (Figure 6.16 (a)). If it is not possible to find a laying sequence: either the final structure cannot be built (Figure 6.16 (b)) or the final structure is not stable (Figure 6.16 (c)). It is however necessary to first compute an algorithm capable of determining the stability of a structure before being able to implement a laying sequence algorithm.

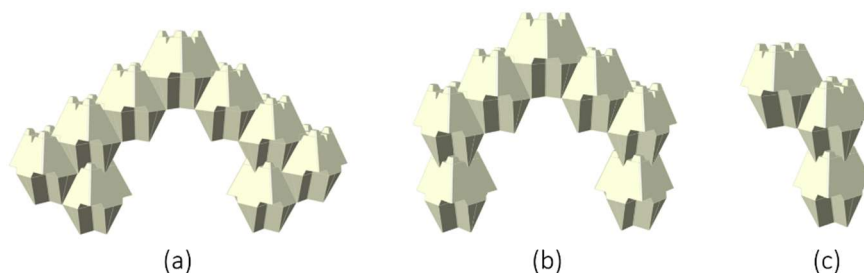


Figure 6.16 - For structure (a) a laying sequence can be found, for (b) and (c) no laying sequences are possible

6.3.2 Generalisation to 3D cases

G. GUILMIN [26] has already implemented the stability algorithm for 3D structures during his master thesis. The algorithm proved to be functional for most cases (the exceptions are tackled in section 6.5). GRÉGOIRE uses *Grasshopper* to physically construct the droxels in the 3D *Rhino*-space and determine the center of gravity of a certain group.

Even though this method works, it leads to large computation times. The program is limited to a maximum of 10 floors (for the concept of floors : see Figure 6.17). It also considers that the droxels of a certain layer are all placed at once. The latter could lead to problems when trying to determine the laying sequence of droxels. In reality, only one droxel can be placed at a time. It is therefore necessary to know the impact of the addition of a single droxel to the structure in order to be able to guarantee its stability during every construction stage.

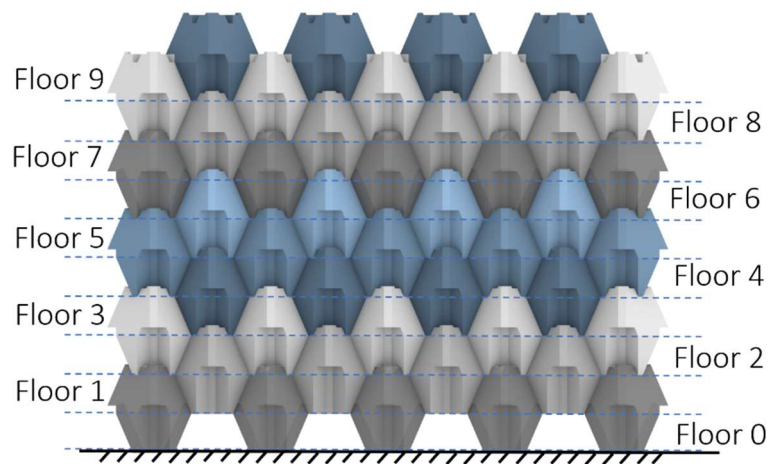


Figure 6.17 - Concept of floors [26]

This is why a different approach is implemented for this thesis: only using the physical base point of the droxels as input of the program. Using these, as well as the 8 droxel parameters, the algorithm then determines whether the structure is stable or not. The complete calculation is done in the *C#* script and the visualisation through *Rhino* is optional at the end.

The construction of the droxels for the visualisation is also done with a *C#* code. First, all the points of the corners of the droxels are calculated. These points are then joined to form the edges of the droxel-faces (Figure 6.18). The physical construction of a droxel is only done once. When visualising two or more droxels, the already computed droxel is moved to its corresponding base point coordinate.

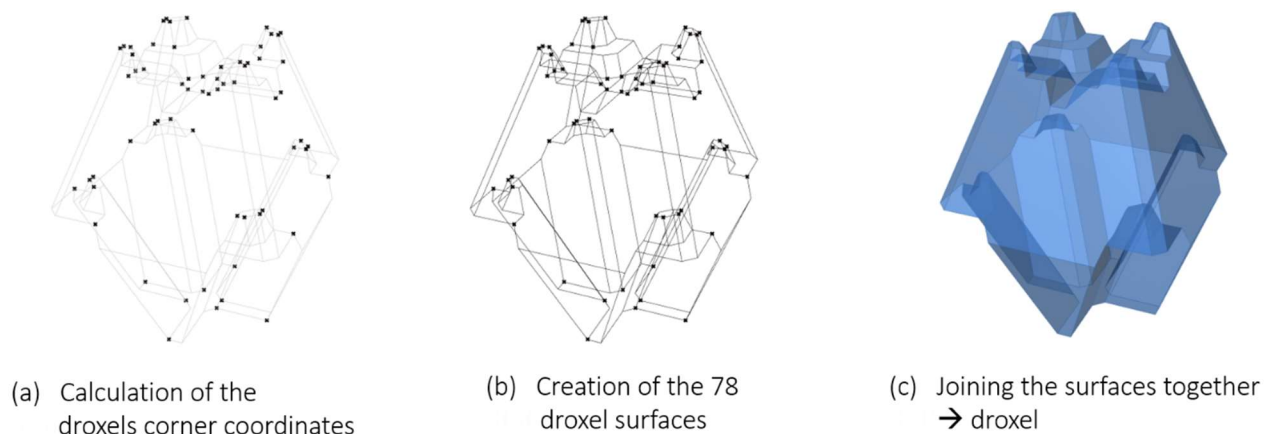


Figure 6.18 - Creation of a droxel in *Grasshopper*, calculation and connection of the corner points

In conclusion, the implementation of the stability algorithm is done in *C#* which has several advantages compared to coding the algorithm directly in *Grasshopper*. The faster computation time, infinite possibility of the number of floors (in the limit of the memory capacity of the computer) and the capacity to add a single droxel at a time can for example be mentioned. The latter is capital when determining the laying sequence of a structure (see CHAPTER 7).

6.4 CODING CHOICES

Some coding choices, which can help to understand the *C#* scripts are detailed below. For an easier visual representation, the normalised coordinates are used in this example.

Following structure is considered:

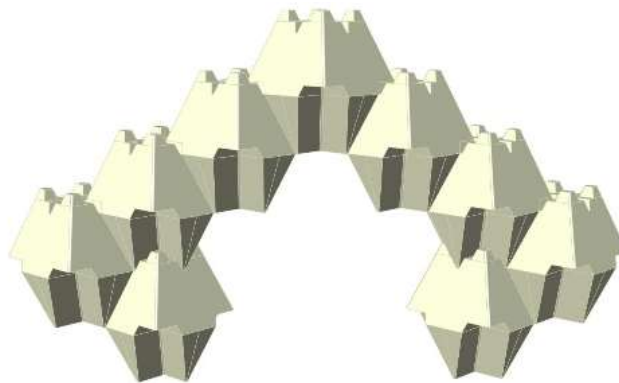


Figure 6.19 - example of an arch-like structure

Index of the droxels

At the very beginning of the code, the droxels are given by ascending z-value. Each droxel has a unique value starting at zero. The droxels of the example are numbered like this:

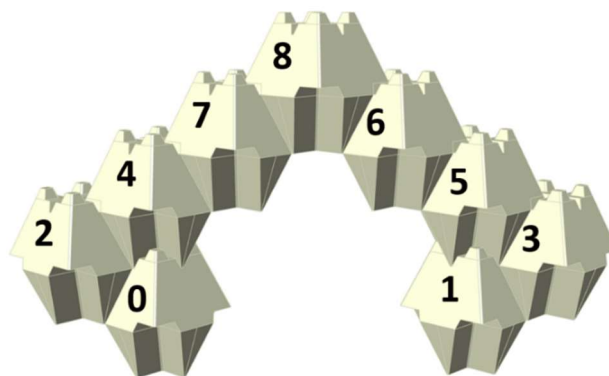


Figure 6.20 - Example: numbering of the droxels

Representation of the droxels

A single droxel is represented by a *Point3d* object. This is a *C#* instance with 3 variables: the x-, y- and z-coordinates. These coordinates correspond to the base of the droxel (see Figure 6.21).

A structure is represented by a list of *Point3d*. The corresponding list of normalised points representing the structure is given in Figure 6.22.

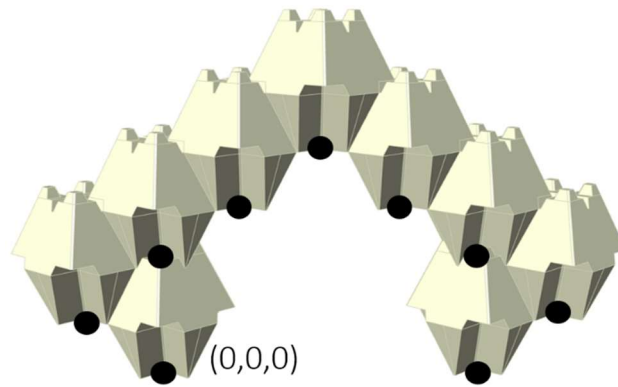


Figure 6.21 - Example: the droxel coordinates correspond to their base point (black points)

2D arch	
{ 2, 2, 4 }	
{ 3, 3, 3 },	{ 1, 1, 3 }
{ 0, 0, 2 },	{ 4, 4, 2 }
{ -1, -1, 1 },	{ 5, 5, 1 }
{ 0, 0, 0 },	{ 4, 4, 0 }

Figure 6.22 - Example: normalised coordinates

Neighbours

Every droxel has a maximum of 10 neighbours. The positions of the neighbours are given in Figure 6.23.

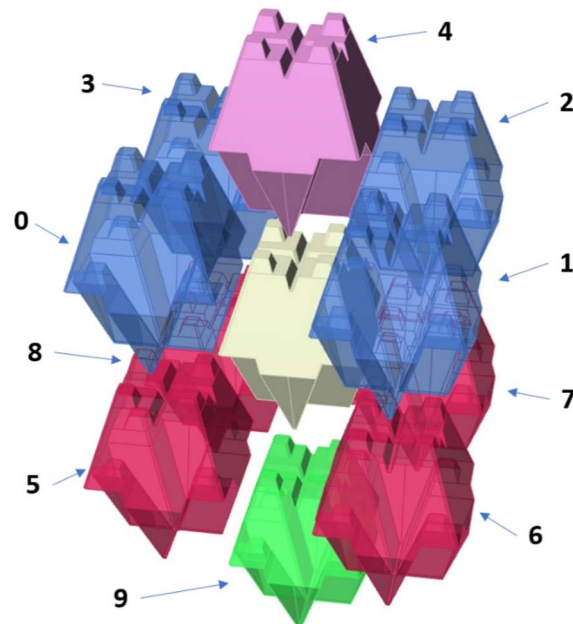


Figure 6.23 - Index of the neighbours (of the beige droxel)

Connectivity tree

In order to be able to easily access the neighbour droxels, a connectivity tree is created. A tree is an object that can be composed of different branches and subbranches. In this case, the connectivity tree has as many branches as the number of droxels in the final structure. The first branch corresponds to the first droxel and so on.

Each branch consists of a list of 10 integers corresponding to the 10 potential neighbours. If the neighbour does not exist in the structure, the value is set at -1. Otherwise, the value corresponds to the index of the corresponding neighbour droxel.

For the 2D example of an arch, the connectivity three of the first two droxels is given in Figure 6.24 :

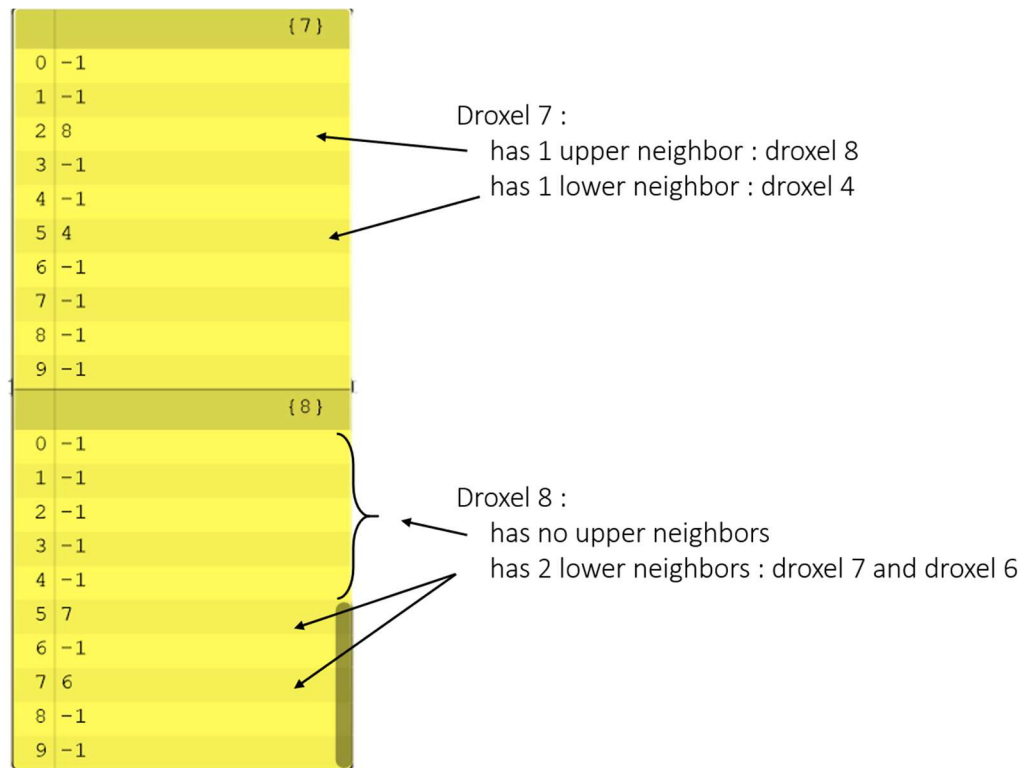


Figure 6.24 - Example: branches of the connectivity tree corresponding to droxel 0 and 1

6.5 LIMITS OF THE ALGORITHM

The algorithm shows some limitations. With more experiments, other limitations will likely be discovered. The following list is thus non-exhaustive.

The found limitations are all due to two assumptions:

- 1) The algorithm neglects the friction between the droxels : the second instability mode (represented in Figure 6.25) is not considered.

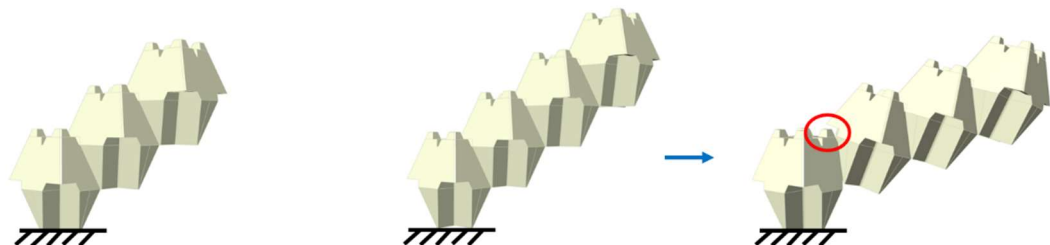


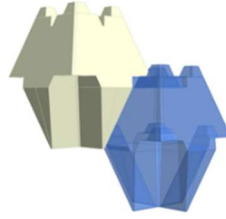
Figure 6.25 - Second instability mode : not considered in the algorithm

- 2) A droxel in the algorithm is either hooked or not. In reality, a droxel can be hooked in one (or more) directions but not in another. At the moment, if a droxel is unhooked in at least one direction, it is considered completely unhooked. This is on the safe side but could be modified.

Examples of the limits are given in section 6.6.

6.5.1 Friction between the blocks neglected

The first assumption, neglecting the friction between the blocks, is not correct. Indeed, this implies that the number of droxels possible in cantilever mode (mode 1A) is infinite for every possible droxel geometry. This is only the case when $A^* \leq \phi$ (with ϕ the friction coefficient) [35].



Mode 1A

Figure 6.26 - Reminder : mode 1A

A relationship between the maximum number of droxels in cantilever mode, the droxel parameters and the friction coefficient has been determined [35]:

$$\frac{d}{p} \cong \frac{1}{2} \left(\frac{1}{\tan(A^* - \phi)} + \tan(A^*) \right)$$

Equation 1 - Relationship to determine the maximum amount of droxels in cantilever mode

The determination of d and p is given in Figure 6.27 and Figure 6.28:

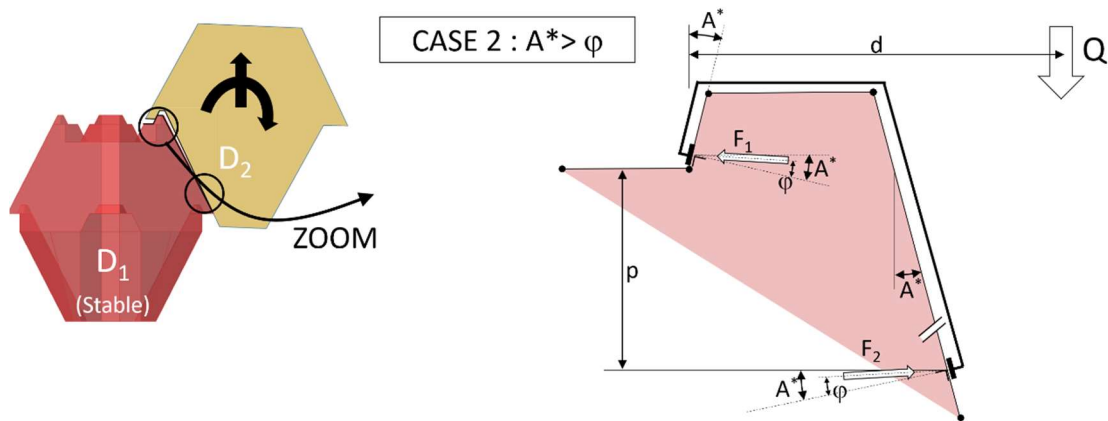


Figure 6.27 - Determination of p [35]

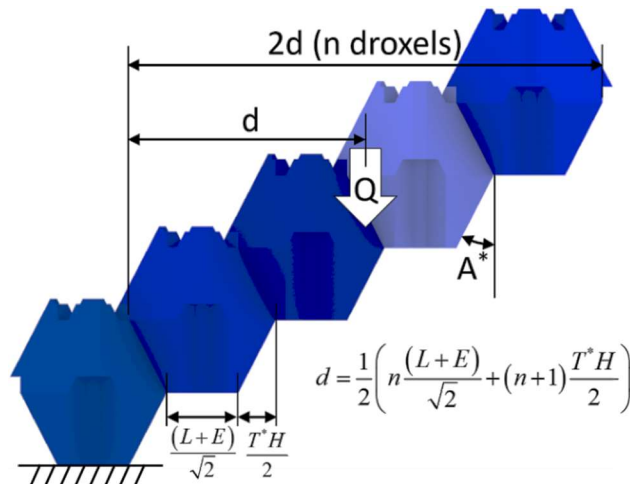


Figure 6.28 - Determination of d [35], $T^* = \tan(A^*)$

The corresponding graph of Equation 1 for the used 3D printed droxel is:

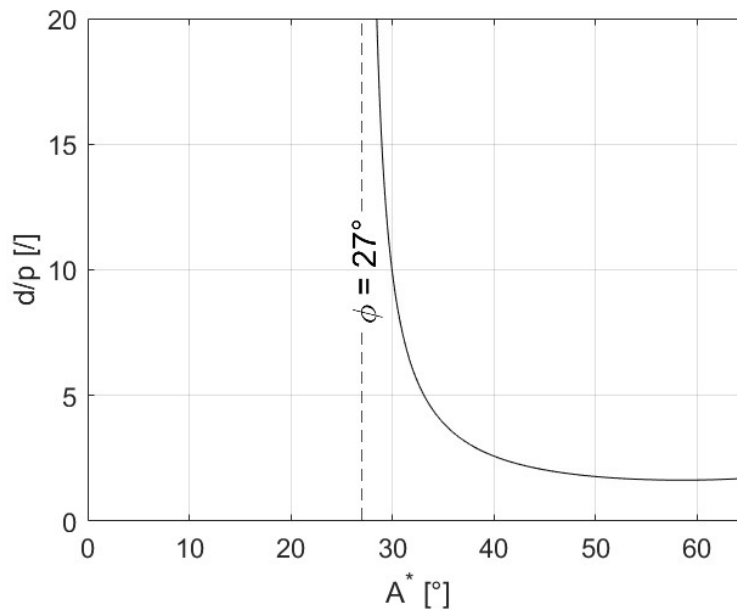


Figure 6.29 - Maximum value of d/p for given values of ϕ and A^*

As has been shown in CHAPTER 4, following values are valid for the printed droxels: $A^* = 27,14^\circ$ and $\phi = 27^\circ$. Since those angles are almost identical, the situation tends to the asymptote in Figure 6.29 where the relationship is likely not valid anymore. It is therefore difficult to compare the graph with the experiments. For the values of A^* and ϕ , an almost infinite amount of droxel in cantilever mode should be possible.

It has been experimentally determined that a maximum of 5 droxels can be put in cantilever mode. The corresponding d - and p -values are respectively 98 [mm] and 21 [mm] (see Figure 6.30). According to the graph A^* should therefore be approximately equal to 33° .

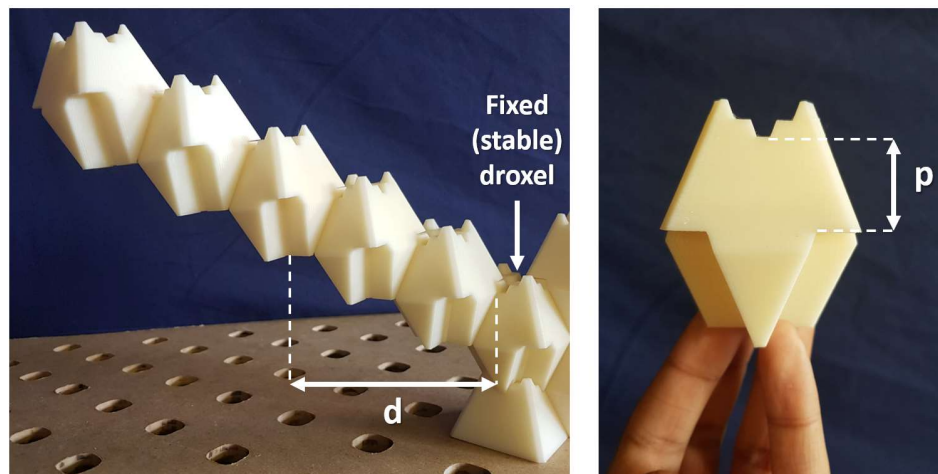


Figure 6.30 - Determining d and p

There are several possible reasons for this difference. Either the graph is not valid anymore starting from a certain d/p . It is also possible that the value of ϕ is overestimated. The fact that the droxels have a certain fabrication tolerance can also explain the difference between the theory and the experiments.

Even though interesting for droxels in mode 1A, this relationship should also be extended to other stacking modes: the sliding between the droxels is indeed not only limited to droxels in mode 1A (see 6.2.1, p. 57). It can also occur for the other stacking modes.

Suggestion

In order to be able to take the friction into account in the algorithm, it might be possible to look for a more general law. It may be possible to define a maximum equivalent moment M_{equ} to which an upper hook can be subjected before creating an instability due to the unhooking of one or more droxels. This moment would most likely be function of the friction coefficient, the geometry as well as the stacking mode of the droxel and could be implemented into the *Grasshopper* code directly once the laws have been determined.

This phenomenon might however be complicated to quantify. It cannot be excluded that numerical simulations will always be necessary in order to take these effect into account, further research on the friction is necessary.

6.5.2 Hooked is binary

At the moment a droxel is either hooked or unhooked. In reality, a droxel can be hooked in 1, 4 or no direction at all. For the simple example represented in Figure 6.31, the upper droxel rests on an unhooked droxel. Since the resultant of the center of gravity of this group is outside of its support area, the algorithm tells us that this structure is unstable. However, both real-life tests and numerical simulations show that it is stable. It might therefore be useful to add the notion of direction to whether a droxel is hooked or not.

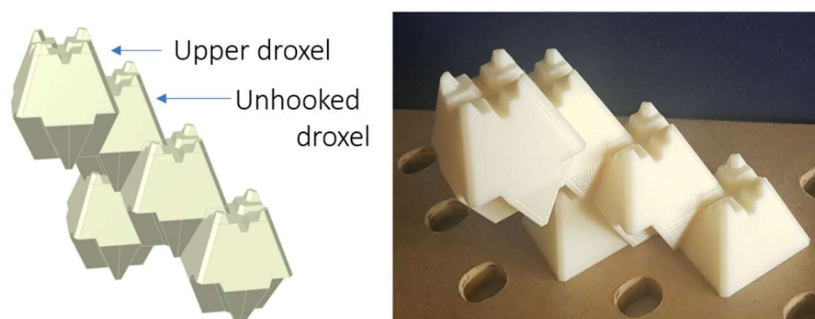


Figure 6.31 - Limitations of the algorithm: hooked is binary

CONCLUSION

It is now possible, for a given droxel geometry and droxelised structure to determine whether that structure is globally stable (no tilting of a group of the structure) or not. The big hypothesis of this chapter is that $A^* \leq \phi$. If this is not the case, a second check must be done in order to assure that no droxels are being “unhooked” due to the droxels sliding on each other. The latter is, according to me, one of the next necessary steps needed in order to generalise the program even more.

The limitations of the program have also been highlighted and clarified. They do need to be taken into account for further works.

6.6 EXAMPLES

An example of a stable and unstable structure is given to illustrate how the algorithm works. In order to validate the algorithm, numerical and experimental verifications are performed.

Three examples, showing the mentioned limitations are also given.

6.6.1 Example 1: stable 2D example

Algorithm

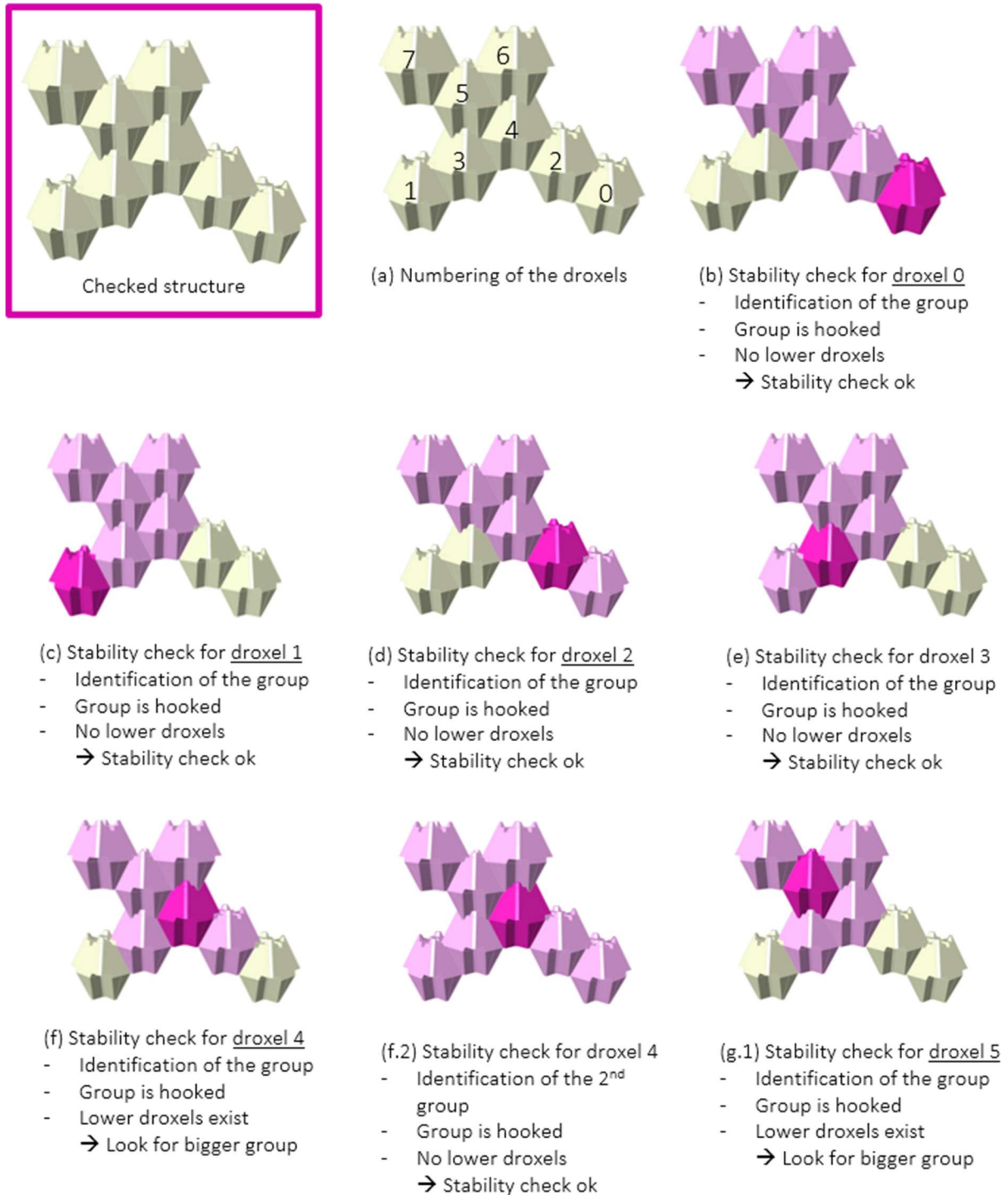


Figure 6.32 - Illustration of the algorithm for a stable example (part I)

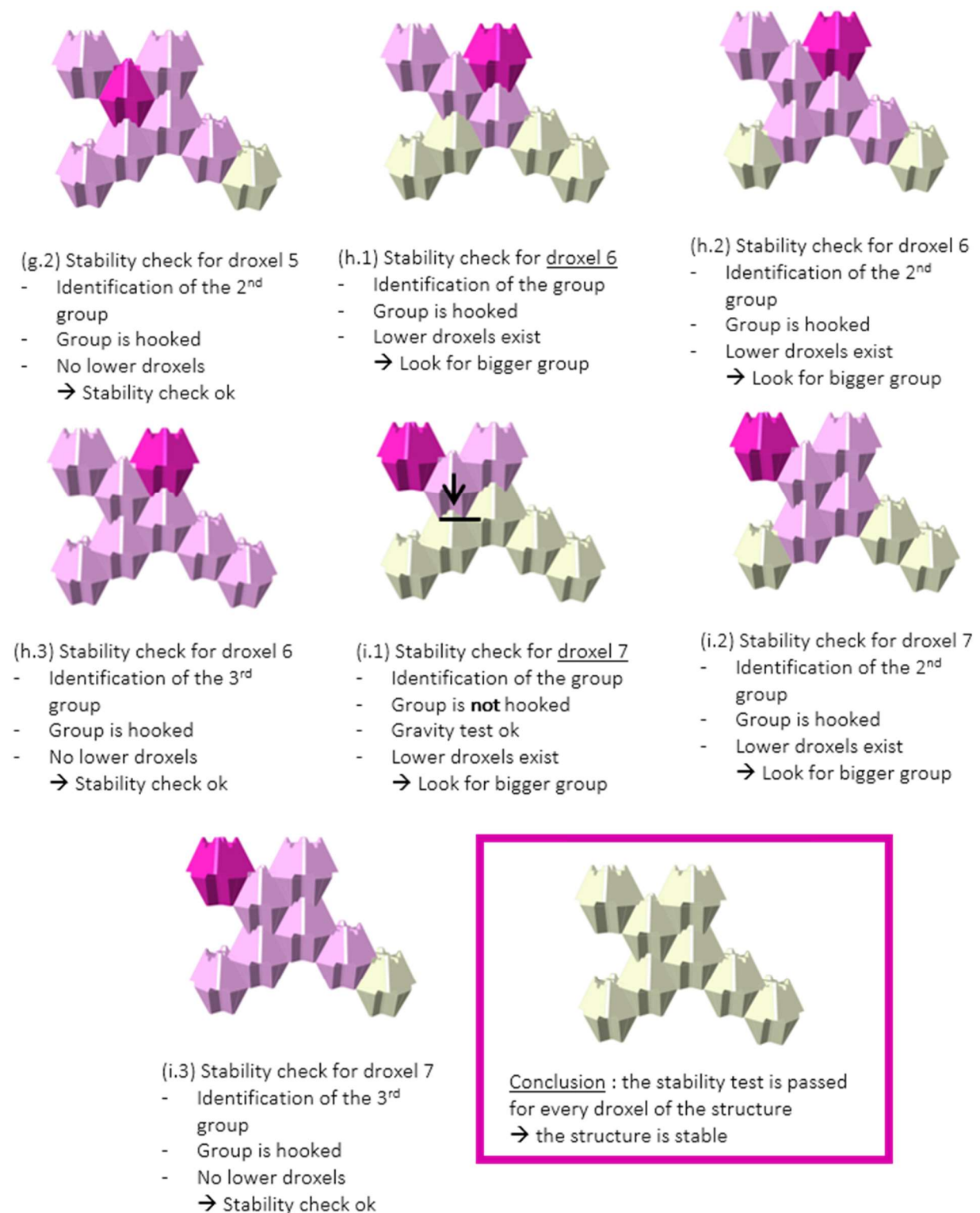


Figure 6.33 - Illustration of the algorithm for a stable example (part II)

Numerical simulation

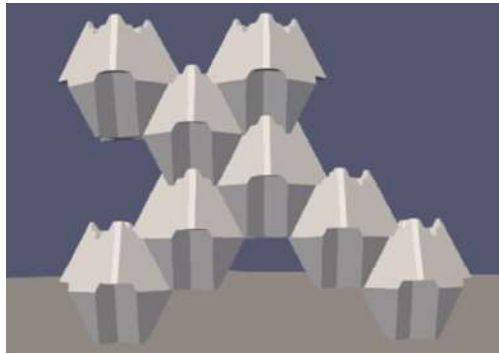


Figure 6.34 - Example 6.1 : numerical simulation after $t = 45$ [sec] : the structure is stable

Experimental test



Figure 6.35 - Example 6.1 : experimental test

Conclusion

The *Grasshopper* algorithm as well as the numerical simulation and experimental test give the same result: the structure is stable.

6.6.2 Example 2: unstable 2D example

Adding an 8th droxel to the previously checked structure, it can be seen that the structure becomes unstable according to the *Grasshopper* algorithm.

Algorithm

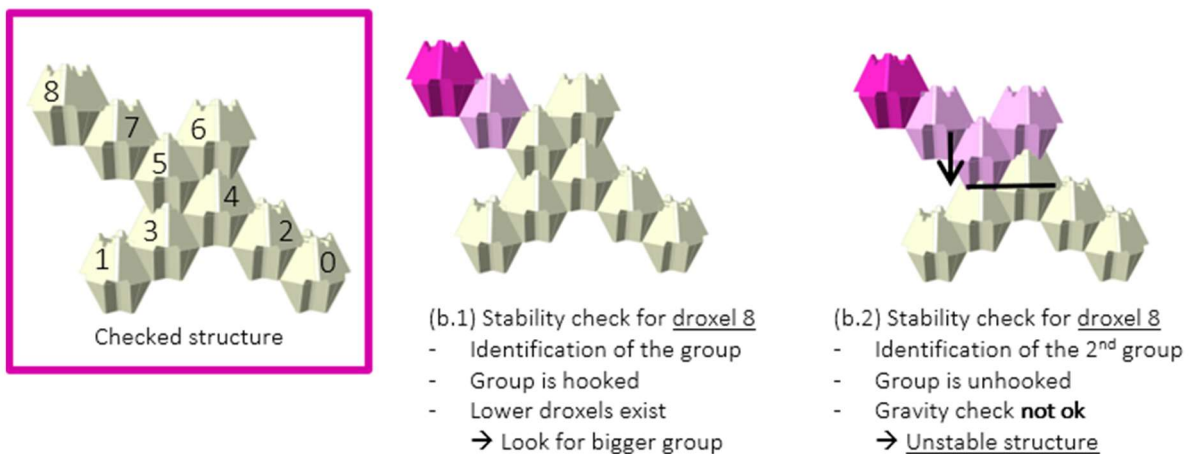


Figure 6.36 - Illustration of the algorithm for an unstable example

Numerical simulation

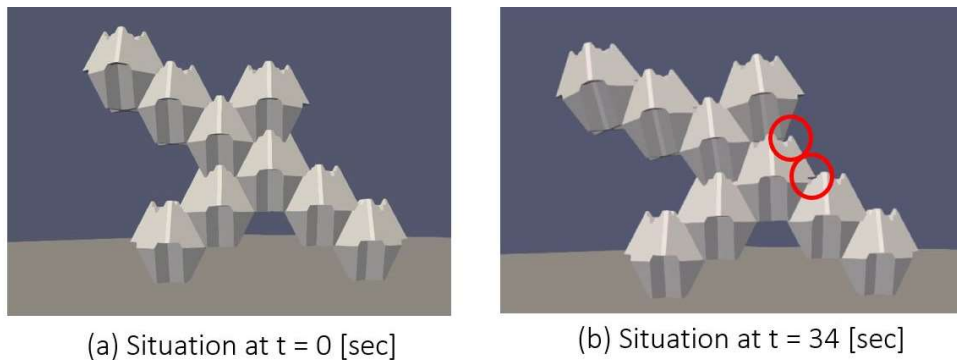


Figure 6.37 - Example 6.2: numerical simulation using LMGC90

Experimental test

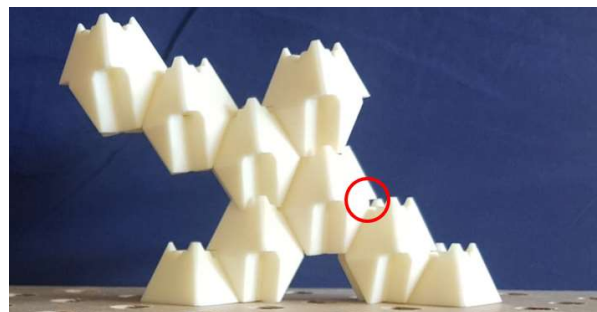


Figure 6.38 - Example 6.2: experimental test

Conclusion

Even though no “real” collapse of the structure occurs, the proposed structure is clearly unstable both for numerical simulation and the test with printed droxels. The fact that the droxels’ heads are somewhat preventing the structure from falling is indeed something that is not taken into account in the *Grasshopper* algorithm. Since this consideration is on the safe side, it can be accepted that this is not taken into account.

6.6.3 Example 3: Limitation of the algorithm : friction

Algorithm

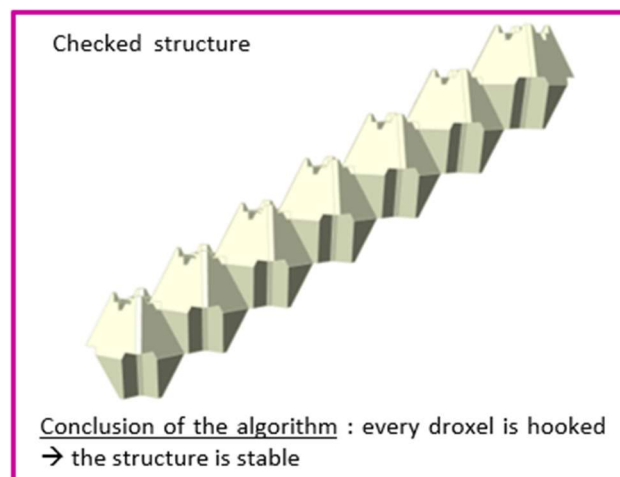


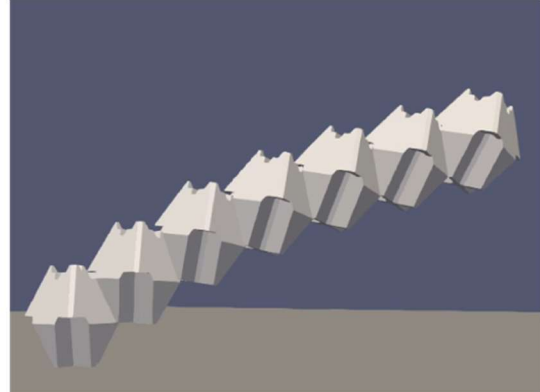
Figure 6.39 - Example 6.3: Grasshopper algorithm

According to the algorithm, which neglects the sliding between the blocks, an infinite number of droxels can be put in cantilever mode. As numerical simulations and experimental tests show, this is not the case.

