Horizon 2020

Societies of Symbiotic Robot-Plant Bio-Hybrids as Social Architectural Artifacts

Deliverable D3.1

Representations and design rules

Date of preparation: 2017/03/31	**Revision: 1 (r792)**
Start date of project: 2015/04/01	**Duration: 48 months**
Project coordinator: UPB	**Classification: public**
Partners:	
lead: **CITA**	*contribution:* **UNIGRAZ, UPB**
Project website:	http://florarobotica.eu/

H2020-FETPROACT-2014

DELIVERABLE SUMMARY SHEET

Grant agreement number:	640959
Project acronym:	*flora robotica*
Title:	Societies of Symbiotic Robot-Plant Bio-Hybrids as Social Architectural Artifacts

Deliverable N^o:	Deliverable D3.1
Due date:	M24
Delivery date:	2017/03/31

Name:	**Representations and design rules**
Description:	We develop appropriate architectural representations (modeling methods, simulation and systems of notation) that integrate models of robot mechanics and its control with relevant biological models (e.g., projection of growth, leaf-cover and structural strength models) to support the design, envisioning and evaluation of architectural *flora robotica* propositions in terms of structural and environmental performance and spatial potential. Data of *flora robotica* will be used to calibrate models and provide existing starting states from which to generate propositional simulations of growth towards desired architectural states. Architectural design rules derived from botanical rules of growth will be established.

Partners owning:	**CITA**
Partners contributed:	**UNIGRAZ, UPB**
Made available to:	public

Contents

1 Introduction

The growth of architectural artifacts with plant-robot bio-hybrids is a central objective of *flora robotica*. These artifacts incorporate robotic elements, biological plants, and mechanical scaffolds. Though the original proposal pointed to a strut and node approach, in which the distinction between robot and scaffold may have potentially been very clear, the consortium has since turned to braid as the organizational logic (see deliverable *D1.2 Evaluation of mechatronics prototype of the robotic symbiont including supporting software*). Braid allows a flexible organization of scaffold and robots. Though individual robotic nodes are a key component on the scaffolds, robotic elements can also be fully integrated into braided structures by interlacing into the weave or embedding into the filaments.

This shift has made the design and construction of mechanical scaffolds a central aspect of WP3. As such, the representational methods reported here in D3.1 are partially influenced by the construction logics being reported in the future deliverable *D3.2 Architectural propositions*. This deliverable therefore briefly introduces concepts from the future *D3.2*, as needed.

The main objective of the task T3.1 reported here is to develop representational methods[1] necessary for:

1. generation of artifact states,
2. realistic geometric modeling of artifact states, and
3. evaluation of artifact states.

Generation of artifact states. Representational methods are needed to integrate biological growth and robotic control with the specification of artifact states, and to steer control toward states that achieve high-level architectural objectives. The overall integration of biological growth and robotic control is addressed through the Integrated Projection. The Formation Space, the VMC, and aggregate artifact morphology serve to specify artifact growth states and steer them towards certain architectural objectives. Robotic control that fabricates braided structures from those artifact states is addressed by the low-level graph representation of braid and by the integration of braided modeling with robotic fabrication methods.

Realistic geometric modeling of artifacts. The interpretation of artifact states into a realistic geometric representation of the resulting physical braids is crucial to support the architectural design, envisioning, and evaluation of the *flora robotica* artifacts. This is achieved through a generalized mesh modeling approach to braid, including pattern generation through tiling, discretization, geometry relaxation, and physics-based simulation. The choice of braid as the overall organizational logic establishes the artifacts' strong relationship with botanical rules of growth, through braid's flexibility and material continuity.