Numerical simulation



Simulation with 6 droxels at $t = 0$ [sec]

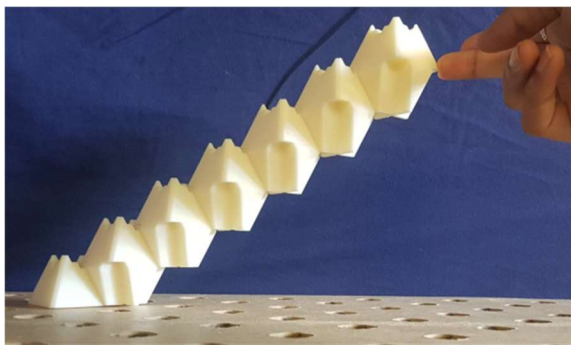


Simulation with 6 droxels at $t = 150$ [sec]

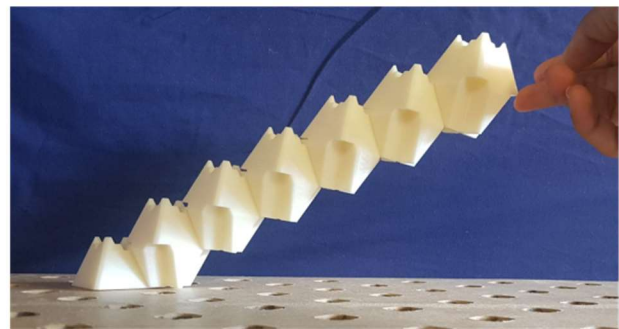
Figure 6.40 - Example 6.3: numerical simulation

The numerical simulations show the situation at the start and at $t = 150$ [sec]. All the droxels have a zero velocity at 150 [sec]. However, the vertical displacement of the droxels is deemed too important and even though the structure has a “stable” final configuration (zero velocity of the droxels), the configuration is not considered as acceptable.

Experimental test



Theoretical placements of the droxels



The droxels collapse when submitted to their self-weight

Figure 6.41 - Example 6.3: experimental: the structure is unstable

Conclusion

As explained, the friction coefficient is not taken into account in *Grasshopper*. This is why the instability is not identified with the implemented algorithm. In order to have a reliable algorithm it is undeniable that taking the friction between the droxels into account is the next necessary step.

Once again, the numerical simulations give an accurate representation of the experimental tests which is reassuring.

6.6.4 Example 4: Limitation of the algorithm: Arch-like structures

Algorithm

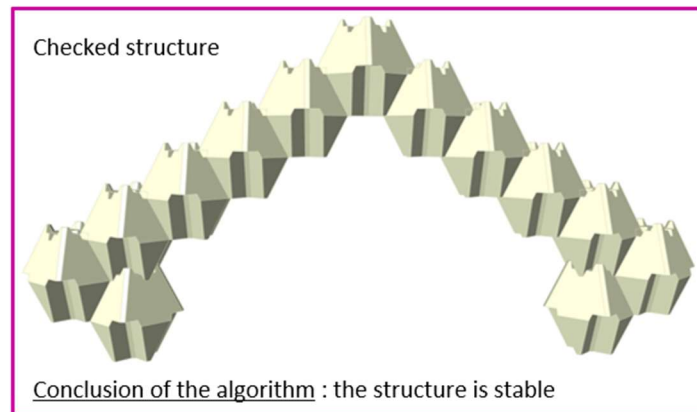


Figure 6.42 - Example 4: Grasshopper algorithm

Experimental test

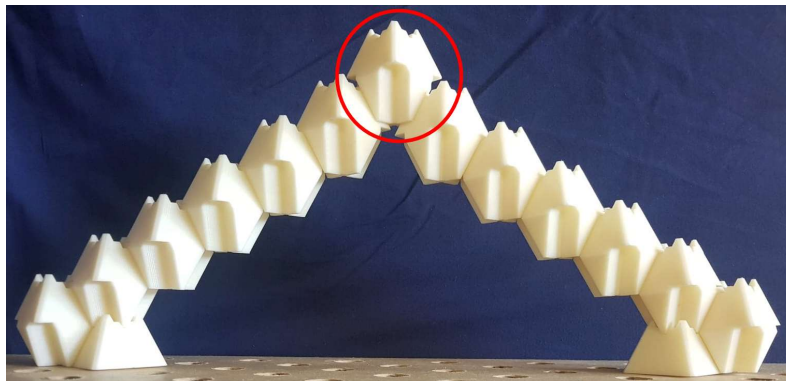


Figure 6.43 - Example 4: lifting of the keystone (experimental test)

For arch-like structures, there is a risk that the keystone raises if the friction is not sufficient to hold the keystone in place. This can lead to a configuration as represented in Figure 6.43: the friction between the droxels and/or the weight of the upper droxel is not sufficient to prevent the keystone from raising.

Solutions for this specific issue could be to add a horizontal buton between the half arches to prevent their collapse, to pull a cable from the keystone to anchor it into the ground or to have a heavier keystone droxel.

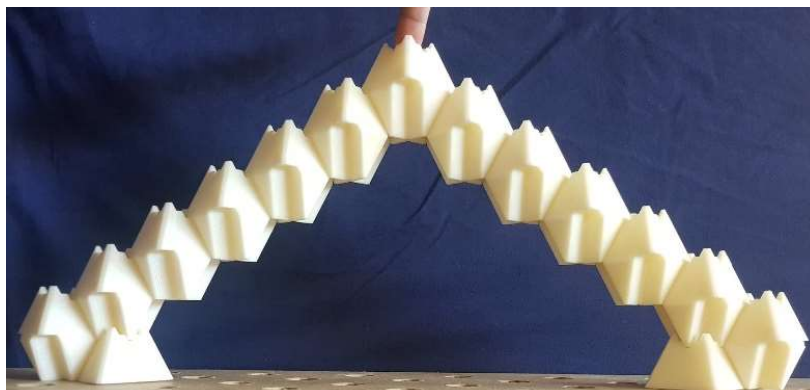
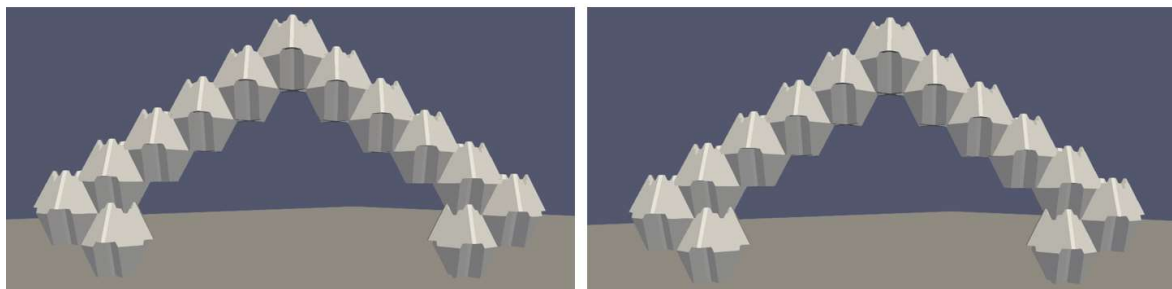


Figure 6.44 - Influence of a heavier keystone or its anchoring to the ground with a cable

Numerical simulation



Situation at $t = 0$ [sec]

Situation at $t = 75$ [sec]

Figure 6.45 - Lifting of the keystone: numerical simulations

Conclusion

The numerical simulation of the structure does not show the settlement effect found in the experimental test.

As explained in CHAPTER 4, the friction coefficient is calibrated in order to match the experimental tests. A different calibration method might be necessary.

Another possible explanation for this is the fact that the effect of the addition or removing of a droxel is not taken into account in the simulations (more on that in 7.4.2).

6.6.5 Example 5: Limitation of the algorithm: hooked is binary

Algorithm

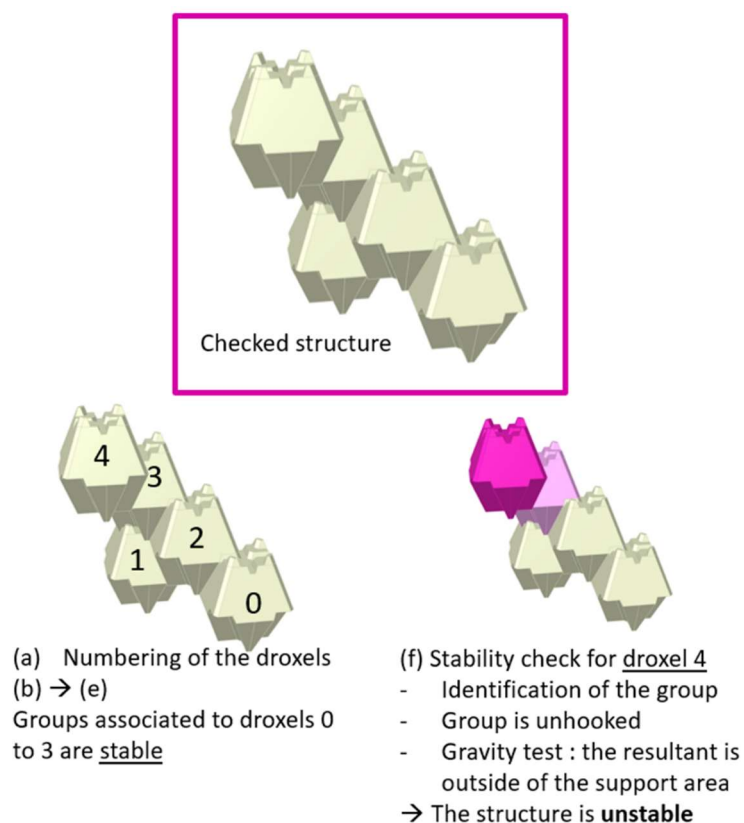


Figure 6.46 - Example 6.5: Grasshopper algorithm

Experimental test

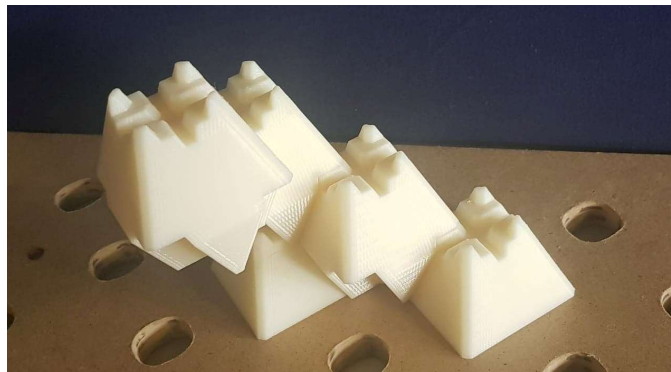
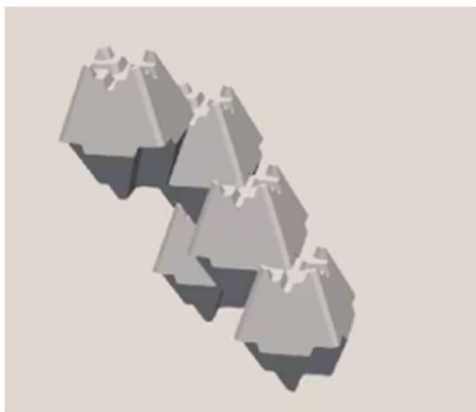


Figure 6.47 - Example 6.5: experimental test : the structure is stable

Numerical simulation



Simulation at $t = 0$ [sec]



Simulation at $t = 30$ [sec]

Figure 6.48 - Example 6.5: numerical simulations : the structure stays stable

Both experimental tests and numerical simulation show that the suggestion made in section 6.5.2 is pertinent. The suggestion: considering that hooked is no longer a Boolean value but considering that a droxel can be hooked in 1, 2, 4 or no direction at all.

EXAMPLES : CONCLUSION

The examples prove that the algorithm works correctly within the given limits. Example 2 even shows that the algorithm is on the safe side for some cases because it does not consider the fact that the droxels' heads can sometimes prevent an unstable group from falling.

The numerical simulations showed that the friction was not always modelled correctly (example 3 and 4). This should be further studied. Two possible reasons for the difference between the experimental and the numerical simulations have been given: the friction coefficient might not be calibrated correctly (but this hypothesis is not consistent with what is seen on example 3) or the fact that the influence of the addition/removing of a droxel is not taken into account in *LMGC90*. More simulations and more thought must be given to this but quantifying those effects might be a good start.

The fact that hooked should not be a binary value anymore but that the direction should be specified was also validated by experimental tests and numerical simulation in example 5.

CHAPTER 7 LAYING SEQUENCE OF A DROXELISED STRUCTURE

It is not sufficient to know whether a given structure is stable or not. The stability during the whole construction of the structure must be assured. This chapter aims to find and implement a method allowing to determine the laying sequence of a structure. If no laying sequence can be found, the final structure cannot be built or is not stable.

This method (searching for the laying sequence of a structure) allows to skip the verification of a structure based on the stability assessment of the groups associated to every single droxel of the structure (as done in CHAPTER 6). Only the influence of the addition of the last added droxel to an already stable structure is verified. This results in an important time gain.

The used method allows to verify whether a structure remains stable after the addition of a single droxel. It is therefore not necessary to check the stability of the final structure before determining its laying sequence. This last characteristic allows a non-neglectable time gain.

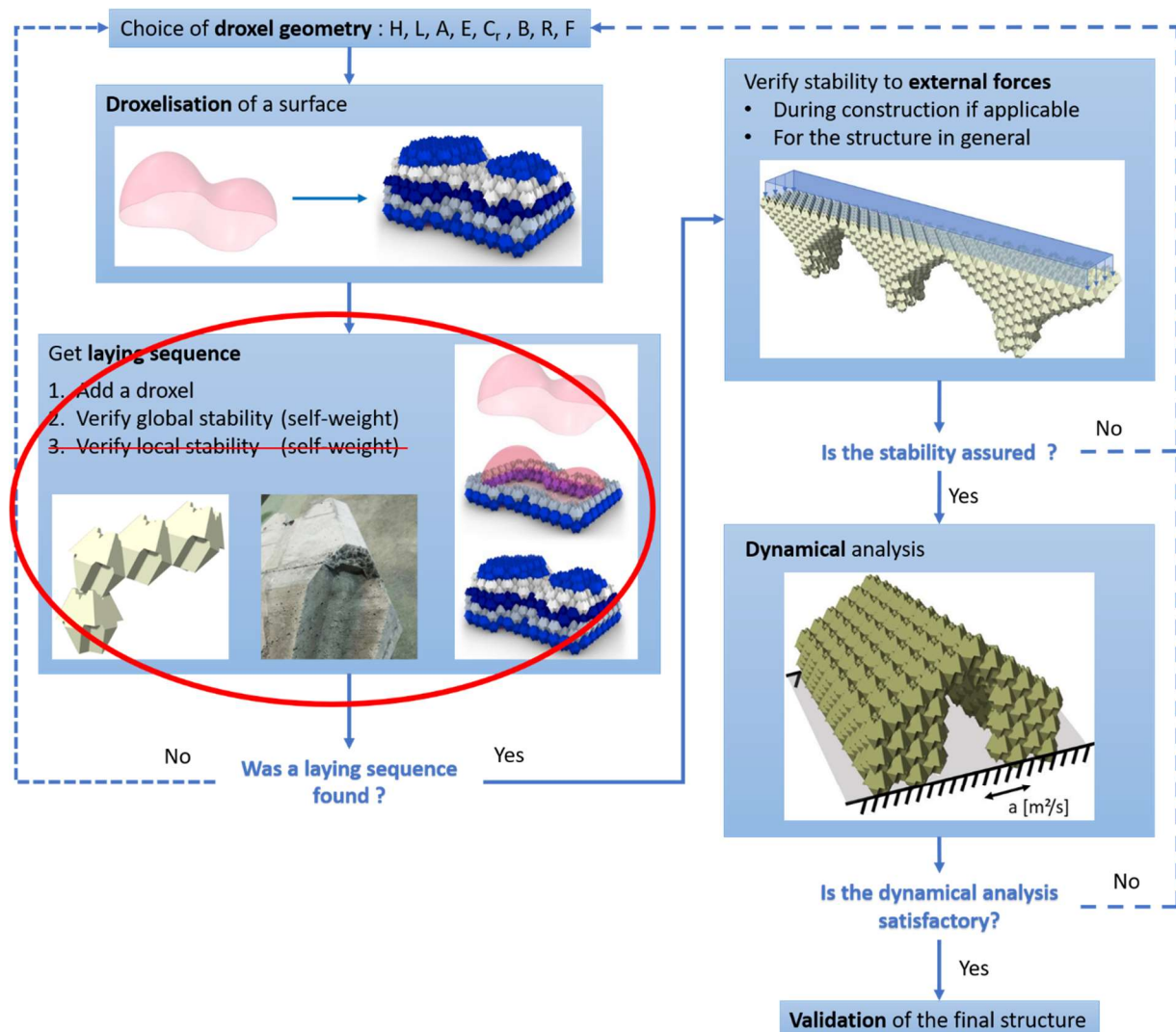


Figure 7.1 - Chapter 7: Laying sequence

The laying sequence is not trivial for all structures. Figure 7.2 represents the building of a stable arch-like structure. As can be seen, this structure requires temporary droxels as counterweight from the

steps (d) to (f). Those are represented in pink in the figure. Without those counterbalancing droxels, the half-arches would collapse before the keystone could be placed.

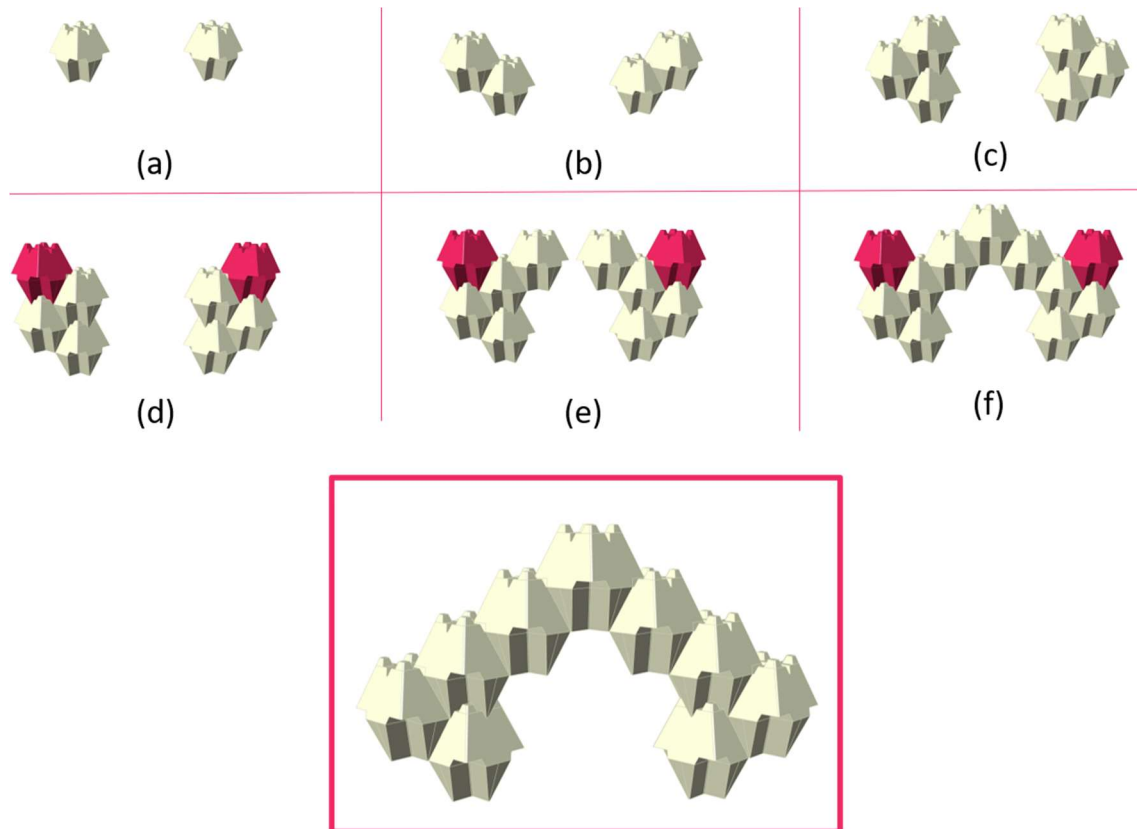


Figure 7.2 - Building of an arch with temporary droxels (in pink) from steps (d) to (f)

Moreover, not all structures which are stable in their final form can be built: an example of such a structure is given in Figure 7.3. Indeed, after step (b), no other droxel can be placed without causing an instability as represented in Figure 7.4. Since only one droxel can be placed at once, the structure cannot be built.

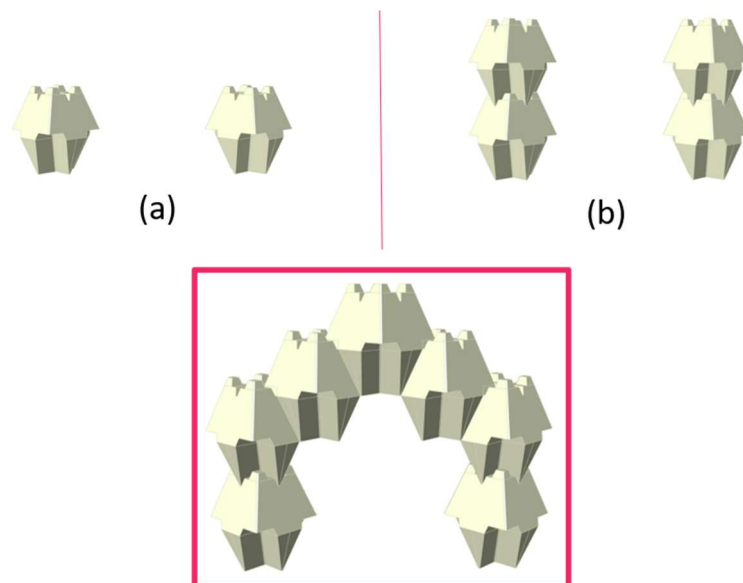


Figure 7.3 - Building of a (stable) gate: impossible to build the structure

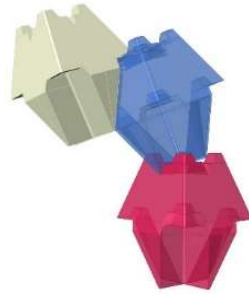


Figure 7.4 - Instability mode caused when trying to execute any step (c) of Figure 7.3

7.1 SOME NOTIONS

In order to explain the laying sequence algorithm, some notions must first be explained.

Cone of influence of a droxel

A first notion is the **cone of influence**. The cone of influence corresponding to a droxel is the cone containing all the base points of all the possible droxels that could be supported by this droxel. In Figure 7.5, the cone of influence of the beige droxel is represented by the blue droxels.

Any instability of the beige droxel might be solved by adding one or more droxels of its influence cone to the structure. The addition of a droxel outside of the cone of influence of a droxel has no influence on the droxel's stability.

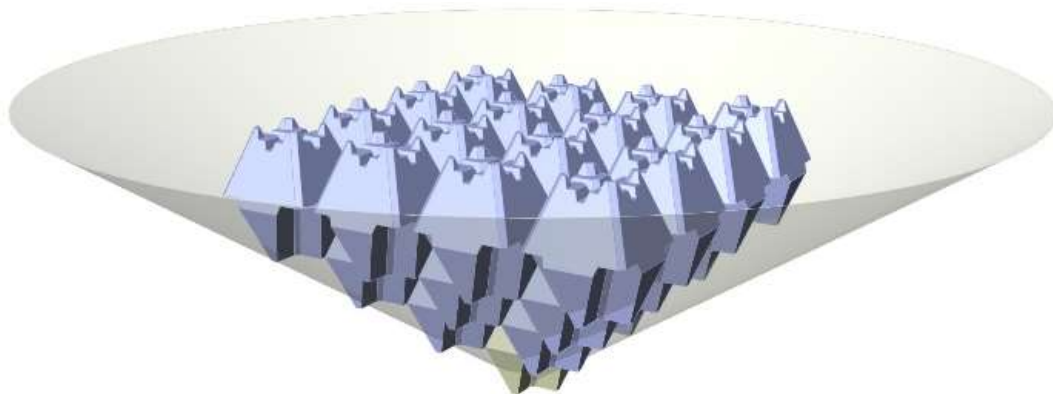


Figure 7.5 - Cone of influence: in blue the cone of influence of the beige droxel

Substructure

The **substructure** corresponding to a given droxel is composed of all the lower droxels of this droxel until the ground floor. Of all the droxels on the ground floor, all the superior droxels are then added (see Figure 7.6).

For some structures, such as the example in Figure 7.6 (b), the substructure corresponds to the whole structure.

Inaccessible positions

For most droxelised structure, there are positions on which no droxels can be added, even as temporary position. These are called the **inaccessible positions** and consist of:

1. All the positions on top of which an upper droxel is placed in the final structure. (Once the upper droxel is placed, it would become impossible to remove the temporary droxel.)

2. All the droxels below the droxels mentioned above. (In order to stay practical and to assure a good execution when removing the temporary droxels.)

Examples of inaccessible positions are given in Figure 7.7.

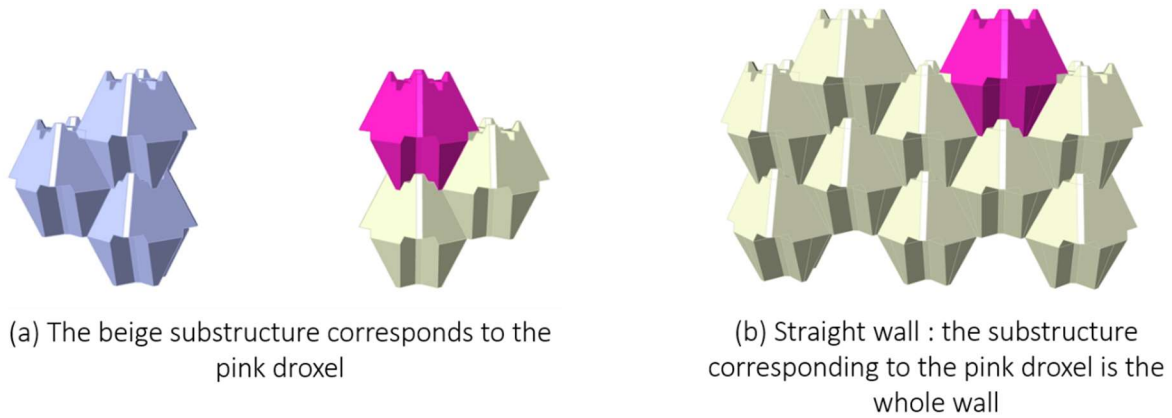


Figure 7.6 - Representation of substructures

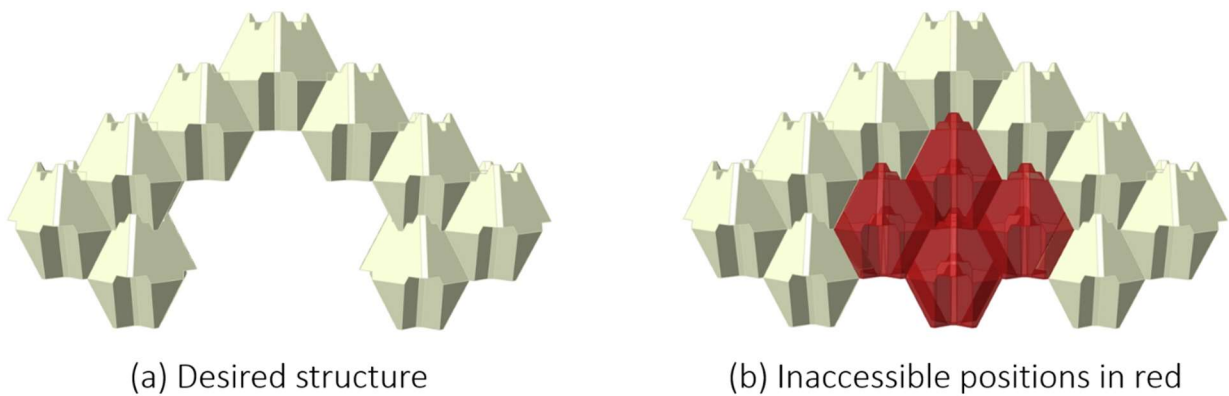


Figure 7.7 - Notion of inaccessible position

Cone of influence of a substructure

The concept of substructure is used when a structure must be stabilised during construction. For example, when building the arch represented in Figure 7.7 : it makes no sense to try to stabilise the right part of the arch by adding a droxel to the left substructure. In order to see which points are pertinent for the stabilisation of the structure, the cone of influence of the substructure is created



Figure 7.8 - Cone of influence of a substructure: the droxels in green are the cone of influence of the right part of the arch

Accessible positions

The **accessible positions** of a (sub)structure are all the positions in the cone of influence of the (sub)structure which are not listed as an inaccessible position and have a lower neighbour in the structure.

This notion is very important to identify where exactly a temporary droxel can be added.

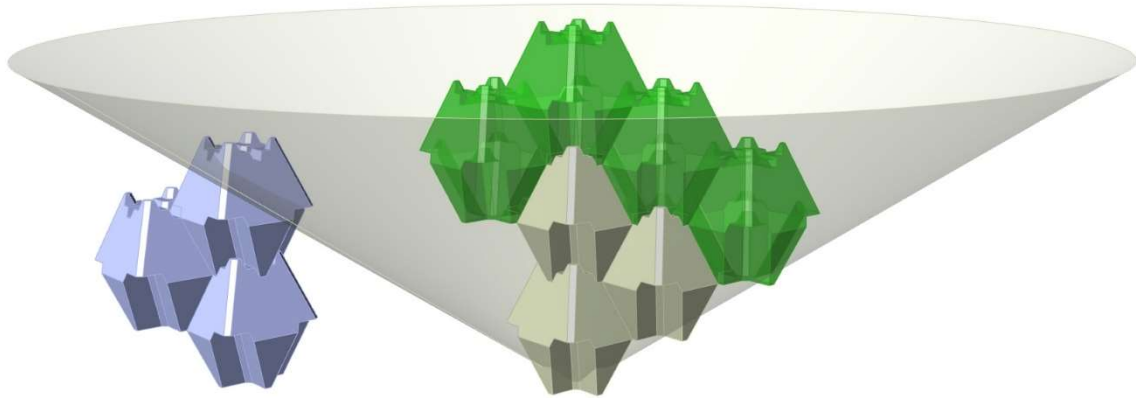


Figure 7.9 - Accessible positions in green: 4 possible candidates (in the plane of the structure) to stabilise the structure

7.2 PRINCIPLES

Some general principles are used in order to develop an algorithm capable of finding a possible laying sequence. If the algorithm does not find a way to build the structure, it means the structure cannot be built.

1) Building floor by floor

A first principle is the fact that a structure is (logically) built from bottom to top. If for a certain floor it is not possible to find a way to build the floor, the final structure cannot be built.

2) Stabilising droxels are removed at the end

In order to reduce the complexity of both the algorithm and the execution, the temporary droxels are removed at the end of (and not during) the construction stage. In the long run, this could easily be modified in order to use less droxels.

3) Main general principle

The main general principle of the algorithm is the following:

1. Placing all the droxels on the ground floor;
2. Placing the next droxel (named D_1) if possible;

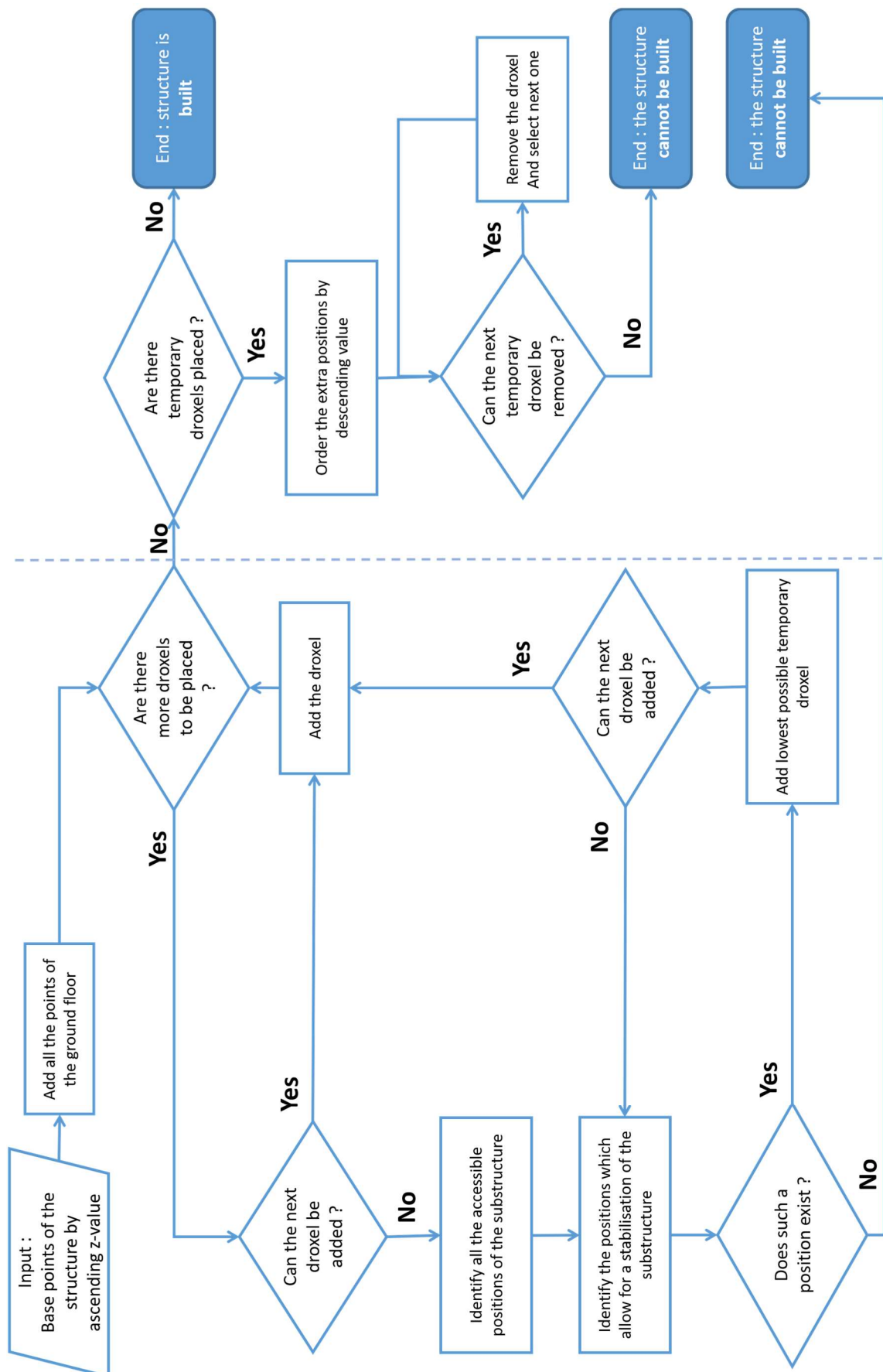
If it is not possible :

- Identification of the substructure corresponding to the droxel D_1 ;
- Identification of the accessible positions in the cone of influence of the substructure;

- Addition of the temporary droxels at the positions which allow to stabilise the substructure;
 - Addition of the droxel D_1 .
3. Repeating of step 2 until a solution to add all the desired droxels is found;
 - If no more temporary droxels allow for a stabilisation, the structure cannot be built.
 4. Removing of all the temporary droxels from top to bottom;
 - If they cannot be removed without causing an instability, the final structure cannot be built.
 5. The structure is built. The deconstruction of the structure is done by applying the inverse scheme.

Section 7.3 gives a flowchart of the complete implemented algorithm. The part on the left of the dotted line corresponds to the first five steps given above. The part on the right concerns the removal of the temporary droxels.

7.3 LAYING SEQUENCE ALGORITHM : FLOWCHART



7.4 ADVANTAGES, LIMITS AND IMPROVEMENT SUGGESTIONS OF THE ALGORITHM

7.4.1 Advantages

The solution found by the algorithm is not always the most effective one (especially for plane structures as shown in 7.5.2). However, the algorithm allows to find a possible way to build the structure if one exists. The main goal has been reached.

Another advantage of this algorithm is that it can spot the structures that cannot be built, even if their final disposition is stable. Thanks to the *Grasshopper* program, a visualisation is also possible allowing to see why a structure cannot be built.

Lastly, having found an algorithm to determine a laying sequence, the (more time consuming) general stability method mentioned in 6.3 does not need to be used. The stability check can just be verified for every individual droxel as they are virtually being added by the program. This allows to reduce the total computation time considerably.

7.4.2 Limits

The first limit is the fact that the laying sequence might not be the one that uses the less possible droxels (see 7.5.2).

The last limit of the laying sequence algorithm is the fact that the impact of droxel addition/removal is not taken into account. Both the *Grasshopper* algorithm and numerical simulations consider initially static systems where all the droxels are at the correct initial position. The possible settlement of some droxels in the structure before the addition of the next droxel is not taken into account. The effect of droxel removal (when removing temporary droxels f. e.) is also not taken into account. This can lead to incorrect results : if all the droxels are at the correct initial position, some structures might indeed be stable. In reality, the droxels are initially not at the correct position. This is not spotted when doing the numerical simulations but can be observed when doing the experimental tests. This limit is one of the limits of the *Grasshopper* algorithm but it is also a limit of the numerical simulations.

As mentioned in 2.3.9, droxels have a certain laying tolerance (see Figure 7.10). However, the program considers that a droxel is always placed exactly at its final position. The case where a droxel is dropped somewhere in the laying zone might have an influence on the stability of some structures and is not taken into account in *Grasshopper*.

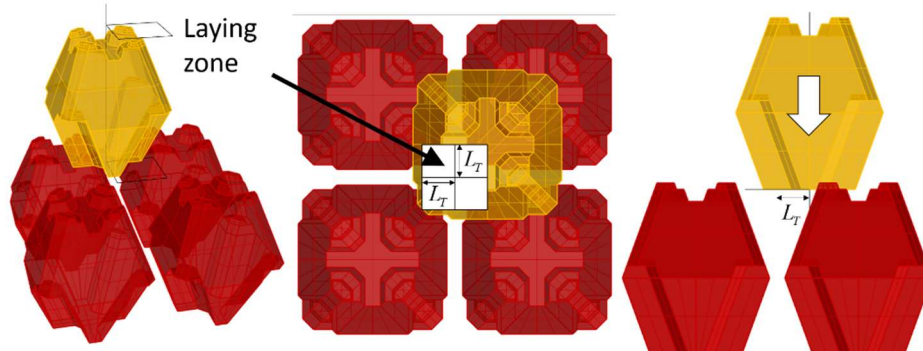


Figure 7.10 - Reminder : laying tolerance of droxels

It is possible that DEM will always be necessary in order to analyse the influence of the dropping zone of a droxel.

The other limits are the same as the first two mentioned in the general stability check (see 6.5) being:

1. Friction is not taken into account (only valid for $A^* \leq \phi$);
2. Hooked is considered as a binary value. In reality, a droxel can be hooked in a direction but unhooked in another.