Evaluation of artifact states. Representational methods needed to support the evaluation of propositions in future deliverable *D3.2*, beyond realistic geometric modeling with some small

[1] The representations reported here are frequently developed in software platforms common in architectural disciplines, as this is important for dissemination and impact in architectural scientific publishing and public or industry dissemination. All representations here have been developed in a way that they can be decoupled from these softwares and implemented in Python, Blender, and similar. Libraries made for the architectural platform Grasshopper3d have been developed in C# or VB.NET, occasionally referencing RhinoCommon objects. The RhinoCommon objects referenced are simple, like a point or line, and can therefore easily be replaced to make the libraries fully independent of the utilized architectural software platforms.

extensions, include visualization methods to support designer decision making and methods for recording the data needed for evaluation as the artifacts grow. Designer decision making regarding evaluation of spatial spatial potential is supported by the realistic modeling of artifacts. Designer decision making about fitness functions regarding the priorities of structural, environmental, and spatial performance is supported by visualization of evolutionary design spaces. The long-term evaluation of growing *flora robotica* artifacts is supported by recording represented histories of natural growth, artificial growth, environmental conditions, and human occupancy. This will support the calibration of models and choice of starting states when defining architectural propositions in *D3.2*.

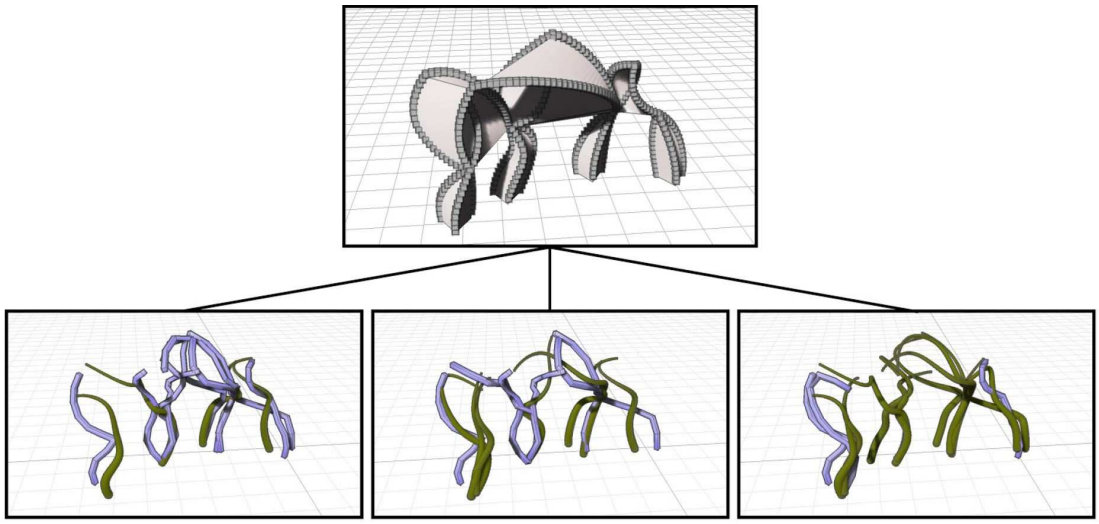


Figure 1: Three geometric models of example artifacts projected from a single system configuration with growth models that include stochasticity.

2 Using a Pluralistic modeling approach for Integrated Projection

Projecting a full geometric model of artifact states is important for evaluating architectural propositions during the process of designing controllers (see Sec. 5 for details on evaluation). The visual legibility of such a geometric model is also useful for end users of the social garden to view when incorporating their preferences through interactive evolution.

Due to the heterogeneity of elements in *flora robotica*, there are multiple models of robotic control and biological growth in use. In order to generate an integrated geometric model of resulting bio-hybrid artifacts, several interacting models need to be combined. Additionally, at least some of the models include stochasticity, meaning that the resulting artifacts can be projected, but not deterministically predicted. Therefore, the approach of the Integrated Projection produces multiple geometric models from a single system configuration, as a group representing the range of physical bio-hybrid artifacts that could possibly result from that configuration (see Fig. 1). The Integrated Projection is developed as part of task *T2.3 Micro-Macro-Models of feedback-based robot-plant-interactions*.

Data recorded to represent histories of natural and artificial growth, described in Sec. 6, will be used not only for long-term architectural evaluation of artifacts, but for calibration of the various models of biological growth and robotic control. This data can also be used to help determine appropriate starting configurations to steer growth to architectural propositions that match specific use cases.