7.4.3 Improvement suggestions

Apart from tackling the mentioned limits, following suggestions might help to have a more efficient algorithm. The algorithm works fine without this adjustment but it could be pertinent to implement them.

The laying sequence of the droxels placed on the same floor is at the moment not optimised. As proposed by S. GOESSENS [24], the DIJKSTRA's [49] shortest path algorithm could for example be implemented once the total laying sequence has been determined. This implementation would allow to reduce the total distance covered by a robot/builder.

When looking to stabilise the structure, the algorithm looks at the cone of influence of the whole corresponding substructure. However, looking at the cone of influence of the biggest unstable group might be more efficient: only the droxels with an influence on the stability of the group are then considered as possible stabilising candidates. To illustrate this, the actual algorithm is compared to the algorithm should the improvement suggestion be included. The studied structure is the one in Figure 7.11.

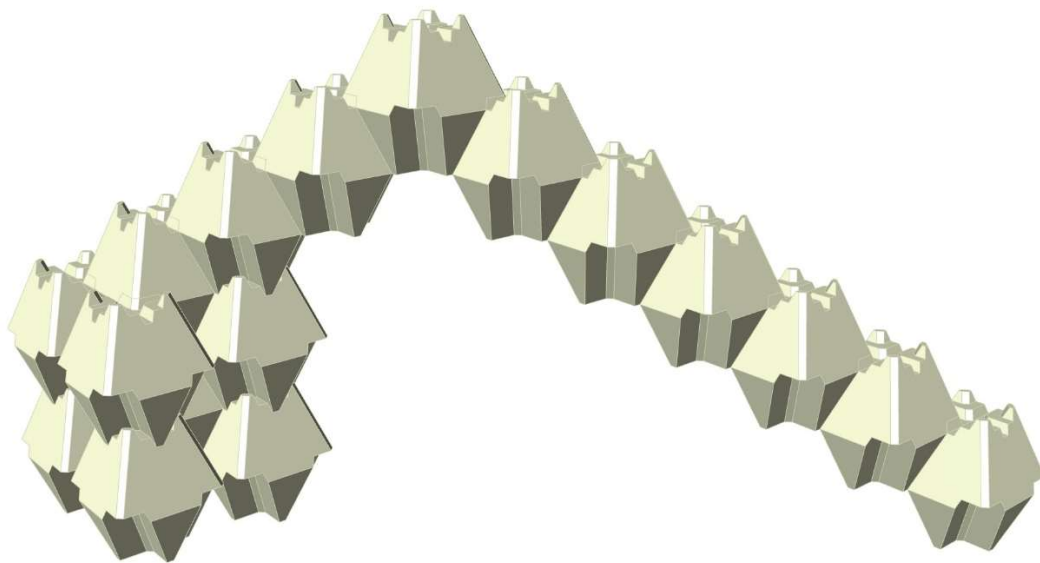


Figure 7.11 - Improvement possibility: example

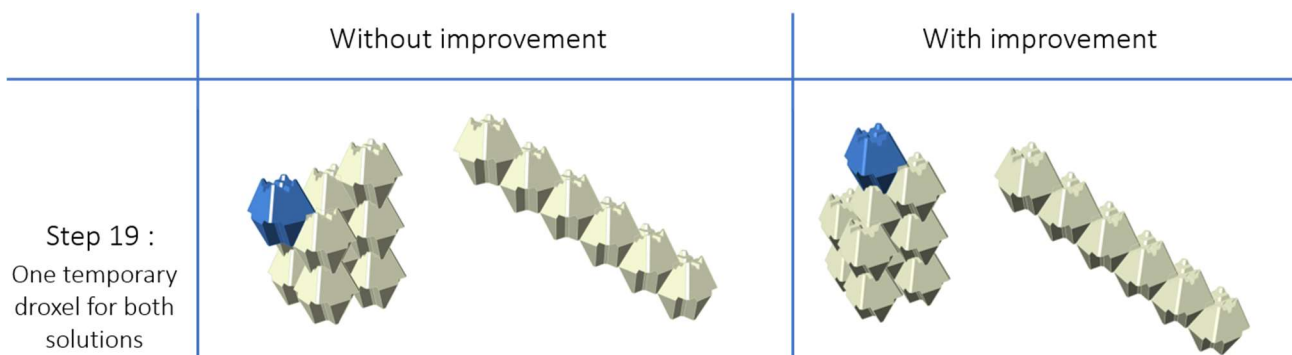


Figure 7.12 - Suggested improvement: example (part I)

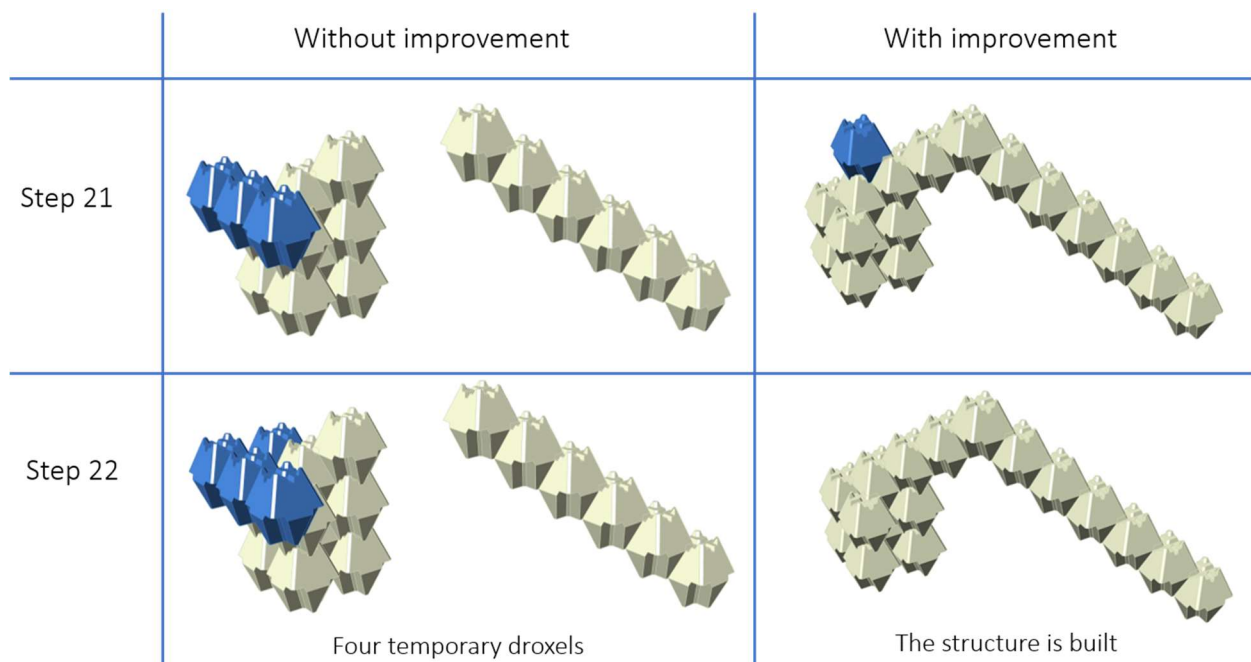


Figure 7.13- Suggested improvement: example (part II)

Whilst the implemented method without improvement needs 37 steps before the structure is built, implementing the improvement could reduce that number to 22. It is thus certainly worth looking into.

CONCLUSION

A laying sequence can now be found for a droxelised structure according to following principles :

1. Building floor by floor;
2. Placing every droxel possible without causing an instability;
3. When no more droxels can be placed without causing an instability, one temporary droxel at a time is added. They allow to stabilise the structure;
4. When the droxel can be placed, the algorithm moves to the next droxel;
If the droxel cannot be added, the structure cannot be built.

Once all the droxels are in place, the temporary droxels are removed, starting at the top. If they can all be removed without causing an instability, the structure can be built. Otherwise it cannot be built.

The limits and improvement suggestions for this algorithm have also been tackled. The limits for the laying sequence that can be mentioned are:

- The proposed solution might not be the “optimal” one;
- The dynamic effect of droxel removal/addition is not taken into account.
- The laying tolerance for the placement of the droxels is not taken into account

7.5 EXAMPLES

In order to show how the algorithm works, five examples are given next:

1. A trivial 2D structure;
2. A non-trivial 2D structure;
3. A non-trivial 3D structure;
4. A structure that cannot be built;
5. A structure showing the limits of the algorithm;
6. A structure showing the limits of the algorithm and of the numerical simulations.

More examples are given in APPENDIX A.

A comparison between the algorithm, the experimental tests and the numerical simulations is done in the next chapter.

7.5.1 Example 1: trivial 2D structure – straight wall

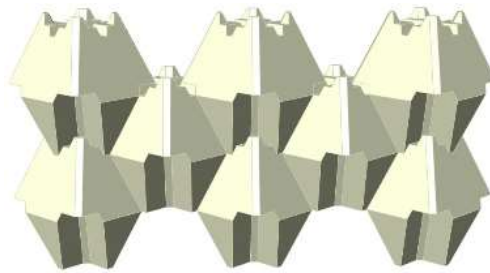


Figure 7.14 - Example of a droxelised wall

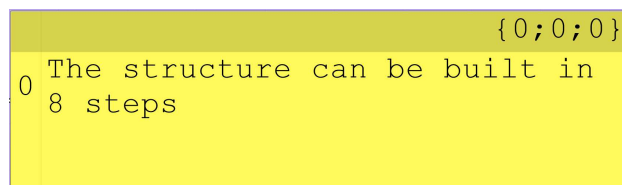


Figure 7.15 - Example 7.1: screenshot of the Grasshopper program

The *Grasshopper* program informs us that this structure of 8 droxels can be built in 8 steps.

Algorithm

The trivial laying sequence found by the program is given in Figure 7.16 and Figure 7.17:

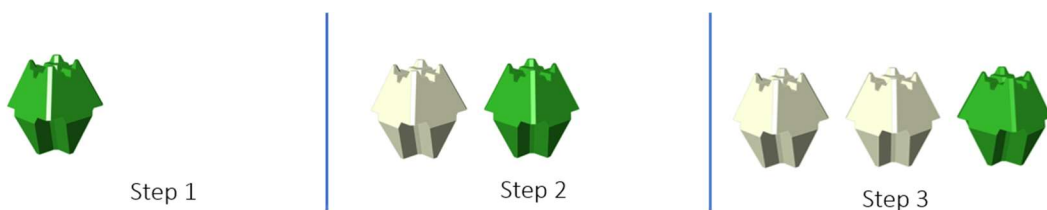


Figure 7.16 - Laying sequence for a droxelised wall. The added droxel at every step is represented in green (part I)

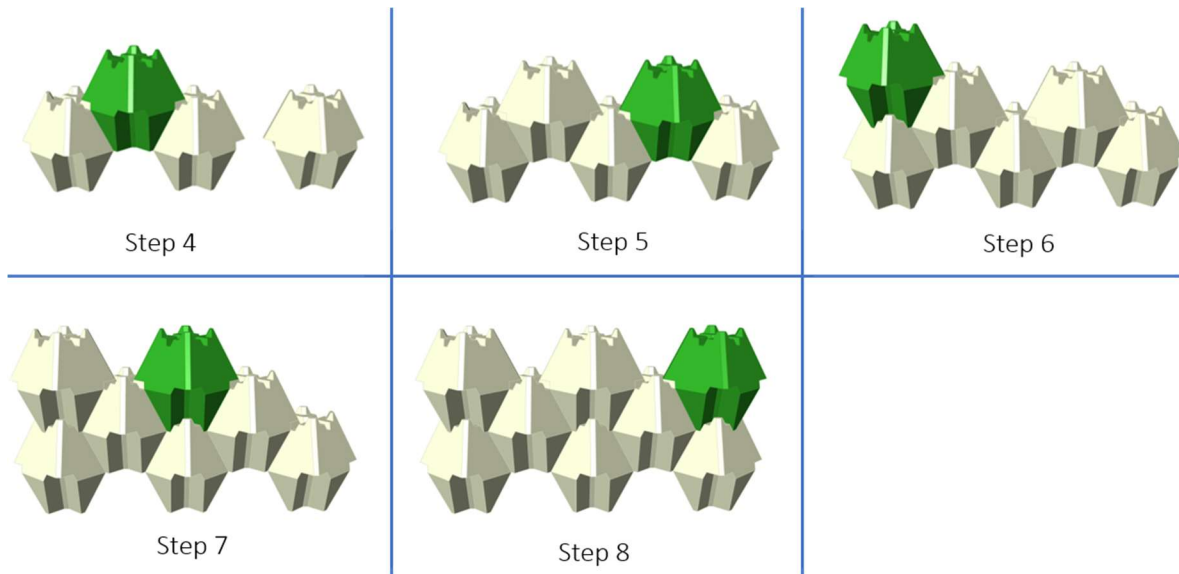


Figure 7.17 - Laying sequence for a droxelised wall. The added droxel at every step is represented in green (part II)

7.5.2 Example 2: non-trivial 2D structure – arch

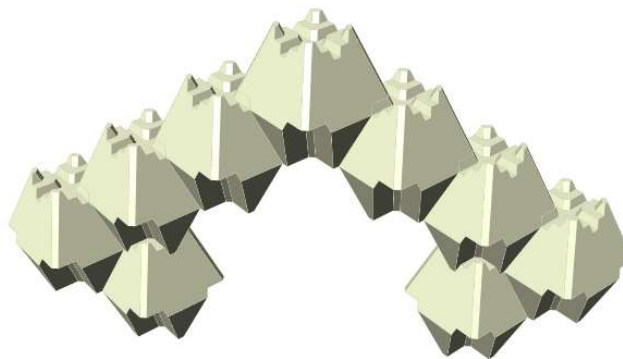


Figure 7.18 - Example of a droxelised arch

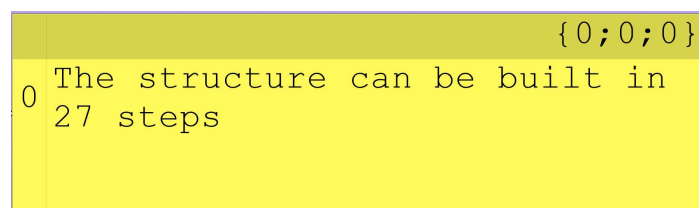


Figure 7.19 - Example 7.2: screenshot of the Grasshopper program

The *Grasshopper* program informs us that this structure of 9 droxels can be built in 27 steps.

The complete laying sequence is given from Figure 7.20 to Figure 7.22 : the last added droxel is represented in green, the temporary droxels are represented in blue. The steps after step 18 are the removal of the temporary droxels.

Since the arch is a plane structure, it can be seen that this solution is not the most efficient one. A solution requiring fewer droxels would be the one proposed in Figure 7.2 (p. 80).

However, since it is likely that most droxelised structures are not plane structures, a more general approach is implemented. It is entirely possible to add a verification checking whether a given structure is plane or not but that is a more specific case and is not covered here. This is also an improvement possibility that might be interesting to explore.

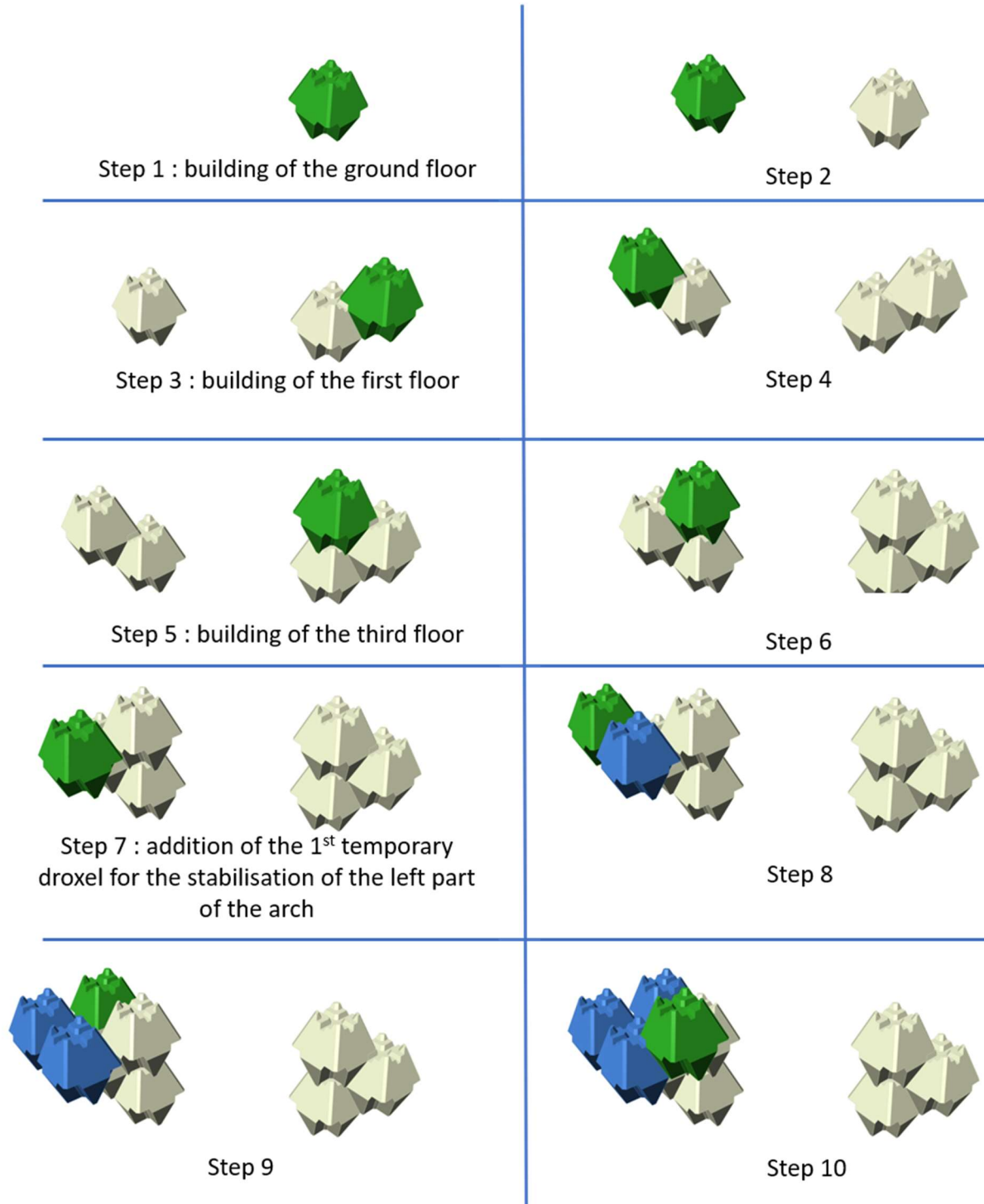


Figure 7.20 - Steps for the realisation of an arch. Last placed droxel in green, temporary droxel in blue. (PART I)

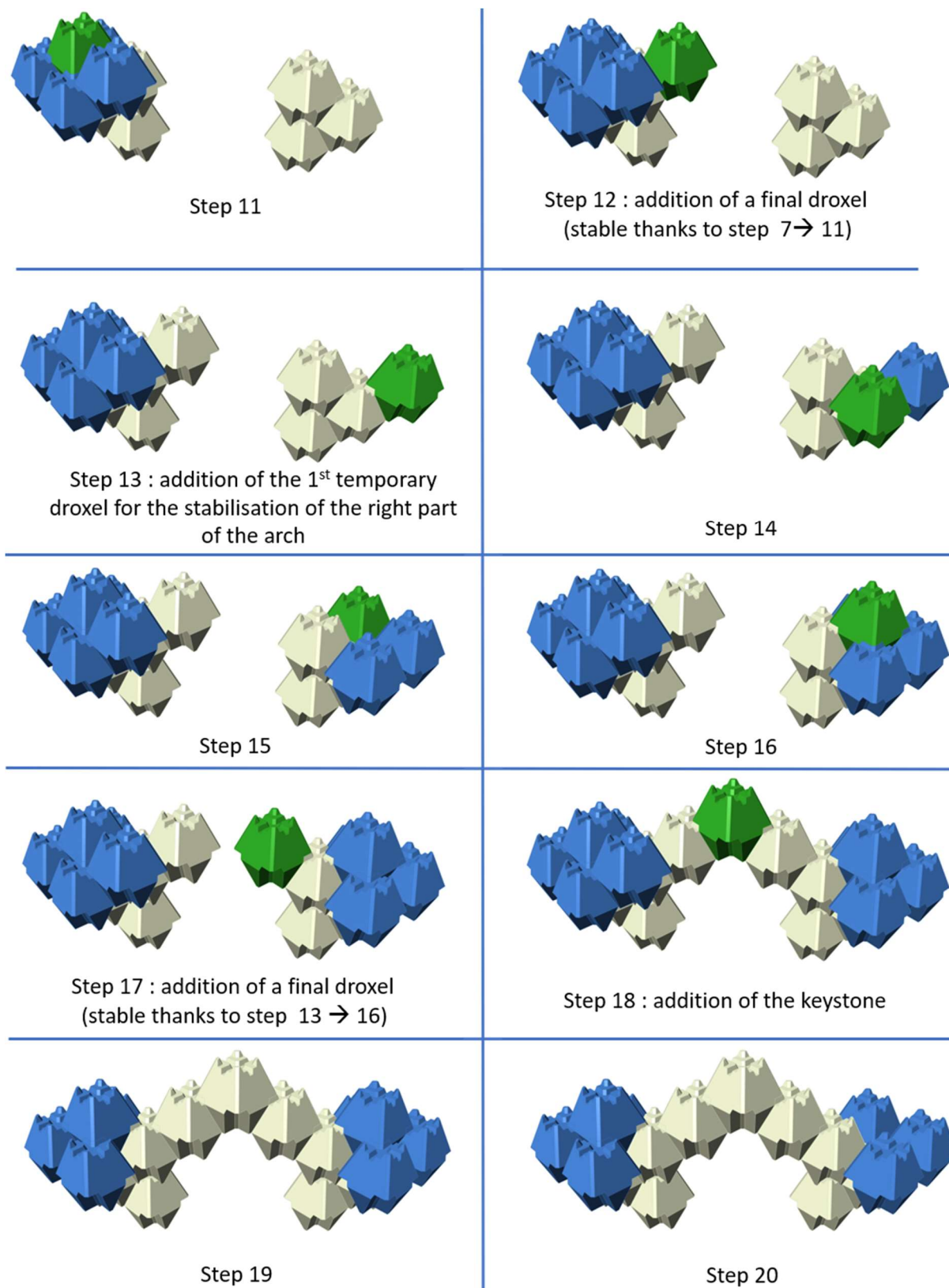


Figure 7.21 - Steps for the realisation of an arch. Placed droxel in green, temporary droxel in blue. (PART II)

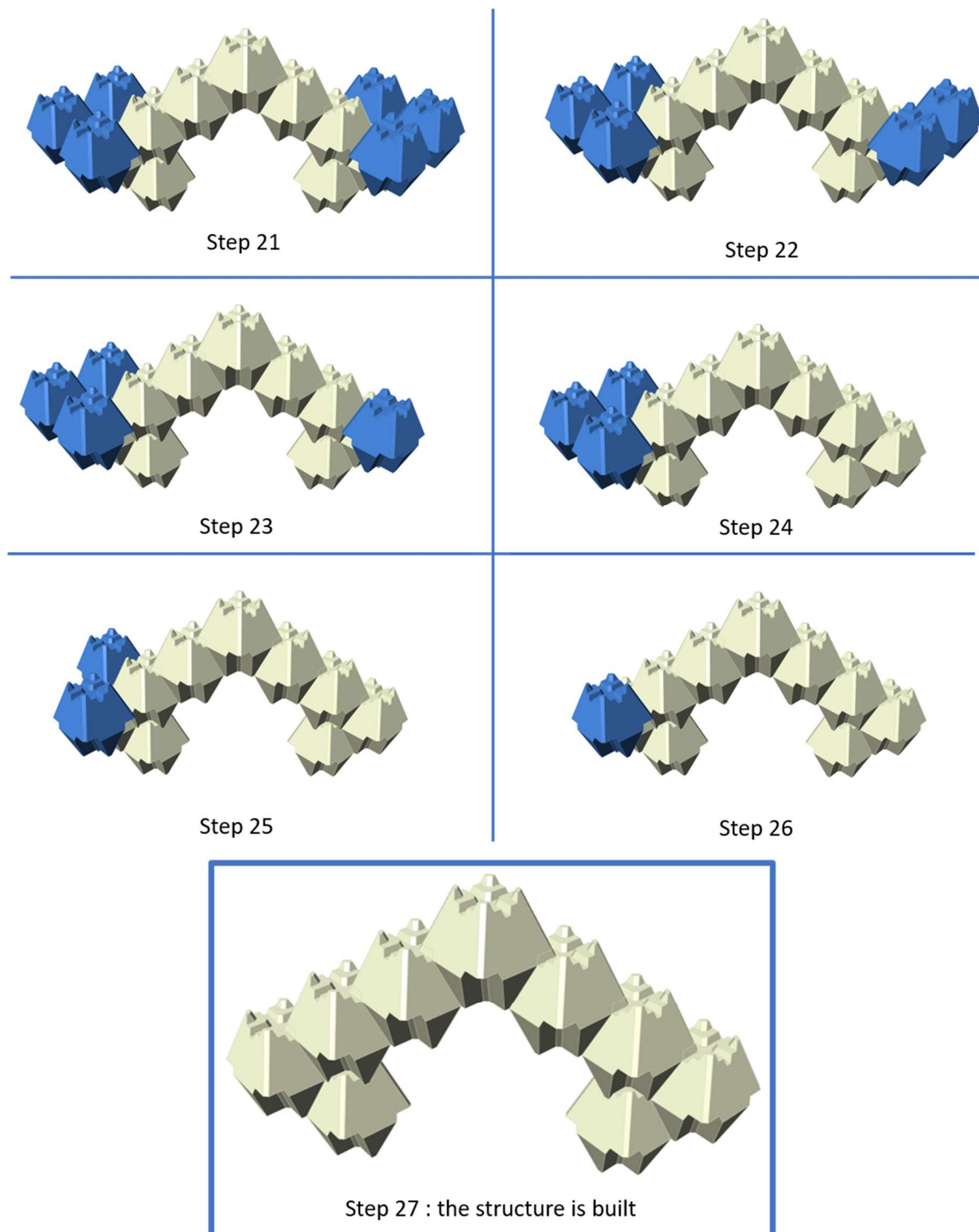


Figure 7.22 - Steps for the realisation of an arch. Placed droxel in green, temporary droxel in blue. (PART III)

7.5.3 Example 3: non-trivial 3D structure – small shelter

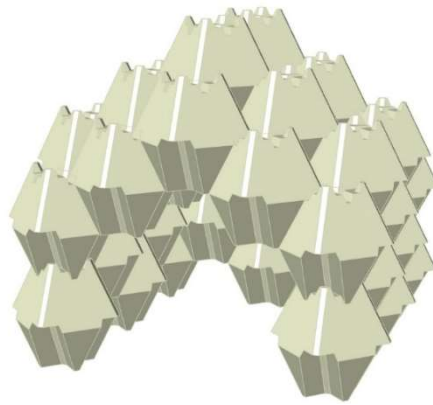


Figure 7.23 - Example of a small shelter

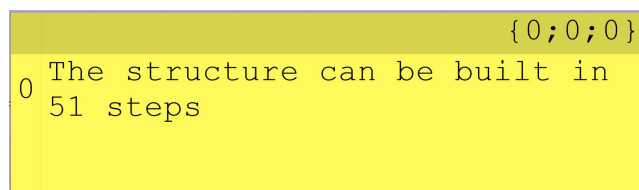


Figure 7.24 - Example 7.3: screenshot of the Grasshopper program

The *Grasshopper* program informs us that this structure of 39 droxels can be built in 51 steps.

The complete laying sequence of this structure is given in APPENDIX A. A shorter version with the important steps is given on the figure below. The steps after step 45 are the removal of the temporary (blue) droxels.

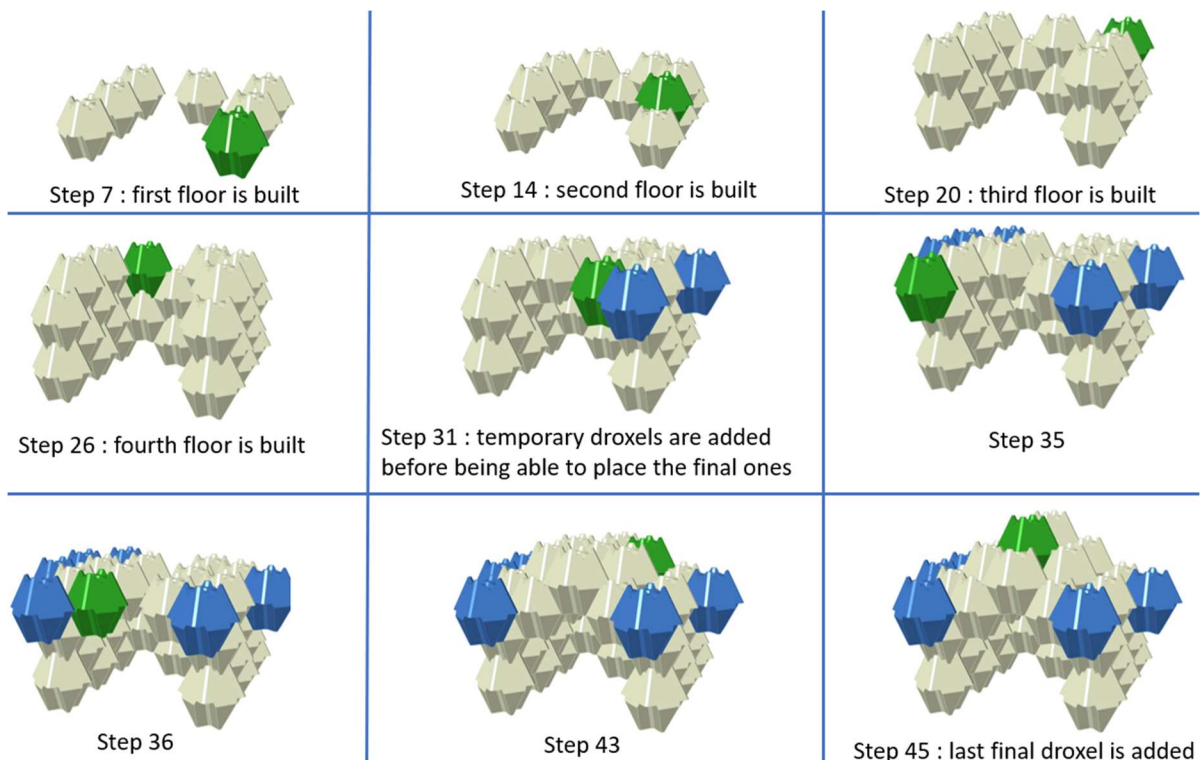


Figure 7.25 - Important steps for the building of the shelter. Placed droxel represented in green, temporary droxel in blue.

7.5.4 Example 4: stable final structure that cannot be build

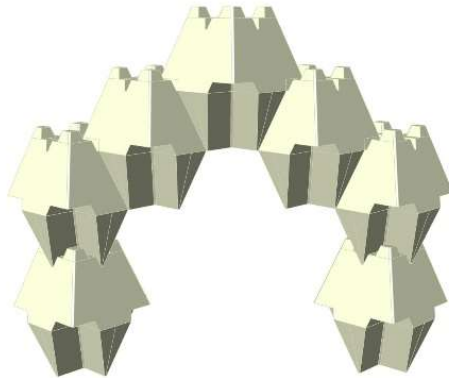


Figure 7.26 - Example of a stable final structure that cannot be built

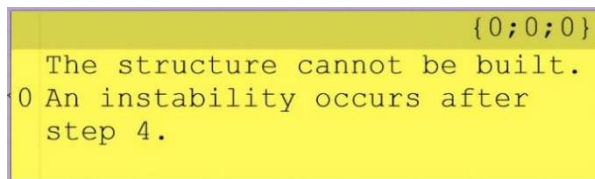


Figure 7.27 - Example 7.4: screenshot of the Grasshopper program

The *Grasshopper* program informs us that this structure of 7 droxels cannot be built. The first 4 steps are given on Figure 7.28.

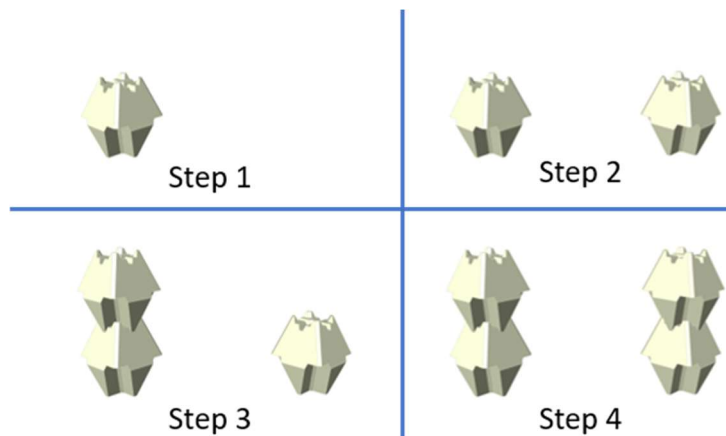


Figure 7.28 - Laying sequence of the structure in Figure 7.26

By manually trying to add a droxel to the structure, it can easily be seen that the structure can indeed not be built. No step is possible after the fourth step (Figure 7.29). This is also confirmed by numerical simulation as is shown in the next chapter.

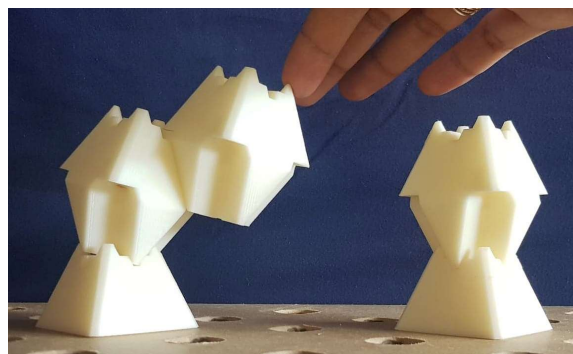


Figure 7.29 - Example 7.4: it is not possible to add a droxel after step 4

7.5.5 Example 5: showing a limit of the algorithm

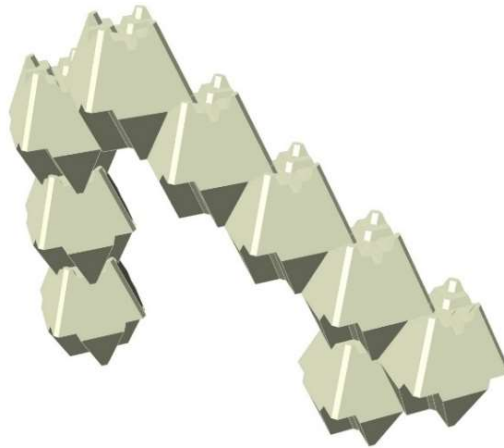


Figure 7.30 - Limits of the algorithm: example

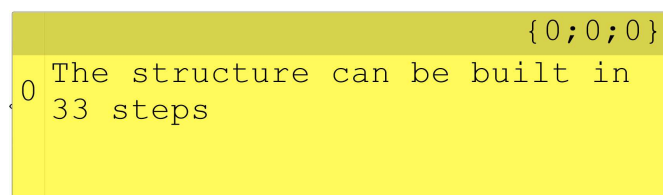


Figure 7.31 - Example 7.5 : screenshot of the Grasshopper program

The *Grasshopper* program informs us that this structure of 9 droxels can be built in 33 steps. The most important steps are given in Figure 7.32. The steps after step 21 are the removal of the temporary droxels.

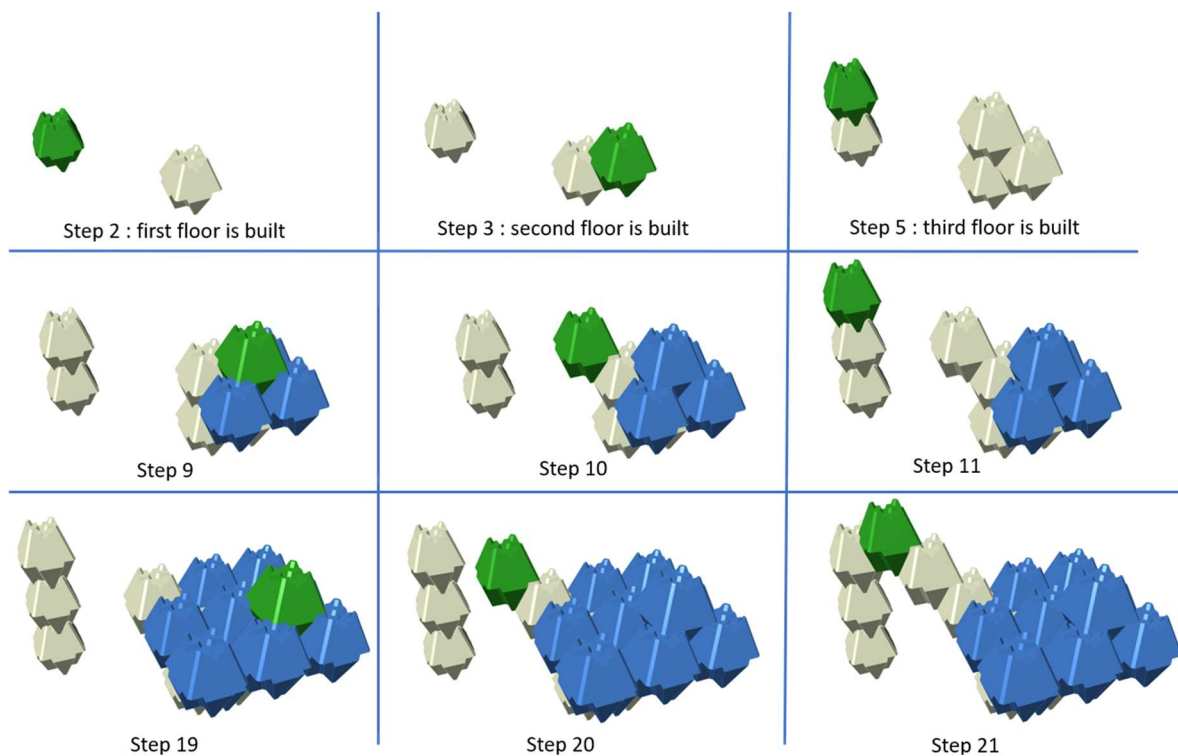
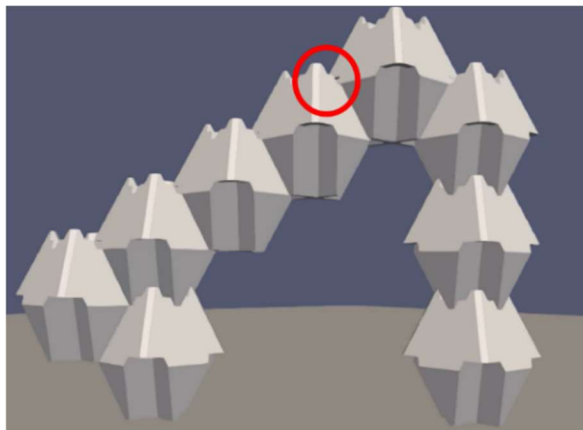
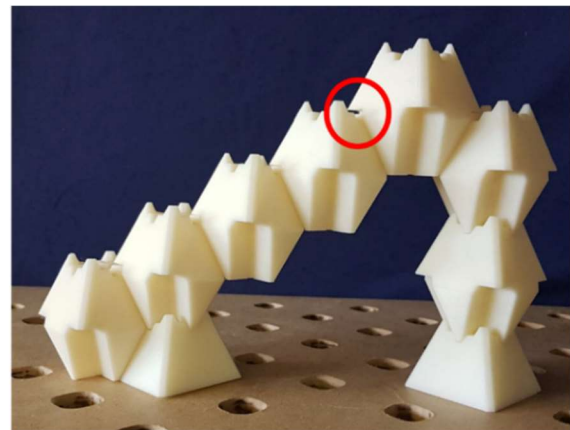


Figure 7.32 - Example 7.5: laying sequence



Simulation at $t = 45$ [sec]

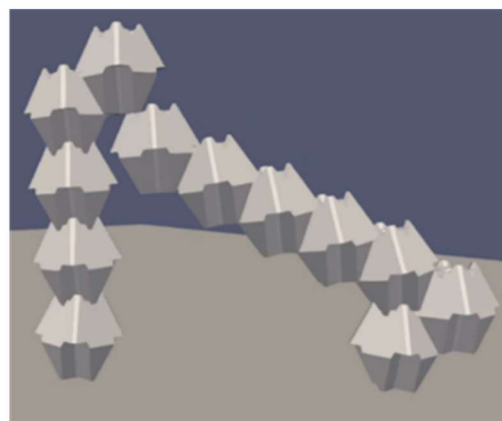


Experimental try

Figure 7.33 - Example 7.5: final structure simulation and experimental test

The last given example is one that illustrates the limits of the algorithm. Even though stable during the construction, the removal of the temporary droxels causes a slight movement of some droxels in the structure (see Figure 7.33).