Connection to braided artifacts. Projecting braided artifacts from robotic morphology controllers takes some interpretation of the controller state. For example, a state of the Vascular Morphogenesis Controller (VMC) described below is interpreted when constructing a realistic geometric model of the braided artifact it represents. When this geometric model of the artifact is used not only for visual analysis but for generating instructions for robotic fabrication of the artifact, then the step of interpreting the controller state partially controls the artifact. This gives increased flexibility to how the robotic controllers are used to generate artifact states. For

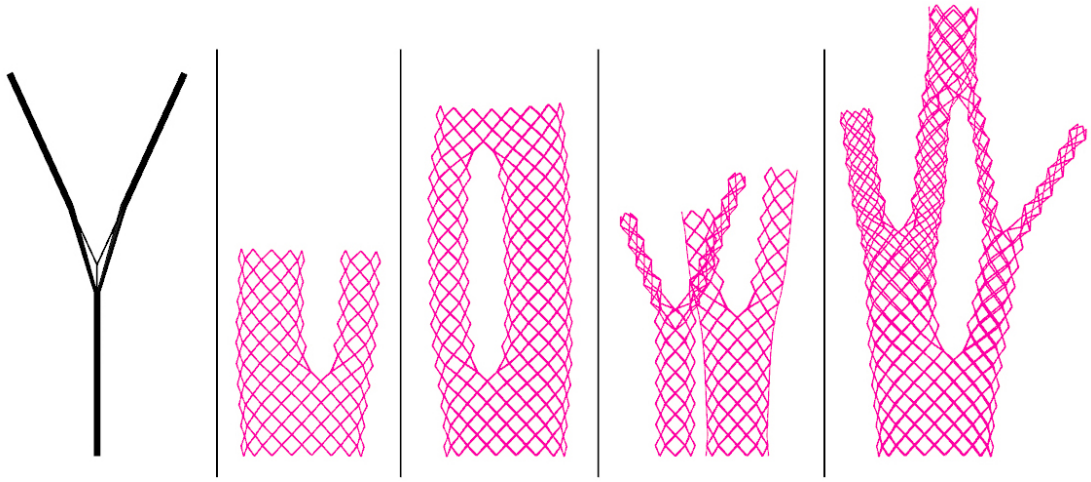


Figure 2: A single VMC robotic controller state interpreted in four different ways, resulting in four distinct possible braided artifacts.

example, a straightforward way to interpret a graph from the VMC is to treat each graph edge as the central axis of a tubular braid. However, the graph could define a braid in many other ways, which may offer various advantages in different architectural use cases (see four distinct interpretations of a single VMC graph state in Fig. 2). The step of interpretation between the controller and the integrated projection allows us to make use of this flexibility, defining the relationship between controller and artifact according to the spatial potential and other performative criteria of architectural propositions.

3 Growth toward desired architectural objectives

Low-level robotic controllers such as the VMC need to be steered for resulting artifacts to meet, among other goals, high-level architectural objectives for desired propositions of use.

3.1 Formation Space: Representation of high-level architectural objectives

For the evolution of *flora robotica* controllers (e.g., in the context of the social garden) fitness evaluation includes high-level architectural objectives. For this purpose, the 'formation space' is developed as a representational method for high-level architectural characteristics that arise from distributed robotic controllers. It can be used to generate artifact states, steering artificial growth towards architectural objectives. The description of the formation space, as detailed here, follows our paper [20].

In architectural design with self-organizing construction, important characteristics are unlikely to be well-represented by internal states of robots. Architectural characteristics rather deal with the details of the construction material deposited as robots progress along their trajectories (similar to the *flora robotica* concept of 'embodied memory'). For this reason, the mathematical formation space is developed, as an extension of the phase space.

3.1.1. Background of active architectural elements

As described in our paper [20], the architectural design problem at hand is that of designing controllers for active/robotic architectural elements that have strong self-organizing behaviors.