Even though this small movement might not be problematic in this case, when trying to “scale up” the structure, friction is not sufficient to prevent a collapse of the structure. Even though the structure seems stable according to the *Grasshopper* algorithm, it is not in reality.



Simulation at $t = 45$ [sec]



Experimental try :
if not withheld, the structure collapses

Figure 7.34 - For a bigger similar structure, a total collapsing of the structure occurs

7.5.6 Example 6: Limitation of the algorithm: bridge-like structure

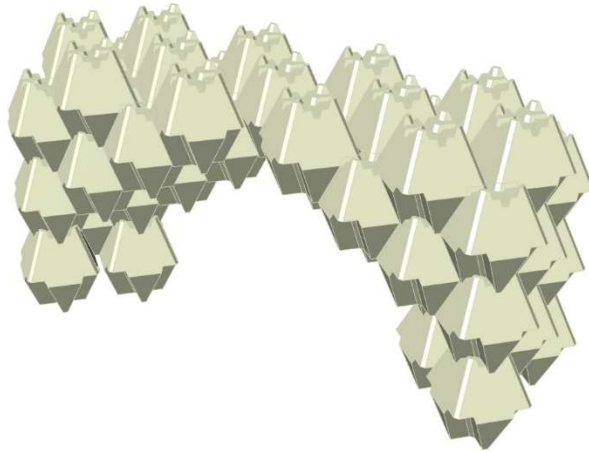


Figure 7.35 - Example 7.6: Grasshopper algorithm

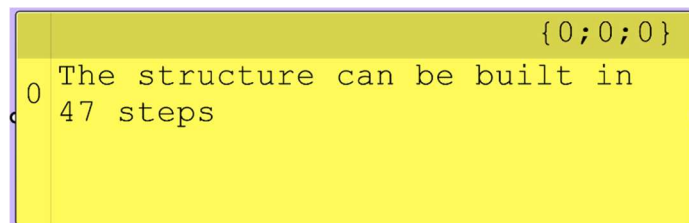


Figure 7.36 - Example 7.6 : screenshot of the Grasshopper program

The *Grasshopper* program informs us that this structure of 47 droxels can be built in 47 steps. For this last structure, only the last step, the addition of the central droxel creating the link between the two half-arches, is studied. The final structure, obtained after the addition of the last droxel, is stable according to the *Grasshopper* algorithm.

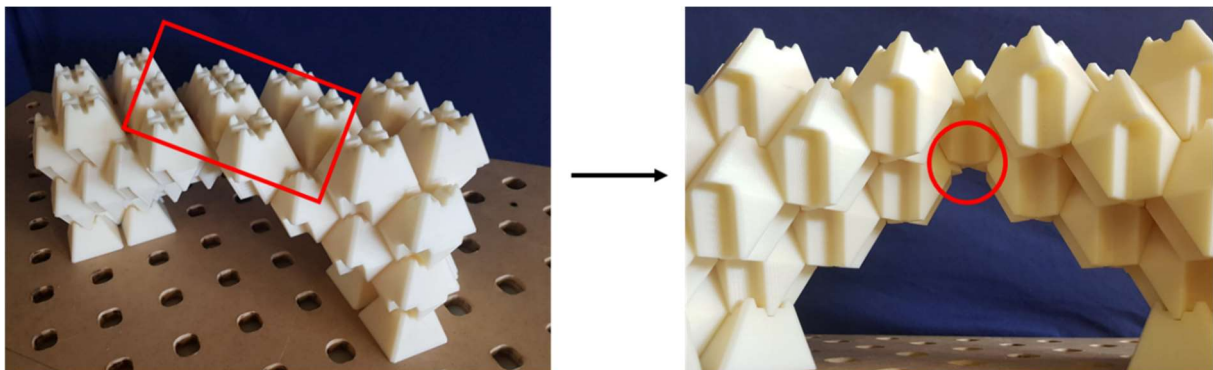


Figure 7.37 - Example 7.6: experimental test

It can be seen that due to the initial settlement of the two half-arches, the addition of the last droxel leads to a final structure where the last added droxel is higher than it should be (see Figure 7.37). Even though the 2 half-arch structures are stable without the keystone, they have a non-neglectable displacement making it difficult to insert the keystone correctly. After applying a vertical force on the keystone, the droxels of the structure all take their “correct” position.

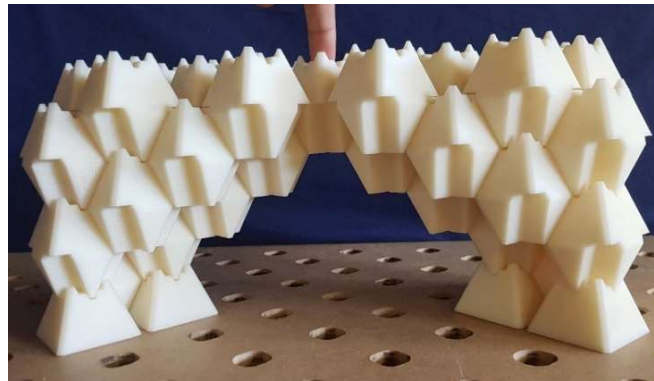


Figure 7.38 - Influence of a heavier keystone

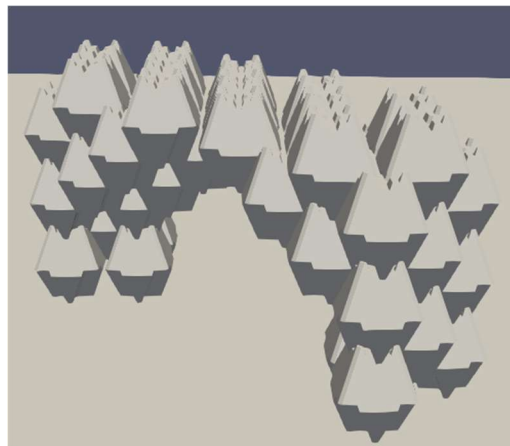


Figure 7.39 - Numerical simulation of the bridge after 45 [sec]

As discussed, the numerical simulation is not able to spot the mentioned limitation. In order to do that, the final positions of the simulation of the previous step should be saved. The last droxel should ideally be added to that deformed structure in the simulations.

The experimental verification of the algorithm and the numerical simulations clearly proves their utility and shows that the numerical model still has to be improved.

EXAMPLES : CONCLUSION

The tackled examples prove that the algorithm works fine within the given limits.

Examples with trivial and non-trivial laying sequences are given. An example of a structure that cannot be built as well as two examples showing the limits of the algorithm and the numerical simulations are also given.

CHAPTER 8 COMPARISON BETWEEN THE 3 METHODS : EXAMPLES

Now that the numerical simulation, the experimental verification and the laying sequence algorithm have been explained, some examples are developed in order to compare the algorithm and its limitations with the numerical simulations and experimental tests.

The simulations are all realised with 200 steps of $\Delta t = 0.15$ [s]. The total simulation time is 30 [sec]. The velocity of the droxels in the structure at the end of the simulations is however always verified : if it is virtually zero (< 1 mm/sec), the simulation is validated. Otherwise, the number of time steps is increased.

Due to the important computation times (see 4.1.3), not all the steps are simulated. Some situations (considered trivially stable) are therefore not represented in the following pages.

8.1 EXAMPLE 1: SIMPLE WALL

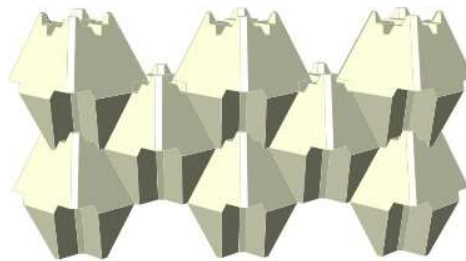


Figure 8.1 - Example 8.1: simple wall

Grasshopper algorithm

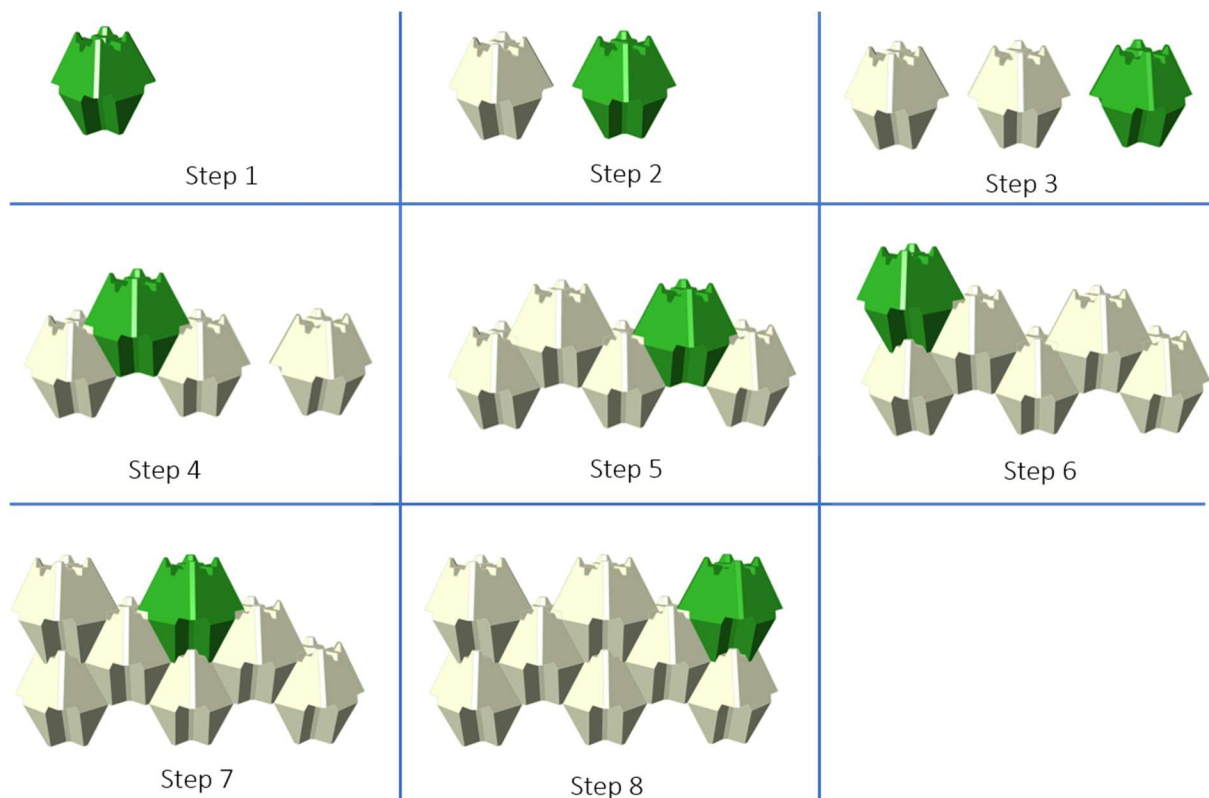


Figure 8.2 - Example 8.1: Laying sequence given by the Grasshopper algorithm

Numerical simulation: *LMGC90*

Due to the important computation times, only the last numerical simulation is performed for this wall because its stability during the construction is quite trivial.



Figure 8.3 - Example 8.1 : simulation of the wall after $t = 30$ [sec]

From now on, it is assumed that straight vertical walls are always stable.

Experimental tests

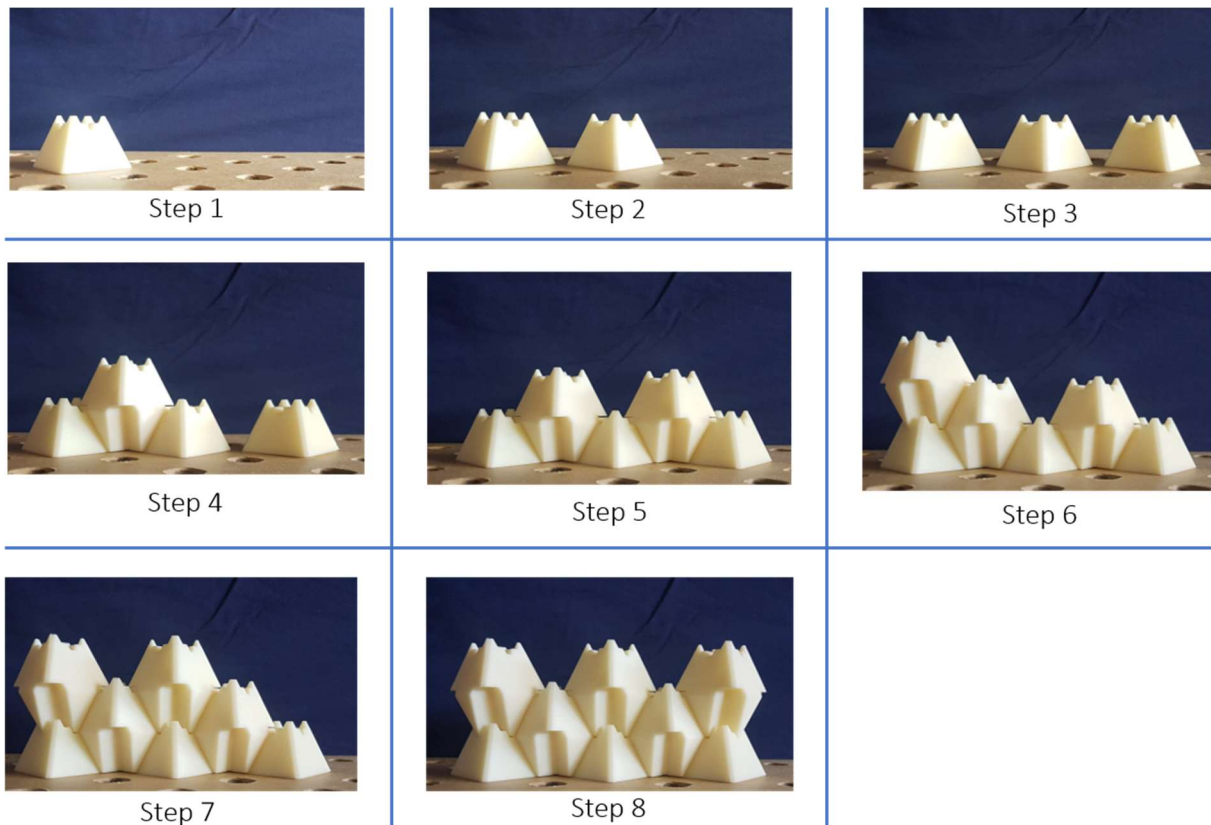


Figure 8.4 - Example 8.1: Laying sequence: experimental test

Conclusion

This first example shows that the Grasshopper algorithm works perfectly. This example is however quite trivial, which is why a non-trivial example is given next.

8.2 EXAMPLE 2: ARCH

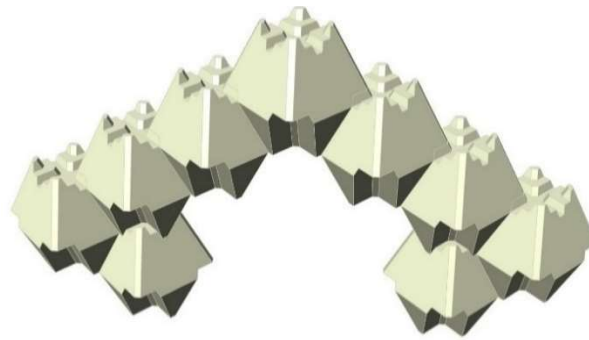


Figure 8.5 - Example 8.2: Arch

Comparison between the three methods

Step	Grasshopper code	Experimental	LMGC90 simulation (t = 30 [sec])
2.			Trivially stable case : droxels embedded in the ground surface No simulation
4.			Trivially stable case : lower droxels are embedded, no possible instability No simulation
6.			
7.			
8.			

Figure 8.6 - Example 8.2 (part I)

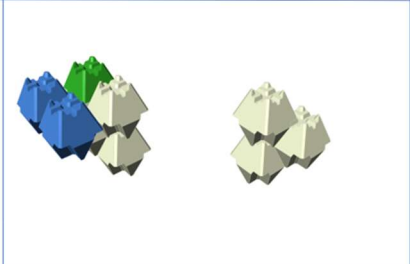
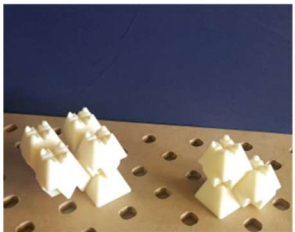
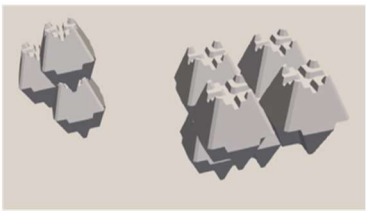
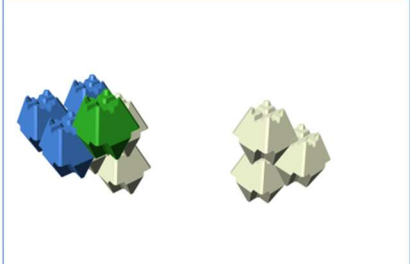
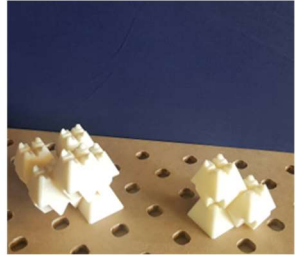
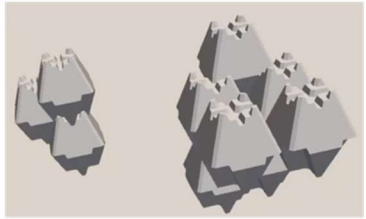
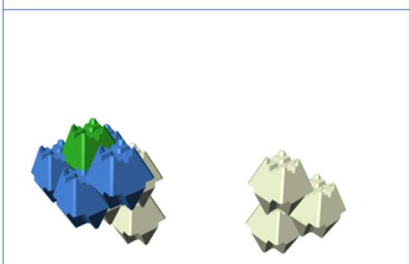
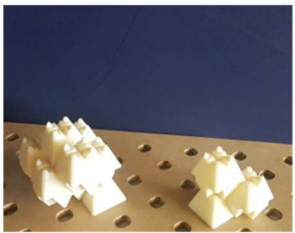
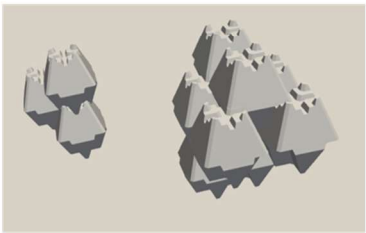
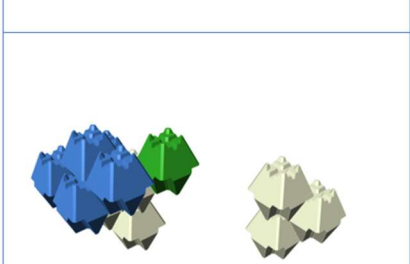
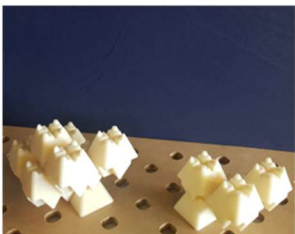
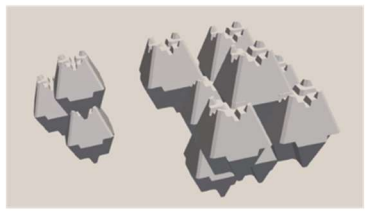
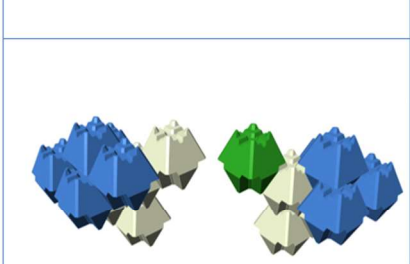
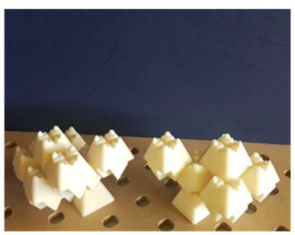
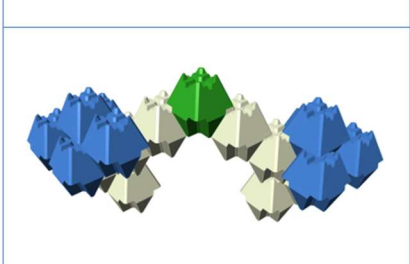
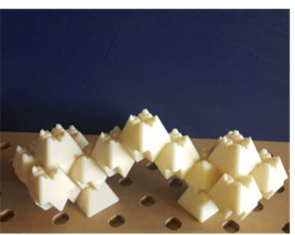
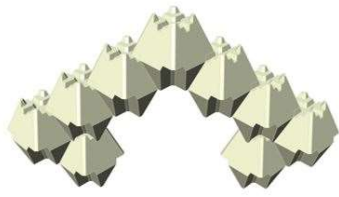
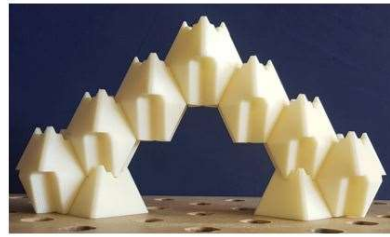
Step	Grasshopper code	Experimental	LMGC90 simulation ($t = 30$ [sec])
9.			
10.			
11.			
12.			
17			Trivially stable case : the two half arches are symmetrical and one has already been simulated at step 12. No simulation
18.			

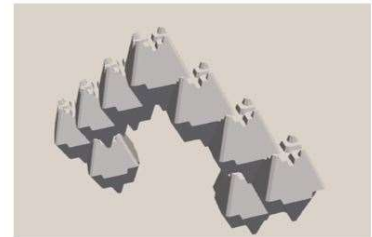
Figure 8.7 - Example 8.2 (part II)



Grasshopper prediction : stable



Experimental try : stable



Simulation at $t = 30$ [sec] : **stable**

Figure 8.8 - Example 8.2 : the three methods predict the same results

Conclusion

The three methods give a stable situation at every step, the *Grasshopper* algorithm and the numerical simulations seem to give the same results as the experimental tests.

8.3 EXAMPLE 3: SMALL SHELTER

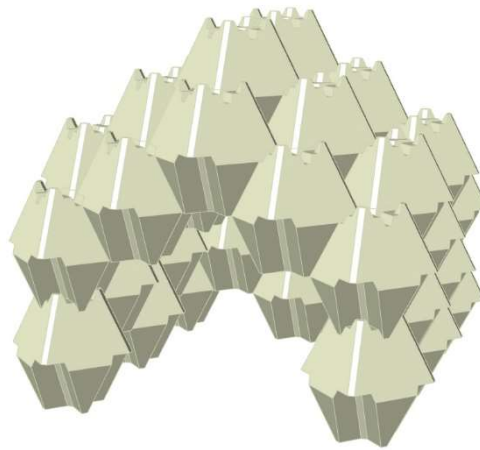


Figure 8.9 - Example 8.3: small shelter

Comparison

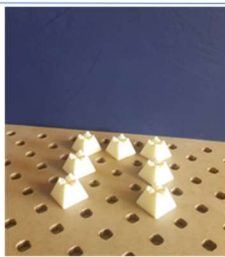
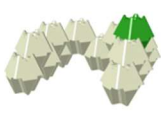
Step	Grasshopper code	Experimental	LMGC90 simulation ($t = 30$ [sec])
7.			Trivially stable case : the droxels of the ground floor are embedded No simulation
14.			

Figure 8.10 - Example 8.3 (part I)

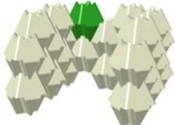
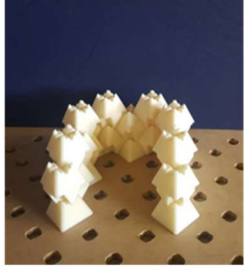
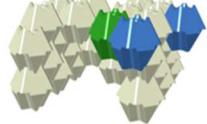
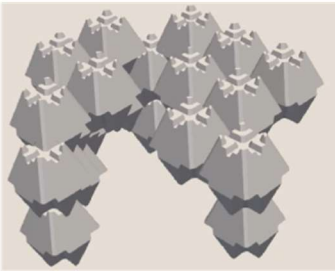
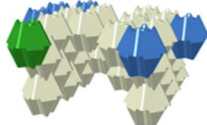
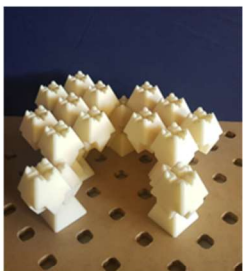
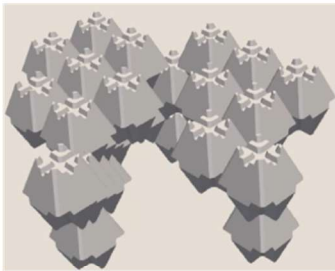
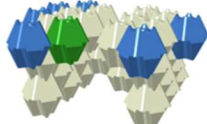
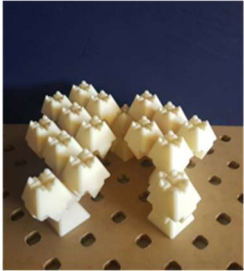
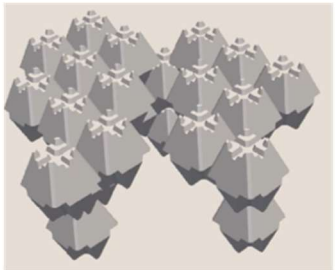
Step	Grasshopper code	Experimental	LMGC90 simulation ($t = 30$ [sec])
20.			Trivially stable case : See step 14. No simulation
26.			Trivially stable case : See step 14. No simulation
31.			
35.			
36.			

Figure 8.11 - Example 8.3 (part II)

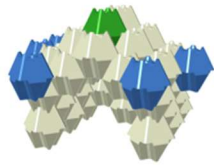
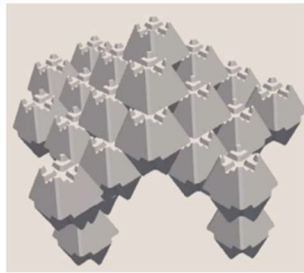
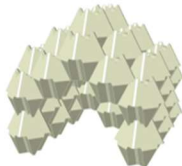
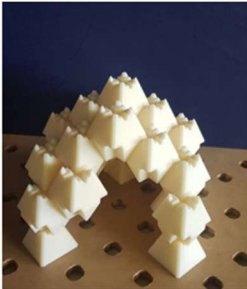
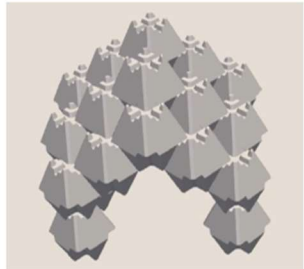
Step	Grasshopper code	Experimental	LMGC90 simulation (t = 30 [sec])
45.			
51.			

Figure 8.12 - Example 8.3 (part III)

Conclusion

The simulations for more complex structure quickly requires a lot of computational time (up to 48 hours for the last step). This being said, the results are very satisfactory since no difference was spotted between the three methods.

8.4 EXAMPLE 4: GATE (CANNOT BE BUILT)

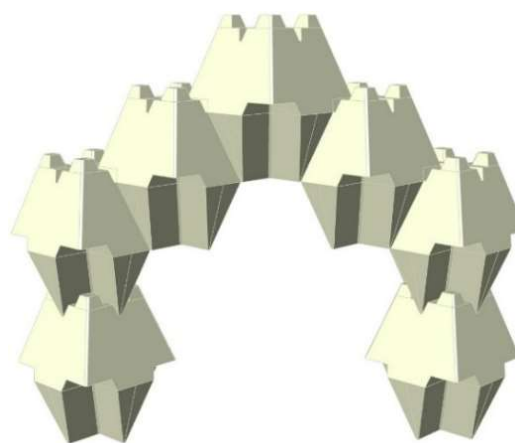


Figure 8.13 - Example 4: gate

The *Grasshopper* program directly gives a message indicating that the structure cannot be built (see Figure 8.14). It is however possible to see up until where a structure can be built. The program shows the first 4 instructions. As shown by numerical simulations and experimental tests, it is not possible to find a fifth step to build the structure.

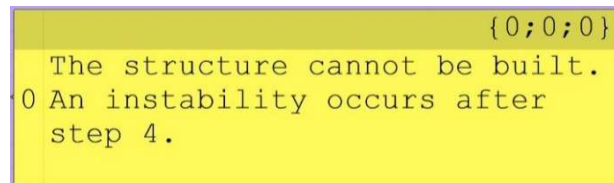


Figure 8.14 - Information message from Grasshopper when the gate is given as input structure

Comparison

Step	Grasshopper code	Experimental	LMGC90 simulation
1.			Trivially stable case : the droxel is embedded in the ground No simulation
2.			Trivially stable case : See step 1. No simulation
3.			Trivially stable case : a column of single droxels is always stable No simulation
4.			Trivially stable case : See step 3. No simulation
5.	No more possible steps found by Grasshopper		 Simulation at t = 6 [sec] : unstable

Figure 8.15 - Example 8.4

Conclusion

The fact that the first 4 steps are stable is trivial and is therefore not simulated. In order to verify that no more droxel can be placed, a numerical simulation of a possible fifth step is realised. Both experimental tests and numerical simulations prove that this cannot be done without causing an instability of the structure. The identification of structures that cannot be built seems to be working accurately.

8.5 EXAMPLE 5: LIMITS OF THE ALGORITHM - EXAMPLE

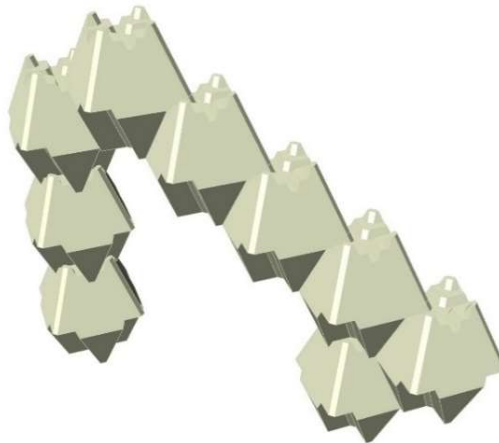


Figure 8.16 - Example 8.5: limits of the algorithm

Comparison

Step	Grasshopper code	Experimental	LMGC90 simulation (t = 30 [sec])
2.			Trivially stable case : the droxels are embedded into the ground No simulation
3.			Trivially stable case : lower droxel is embedded, no possible instability No simulation
5.			Trivially stable case : a column of single droxels is always stable under its self-weight No simulation

Figure 8.17 - Example 8.5 (part I)

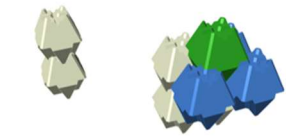
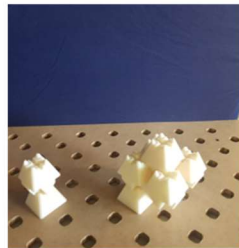
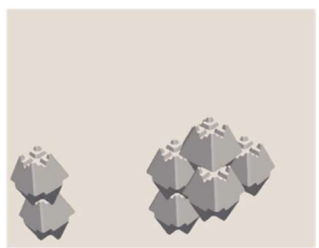
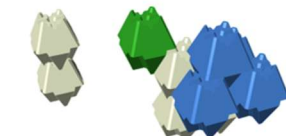
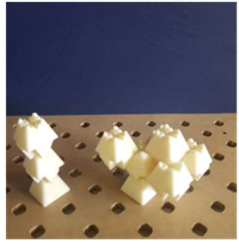
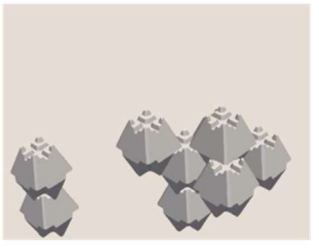
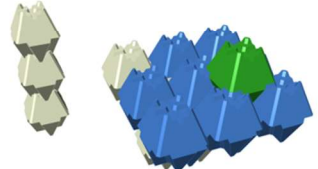
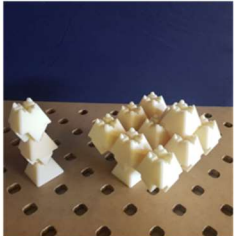
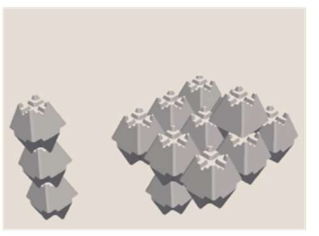
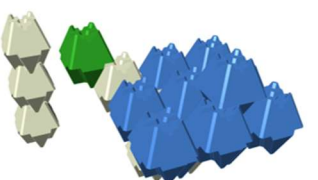
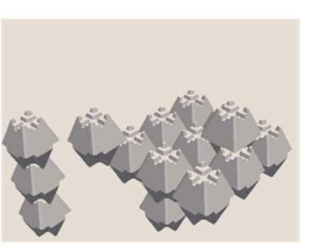
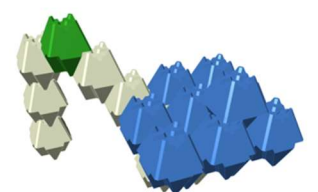
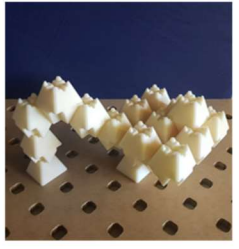
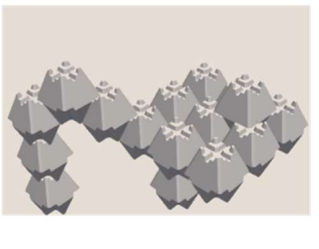
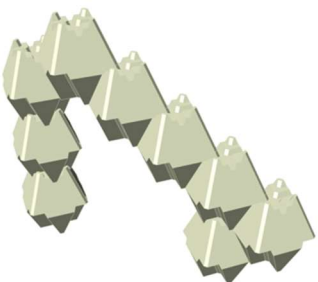
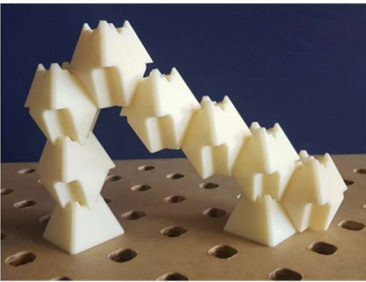
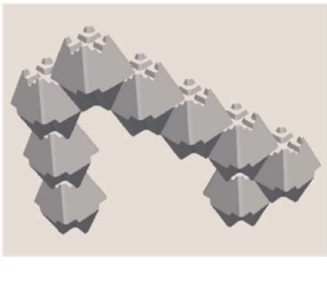
Step	Grasshopper code	Experimental	LMGC90 simulation (t = 30 [sec])
9.			
10.			
19.			
20.			
21.			
33.			

Figure 8.18 - Example 8.5 (part II)

Conclusion

The same comment as in section 7.5.5 can be made. The numerical simulations do not give an accurate result : the friction between the elements seems to be overestimated compared to the experimental tests. For the rest, the algorithm proves to be working and proposes a possible laying sequence.

CONCLUSION

All the tried examples proved to be working. Apart from the limitations, already mentioned in section 6.5 which are mainly due to the neglecting of the friction between the droxels, no new limitation was encountered when confronting the algorithm with experimental tests and numerical simulations.

CONCLUSION

The comparison between the *Grasshopper* algorithm, the experimental tests and the numerical simulations is what concludes this master thesis.

Conclusion of the master thesis

Three goals were set for this thesis :

1. Implementing the stability algorithm for 3D structures;
2. Implementing a laying sequence algorithm;
3. Comparing the implemented algorithms with numerical simulations and experimental tests.

These objectives have been met by implementing *C#* codes into a *Grasshopper* interface. The implemented codes, allow to:

- (a) Define a certain *droxel geometry* (depends on 8 different parameters) – CHAPTER 2
 - (b) Choose one out of the two *droxelisation* methods – CHAPTER 5
 - (c) Determine whether a given structure is *globally stable* or not – CHAPTER 6
 - (d) Determine the *laying sequence* of a given structure – CHAPTER 7
- If no laying sequence can be found, it is indicated by an output message on the interface. The message gives information on whether the structure is unstable or if it cannot be built.

Stability algorithm

The stability algorithm had already been implemented geometrically. In this thesis, this algorithm has been implemented analytically which allows to considerably reduce the computation time.

The main limits of the stability algorithm are due to the fact that the friction is not taken into account. This means that the instabilities due to the sliding between the droxels are not spotted. More research is needed to determine whether this feature can be included directly into the algorithm by finding some behavioural laws for droxels or if *Discrete Element Methods (DEM)* will always be necessary to verify this.

A second limit that has been identified is the fact that a droxel is either considered *hooked* or *unhooked*. In reality, a droxel can be hooked in 1, 2, 4 or no direction at all.

Laying sequence algorithm

The laying sequence algorithm was implemented by relying on following main principles :

- The structure is built floor by floor;
- If the addition of the next droxel causes an instability, find a way to stabilise the structure;
 - If it is possible, move to the next droxel;
 - If it is not possible, the structure cannot be built.
- The temporary droxels are all removed at the end of (and not during) the construction.

Comparison

After implementing both algorithms, a [comparison](#) between the *Grasshopper* algorithm, *DEM* and *experimental tests* is done. Apart from the previously mentioned limitations, the conclusions of this comparison are positive. The *Grasshopper* algorithm proves to match both experimental and numerical simulations.

Prospects

Several improvements are possible (and some even necessary) before being able to realise real scale droxelised structures. The possible improvements for future works are split in 3 categories: improvement for the droxels, for the *Grasshopper* algorithm and for the numerical simulations.

Future works: droxels

As shown in CHAPTER 2, one of the disadvantages of droxel structures is the fact that there are openings in them (see Figure 6). It could be possible to imagine a slightly different new geometry allowing to reduce these openings.

However, before looking into that it might be interesting to try to define the scope of use of droxels. If droxels are only used for temporary structures or structures that do not need to have an important isolation, the openings might not be a problem: a covering/isolation system can easily be found. If however droxels are considered for permanent structures needing important isolation it might be relevant to review their geometry.

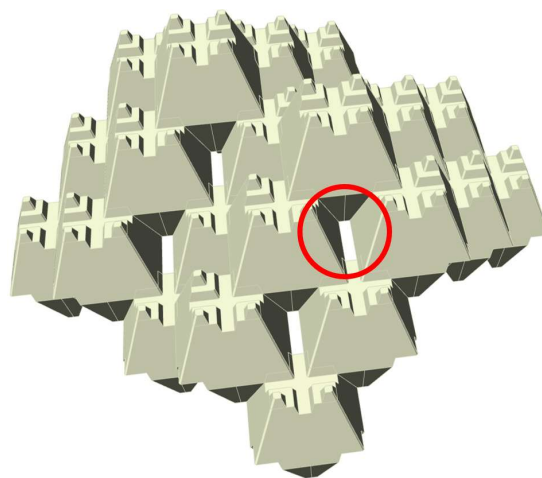


Figure 6 - Openings in the structure: proportional to E

Currently, the production method considered for the construction of droxels is plastic injection (for plastic droxels) or moulding (for concrete droxels). The major disadvantage of plastic injection is the fact that the moulding is very expensive. The thermal effects have to be taken into account in order to avoid warping or local defaults on the droxels' surfaces. On the other hand, plastic droxels can be filled with ballast later on, are lighter and are therefore more easily transported than concrete droxels. This is an advantage of the expensive plastic droxels over the cheaper concrete ones.

A study on which technology to use is therefore needed before being able to think about the commercialisation of the droxel. Defining the scope of use of the droxels should already be a great

help in order to choose one out of the two production methods. Maybe other options will also be found and will prove to be more pertinent.

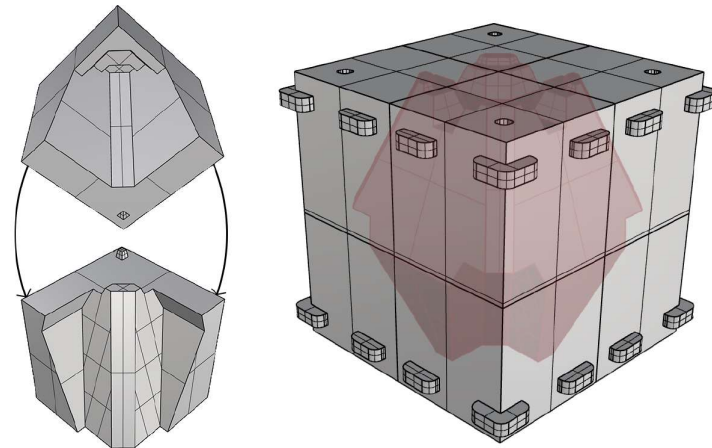


Figure 7 - Moulding of a droxel [40]

Future works: stability algorithm

At the moment, either a 3D surface is inputted in the program or the base points of the droxels are added manually. It could certainly be interesting to implement a third method. For instance by representing the possible base points of the droxels and selecting them with the computer mouse. This implementation would assure the most user friendly droxelisation method.

A feature that must absolutely be implemented into the *Grasshopper* program is the friction. How to do this is not yet very clear. Either empirical laws can be determined allowing to add the friction into the algorithm. Another option is to implement the *DEM* directly into the *Grasshopper* code. Finding empirical laws covering all the possible cases does not seem very effective. This is why I believe that finding a way to implement *DEM* into *Grasshopper* could be a relevant research subject.

During this whole master thesis, it was assumed that only one droxel geometry was possible for a given structure. It is however possible to introduce slope changing droxels [35]. The algorithm could thus be modified in order to, if required, allow for the use of different droxel geometries in one structure.

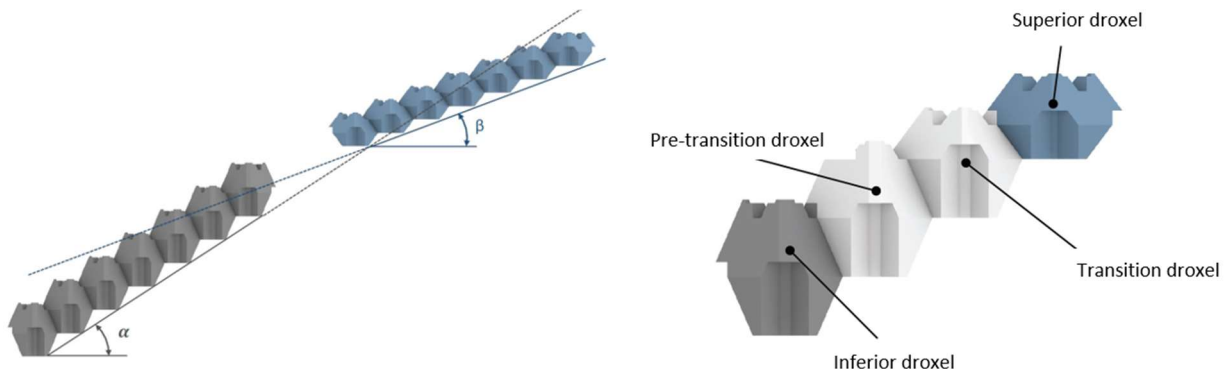


Figure 8 - Slope change in droxelled structures [26]

Future works: numerical simulations

As mentioned, more experiments have to be realised in order to validate the used interaction model, as well as the friction coefficient. Ideally, the calculation will be optimized needing less computation time (for example by implementing a specific contact detection method). The computation time might

also be improved by implementing multithreaded⁵ calculations. This feature is however not supported by *Windows* for *LMGC90*. It is necessary to switch to *Linux*.

Once the computation time has been significantly reduced, the implementation of the numerical simulations directly into *Grasshopper* would be an undeniable asset. It would directly be possible for the program to work autonomously and to launch simulations of the steps which are deemed to need a *DEM* analysis. For the moment, the structures have to be inputted manually in the *DEM* analysis. It is of course feasible but it would be more comfortable if it were done automatically by the program.

Last words

With this recommendations, hopefully useful for the next persons who will work on droxels, I am ending my master thesis. I certainly believe that the full potential of droxels is yet to be discovered and hope that this subject will raise the curiosity of more people in the future.

⁵ “Multi-threading is a process that contains multiple threads within a single process. [...] the major advantage of multithreading is it works simultaneously, which means multiple tasks execute at the same time [...] maximizing the utilization of the CPU [...]” [14]

Appendices

APPENDIX A: LAYING SEQUENCES - FULL EXAMPLES

A.1 ARCH

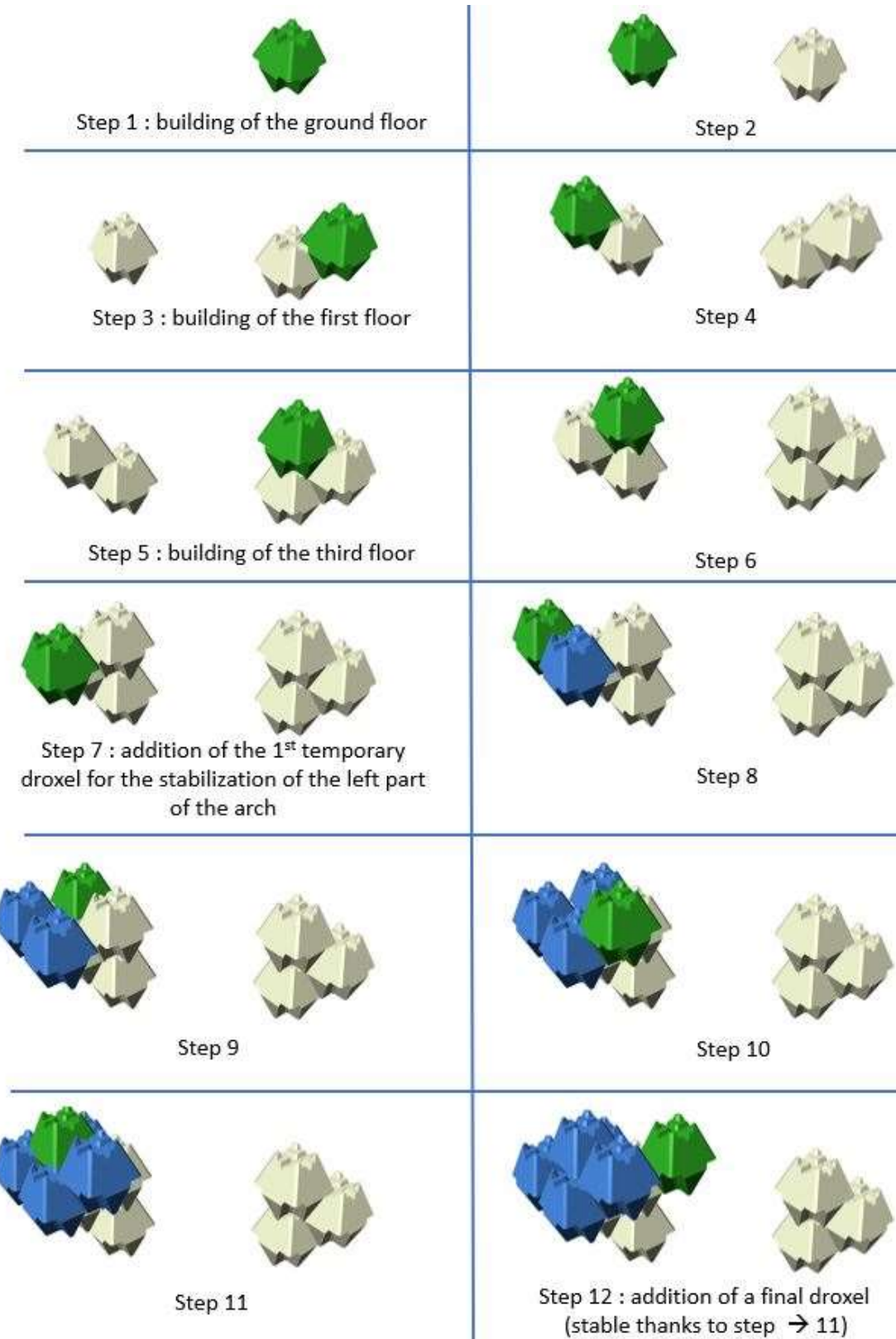


Figure A.1 - Example 2 (arch): Laying sequence (part I)

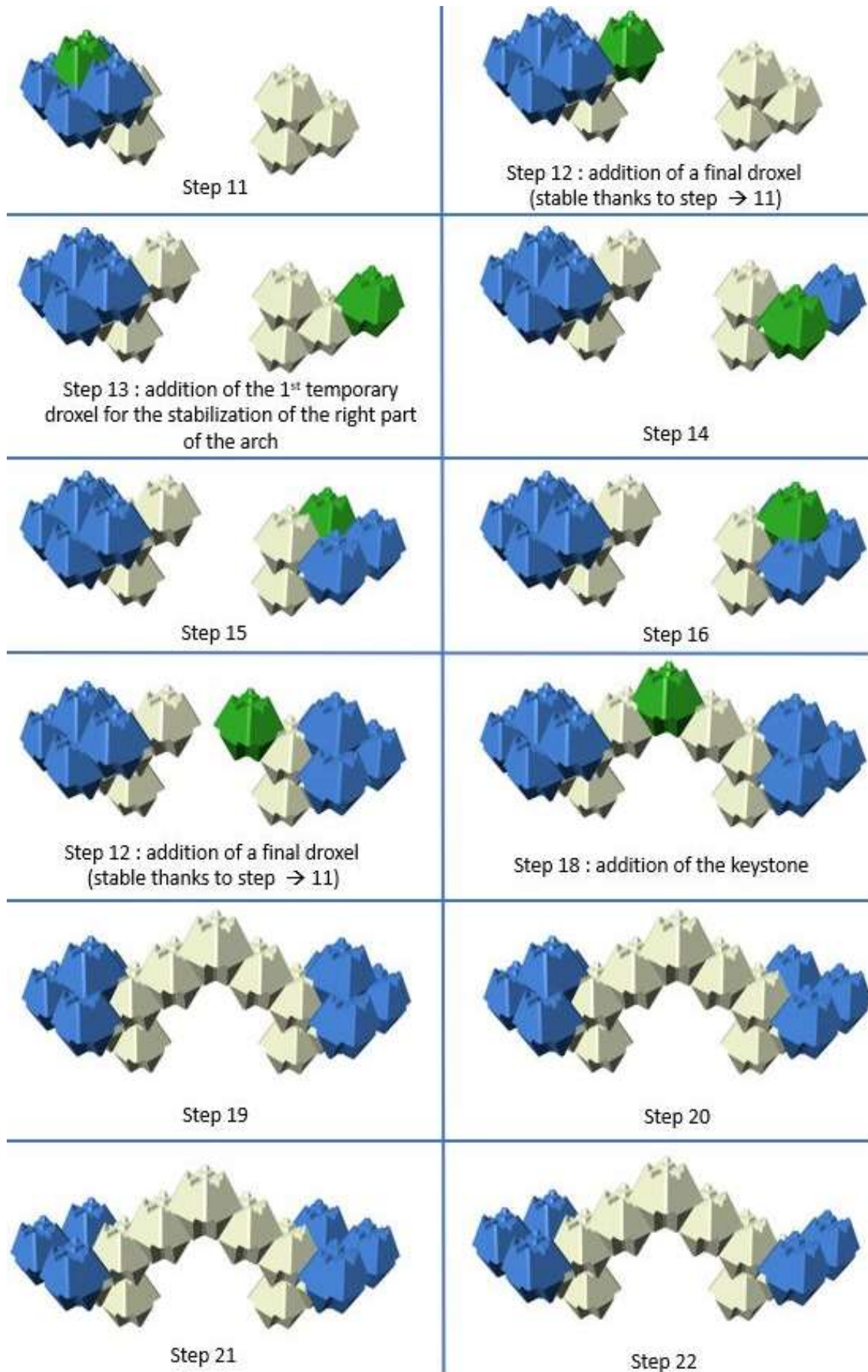


Figure A.2 - Example 2 (arch): Laying sequence (part II)

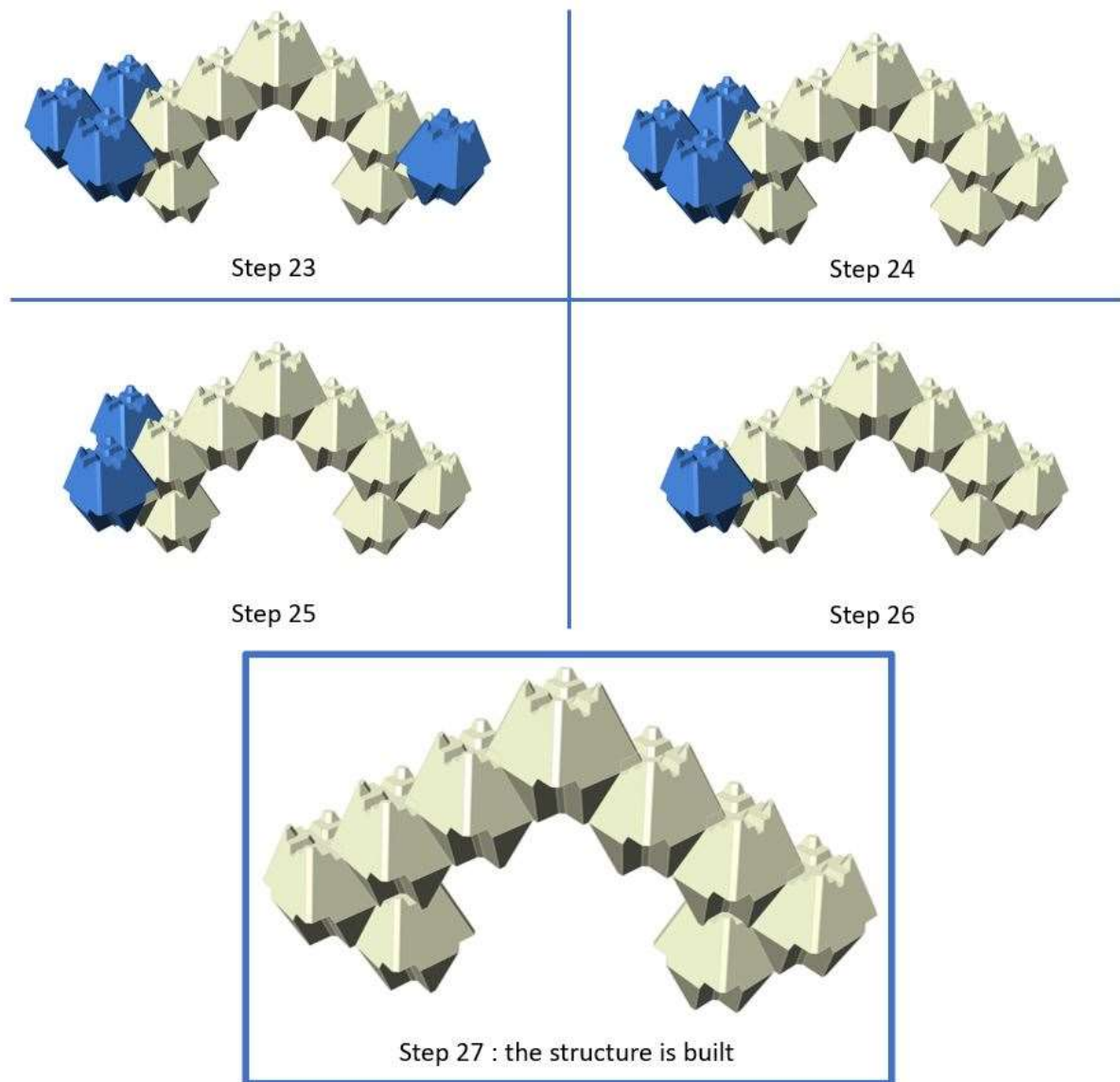


Figure A.3 - Example 2 (arch): Laying sequence (part III)

A.2 SMALL SHELTER

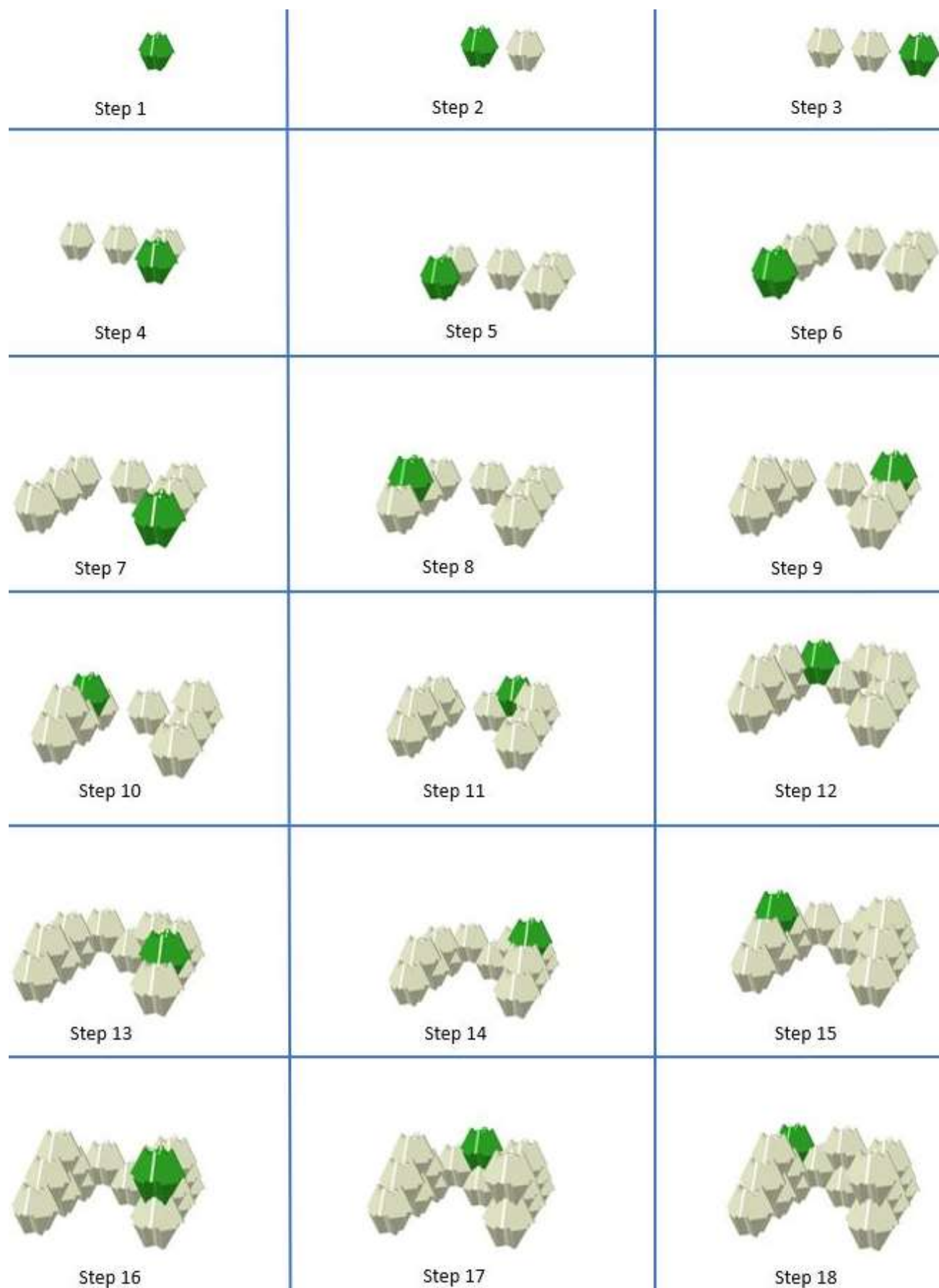


Figure A.4 - Example 3 (small shelter): Laying sequence (part I)

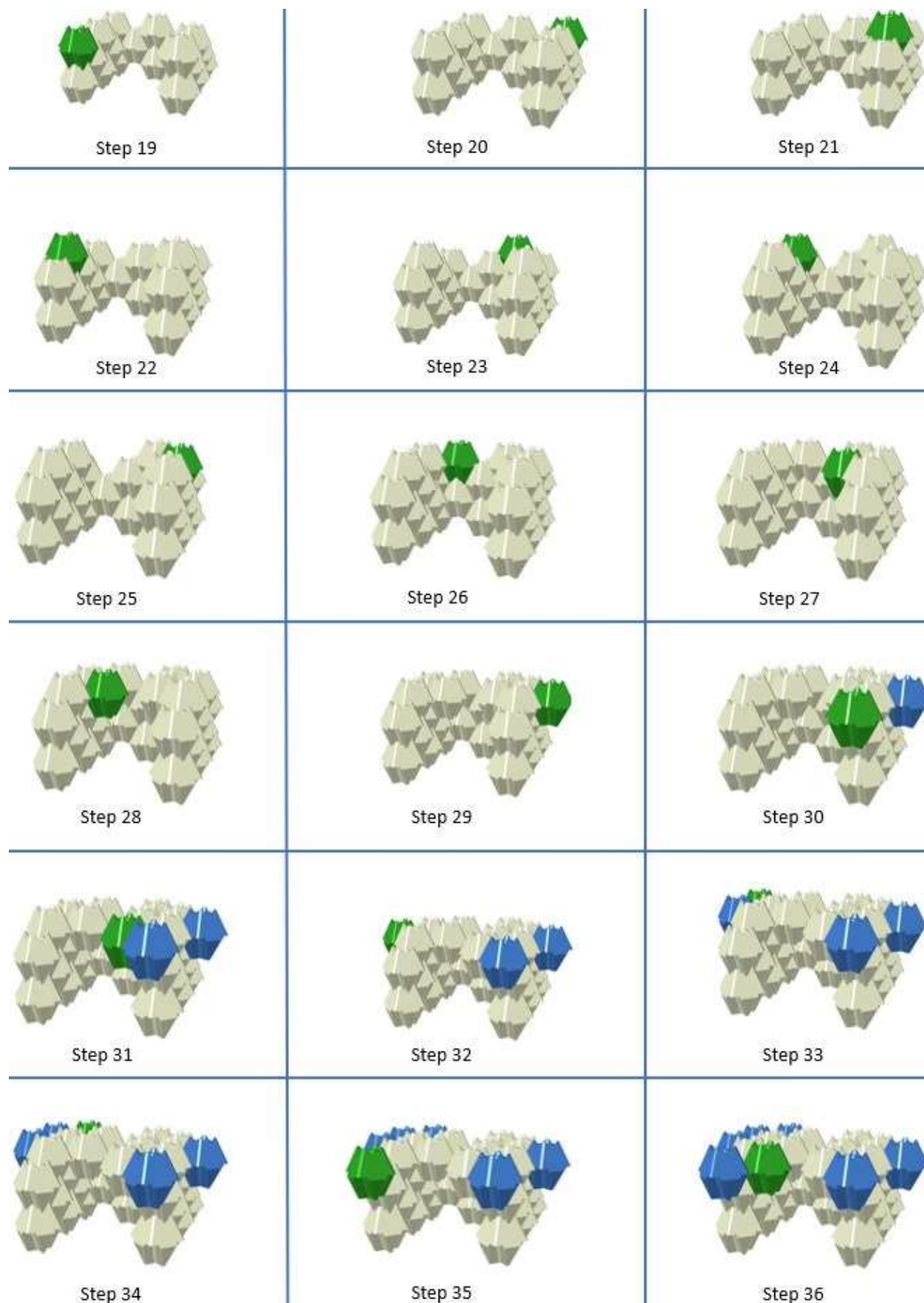


Figure A.5 - Example 3 (small shelter): Laying sequence (part II)

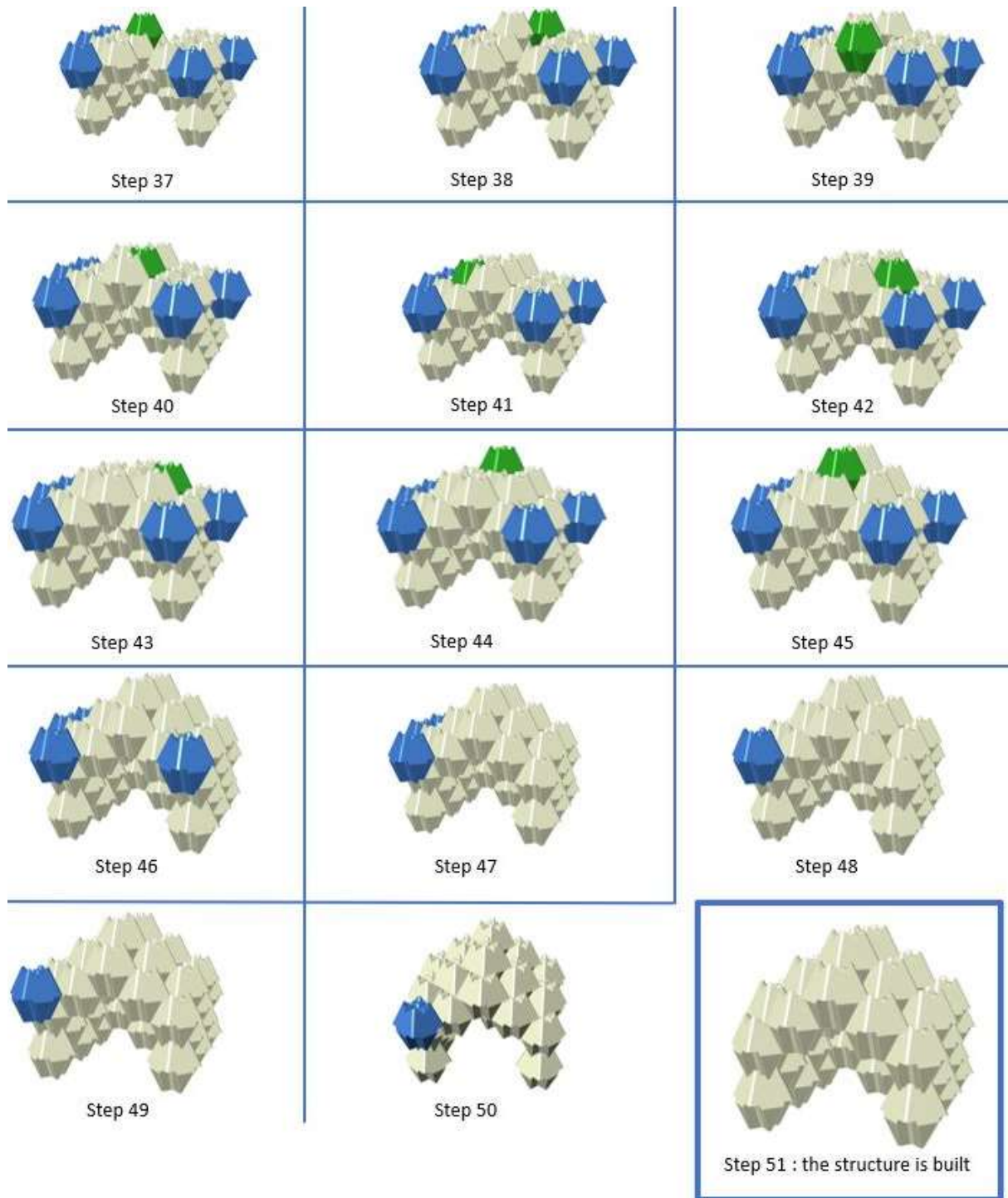


Figure A.6 - Example 3 (small shelter): Laying sequence (part I)

A.3 LIMITS OF THE ALGORITHM: EXAMPLE

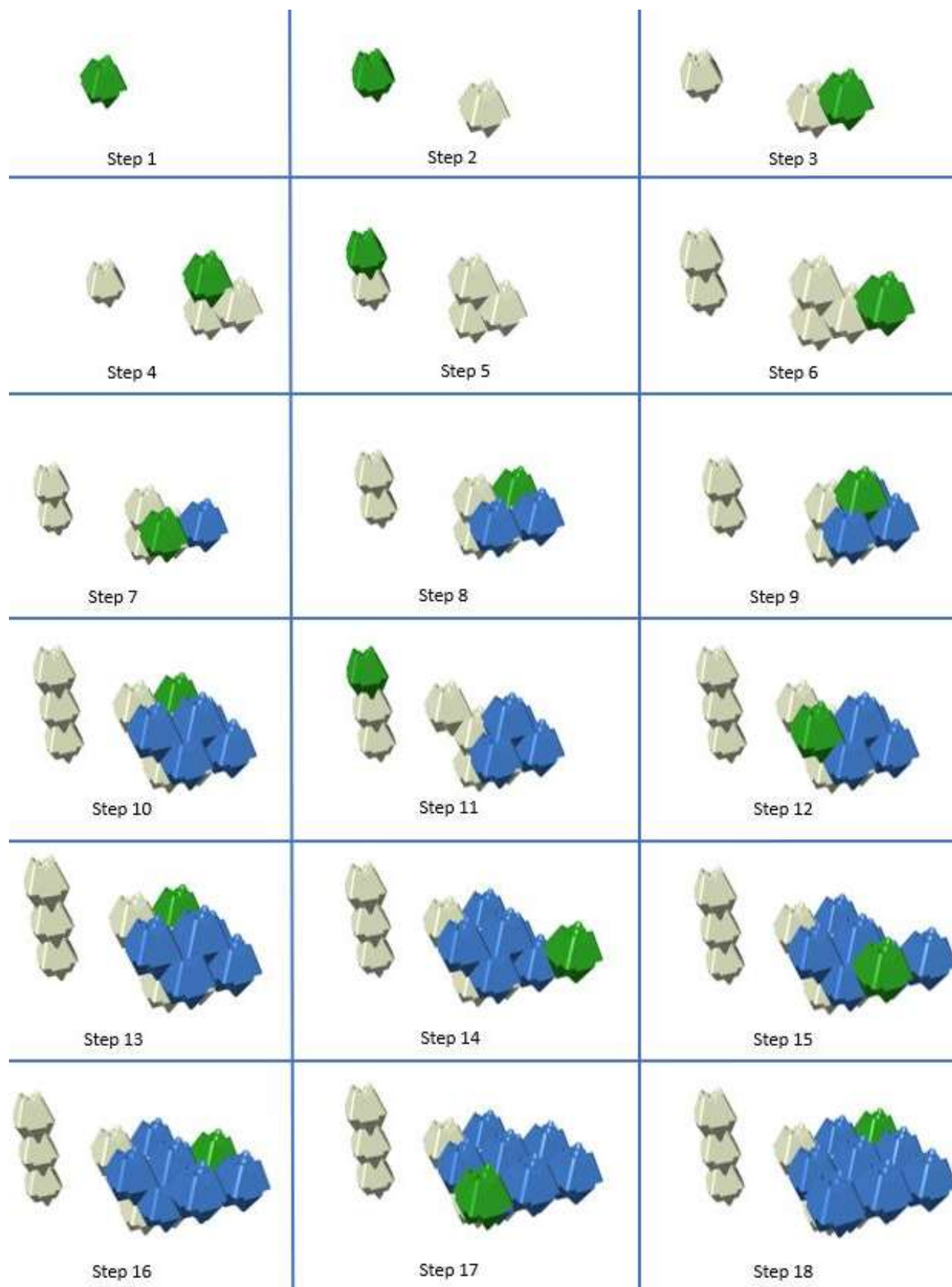


Figure A.7 - Example 5 (limit of the algorithm): Laying sequence (part I)

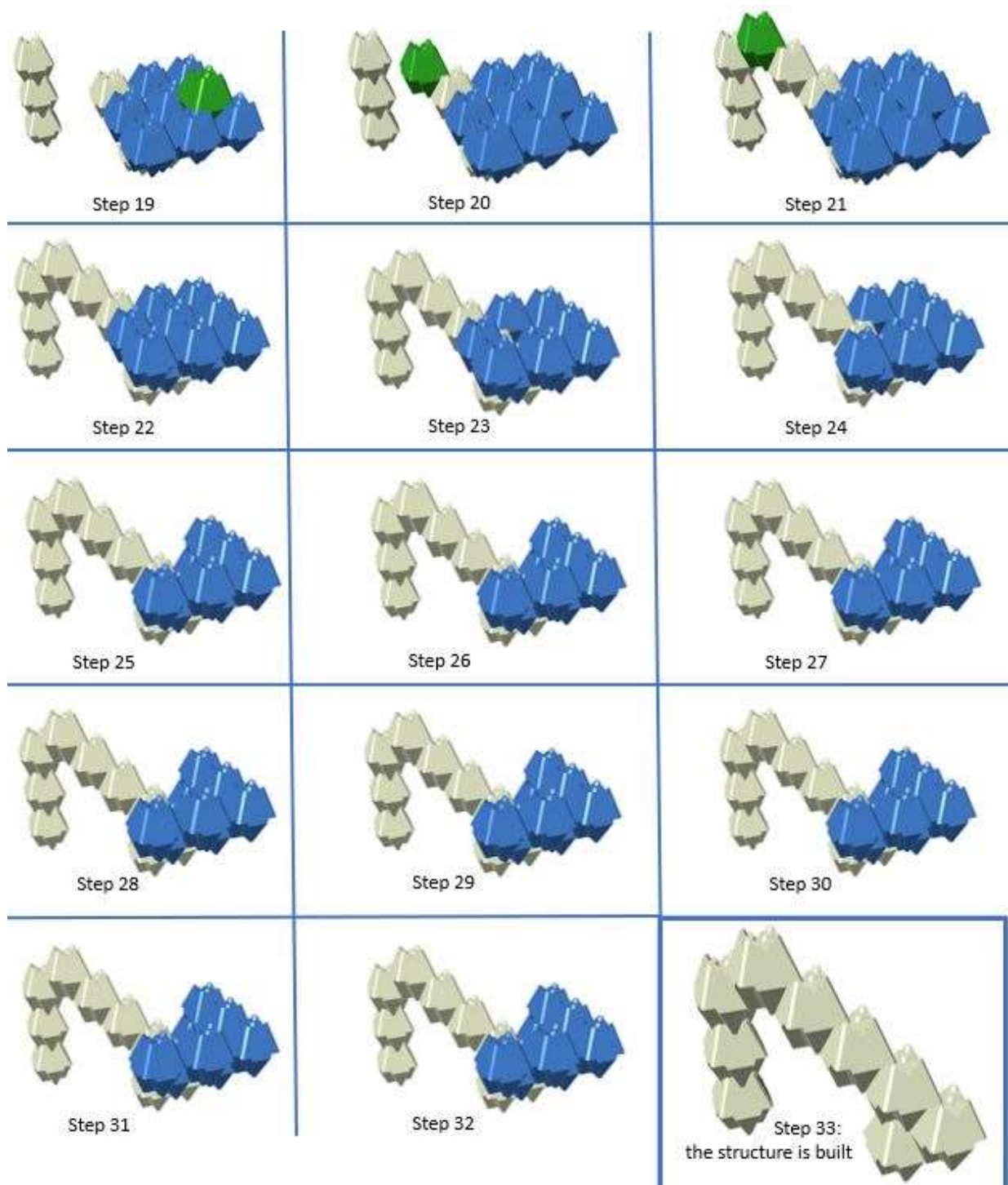


Figure A.8 - Example 5 (limit of the algorithm): Laying sequence (part II)

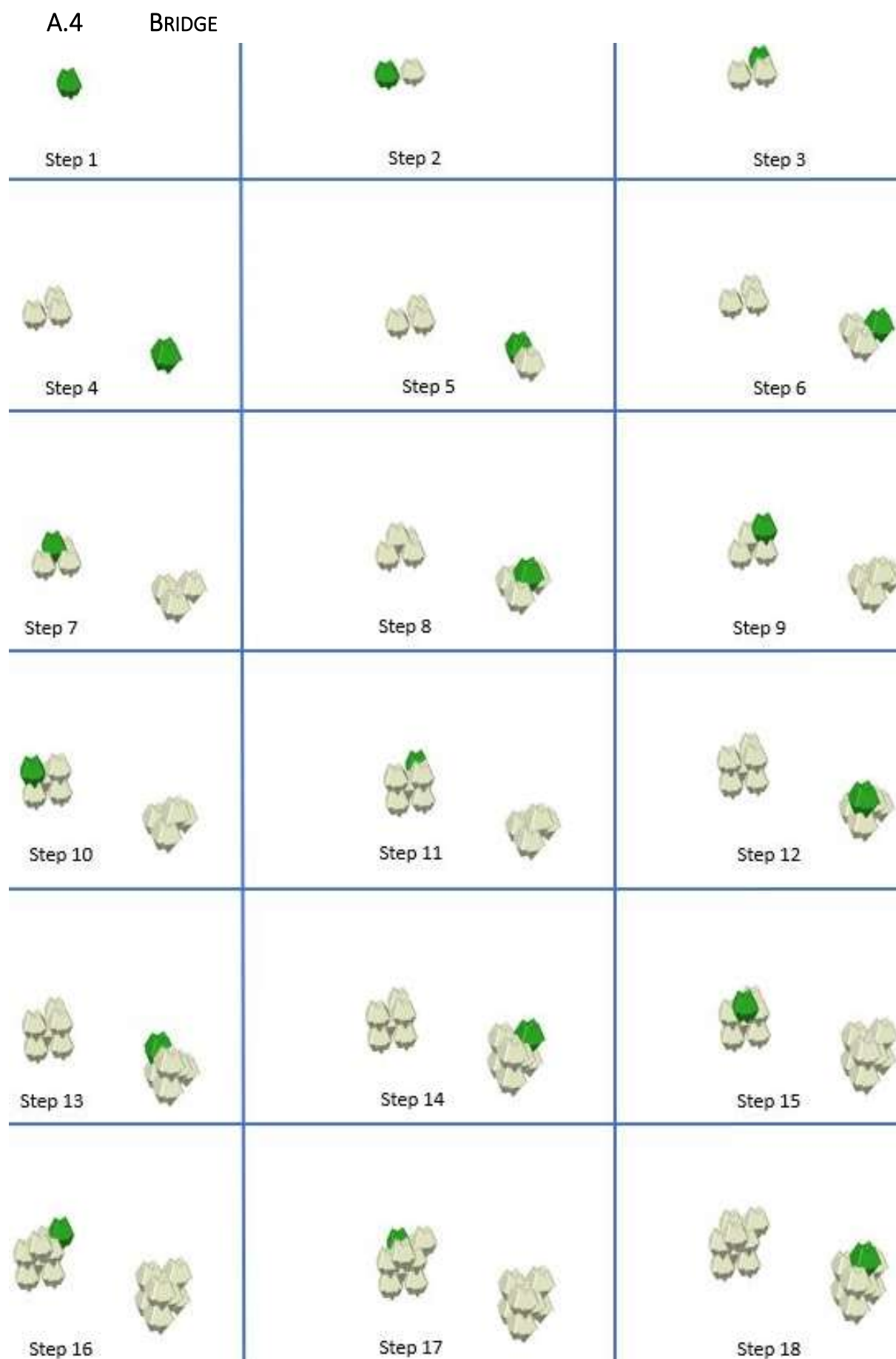


Figure A.9 - Example 6 (bridge): Laying sequence (part I)

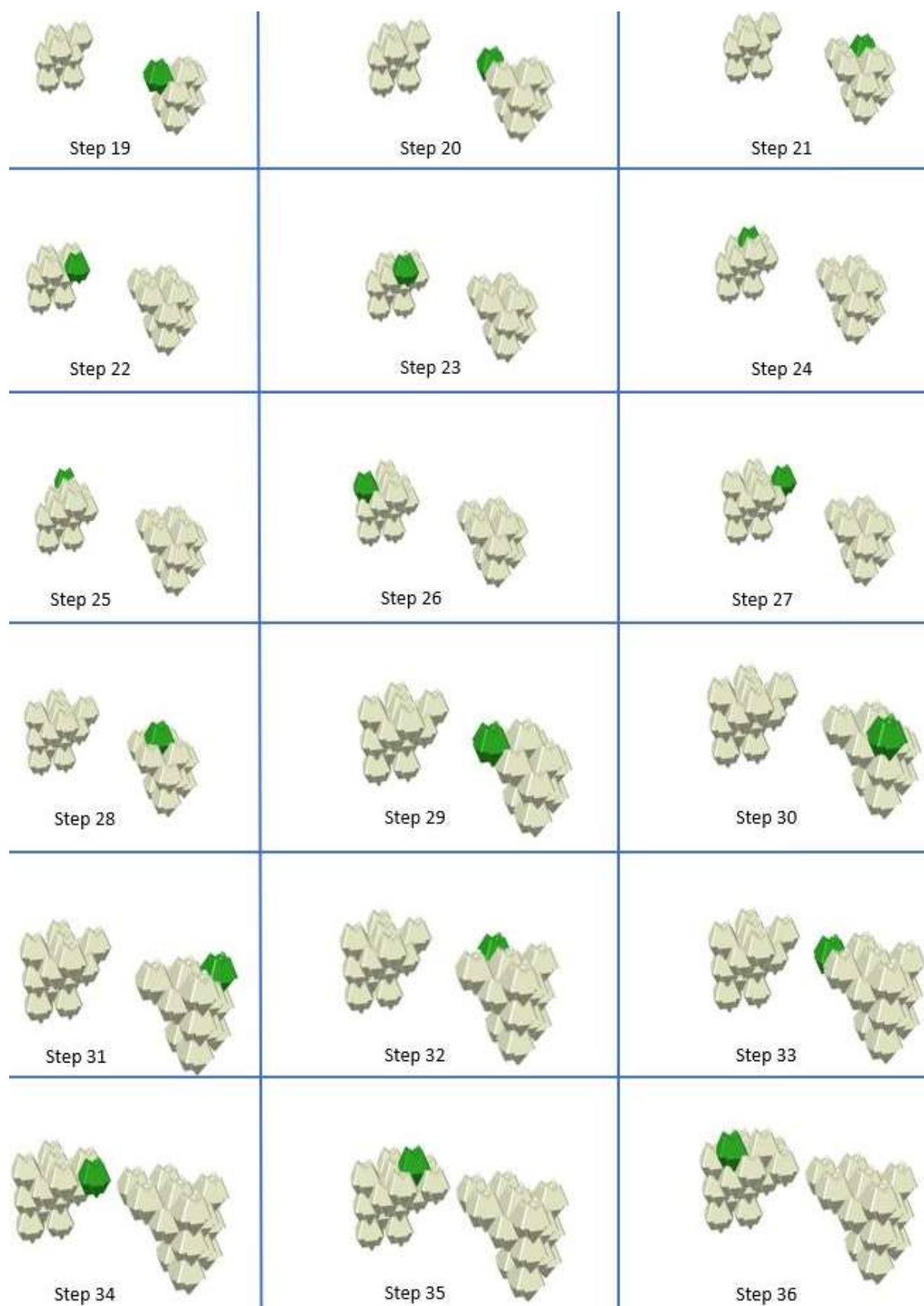


Figure A.10 - Example 6 (bridge): Laying sequence (part II)

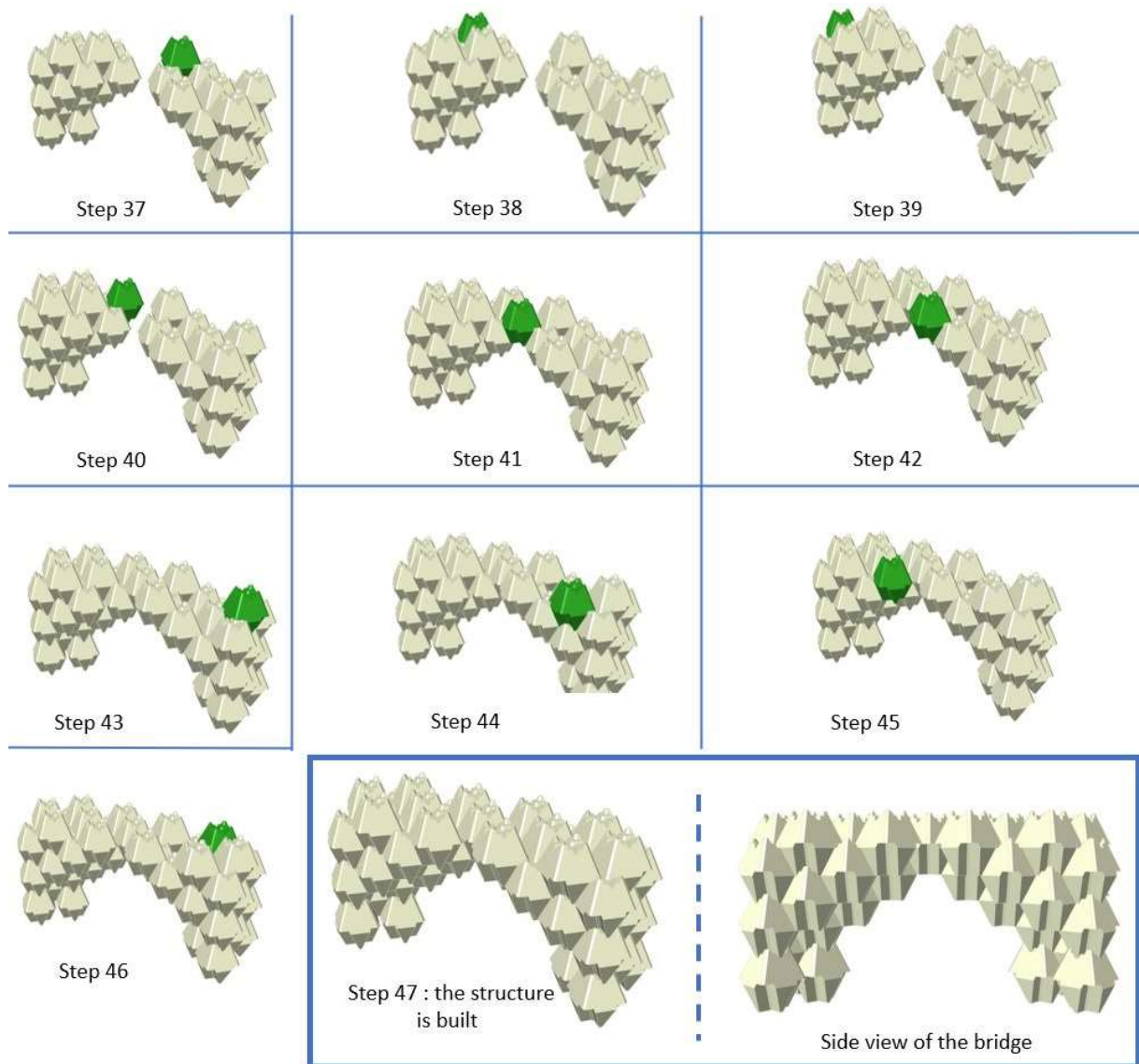


Figure A.11 - Example 6 (bridge): Laying sequence (part III)

A.5 SQUARE FENCE

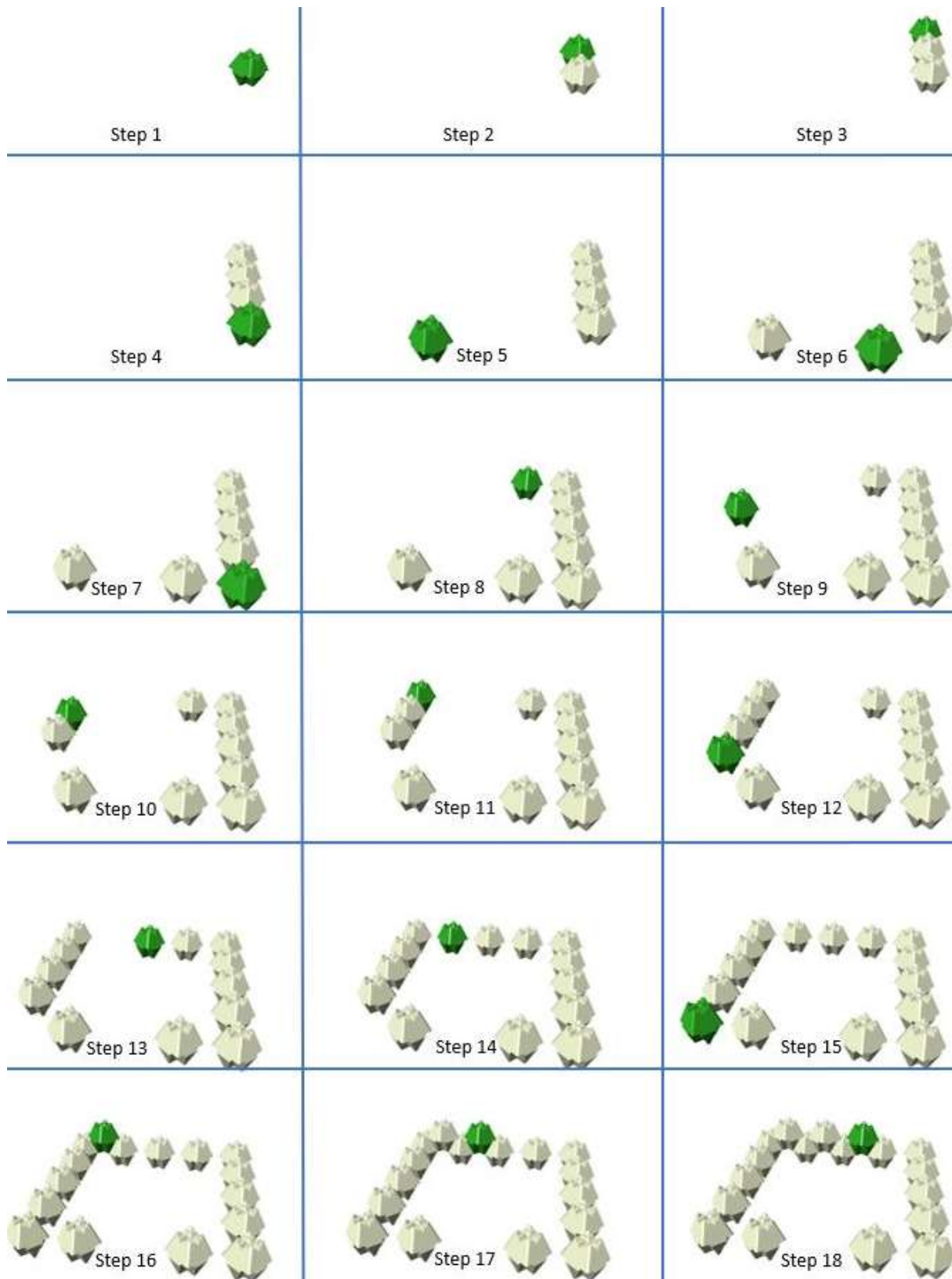


Figure A.12 - Example 7 (square fence): Laying sequence (part I)

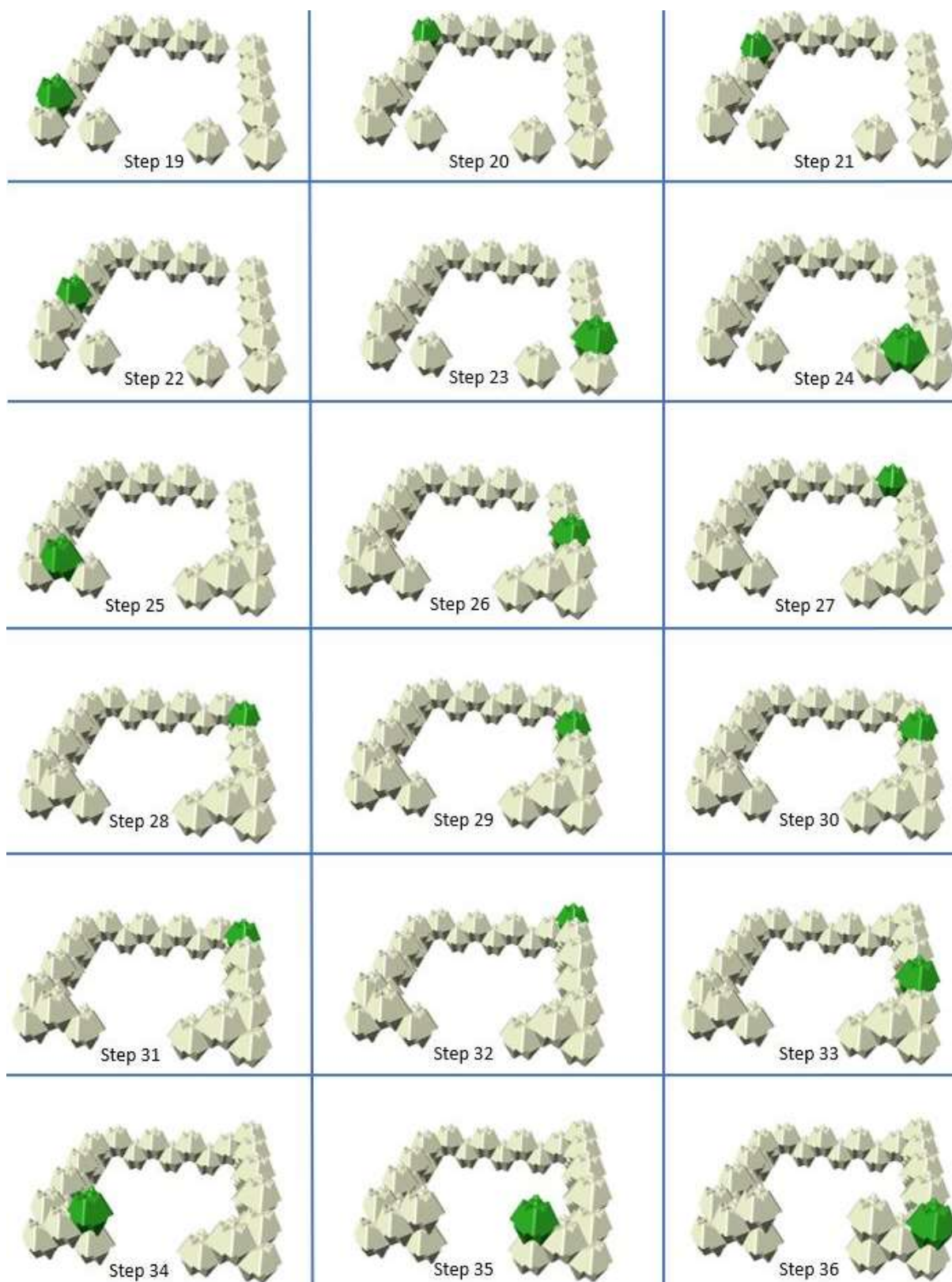


Figure A.13 - Example 7 (square fence): Laying sequence (part II)

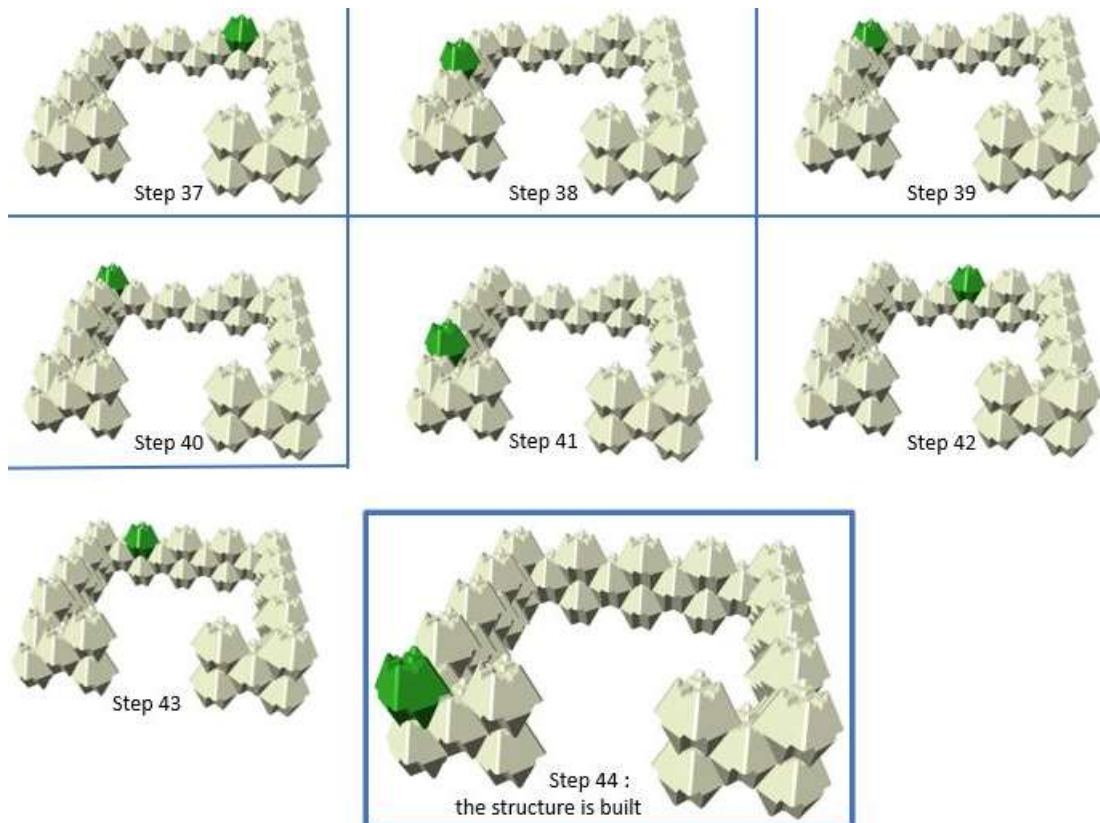


Figure A.14 - Example 7 (square fence): Laying sequence (part III)

APPENDIX B: NUMERICAL MODELLING OF RIGID DISCRETE STRUCTURES

INTRODUCTION TO MODELLING OF DISCRETE STRUCTURES

Discrete, discontinuous, or multibody structures are structures composed of different independent but interacting bodies. Unlike continuous structures, in which the quasistatic hypothesis can be made, the interaction between the bodies must be considered as external forces acting on each individual body. The classical Finite Element Methods (FEM), which considers a structure like a continuous media can thus no longer be used for discrete structures. Indeed, the two main features of discrete structures: unilaterality and friction between the elements are not met using FEM [15]. This explains the introduction of Discrete Element Methods (DEM) about fifty years ago.

The motivation for the development of DEM was the modelling of granular materials which is a very complex issue [14]. Discrete structures can be seen as a granular material in which each grain is one of the bodies of the structure.

It is important to note that many methods exist for the simulation of granular material, a numerical method in itself is actually a model. Even though a method will (almost) always yield some results, these must thus be considered with caution and ideally be compared to real-life experiments to validate the model [14].

The approach used in DEM is to integrate the equation of motion for each individual body of the structure. The resolution of the integrals is done simultaneously for every element. Since the elements interact with each other, contact laws between the elements must correctly be determined. In collections with a large number of bodies, controlling the possible interaction would be too costly in terms of computational effort [15]. In order to limit the number of interactions needing to be checked, several sorting algorithms can be implemented. Choosing an appropriate sorting algorithm is a way to reduce the important computational cost of DEM.

The contact interactions can be modelled in different ways depending on the modelling of the bodies themselves. One can for example consider the overlap between the bodies to calculate the repulsive forces between them. This interaction model is called molecular dynamics (MD) [15].

Another possibility is to use contact dynamics (CD), which allow for an exact treatment of frictional contacts between solid bodies. This is the contact interaction further studied in this document [15].

When systems become very dense, it might be interesting to consider quasistatic (QS) discrete element methods (the inertial effects of the bodies are then neglected). But when the interactions of the bodies are governed by the equation of dynamics, the use of this method cannot be justified [15].

Finally, event-driven (ED) dynamics are relevant when studying dilute systems (granular gases) [15].

General principle of the contact dynamic (CD) method

The general principle of CD can be easily explained with *Figure 4.2* which shows the 3 tasks executed over a certain imposed time step Δt .

Let's consider a collection of bodies. Their displacement can be found solving the equation of motion for each body:

$$\mathcal{M}(q, t)\dot{V} = F_{ext}(q, V, t) + F_{int}(t) + R$$

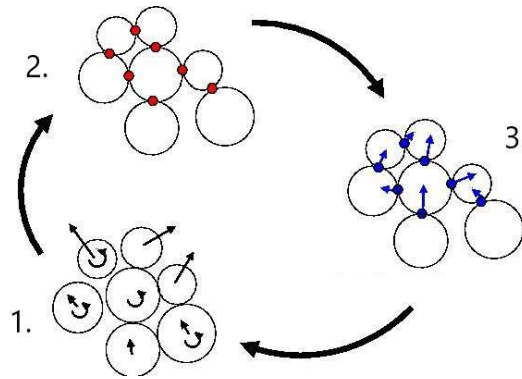


Figure B.1 - Tasks executed over a time step [16]

1. Bodies in movement
2. Detection of the bodies in interaction with other bodies
3. Computation of the contact forces between the bodies

Generally, the iteration stops when t_{tot} has been reached but it could also be possible to impose other conditions to stop the simulation (maximum displacement/velocity/...).

Content of this appendix

This document is divided in 3 main parts covering the three steps presented in Figure 4.2.

In the first part, the computation of the movement of the bodies as well as the possible interaction laws between them will be covered. Some elements which will later be needed for contact detection and description, will also be introduced.

The second part concerns the computation of contact forces between the bodies. These forces are necessary to compute the movement of the bodies after an impact.

Next, the contact detection method between the bodies will be mentioned. The choice of contact detection method is important to reduce the computation time of DEM, one of its main drawbacks.

The last part of this document will briefly describe the modelling of droxels in the *LMGC90* platform.

Some considerations done in this document

In this document, only rigid bodies will be considered. Some adaptations have to be made to consider elastic bodies but they won't be detailed. We can however mention that the main differences are present in the modified mass matrix and some velocity adaptations necessary after an impact.

The contact dynamics can be classified into two main categories: smooth and non-smooth dynamics. Non Smooth Contact Dynamics (NSCD) allow for the description of collisions and other non-smooth phenomena. It also has the advantage of accepting rather large time steps (in comparison to smooth methods) [16]. This is why smooth dynamics will not be covered in this document.

Lastly, only convex bodies will be considered. The generalisation for non-convex bodies will however be treated in the *Contact detection* part.

1. CONTACT DYNAMICS METHOD

1.1 Introduction

The two main features considered for contact dynamics are the unilaterality (bodies cannot penetrate one another) and dry friction (a certain force is necessary before the bodies start sliding). These two features allow the implementation of the contact laws necessary for the computation of the equation of motion of the bodies.

The movement of the bodies can afterward be found by integrating these equations of motion.

1.2 Variables involved

Several variables are used for the resolution of a frictional contact problem : the bodies, the contactors (useful for complex or non-convex particles) and the candidates to contact.

First of all, the collection of rigid bodies q has to be given as an input. The components of q can for example be the position of the gravity centres as well as the rotation vector of each body.

Once we have the collection of bodies, several pair of bodies (one candidate and one antagonist body each time) can be considered : σ_m and σ_n . For each of these bodies, there is a unique pair of points P ($\in \partial\sigma_m$) and P' ($\in \partial\sigma_n$) minimizing the distance MM' between the two boundaries of the bodies (see *Figure B.2*) [16].

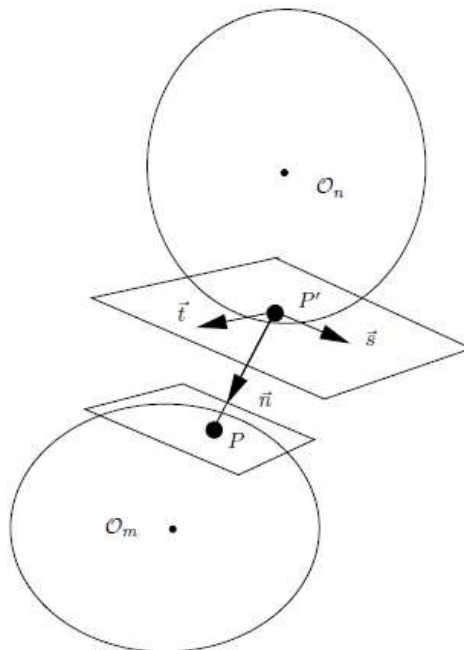


Figure B.2 - Distance between two objects, definition of the local frame [16]

Since strictly convex bodies are considered, it is assumed that there exists a unique unit normal vector to $\partial\sigma_m$ at P and to $\partial\sigma_n$ at P' . This is not the case for non-convex particles (see later) where the notion of contactor becomes interesting : “A contactor is a body, or part of this body, or a virtual part attached to this body, where the candidate-antagonist points are fixed by an appropriate rule.” [15]

For small collection of bodies, one could check all the possible contacts between all the bodies. However in most cases it is interesting to implement a sorting algorithm allowing to select only the bodies close enough to risk a contact. (More on that in 3.3)

1.3 Kinematics

Rigid body kinematics

For a motion with respect to a fixed frame we can write the velocity vector of a point M for a rigid body σ [16]:

$$\vec{V}(M) = \vec{X} + \vec{\omega} \wedge \overrightarrow{OM} \quad (1)$$

Local variables and distance function

The algebraic distance PP' is called the *gap*. It is positive and vanishes when the bodies are contacting (however, for numerical convenience negative values are allowed). In order to determine the reactions from one body to another, a local frame (f_t, f_N, f_s) is defined between the two elements. The tangential components f_t, f_s are often referred to as f_T .

The *relative velocity vector* of the candidate body with respect to the antagonist body is defined by : $\vec{U} = \vec{V}(P) - \vec{V}(P')$. The tangential component is called the *sliding velocity*.

If α is a candidate to contact⁶, the distance between the possible contractors can be defined as a continuous function g :

$$g^\alpha = \text{Gap}^\alpha(q)$$

An important relation is the fact that *the time derivate of the distance function is the normal component of the relative velocity* :

$$\dot{g}^\alpha = U_N^\alpha$$

This relation allows for the construction of the distance function as well as a numerical approximation of the gap over a time step $h = [t_i, t_{i+h}]$.

$$\text{Gap}^\alpha(q(t_i + h)) \sim \text{Gap}^\alpha(q(t_i)) + h U_N^\alpha(t_i + h) \quad (2)$$

Relations between generalised and local variables

Local variables are represented by a lower case symbol, generalised variables with an uppercase. Following relations are valid :

- Relative velocity : $U^\alpha = G^\alpha(q)u$ or $U^\alpha = H^{*\alpha}(q)u$
- Local reaction : $r^\alpha = G^{*\alpha}(q)R^\alpha$ or $r^\alpha = H^\alpha(q)R^\alpha$

With following mappings :

- Mapping $G(q)$: a linear mapping from $\mathbb{R}^N \rightarrow \mathbb{R}^3$ with N the number of degree of freedoms of the whole system ($= H^{*\alpha}(g)$)
- Transposed mapping $G^*(q)$: mapping from $\mathbb{R}^3 \rightarrow \mathbb{R}^N (= H^\alpha(g))$
 - o This mapping is not injective, meaning that several possible values can be found in $\mathbb{R}^N$. This leads to a non-uniqueness of the solution. This non-determination is mostly a problem in very dense, quasistatic systems. [3]

1.4 Equations of Dynamics

The Lagrange equation is [16]:

⁶ Notation : a possible contact will as of now be indicated with the Greek symbol (α or β).

$$M(q)\ddot{q} = F(q, \dot{q}) + E(t) + r \quad (3)$$

With :

- q the Lagrange variable
- $M(q)$ the mass matrix
- E the external forces explicitly known as functions of t
- r the generalized forces representing the frictional contact reactions (represented as unknown external forces)

If a rigid body is considered, the Newton-Euler equations are used because the calculation is faster⁷ :

Newton-Euler equation [16]:

$$\begin{cases} \mathbb{M}_R \ddot{q} = F(t) + R \\ \mathbb{J}_R \dot{\omega} = M + M_R \end{cases} \quad (4)$$

1.5 Shocks

If we want to add a possibility of shocks (velocity discontinuities) into our rigid body collection, the Lagrange equation becomes [16]:

$$\begin{cases} M(q)du = F(q, u)dt + E dt + r dv \\ q(t) = q(\tau) + \int_{\tau}^t u dt \end{cases} \quad (5)$$

1.6 Frictional contact laws

The main contact laws are unilaterality and dry friction. However, to determine the motion of the bodies, those two conditions are not enough and usually a shock law is also introduced. Other phenomena such as cohesion can also be added but they aren't conceptually more difficult to implement as unilaterality and dry friction.

A frictional contact law $C_{\text{act}}(g, U, R)$ can be defined as a relationship between the local variables : gap, relative velocity... In some cases other parameters can also be involved (rotation, wear, temperature...)

Let's note that an infinite number of contact laws are possible. Only the most recurring one is specified here but the choice of the contact law is obviously an important one. Choosing a certain contact law instead of another results in a different physical model and thus different results [16].

Unilaterality : Signorini gap condition

In order to assure the unilaterality, the gap has to be non-negative. To assure that, the normal reaction exerted by the antagonist is positive and becomes zero when the bodies separate (since no cohesion is considered). This relation is known as the *Signorini gap condition* : either the bodies touch and the gap is zero, or the bodies are separated and the normal reaction is zero.

⁷ The rest of the document will be done with the Lagrange's equations in order to be as general as possible. The principles are however exactly the same.

This condition supposes that, when a non-vanishing pressure is exerted at the contact, the relative velocity vector belongs to the tangent plane at the contact point. This hypothesis can be accepted, especially for rigid bodies.

The law can be written as follows [16]:

$$g(t) \geq 0, R_N(t) \geq 0, g(t)R_N(t) = 0 \quad (6)$$

Graphically it can be represented like this :

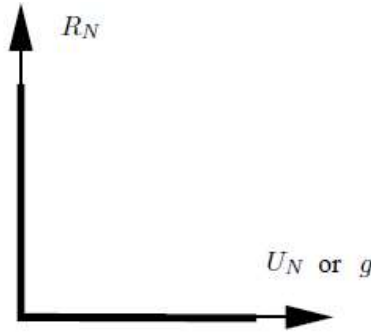


Figure B.3 - Graphical representation of the Signorini gap condition [16]

J.J. MOREAU also showed that if velocity discontinuities are being dealt with in the space of locally bounded variation functions (functions that are integrable for all measures/loads and reactions), an equivalent condition could be formulated in terms of relative velocity. [16]

The equivalent *velocity Signorini condition* is :

$$\begin{cases} g(t) \geq 0 \Rightarrow R_N(t) = 0 \\ \text{at initial time } t_0, g(t_0) \geq 0; \\ \text{for all } t \in I, \text{ if } g(t) \leq 0 \text{ then} \\ U_N^+(t) \geq 0, R_N(t) \geq 0, U_N^+(t)R_N(t) = 0 \end{cases} \quad (7)$$

Other equivalent forms of this condition exist. The generic form is given by :

$$\mathcal{U}_N \geq 0, \mathcal{R}_N \geq 0, \mathcal{U}_N \mathcal{R}_N = 0 \quad (8)$$

Each of the variations can be written as the *standard Signorini relation* : $S_{ignor}(\mathcal{U}_N, \mathcal{R}_N)$

Friction laws : Coulomb's law

A friction law is a relation between friction force (tangential component of the reaction) and the sliding velocity (tangential component of the relative velocity).

A difference is made between viscous and dry friction. Viscous friction occurs between lubricated parts and implies that a tangential force, no matter how small will generate sliding. This is not applicable for discrete structures and will thus not be further detailed.

Coulomb's law takes dry friction into account with friction coefficient $\mu \geq 0$ [16]:

$$||R_T|| \leq \mu R_N, ||U_T^+|| \neq 0 \Rightarrow R_T = -\mu R_N \frac{U_T^+}{||U_T^+||} \quad (9)$$

Using Convex Analysis and if the sliding velocity in Coulomb's law is the right limit velocity (velocity after time step Δt), a generic form can again be found [15]:

$$||\mathcal{R}_T|| \leq \mu \mathcal{R}_N, ||\mathcal{U}_T|| \neq 0 \Rightarrow \mathcal{R}_T = -\mu \mathcal{R}_N \frac{\mathcal{U}_T}{||\mathcal{U}_T||} \quad (10)$$

The graphical representation of Coulomb's law is given in the figure below.

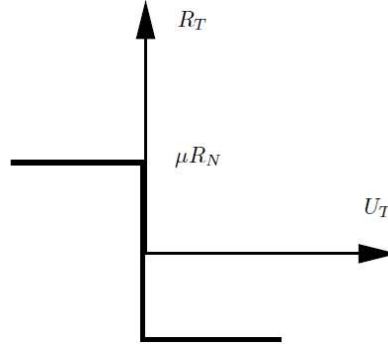


Figure B.4 - Graphical representation of Coulomb's law [16]

Each of the variations can be written as the *standard Coulomb's law* : $C_{oul\mu\mathcal{R}_N}(\mathcal{U}_T, \mathcal{R}_T)$

Shock laws

Even if we consider rigid bodies, all bodies are in reality slightly elastic. In case of shock, several local phenomena occur (local elastic/plastic deformation, wave propagation...). The time and space scale of these phenomena are much smaller than the ones at which the rigid body motions are analysed.

In case of binary impact, three kinds of restitutions can be used [16]:

- Newton restitution linking the velocities before and after the impact ($\mathcal{V}^+ = -e_n \mathcal{V}^-$);
- Poisson restitution linking the compression and restitution impulse ($\mathcal{J}_r = e_p \mathcal{J}_c$);
- Energy or *Stronge* restitution linking the restitution to the compression energy ($e_s = \frac{\int_{t_c}^{t_r} \mathcal{R} \mathcal{V} dt}{\int_{t^*}^{t_c} \mathcal{R} \mathcal{V} dt}$).

The choice of restitution coefficient is a difficult task for complex structures. In the case of a dense collection of rigid bodies, the effects of the impact can be neglected and an inelastic shock law can be used ($e = 0$).

If the kinematics at the local scale are simple and the function g is considered to be a continuous function, Coulomb's law and the Signorini conditions stay relevant even for small shocks.

1.7 Conclusion : equations for frictional contact problem

The equations governing collection of rigid bodies interacting with unilateral contact and Coulomb's friction are :

- Lagrange's equation of dynamics (a)
- Kinematic relations relating the local and global variables (b)

- Frictional contact relation (c)
- Boundary and initial conditions (d)

$$\begin{cases}
 M(q) du = F(q, u) dt + E dt + r dv & (a) \\
 q(t) = q(\tau) + \int_{\tau}^t u dt & (a) \\
 U^{\alpha} = H^{*\alpha}(q)u, r = \sum_{\alpha} r^{\alpha} = \sum_{\alpha} H^{\alpha}(q)R^{\alpha} & (b) \\
 \dot{g}^{+\alpha} = U_N^{+\alpha} \text{ or } g^{\alpha} = \text{Gap}^{\alpha}(q) & (b) \\
 C_{tact}^{\alpha}(q^{\alpha}, U^{\alpha}, U^{\alpha}) & (c) \\
 \text{Boundary and initial conditions} & (d)
 \end{cases}$$

These equations can also be seen as the following scheme :

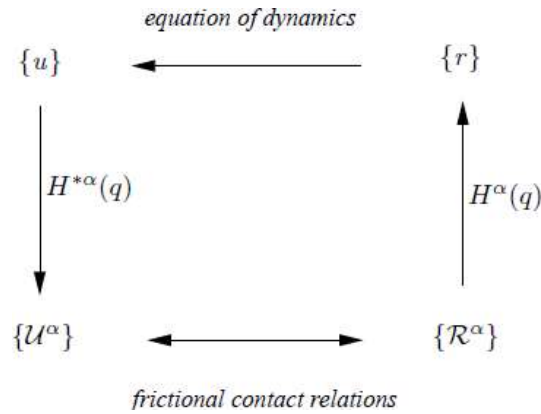


Figure B.5 - Duality scheme : link between the different equations [16]

In order to solve these equations, the impact due to the shocks as well as the new velocities are necessary. This is where the Non Smooth Dynamic (NSCD) methods comes in.

2 NON SMOOTH CONTACT DYNAMICS METHOD (NSCD)

As a reminder, contact dynamics treatment can be divided in two categories : smooth and non-smooth contact dynamic. As explained on page 135, only the non-smooth contacts will be exposed since these are the one that interest us.

2.1 Resolution scheme of NSDC

The general resolution scheme of NSDC goes as follows :

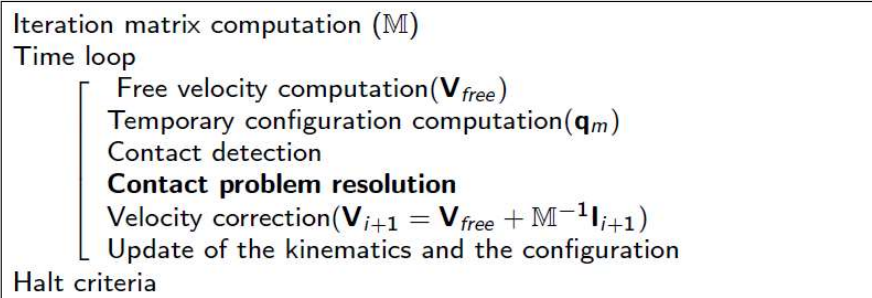


Figure B.6 - Resolution scheme of NSCD [14]

Since the mass matrices of the bodies are constant for the whole simulation, they can be computed once outside the loop. The computation of the free velocity and temporary configuration is done with the equations of dynamics, the same goes for the velocity correction and the update of the kinematics.

Contact detection (or prediction criteria) often use gap or gap-prediction in order to determine whether a contact between two bodies is possible. A possible (pessimistic) criterion could be : *if $g^\alpha(i) \leq 0$, a contact occurs*. Contact detection methods are a vast subject. A (non-exhaustive) overview of the different types of methods (first for convex and then for non-convex particles) is given in *Contact detection*, p. 150 [14].

Assuming a prediction criteria is found and the possible interacting candidates are known, let's tackle the resolution itself of the contacts.

2.2 Frictional contact problem resolution

The contact problem resolution can be done with several algorithms : Non Linear Gauss Seidel (NLGS), quasi NLGS, the Conjugate Projected Gradient Algorithm, Lemke (for small number of rigid bodies)...

The classical approach of NSCD relies on a NLGS algorithm which is why it is the one that will be detailed but literature and implementations of the other methods can also be found.

The frictional contact problem is solved computing the following steps :

1. Discrete form of the equation of dynamics
2. Discrete form of the kinematic relations
3. Discrete form of the frictional contact relations
4. Equation of dynamics restricted to candidates to contact
5. Standard frictional problem for a single contact
6. Frictional contact for a collection of bodies

Discrete form of the equation of dynamics

With a constant mass matrix, the quadratic acceleration terms in the equations of Lagrange vanish and assuming that the external forces are known as a function, of time t , the equation of dynamics can be rewritten as [16]:

$$\begin{cases} \mathbb{M} du = E dt + r dv \\ q(t) = q(\tau) + \int_{\tau}^t u dt \end{cases} \quad (11)$$

Integration of this equations gives :

$$\begin{cases} \mathbb{M}(u^+(t_{i+1}) - u^+(t_i)) = \int_{]t_i, t_{i+1}]} E dt + \int_{]t_i, t_{i+1}]} r dv \\ q(t_{i+1}) = q(t_i) + \int_{t_i}^{t_{i+1}} u dt \end{cases} \quad (12)$$

The impulse, accounting for all the reactions in the interval is the sum of all the reactions exerted at all candidates to contact α : $\int_{]t_i, t_{i+1}]} r dv = p(i+1) = \sum_{\alpha} p^{\alpha}(i+1) = \sum_{\alpha} \int_{]t_i, t_{i+1}]} r^{\alpha} dv$

Resolution scheme:

The choice of the numerical integration scheme is important in order to assure stability of the method as well as a reasonable computation time. For non-smooth systems, two groups of integration schemes exist : event-driven and time-stepping.

Event-driven methods are more accurate for free flight smooth motions having few impacts but aren't able to handle multiple transitions in a short time. Furthermore, they have no convergence proof. This makes them in practice only applicable for small multibody system with few events/interactions.

Time-stepping methods however don't adapt their time step size in function of the events but only if the accuracy requirements haven't been met. Even though their accuracy is less than for an event-driven method (unless a very small time step is used), they have been proven to be convergent and robust. This is why they are used a lot for non-smooth models.

Two types of time stepping integrating schemes have been developed: the Schatzman–Paoli, based on a central difference method and Moreau-Jean scheme, based on a θ -method (Crank-Nicholson). θ -method [34] :

$$y_{i+1} = y_i + \theta h f_n + (1 - \theta) h f_{n+1} \quad (13)$$

If $\theta \in [0.5, 1]$, the method is implicit and unconditionally stable. For $\theta = 1$, the scheme corresponds to an explicit Euler scheme, for $\theta = 0$ the method becomes the implicit Euler scheme. The Moreau-Jean scheme corresponds to one with $\theta = 0,5$.

If we admit that $E_{(i,i+1)}$, $u(i)$ and $q(i)$ are an approximations of respectively $\int_{[t_i, t_{i+1}]} E dt$ (which is a known quantity), $u^+(t_i)$ and $q(t_i)$ we can find $\int_{[t_i, t_{i+1}]} E dt$ and $q(t_{i+1})$. These are obtained by solving the Moreau-Jean scheme numerically, making it possible to complete the discrete equations of dynamics.

Discrete form of the kinematic relations

If the mapping $H^{*\alpha}$ keeps a constant value on the duration of the integration step the kinematic relations become [16]:

$$\begin{aligned} - \text{Gap function :} & \quad \dot{g}^\alpha = U_N^\alpha & \rightarrow g^\alpha(i+1) = U_N^\alpha(i+1) \\ - \text{1}^{\text{st}} \text{ kinematic relation :} & \quad U^\alpha = H^{*\alpha}(q)u & \rightarrow U^\alpha(i) = H^{*\alpha}(q(i))u(i) \\ - \text{2}^{\text{nd}} \text{ kinematic relation :} & \quad r^\alpha = H^\alpha(q)R^\alpha & \rightarrow p^\alpha(i+1) = H^\alpha P^\alpha(i+1) \\ & & \text{with } P^\alpha(i+1) = \int_{[t_i, t_{i+1}]} R^\alpha dv \end{aligned}$$

If however the mapping is not constant but nearly constant, using $H^\alpha(\hat{q})$ with $\hat{q}$ the value of q used to construct the local frame is still acceptable. When the relative displacement during a time step is small with respect to the boundary curvatures of the contactors, adapting q has little influence on the results.

Discrete form of the frictional contact relations

A) Velocity Signorini condition (unilaterality)

A possible discretisation is :

If a contact is predicted,

$$- S_{ignor}(U_N^\alpha(i+1), P_N^\alpha(i+1))$$

Else

$$- P^\alpha(i+1) = 0$$

B) Coulomb's law (friction)

A possible discretisation is :

If a contact is predicted,

$$- C_{oul} \mu^\alpha P_N^\alpha(i+1)(U_T^\alpha(i+1), P_T^\alpha(i+1))$$

Equation of dynamics restricted to candidates to contact

Only the pairs of candidates who have a predicted contact (see previous step) are concerned by this.

Using the discrete forms of the kinematic relations, the discrete equations of dynamic can be rewritten [16]:

$$\begin{aligned} U^\alpha(i+1) &= U_{free}^\alpha + \sum_{\beta} W^{\alpha\beta} P^\beta(i+1) \\ U_{free}^\alpha &= H^{*\alpha}(u(i) + M^{-1}E(i)) \end{aligned} \quad (14)$$

With $W^{\alpha\beta} = H^{*\alpha} M^{-1} H^\beta$ and where U_{free}^α represents the value taken by $U^\alpha(i+1)$ when all reactions vanish.

Standard Signorini μ -Coulomb problem : single contact

The generic form of a frictional contact problem can be given by [16]:

$$\begin{aligned} - \text{Find } \mathcal{U}, \mathcal{P} \text{ such that} \\ \mathcal{U} &= \mathcal{U}_{locfree} + \mathcal{W}\mathcal{P} \\ S_{ignor}(\mathcal{U}_N, \mathcal{P}_N) \\ C_{oul} \mu \mathcal{P}_N(\mathcal{U}_T, \mathcal{P}_T) \end{aligned}$$

With $\mathcal{U}, \mathcal{P}, \mathcal{U}_{locfree} \in \mathbb{R}^3$, $\mathcal{W}$ a 3x matrix

The contact is solved as followed [16]:

- Provisional assumed values : $P^\beta(i+1), \beta \neq \alpha$
- Setting : $\mathcal{U}_{locfree} = U_{locfree}^\alpha, \mathcal{W} = W^{\alpha\alpha}$
- If the contact α is predicted, the solutions of the problem are found :

$$\mathcal{U} = U^\alpha(i+1), \mathcal{P} = P^\alpha(i+1)$$

In 2D (or for 3D spherical bodies), the solution can be found explicitly. For the general 3D case, an iterative method is needed to converge to the solution (e. g. generalized Newton method/projection method).

Solution of the frictional contact problem : collection of contacts (NLGS)

Let α be a candidate to contact (candidate-antagonist pair), an approximation of the solution $P^\alpha(i + 1)$ is sought, within a maximal number of iterations k_{max} . Each iteration can be split up into 3 parts [16]:

1) Approximate solution for the contact α

$$U_{locfree}^\alpha = U_{free}^\alpha + \sum_{\beta < \alpha} W^{\alpha\beta} P^\beta(k + 1) + \sum_{\beta > \alpha} W^{\alpha\beta} P^\beta(k) \quad (15)$$

If the contact α is *not predicted*, $P^\alpha(k + 1) = 0$, else it is the one provided by the resolution of the standard Signorini μ -Coulomb problem (see previous section).

2) The data are updated, the next candidate to contact is now analysed

3) These steps are performed as many times as necessary (k_{max}) to satisfy an accuracy criterion

In most cases, the algorithm converges to a solution.

A non-linear *block Gauss-Seidel method* also exists where several contacts are being checked at once instead of one single contact per iteration.

2.3 Implementation of NLGS (contact problem resolution)

Several implementations are possible, a few of them are :

- Stored Delassus Loop strategy (SDL) :

```

(0) Evaluating all the matrices  $W^{\alpha\beta} \forall \beta$ 
   $k = k + 1$  (NLGS iteration)
     $\alpha = \alpha + 1$  (Contact index)
    (a) Evaluating the right-hand side
         $\mathcal{V}_{loc}^\alpha = \mathcal{V}_{free}^\alpha + \sum_{\beta < \alpha} W^{\alpha\beta} \mathcal{I}^{\beta,k+1} + \sum_{\beta > \alpha} W^{\alpha\beta} \mathcal{I}^{\beta,k}$ 
    (b) Solving the local problem
    Convergence test for  $k = 0 \dots k_{max}$ 
    
```

Figure B.7 - Stored Delassus Loop Strategy [14]

This method is performant for rigid and elastic bodies. Exchange Local Global Strategy (ELS) :

```

(0) Evaluating the  $W^{\alpha\alpha}$  matrix
   $k = k + 1$  (NLGS iteration)
     $\alpha = \alpha + 1$  (Contact index)
    (i) Identifying the contact bodies( $\alpha = jl$ )
        Computing an auxiliary value
         $\mathcal{V}_{aux}^\alpha = \mathbb{H}^{*,\alpha}(\mathbb{M}^{-1,j} \mathbf{p}^{j,k} - \mathbb{M}^{-1,l} \mathbf{p}^{l,k})$ 
    (a) Evaluating the right-hand side
         $\mathcal{V}_{loc}^\alpha = \mathcal{V}_{free}^\alpha + \mathcal{V}_{aux}^\alpha - W^{\alpha\alpha} \mathcal{I}^{\alpha,k}$ 
    (b) Solving the local problem
    (i) Updating the resultant on bodies( $\alpha = jl$ )
         $\begin{bmatrix} \mathbf{p}^j \\ \mathbf{p}^l \end{bmatrix}^{k+1} = \begin{bmatrix} \mathbf{p}^j \\ \mathbf{p}^l \end{bmatrix}^k + \mathbb{H}^\alpha(\mathcal{I}^{\alpha,k+1} - \mathcal{I}^{\alpha,k})$ 
    Convergence test for  $k = 0 \dots k_{max}$ 
    
```

Figure B.8 - Exchange Local Global Strategy [14]

Implementations are also possible in parallel treatment : other CPU consuming parts of the code are executed at the same time. An schematic representation of a parallel SDL treatment is given below :

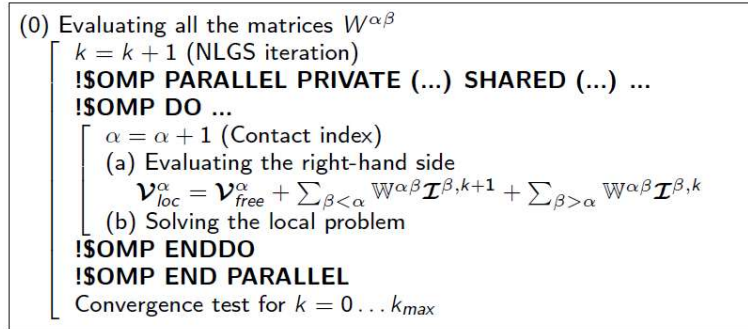


Figure B.9 - Parallel SDL treatment [14]

2.3 Conclusion

With as input the possible bodies being in interaction, the Non-Smooth Contact dynamics allow for the computation of the impacts and changed velocities following the interactions or following external forces being applied to one or several bodies.

3 CONTACT DETECTION

3.1 Introduction

Contact detection might be the most complex issue in contact dynamics and several methods exist to deal with it. The goal of this section is to give an overview of the possible methods without going into too much detail. For more information, a state of the art on contact detection is realised in [6].

Contact detection is important in order to reduce the computation time for DEM. Let's however note that a complex contact detection taking more time to run than the contact dynamics itself without the complex contact detection has no sense. An equilibrium must be found.

The dynamic method explained before considered strictly convex, spherical-like shapes. However, some specifications must be made if we want to consider complex or non-convex shapes.

First, the possible methods to implement a geometrical description of particles will briefly be discussed. This first step will allow us to get to some more elaborate contact detection methods of convex particles. Lastly, the issue of non-convex particles will be tackled.

3.2 Geometrical description

Several approaches are possible for the modelling of granular material [15]: considering spherical particles and adapting the interaction laws to match the real behaviour of the material, considering the particles as an assembly of spheres or representing them as super-ellipsoids, spheroids... A last possible method is to consider a polyhedral description of the grains.

Different geometrical description of the bodies are possible : Constructive Solid Geometry (CSG) which describes an object as a combination of several basic volumes (spheres, cylinder, cone...) or boundary Representation (*Brep*) containing topological and geometrical data of the bodies.

However both these methods are either too cumbersome or need a very effective computation of the distance between those surfaces. In order to overcome these drawback, a polyhedral⁸ description of

⁸ Polyhedron : 3D volume composed of flat polygonal faces, straight edges and sharp corners or vertices. [34]

the objects is obtained from the surface or volume meshes. This method also allows for an easier computation of mechanical quantities such as : centre of inertia, inertia matrix, global frame...

3.3 Sorting algorithms

In order to save some precious computation time, sorting algorithms are used in order to select only the close enough candidates to verify contact.

For simple convex bodies as considered up until now, a simple gap-prediction strategy allows to estimate which bodies are in contact with one another. For polyhedral particles other algorithms are implemented.

Contact detection methods are divided into 3 phases in order to reduce the complexity of the algorithms [14].

- 1) Broad phase : Building of a list with adjacent object within a certain distance
- 2) Narrow phase (optional) : improving the broad phase by removing trivial undesirable items or specifying which parts of the object are potentially concerned by the contact (notion of contactors mentioned in 1.2 *Variables involved*).
- 3) Contact determination : Determination of the pairs of touching objects and of the contact loci, relative velocities, distances...

Broad phase

The goal of this phase is to quickly build a list of neighbouring particles.

In the first part of this phase, bounding volumes (BV) are determined for the particles. Different examples of possible BV are shown in *Figure B.10*.

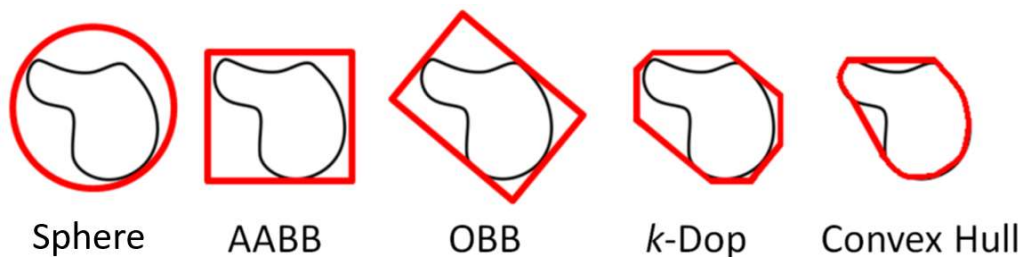


Figure B.10 - Examples of possible bounding volumes [14]

Once the bounding volumes are determined, a proximity method is applied to identify potential contactors. There are three proximity methods : triangulation, space subdivision and shadow overlap on an axis (sweep and prune method).

The triangulation method is not much used and mainly for spherical volumes.

The space subdivision consists of dividing the space into a box-mesh for which the size is equal to the diameter of the largest object. The potential contactors of an object inside a mesh elements are the objects present in the same cell as well as the surrounding cells (see *Figure 4.3*). Since this technique has a high computational cost, it is scarcely used.

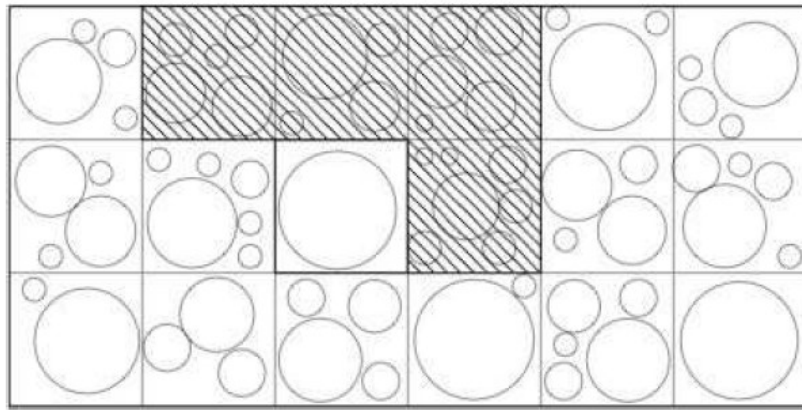


Figure B.11 - Broad detection : Space division [16]

The sweep and prune method (see Figure 4.5) is the most used method because it is efficient and doesn't depend on the complexity of the objects. It is a method that consists of projecting the bounding boxes of the elements on axes, if the projection overlaps on all the axes, it can be assumed that a contact may be happening and the pair of bodies is checked for contact detection.

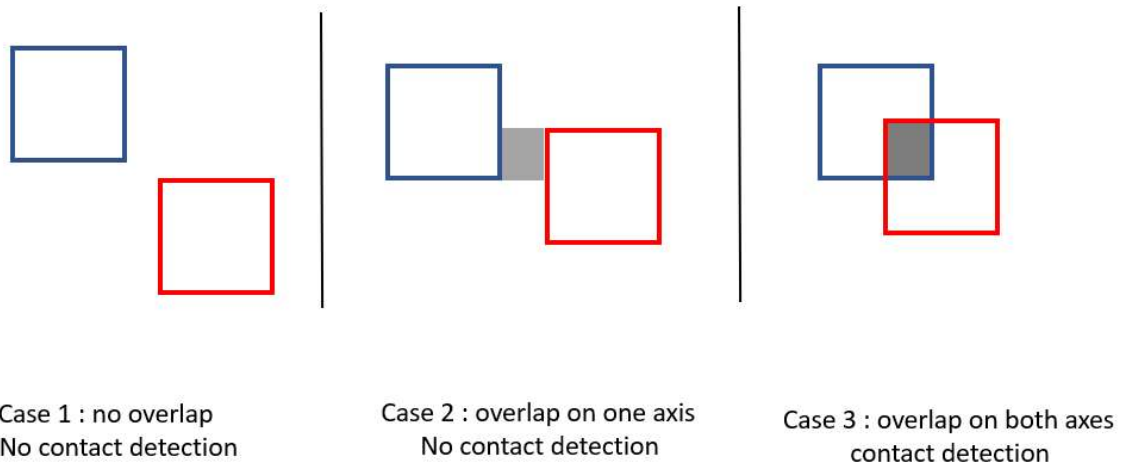


Figure B.12 - 2D representation of the Sweep and Prune method

Narrow phase

For simple convex objects, the broad phase can be repeated (for example : combination of space partition with sweep and prune method).

For more complex, non-convex object, a boundary volume hierarchy (BVH) can be created. This allows to divide an object in subparts represented by a boundary volume (BV), later a BV-BV detection is then used. An example of this method is given in the Figure below:

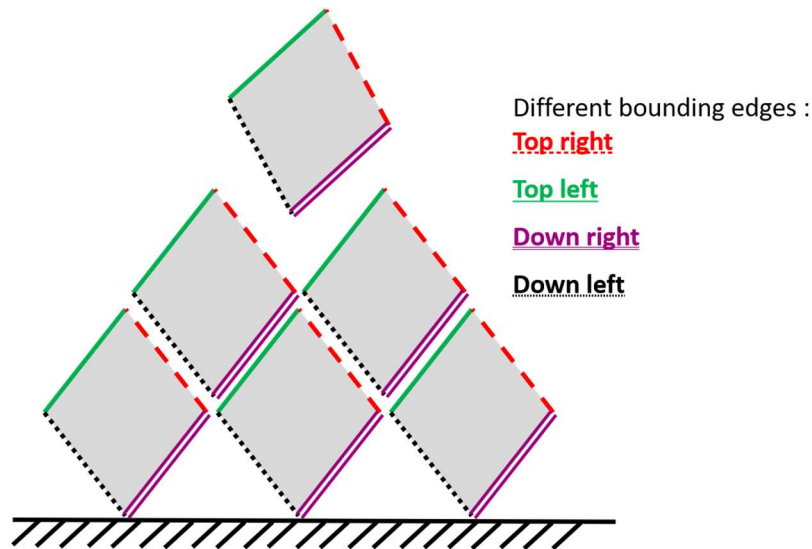


Figure B.13 - 2D Representation of bounding box hierarchy

The rhombus' edges have been split up in 4 parts : top left – top right – down right – down left. This makes it possible to create an interaction table : during the contact detection step only the possible contacts (*top right, bottom left*) and (*top left, bottom right*) will be checked.

Contact detection : convex polyhedral

Three types of contact between bodies are possible : grazing, separated and penetrated contact. Since grazing contact alone is a rare situation and too restrictive, methods were first developed for grazing or separated contact (the two objects might be in contact after the time step Δt but aren't at the beginning of it). All these methods have now also been extended to penetrating contact.

A first group of methods consists of finding a separating plane (dividing the space into two subspaces, each one containing one object). An example of such planes is given in Figure B.14. Finding such a (non-unique) plane means that the objects are not overlapping. This method is called the separating axis or shadow overlap method (for more information, see [17] as cited by F. DUBOIS). However, having such a plane doesn't give any information on the distance between the objects. Several methods are possible to compute the minimum distance between two convex bodies : an overview of these methods is given in [16].

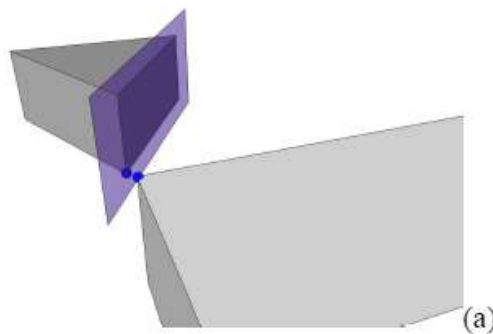


Figure B.14 - Separating plane between two objects (part I) [16]

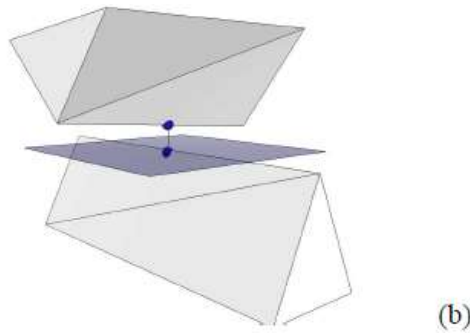


Figure B.15 - Separating plane between two objects (part II) [16]

Once the distance (and contact normals) between the objects are determined, the contact locus where the forces will have to be defined. This can be done as follows (see Figure B.16) :

- (a) Build a common plane between the two objects
- (b) Map a subset of the vertices at a certain maximal distance of the plane
- (c) Build the convex hull of the nodes
- (d) Compute the intersection of these convex hulls
- (e) Chose contact points in this domain and rebuild the necessary parameters (distance, normal...)

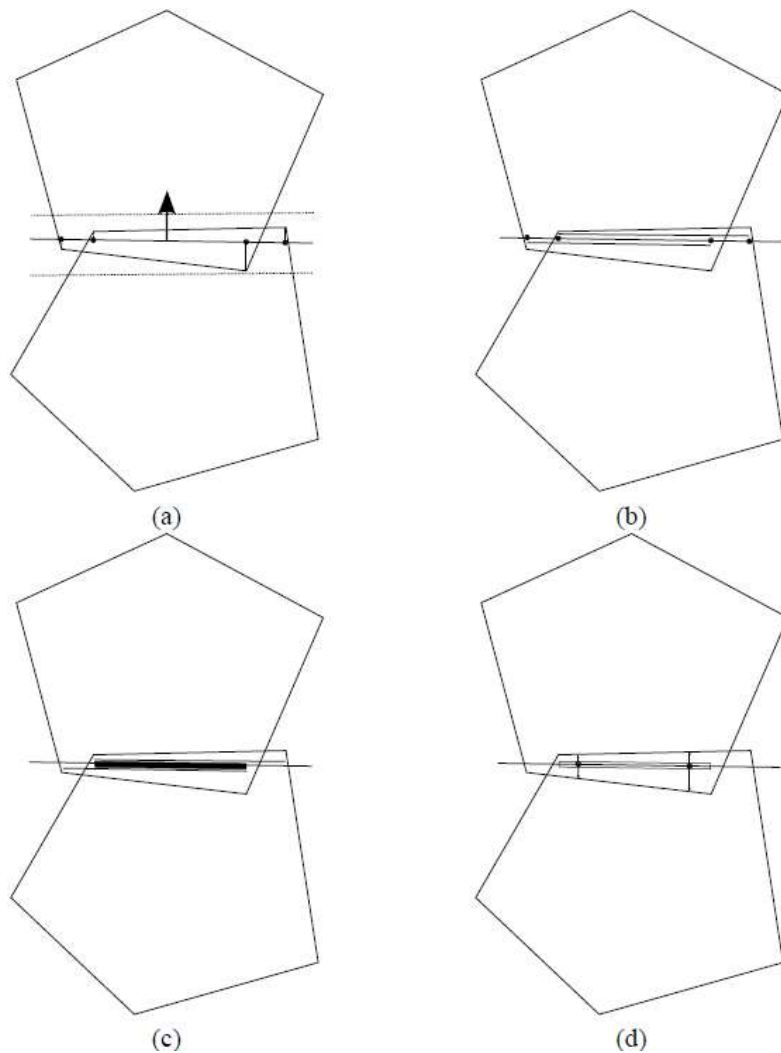


Figure B.16 - Steps to define the contact locus [16]

Contact detection : nonconvex polyhedral

The two most common approaches to deal with nonconvex polyhedral are the following :

- Decomposition of nonconvex objects into a set (cluster) of convex objects. This can be done either by hand or through a suitable algorithm.

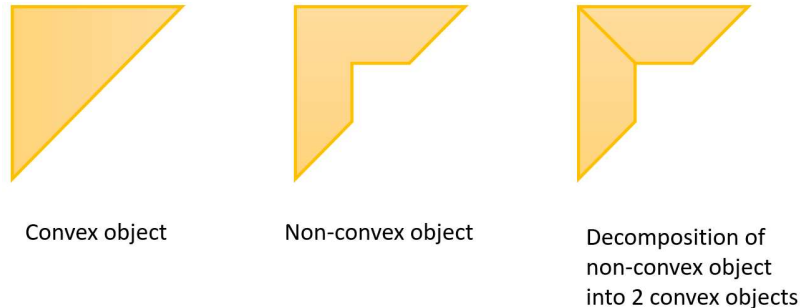


Figure B.17 - Example of object decomposition

- Using a bounding box hierarchy as explained before and represented in *Figure 4.4*.

Once the possible contact have been identified, the contact problem can be solved as explained before.

4 THE DROXEL AS A BODY IN LMGC90

The goal of this last part is to present how the geometry of the droxel is implemented into *LMGC90*. In order to reduce the computation time of the program, the droxel is divided into different boundary volumes as described before.

4.1 Brief introduction to LMGC90

The choice for a platform used for the computation of droxelised structures is *LMGC90*, its coding language is *Python*.

As explained by its creators, “LMGC90 is an open platform dedicated to the modelling of large collections of interacting objects (2D/3D). It aims at modelling objects of any shape with various mechanical behaviour and to take into account interaction laws as complex as necessary. Furthermore multi-physics couplings (thermal effects, fluids, etc) are progressively taken into account.” [15]

LMGC90 being open-source, it has the advantage of allowing the addition of new geometries, behaviours and interaction laws as well as contact detection methods and numerical strategies. *LMGC90* works both with rigid and elastic bodies.

The structure and explanation of the code for rigid (and elastic) bodies has already been done by Mathieu Mullier during his master thesis, it implements the parts : Contact dynamic method, Non Smooth Contact Dynamics Method (NSCD) and contact detection. [40]

4.2 Modelling of a droxel

Geometrical parameters of a droxel

The general geometry of a droxel is determined by 8 parameters : L, E, C_r, A, H, R, B, F [18].

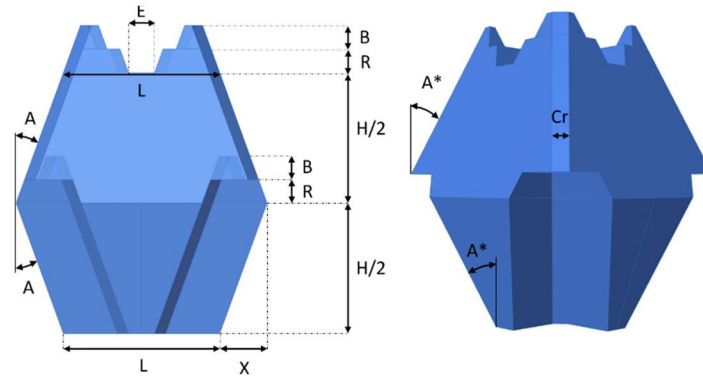


Figure B.18 - 8 parameters of a droxel

The parameter F is defined by : $\frac{B_1}{B_2}$ [%] (see Figure B.19).

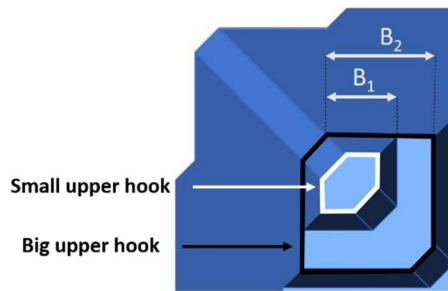


Figure B.19 - F parameter of a droxel

Geometrical description in LMGC90

For the moment, only one specific geometry of droxels has been implemented in *LMGC90*. This has been done in M. MULLIER's thesis. The parameters of this droxel are the following : $h = 250$ [mm], $L = 150$ [mm], $E = 25$ [mm], $Cr = 23$ [mm], $A = 20$ [°], $R = 25.2$ [mm], $B = 7.6$ [mm], $F = 56$ [%]

This *.gmsh* file has been created by importing a *.stp* file containing the mesh of the droxel. The physical surfaces, used for contact detection are then visualised and named in the *.gmsh* file. For the command lines to execute this, see [40]. The different surfaces are defined as shown in Figure B.22.

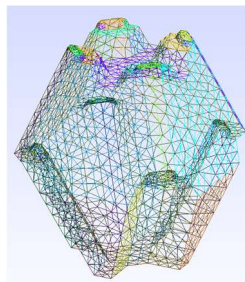


Figure B.20 - Meshing of the droxel (1771 nodes)

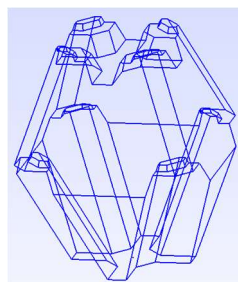


Figure B.21 - 3D view of the gmsh surfaces of a droxel

Division of the droxel geometry for contact detection

Dividing the total surface of the droxel makes it possible to create interaction tables to indicate which surface of a droxel can enter in contact with another surface from another droxel. Those protagonist-antagonist pairs are : *down – top*; *down v – top v*; *down ev – top ev*; *oblique bot – oblique top*; *mid e – top e*; *mid p – top p*; *encoches – top top*. They are represented in *Figure B.22*.

Physical model of a single droxel

Using a Signorini-Coulomb frictional contact law (no interpenetration and dry friction, see 1.6 *Frictional contact laws*) gives us the resultant reactions on the droxel for each subsurface (see *Figure B.23*). In reality, the forces are computed at every point of the meshing of the droxel (meshing represented in *Figure B.20*), only the resultant are represented in *Figure B.23*.

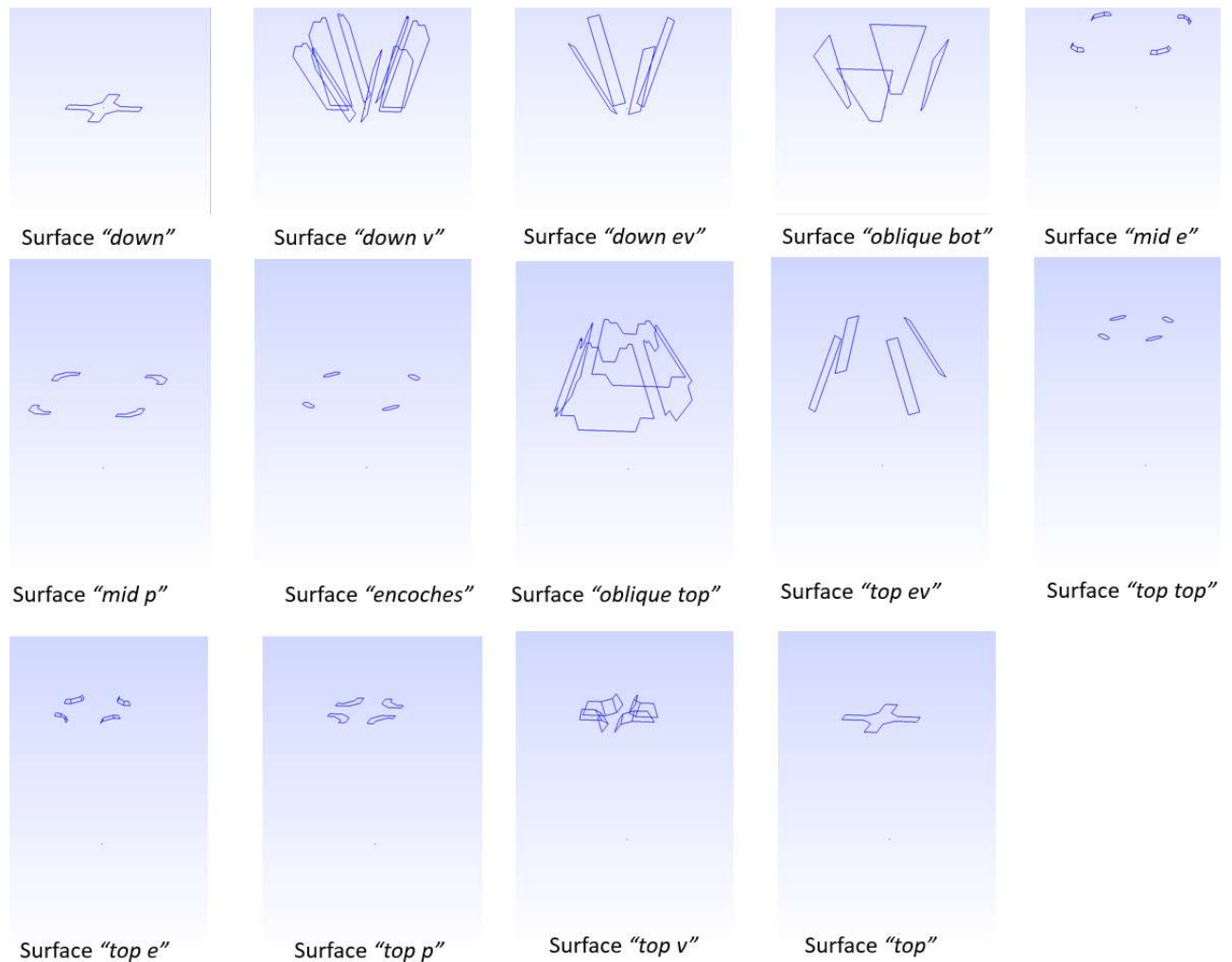


Figure B.22 - Boundary volumes (BV) for droxels

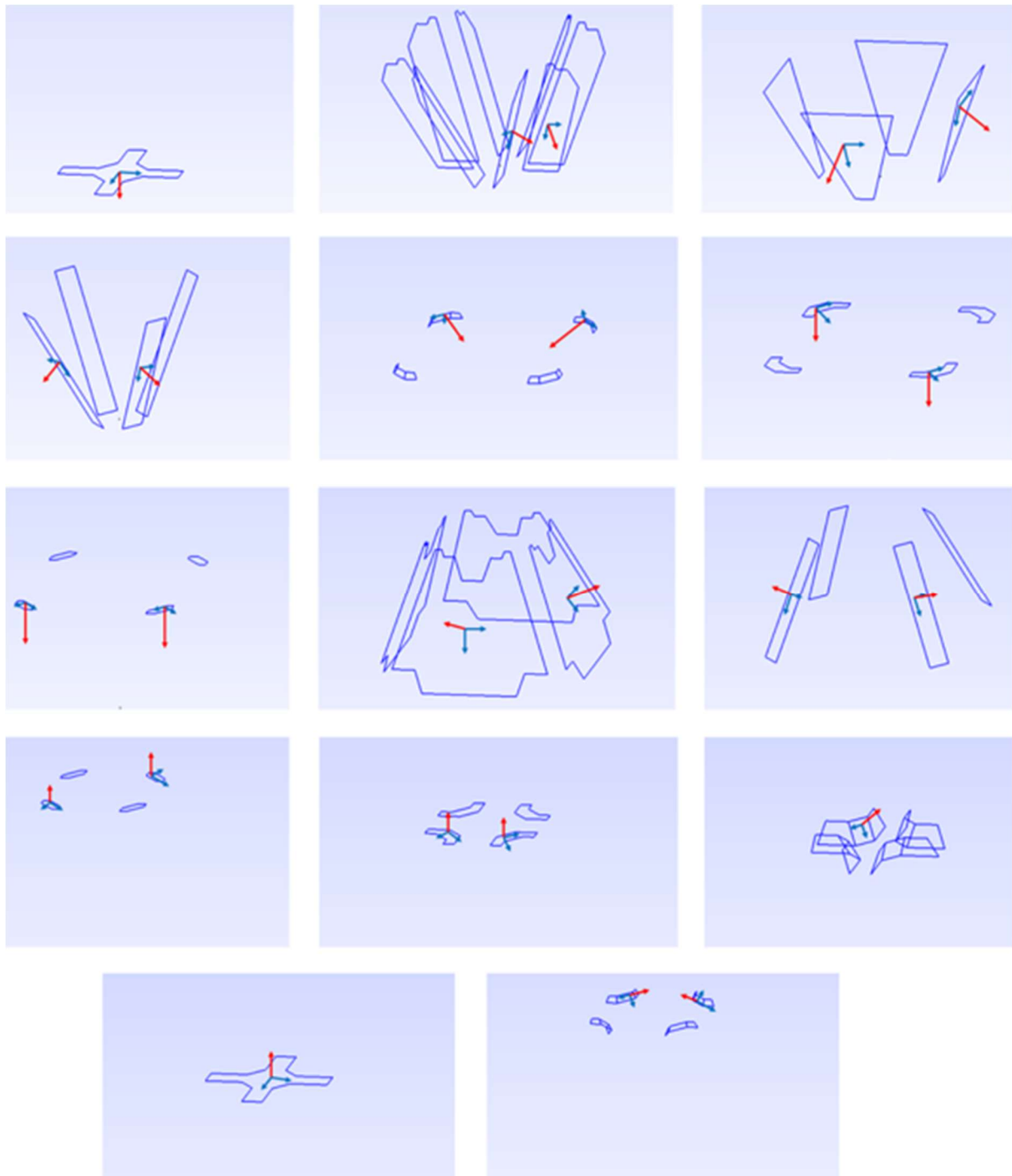


Figure B.23 - Representation of the resultant forces acting on the surfaces : friction forces in blue, normal forces in red

APPENDIX C: EVOLUTION OF THE PRINTED DROXEL GEOMETRY

Seven different droxel geometries were tried before deciding the final one. The choices between the evolution of the geometry is given below :

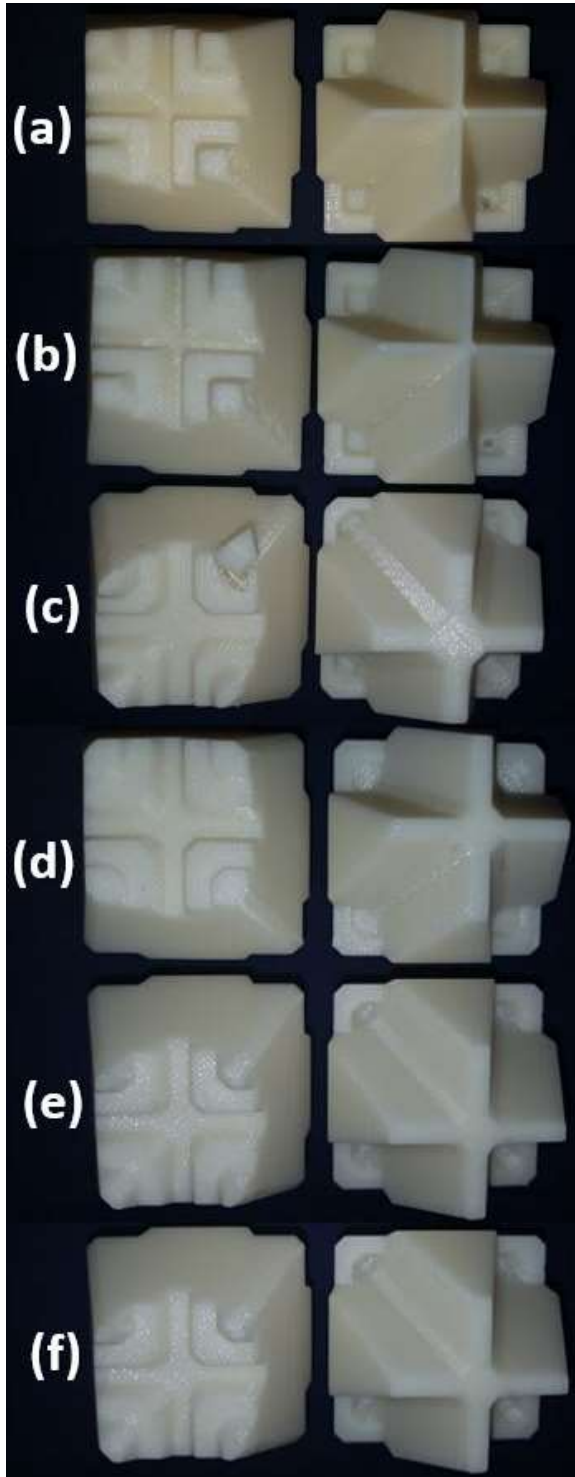


Figure C.1 - Evolution of the different droxel geometries

(a) → (b)

Modification : Augmenting E and Cr

Reason : Avoid thin edges and sharp angles

Conclusion : edges still rather thin, no noticeable difference

(b) → (c)

Modification : Augmenting Cr , B and E

Reason : Allow more droxels in cantilever mode, avoid thin edges and sharp angles

Conclusion :

- 1) Since the droxels are printed upside down, several heads broke down because they could not support the weight of the droxels during the printing.
- 2) Their interlocking is not optimal (see Figure C.2)

(c) → (d)

Modification : Decreasing B

Reason : Avoid broken heads

Conclusion :

- 1) The maximum number of droxels allowed in cantilever mode is considerably smaller than before (as explained in 2.3.6, decreasing B decreases the maximum number of droxels in cantilever mode).
- 2) The interlocking is still not optimal (see Figure C.3).

(d) → (e)

Modification : Same geometry as in c), printing the droxels from bottom to top (using more support material), small offset on some faces (see Figure C.4)

Reason : improve the interlocking capacities, impose 3 contact faces (see Figure C.5).

Conclusion :

- 1) The geometry is ok
- 2) The maximum number of droxels allowed in cantilever has increased
- 3) The interlocking is still not optimal

(e) → (f) : final geometry is determined

Modification : Lifting the central octagon, maximal extrusion of the 4 holes

Reason : counterbalance the warping effects during impression, improve impression quality. (see Figure C.6).

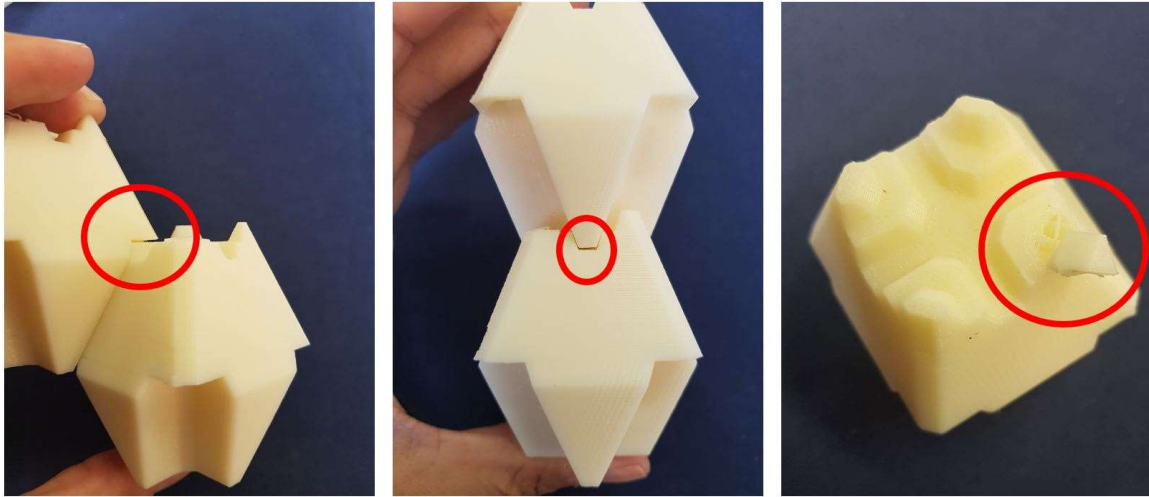


Figure C.2 - Configuration (c): the interlocking is not optimal, some droxel heads broke down during the printing

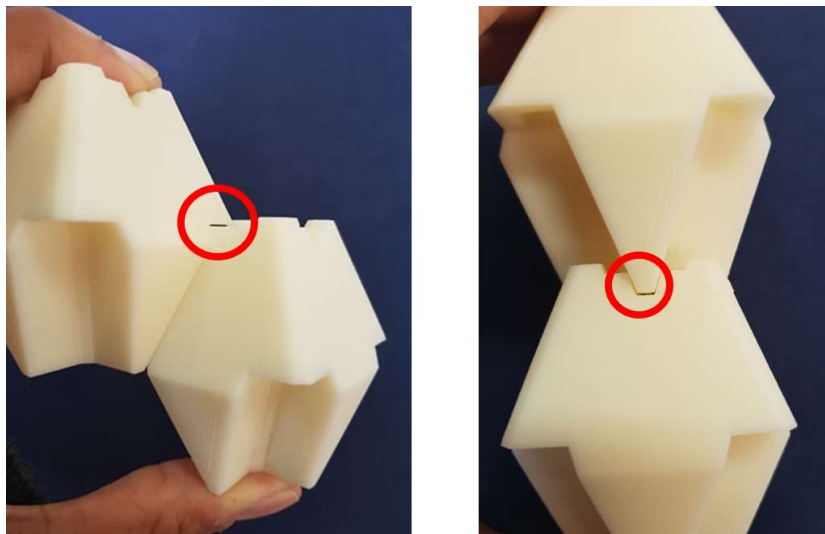


Figure C.3 - Configuration (d): the interlocking is not optimal

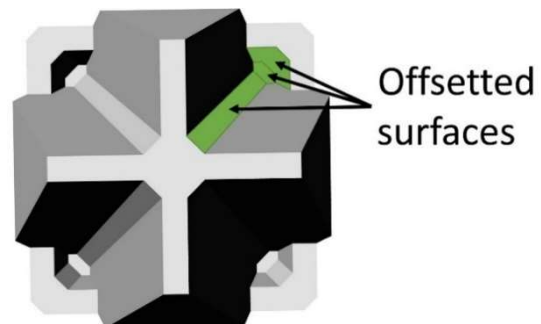


Figure C.4 - Configuration (e): Offsetted surfaces

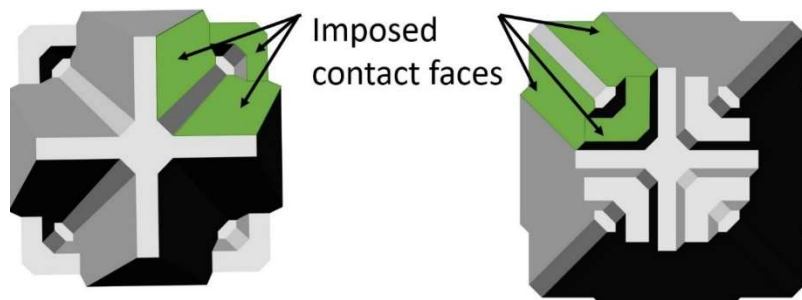


Figure C.5 - Configuration (e): Imposed contact faces for droxels

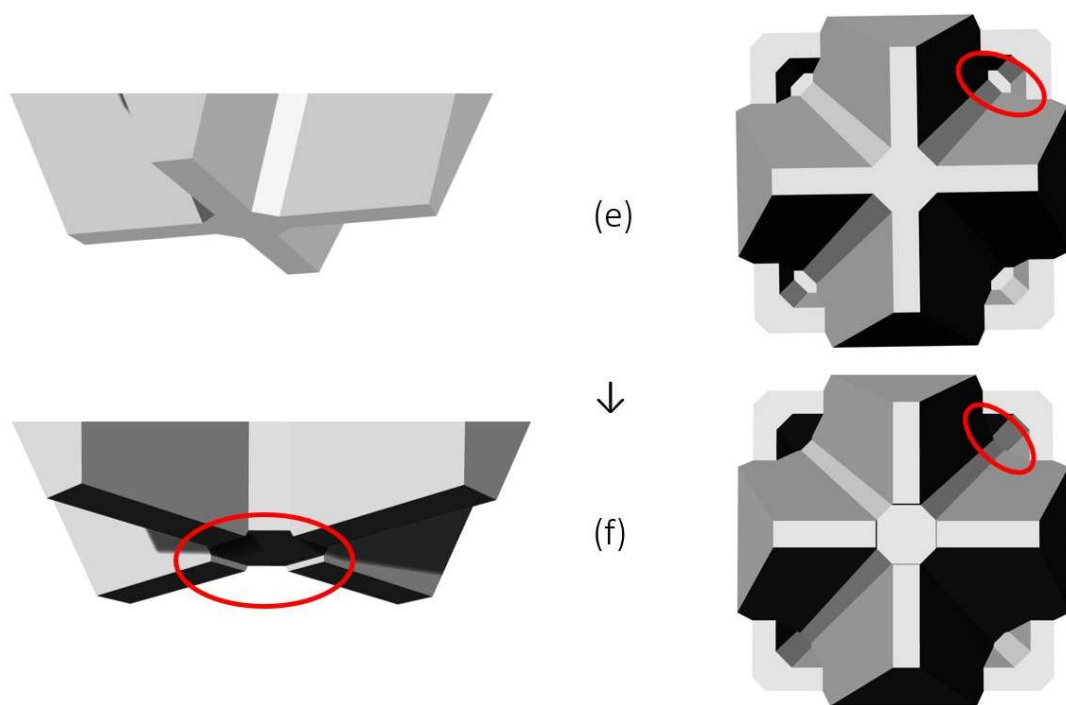


Figure C.6 - Configuration (f): Lifting of the central octagon and complete extrusion of the holes

TABLE OF APPENDICES

Appendix A: Laying sequences - full examples.....	121
A.1 Arch	121
A.2 Small shelter	124
A.3 Limits of the algorithm: example	127
A.4 Bridge	129
A.5 Square Fence	132
Appendix B: Numerical modelling of rigid discrete structures	135
Introduction to modelling of discrete structures.....	135
General principle of the contact dynamic (CD) method	135
Content of this appendix	136
Some considerations done in this document.....	136
1. Contact Dynamics method	137
1.1 Introduction.....	137
1.2 Variables involved	137
1.3 Kinematics	138
1.4 Equations of Dynamics	138
1.5 Shocks.....	139
1.6 Frictional contact laws.....	139
1.7 Conclusion : equations for frictional contact problem.....	141
2 Non Smooth Contact Dynamics method (NSCD).....	142
2.1 Resolution scheme of NSDC	142
2.2 Frictional contact problem resolution.....	143
2.3 Implementation of NLGS (contact problem resolution).....	146
2.3 Conclusion	147
3 Contact detection.....	147
3.1 Introduction.....	147
3.2 Geometrical description	147
3.3 Sorting algorithms	148
4 The droxel as a body in LMGC90.....	152
4.1 Brief introduction to LMGC90	152
4.2 Modelling of a droxel.....	152
Appendix C: Evolution of the printed droxel geometry	157

TABLE OF FIGURES

Figure 1 - Droxel geometry : perspective and bottom view.....	1
Figure 2 - Example of possible self-made agricultural shelters out of droxels [15]	1
Figure 3 - Third goal : comparing the stability algorithm with the numerical simulations and experimental tests.....	2
Figure 4 - Example of the droxelisation of a surface.....	2
Figure 5 - Concrete or recycled plastic droxels for housing [15].....	3
Figure 1.1 - Traditional wall: bricks and mortar [10].....	5
Figure 1.2 - Interlocking blocks developed by R. Thallon [45]	6
Figure 1.3 - Mecano block system, Cetholic [11]	6
Figure 1.4 - Azar block	7
Figure 1.5 - Haener hollow blocks [28].....	7
Figure 1.6 - Example of 3 types of PUTRA BLOCKS [46]	7
Figure 1.7 - IITM HILBLOCK (dimensions in mm) [6]	7
Figure 1.8 - FCS Block, example of a house built with FCS blocks [20][21]	8
Figure 1.9 - Stepoc blocks [21]	8
Figure 1.10 - Spar-lock blocks [21]	8
Figure 1.11 - Example of a corner wall with Armo system [48]	8
Figure 1.12 - Example of a wall built with hydraform blocks [21].....	9
Figure 1.13 - Tanzania Interlocking Brick [28]	9
Figure 1.14 - CFRC Interlocking Blocks. On the right the dynamical test of a CFRC column [5]	9
Figure 1.15 - Muraces: an example of blocks developed for retaining walls.....	10
Figure 1.16 - LEGO inspired blocks by Abang A. et al [2].....	10
Figure 1.17 - Universiti Teknologi Malaysia [40]	10
Figure 1.18 - Osteomorphic blocks in different configurations [42]	11
Figure 1.19 - Tetraloc blocks in different configurations [47].....	11
Figure 1.20 - Blocks used by the architect Kengo Kuma [24][37].....	11
Figure 1.21 - Conical drick with a laying tolerance of 6.5 cm. [24]	12
Figure 1.22 - 8 types of rectangular dricks allowing to realise straight wall configurations [24]	12
Figure 1.23 - Dimensions of an extrudable drick [24]	13
Figure 2.1 - Step-by-step design philosophy of the droxel (part I) [24]	15
Figure 2.2 - Step-by-step design philosophy of the droxel (part II) [24]	16
Figure 2.3 - Droxel parameters, X and A*	16
Figure 2.4 - Top view of $\frac{1}{4}^h$ droxel: droxel parameter $F = B_1/B_2$	17
Figure 2.5 - Different possible droxels shapes obtained by varying the 8 parameters.....	17
Figure 2.6 - Influence of H on the droxel geometry	17
Figure 2.7 - Influence of H on the final geometry of a droxel wall.....	18
Figure 2.8 - Influence of L on the droxel geometry.....	18
Figure 2.9 - Influence of L on the final geometry of a droxel wall	19
Figure 2.10 - Influence of L on the ground surface of a structure (top view)	19
Figure 2.11 - Influence of A on the droxel geometry	19

Figure 2.12 - Maximum number of droxels in cantilever mode depends on A (and on the friction coefficient).....	19
Figure 2.13 - Possible angles in function of the orientation	20
Figure 2.14 - Slope change with transition droxels, as represented by G. GUILMIN [26]	21
Figure 2.15 - Influence of E on the droxel geometry, perspective and top view	21
Figure 2.16 - Influence of E on the openings in the structure.....	21
Figure 2.17 - Influence of R on the droxel geometry	22
Figure 2.18 - Influence of B on the droxel geometry	22
Figure 2.19 - Example of a cantilever structure (possible for $B > 0$)	22
Figure 2.20 - Influence of C_r on the droxel geometry.....	23
Figure 2.21 - Influence of F on the droxel geometry.....	23
Figure 2.22 - Laying tolerance of a droxel, as represented in [35].....	24
Figure 2.23 - Summary chart with the influences of the different parameters [35].....	24
Figure 2.24 - Example of the vault effect in a structure of droxels.....	25
Figure 2.25 - Example of the realisation of an arch in droxels: temporary counterbalancing droxels in pink	26
Figure 2.26 - Realisation of an arch in droxels: temporary (ballasted) droxels in blue (part I).....	26
Figure 2.27 - Realisation of an arch in droxels: temporary (ballasted) droxels in blue (part II).....	27
Figure 2.28 - Straight columns composed of one droxel: risk of instability	28
Figure 3.1 - General steps necessary to realise a droxelised structure.....	29
Figure 3.2 - Example of an unstable droxel configuration and a possible stabilisation mode.....	30
Figure 3.3 - Global structure of the Grasshopper code.....	30
Figure 3.4 - Example of the visualisation of a structure with Rhino	31
Figure 3.5 - Grasshopper interface (part I).....	31
Figure 3.6 - Grasshopper interface (part II).....	32
Figure 4.1 - Comparison between the three methods.....	33
Figure 4.2 - DEM : representation of the tasks executed over a time step [16]	34
Figure 4.3 - Broad detection: Space division for interacting spheres [17]	35
Figure 4.4 - Narrow phase: bounding box hierarchy.....	35
Figure 4.5 - Example of a 2D contact detection method: Sweep and Prune method.....	35
Figure 4.6 - Coulomb's friction model.	36
Figure 4.7 - Simulation of a structure with different time steps.....	37
Figure 4.8 - Computation time for different (stable) droxel structures ($dt = 0.15$, 300 steps)	37
Figure 4.9 - Droxels on the ground floor are fixed during the whole simulation.....	38
Figure 4.10 - GMSH program: from .stp volume to .msh meshing	38
Figure 4.11 - Resultant of forces on droxel surfaces. Normal force in red, friction force in green	39
Figure 4.12 - Final droxel geometry	39
Figure 4.13 - Reminder: geometry of a droxel (F-factor not represented)	40
Figure 4.14 - Lattice-like structure in a droxel (left), determination of μ (right)	40
Figure 4.15 - Numerical simulation with $\mu = 0.61$: no collapsing. Friction coefficient overestimated .	41
Figure 4.16 - Droxels for the base layers.....	41
Figure 4.17 - Wooden base plate for the droxels.....	41

Figure 4.18 - Example of droxels embedded in the wooden base	42
Figure 5.1 - Chapter 5: droxelisation of a surface	43
Figure 5.2 - Droxelisation examples: domes, bridges, tunnels, shelters.....	44
Figure 5.3 - Example of a floating droxel in red	44
Figure 5.4 - Possible solutions when the angle cannot be realised with the chosen droxel	45
Figure 5.5 - Representation of an unstable droxelised structure and its instability mode.....	45
Figure 5.6 - Reminder: Geometry of a droxel, $F = B_1/B_2$	46
Figure 5.7 - Scaling of a droxel: only the angle A and the parameter F remain the same	46
Figure 5.8 - Influence of the droxel size on the resolution of the structure	46
Figure 5.9 - Influence of the width on the droxelised structures.....	47
Figure 5.10 - Influence of the rotation of the grid	47
Figure 5.11 - First droxelisation possibility.....	48
Figure 5.12 - Representation of the second droxelisation method	49
Figure 5.13 - Droxelisation example: 3D surface centered around the origin	50
Figure 5.14 - Droxelisation example: computation of all the possible base points around the surface	50
Figure 5.15 - Droxelisation example: keeping points at distance d_{max} from the surface	50
Figure 5.16 - Droxelisation example: identification of floating droxels	51
Figure 5.17 - Droxelisation example: normalised coordinates of the structure	51
Figure 5.18 - Droxelisation example: computation of the real base point coordinates	51
Figure 5.19 - Droxelisation example 2: from normalised to real point coordinates.....	52
Figure 5.20 - Flowchart of the two possible droxelisation methods.....	53
Figure 5.21 - Part of the Grasshopper program corresponding to the droxelisation	54
Figure 6.1 - Chapter 6: Global stability of a structure	55
Figure 6.2 - Local rupture on a (concrete) droxel [40]	56
Figure 6.3 - Example of a globally stable and unstable structure, representation of the instability mode	56
Figure 6.4 - Stability of a structure: principle [24].....	57
Figure 6.5 - Different possible stacking modes of droxels	57
Figure 6.6 - Possible unhooking of the droxels in modes 1B, 1AB and 2AB.....	58
Figure 6.7 - Determination of a group of droxels (part I).....	58
Figure 6.8- Determination of a group of droxels (part II).....	59
Figure 6.9 - Example of an unhooked and hooked group	59
Figure 6.10 - Convex hull of a set of points : example	59
Figure 6.11 - Determination of the complex hull for droxels.....	60
Figure 6.12 - Example of the gravity test for the light pink group. The test is passed.....	60
Figure 6.13 - Experimental validation with 3D printed droxels: the structure is stable	61
Figure 6.14 - Stability determination	61
Figure 6.15 - Flowchart : stability algorithm check of a group.....	62
Figure 6.16 - For structure (a) a laying sequence can be found, for (b) and (c) no laying sequences are possible.....	62
Figure 6.17 - Concept of floors [26].....	63

Figure 6.18 - Creation of a droxel in Grasshopper, calculation and connection of the corner points..	63
Figure 6.19 - example of an arch-like structure	64
Figure 6.20 - Example: numbering of the droxels	64
Figure 6.21 - Example: the droxel coordinates correspond to their base point (black points)	65
Figure 6.22 - Example: normalised coordinates.....	65
Figure 6.23 - Index of the neighbours (of the beige droxel)	65
Figure 6.24 - Example: branches of the connectivity tree corresponding to droxel 0 and 1.....	66
Figure 6.25 - Second instability mode : not considered in the algorithm	66
Figure 6.26 - Reminder : mode 1A.....	67
Figure 6.27 - Determination of p [35]	67
Figure 6.28 - Determination of d [35], $T^* = \tan(A^*)$	67
Figure 6.29 - Maximum value of d/p for given values of ϕ and A^*	68
Figure 6.30 - Determining d and p	68
Figure 6.31 - Limitations of the algorithm: hooked is binary	69
Figure 6.32 - Illustration of the algorithm for a stable example (part I)	70
Figure 6.33 - Illustration of the algorithm for a stable example (part II)	71
Figure 6.34 - Example 6.1 : numerical simulation after $t = 45$ [sec] : the structure is stable.....	72
Figure 6.35 - Example 6.1 : experimental test.....	72
Figure 6.36 - Illustration of the algorithm for an unstable example	72
Figure 6.37 - Example 6.2: numerical simulation using LMGC90	73
Figure 6.38 - Example 6.2: experimental test.....	73
Figure 6.39 - Example 6.3: Grasshopper algorithm.....	73
Figure 6.40 - Example 6.3: numerical simulation	74
Figure 6.41 - Example 6.3: experimental: the structure is unstable	74
Figure 6.42 - Example 4: Grasshopper algorithm.....	75
Figure 6.43 - Example 4: lifting of the keystone (experimental test).....	75
Figure 6.44 - Influence of a heavier keystone or its anchoring to the ground with a cable	75
Figure 6.45 - Lifting of the keystone: numerical simulations	76
Figure 6.46 - Example 6.5: Grasshopper algorithm.....	76
Figure 6.47 - Example 6.5: experimental test: the structure is stable	77
Figure 6.48 - Example 6.5: numerical simulations : the structure stays stable.....	77
 Figure 7.1 - Chapter 7: Laying sequence	 79
Figure 7.2 - Building of an arch with temporary droxels (in pink) from steps (d) to (f)	80
Figure 7.3 - Building of a (stable) gate: impossible to build the structure.....	80
Figure 7.4 - Instability mode caused when trying to execute any step (c) of Figure 7.3	81
Figure 7.5 - Cone of influence: in blue the cone of influence of the beige droxel.....	81
Figure 7.6 - Representation of substructures.....	82
Figure 7.7 - Notion of inaccessible position	82
Figure 7.8 - Cone of influence of a substructure: the droxels in green are the cone of influence of the right part of the arch	82
Figure 7.9 - Accessible positions in green: 4 possible candidates (in the plane of the structure) to stabilise the structure.....	83
Figure 7.10 - Reminder : laying tolerance of droxels	86

Figure 7.11 - Improvement possibility: example	87
Figure 7.12 - Suggested improvement: example (part I)	87
Figure 7.13- Suggested improvement: example (part II)	88
Figure 7.14 - Example of a droxelised wall	89
Figure 7.15 - Example 7.1: screenshot of the Grasshopper program	89
Figure 7.16 - Laying sequence for a droxelised wall. The added droxel at every step is represented in green (part I).....	89
Figure 7.17 - Laying sequence for a droxelised wall. The added droxel at every step is represented in green (part II).....	90
Figure 7.18 - Example of a droxelised arch	90
Figure 7.19 - Example 7.2: screenshot of the Grasshopper program	90
Figure 7.20 - Steps for the realisation of an arch. Last placed droxel in green, temporary droxel in blue. (PART I)	91
Figure 7.21 - Steps for the realisation of an arch. Placed droxel in green, temporary droxel in blue. (PART II)	92
Figure 7.22 - Steps for the realisation of an arch. Placed droxel in green, temporary droxel in blue. (PART III)	93
Figure 7.23 - Example of a small shelter	94
Figure 7.24 - Example 7.3: screenshot of the Grasshopper program	94
Figure 7.25 - Important steps for the building of the shelter. Placed droxel represented in green, temporary droxel in blue.....	94
Figure 7.26 - Example of a stable final structure that cannot be built.....	95
Figure 7.27 - Example 7.4: screenshot of the Grasshopper program	95
Figure 7.28 - Laying sequence of the structure in Figure 7.26	95
Figure 7.29 - Example 7.4: it is not possible to add a droxel after step 4	95
Figure 7.30 - Limits of the algorithm: example	96
Figure 7.31 - Example 7.5 : screenshot of the Grasshopper program	96
Figure 7.32 - Example 7.5: laying sequence	96
Figure 7.33 - Example 7.5: final structure simulation and experimental test.....	97
Figure 7.34 - For a bigger similar structure, a total collapsing of the structure occurs	97
Figure 7.35 - Example 7.6: Grasshopper algorithm.....	98
Figure 7.36 - Example 7.6 : screenshot of the Grasshopper program	98
Figure 7.37 - Example 7.6: experimental test.....	98
Figure 7.38 - Influence of a heavier keystone	99
Figure 7.39 - Numerical simulation of the bridge after 45 [sec]	99
Figure 8.1 - Example 8.1: simple wall	101
Figure 8.2 - Example 8.1: Laying sequence given by the Grasshopper algorithm.....	101
Figure 8.3 - Example 8.1 : simulation of the wall after $t = 30$ [sec].....	102
Figure 8.4 - Example 8.1: Laying sequence: experimental test.....	102
Figure 8.5 - Example 8.2: Arch.....	103
Figure 8.6 - Example 8.2 (part I)	103
Figure 8.7 - Example 8.2 (part II)	104
Figure 8.8 - Example 8.2 : the three methods predict the same results	105

Figure 8.9 - Example 8.3: small shelter	105
Figure 8.10 - Example 8.3 (part I)	105
Figure 8.11 - Example 8.3 (part II)	106
Figure 8.12 - Example 8.3 (part III)	107
Figure 8.13 - Example 4: gate	107
Figure 8.14 - Information message from Grasshopper when the gate is given as input structure	108
Figure 8.15 - Example 8.4	108
Figure 8.16 - Example 8.5: limits of the algorithm	109
Figure 8.17 - Example 8.5 (part I)	109
Figure 8.18 - Example 8.5 (part II)	110
Figure 6 - Openings in the structure: proportional to E	114
Figure 7 - Moulding of a droxel [40]	115
Figure 8 - Slope change in droxelised structures [26].....	115
Figure A.1 - Example 2 (arch): Laying sequence (part I)	121
Figure A.2 - Example 2 (arch): Laying sequence (part II)	122
Figure A.3 - Example 2 (arch): Laying sequence (part III)	123
Figure A.4 - Example 3 (small shelter): Laying sequence (part I)	124
Figure A.5 - Example 3 (small shelter): Laying sequence (part II)	125
Figure A.6 - Example 3 (small shelter): Laying sequence (part I)	126
Figure A.7 - Example 5 (limit of the algorithm): Laying sequence (part I).....	127
Figure A.8 - Example 5 (limit of the algorithm): Laying sequence (part II).....	128
Figure A.9 - Example 6 (bridge): Laying sequence (part I).....	129
Figure A.10 - Example 6 (bridge): Laying sequence (part II).....	130
Figure A.11 - Example 6 (bridge): Laying sequence (part III).....	131
Figure A.12 - Example 7 (square fence): Laying sequence (part I)	132
Figure A.13 - Example 7 (square fence): Laying sequence (part II)	133
Figure A.14 - Example 7 (square fence): Laying sequence (part III)	134
Figure B.1 - Tasks executed over a time step [16].....	136
Figure B.2 - Distance between two objects, definition of the local frame [16]	137
Figure B.3 - Graphical representation of the Signorini gap condition [16]	140
Figure B.4 - Graphical representation of Coulomb's law [16]	141
Figure B.5 - Duality scheme : link between the different equations [16]	142
Figure B.6 - Resolution scheme of NSCD [14].....	142
Figure B.7 - Stored Delassus Loop Strategy [14]	146
Figure B.8 - Exchange Local Global Strategy [14]	146
Figure B.9 - Parallel SDL treatment [14]	147
Figure B.10 - Examples of possible bounding volumes [14].....	148
Figure B.11 - Broad detection : Space division [16]	149
Figure B.12 - 2D representation of the Sweep and Prune method.....	149
Figure B.13 - 2D Representation of bounding box hierarchy.....	150
Figure B.14 - Separating plane between two objects (part I) [16]	150
Figure B.15 - Separating plane between two objects (part II) [16]	151

Figure B.16 - Steps to define the contact locus [16]	151
Figure B.17 - Example of object decomposition.....	152
Figure B.18 - 8 parameters of a droxel.....	153
Figure B.19 - F parameter of a droxel.....	153
Figure B.20 - Meshing of the droxel (1771 nodes).....	153
Figure B.21 - 3D view of the gmsh surfaces of a droxel	153
Figure B.22 - Boundary volumes (BV) for droxels	154
Figure B.23 - Representation of the resultant forces acting on the surfaces : friction forces in blue, normal forces in red	155
Figure C.1 - Evolution of the different droxel geometries	157
Figure C.2 - Configuration (c): the interlocking is not optimal, some droxel heads broke down during the printing	158
Figure C.3 - Configuration (d): the interlocking is not optimal.....	158
Figure C.4 - Configuration (e): Offsetted surfaces.....	158
Figure C.5 - Configuration (e): Imposed contact faces for droxels.....	158
Figure C.6 - Configuration (f): Lifting of the central octagon and complete extrusion of the holes...	159

BIBLIOGRAPHY

- [1] "A tour of the C# language", <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>, January 2021, last accessed June 2021
- [2] ABANG ALI A. A. and ABDUL KADIR M. R., "Strength properties and structural performance of interlocking hollow block walls," *Jurnal Institut Jurutera Malaysia*, no. Bil.53, pp. 25–35, 1987.
- [3] "About Grasshopper...", <https://www.grasshopper3d.com/>, last accessed May 2021
- [4] ALART P., "Comment gérer en dynamique non régulière des contacts l'indétermination et l'interpénétration dans les milieux granulaires?," *Csma* 2013, pp. 1–8, 2013.
- [5] ALI M., BRIET R., BAI S. and CHOUW N., "Seismic Behaviour of Mortar-free Interlocking Column," *New Zealand Society for Earthquake Engineering Conference*, no. 40, p. 78, 2013.
- [6] ANAND K. B., VASUDEVAN V. and RAMAMURTHY K., "Water permeability assessment of alternative masonry systems," *Building and Environment*, vol. 38, no. 7, pp. 947–957, 2003, doi: 10.1016/S0360-1323(03)00060-X.
- [7] AVRIL Q., "Détection de Collision pour Environnements Large Échelle: Modèle Unifié et Adaptatif pour Architectures Multi-coeur et Multi-GPU," 2011.
- [8] "BLOC TITAN®, le bloc béton empilable haute qualité." <https://bloctitan.fr/>.
- [9] "BLOCS DE BETON EMPILABLES." <https://www.btp-werey.fr/actualites/nouveaute/>.
- [10] "Brick wall." https://www.designingbuildings.co.uk/wiki/Brick_wall.
- [11] Cetholic, *Mortarless Masonry—The Mecano System*. Lima, Peru: Housing Science, Vol. 12, No. 2, 1998.
- [12] COHEN J. D., LIN M. C., MANOCHA D. and PONAMGI M., "I-COLLIDE: An interactive and exact collision detection system for large-scale environments," *Proceedings of the Symposium on Interactive 3D Graphics*. pp. 189–196, 1995.
- [13] "Convex Hull using Divide and Conquer Algorithm," 2018. <https://www.geeksforgeeks.org/convex-hull-using-divide-and-conquer-algorithm/>.
- [14] "C# Multithreading", <https://www.geeksforgeeks.org/c-sharp-multithreading/>, updated April 2020, last accessed June 2021
- [15] "DROXELS : the universal construction component", <https://www.youtube.com/watch?v=EwIXBx9F1Uk>, last accessed in May 2021
- [16] DUBOIS F. and RADJAÏ F., "Discrete numerical modeling of granular materials." 2010.
- [17] DUBOIS F. and RENOUF M., "Discrete Element Methods for the simulation of divided media," 2011.
- [18] DUBOIS F., JEAN M., RENOUF M., MOZUL R., MARTIN A., et al.. "LMGC90". 10e colloque national en calcul des structures, May 2011, Giens, France. hal-00596875.

- [19] EBERLY D., "Intersection of convex objects: the method of separating axes," Geometric Tools Inc, no. 1, pp. 1–20, 2003, [Online]. Available: <http://geomerictools.com/Documentation/MethodOfSeparatingAxes.pdf>.
- [20] FAY L., COOPER P. and DE MORAIS H. F., "Innovative interlocked soil-cement block for the construction of masonry to eliminate the settling mortar," Construction and Building Materials, vol. 52, pp. 391–395, 2014, doi: 10.1016/j.conbuildmat.2013.11.030.
- [21] FAYEZ M. K. M., "Behavior of Masonry Walls Constructed Using Locally Produced Dry-Stack Interlocking Units," no. May 2015, 2009, doi: 10.13140/RG.2.2.21744.35848.
- [22] "FCS INTERLOCKING BRICK SYSTEM (Faster Construction Solutions)." <https://www.youtube.com/watch?v=WmOKUACg4ul>.
- [23] GEUZAIN C. and REMACLE J.-F., "Hmsh : A three-dimensional finite element mesh generator with built-in pre- and post-processing facilities." <https://gmsh.info/> (accessed Apr. 17, 2021).
- [24] GOESSENS S., "Drone-compatible additive manufacturing of building-scale structures," 2019.
- [25] GOESSENS S., MUELLER C., and P. LATTEUR P., "Feasibility study for drone-based masonry construction of real-scale structures," Automation in Construction, vol. 94, no. June, pp. 458–480, 2018, doi: 10.1016/j.autcon.2018.06.015.
- [26] GUILMIN G. Conception et calcul des structures composées d'éléments empilés de type droxels sur base d'une approche paramétrique. Ecole polytechnique de Louvain, Université catholique de Louvain, 2021. Prom. : Latteur, Pierre. <http://hdl.handle.net/2078.1/thesis:28342>
- [27] "Hadrian X® Completes Two Storey Structure | FBR", <https://www.youtube.com/watch?v=l1O1yRZO2Fg>, last accessed February 2021
- [28] HAENER, "Stacking Mortarless Block System," Engineering Design Manual, Hamilton, Ontario, 1984.
- [29] HASHEMI A., CRUICKSHANK H., and CHESHMEHZANGI A., "Environmental impacts and embodied energy of construction methods and materials in low-income tropical housing," Sustainability (Switzerland), vol. 7, no. 6, pp. 7866–7883, 2015, doi: 10.3390/su7067866.
- [30] HORIKAWA J., "Graham Scan Algorithm (Convex Hull) with Grasshopper C#," 2020. <https://www.youtube.com/watch?v=w6K0QxYiUvs> (accessed Jun. 02, 2020).
- [31] HUANG L., KRIGSVOLL G., JOHANSEN F., LIU Y. and ZHANG X., "Carbon emission of global construction sector," Renewable and Sustainable Energy Reviews, vol. 81, no. June 2016, pp. 1906–1916, 2018, doi: 10.1016/j.rser.2017.06.001.
- [32] "INTERLOCKING BRICKS." <http://www.ati.com.ph/fcs.bricks/>.
- [33] KADHUM Y. et al., "INTRODUCING A NEWLY DESIGNED INTERLOCKING MORTARLESS BLOCK SYSTEM FOR HOUSE," no. June, 2018.

- [34] KINTINGU S.H., "Design of interlocking bricks for enhanced wall construction, flexibility, alignment accuracy and load bearing," *Technology*, p. 262, 2009, [Online]. Available: <http://webcat.warwick.ac.uk/record=b2317844~S15>.
- [35] LATTEUR P., GOESSENS S. and FASCETTI A., "The droxel: a universal parametric construction component," *Construction and Building Materials*, submitted November 2020.
- [36] LEGAT V., "Méthodes Numériques : cours 8." p. 11, [Online]. Available: <https://perso.uclouvain.be/vincent.legat/documents/fsab1104//cours8/fsab1104-cours8.pdf>.
- [37] "Minimal Shelter." <https://campus.burg-halle.de/id-neuwerk/minimal-shelter/water-branch-house-detail/>.
- [38] MOREAU J. J., "Indetermination in the numerical simulation of granular systems," *Powders and Grains 2005 - Proceedings of the 5th International Conference on Micromechanics of Granular Media*, vol. 1, pp. 109–112, 2005.
- [39] MORRIS, J. Morris, "Finding the Convex Hull," 2004. https://www.cs.auckland.ac.nz/software/AlgAnim/convex_hull.html.
- [40] MULLIER M. Développement de structures composées de droxels. EPL, Université catholique de Louvain, 2018. Prom. : Latteur, Pierre.
- [41] "MURS DE SOUTÈNEMENT ET STRUCTURE VÉGÉTALISABLE." https://www.interbloccs.com/fr/page/murs_de_soutenement_et_structures_vegetalisable
- [42] "Polyhedron." <https://en.wikipedia.org/wiki/Polyhedron>.
- [43] "Python", <https://www.python.org/about/> , last accessed June 2021
- [44] "Stepoc : blocs de coffrage." <https://betondelalomme.be/fr/nos-produits/blocs-de-coffrage-stepoclomme/>.
- [45] THALLON R., "Dry-Stack Block," *Fine Homebuilding Magazine*, USA, pp. 50–57, Aug. 1983.
- [46] THANOON W. A., JAAFAR M. S., ABDUL KADIR M. R., ABANG ALI A. A., TRIKHA D. N., and NAJM A. M. S., "Development of an innovative interlocking load bearing hollow block system in Malaysia," *Construction and Building Materials*, vol. 18, no. 6, pp. 445–454, 2004, doi: 10.1016/j.conbuildmat.2004.03.013.
- [47] "Tetraloc : interlocking bricks." <https://tetraloc.com/the-tetraloc-system/>.
- [48] "Wall with armo buttress." <https://www.armo-system.com/products/contra-fuerte-armo/>.
- [49] RENIERS M., master thesis under the supervision of P. Latteur and S. Goessens, Development of elementary BIM tools for drone compatible construction systems, June 2016, UCL, Belgium. <http://hdl.handle.net/2078.1/thesis:14970> (last access June, 2, 2020)

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | www.uclouvain.be/epl