Existing buildings containing active products (such as automated window shades or automated HVAC) are often controlled centrally, through readily available technology known as building automation systems or smart building management systems (see Siemens white paper[2]). These centralized systems deal with changes in environmental conditions by existing design methodologies of averaging performance or maximizing peak performance, at the scale of perhaps a wall or a room. In addition to these industry products, there are existing buildings featuring bespoke active facade elements. There are many built examples incorporating matrices of LED lighting, such as the centrally controlled BIX facade of Kunsthaus Graz [13]. There are also examples of built facades incorporating kinetic elements, including Media-ICT (Cloud 9[3]), IBA Soft House (KVA MATx[4]), BIQ Algae House (Splitterwerk[5]), Yeosu Theme Pavilion (SOMA[6]), Al Bahr Towers (AHR[7]), and MegaFaces (Asif Khan[8]). Some of these facades incorporate limited decentralized control but, to our knowledge, do not incorporate decentralized communication between elements. There are some departures from the centralized communication paradigm at the scale of installation (e.g., Sentient Chamber, PBAI[9]) [11]. There are also relevant architectural demonstrators, prototypes, or probes [32] in the literature (cf., [18], [5]). We therefore understand the question of designing distributed/decentralized control and communication in architectural elements to be relevant to CAAD (computer-aided architectural design) state of the art, in addition to its relevance to *flora robotica*.

[2] http://www.usa.siemens.com/intelligent-infrastructure/assets/pdf/smart-building-white-paper.pdf
[3] http://www.ruiz-geli.com/projects/built/media-tic
[4] http://www.kvarch.net/projects/87
[5] http://www.biq-wilhelmsburg.de/
[6] http://www.soma-architecture.com/index.php?page=theme_pavilion&parent=2
[7] http://www.soma-architecture.com/index.php?page=theme_pavilion&parent=2
[8] http://www.asif-khan.com/project/sochi-winter-olympics-2014/
[9] http://www.philipbeesleyarchitect.com/sculptures/Sentient-Chamber/

3.1.2. Complexity and predictability in architectural design

As described in our paper [20], our concept for architectural design methods is based on the following. In the description and study of dynamic systems, the 'phase space' is the representation of all possible instantaneous states that can occur in a physical system [9, 30]). Each of these states, and its associated characteristics, corresponds to a single point in the mathematical phase space. In existing design rhetoric, we sometimes refer to the parameter space and the solution space of a model. These spaces can be approximated and understood analytically, because any particular combination of parameters always results in the same unique solution. The phase space is distinct from these in that it describes the full set of possible states that may eventually arise from a set of behaviors, and therefore, in certain types of systems, cannot be derived purely analytically [3]. In the context of architectural design, this means that design solutions must be modeled and simulated before visual interrogation and analysis of the design can occur. These types of systems, which can be described through modeling and the analysis of model results, are categorically complex systems (or complex adaptive systems). The scientific study of complex systems deals in part with the principles governing the emergence of such systems from simple components [3]. This section details an approach of applying a particular aspect of complexity science to the realm of computational design (or design computation), specifically by employing the mathematical phase space as an approach for architectural design, modeling, and simulation. Due to the context of architectural design, we assume that only spatially distributed models of systems are relevant to this design methodology, which can be an important distinction in modeling (e.g., work on the breakdown of mean-field approximation and work on spatially distributed models [34, 4, 16]).

It is sometimes assumed in architectural discourse that incorporating methods of emergence, distributed control, and evolutionary robotics will largely equate to unpredictability in the architectural result [18], [25], [26]. However, we know from complexity science that this varies depending on which scale of the system you are describing, which characteristics you are concerned with, and at which point in time you are viewing the system.

The amount of predictability varies greatly by system. Some complex systems have attractors, which will cause the system to move toward a particular configuration or set of configurations, depending on the number of attractors and the expanse of the basins associated with those attractors. For instance, if a particular system has two attractors, and the basins of attraction from these two cover the phase space equally and entirely, then the system will tend to one of these two states with equivalent probability, despite an expansive array of possible start conditions [22, 30]. In other words, the phase space of possible configurations is predictable, if not the exact configuration, and the expansiveness of that predictable phase space depends on the complexity of the system. The amount of predictability of the exact configuration that a system with attractors will move toward depends on the number of attractors and basins of attraction. Therefore, in looking to design rules for distributed control and communication in architectural sensor-actuator systems, we do not design a solution by finding and evaluating individual instances. We instead work toward designing a system's (i.e., set of behaviors) full set of possibilities (the phase space), and its attractors. The description used to design an architectural complex system should occur at the scale that is pertinent to the particular architectural design problem at hand (this does not change the scale of rule-based interactions; scale of rule and outcome are still distinct). The system should be designed so that coherent behaviors or correlated behaviors (see complexity profile and related concepts [3]) occur at that pertinent scale (see Fig. 3). There could be a prohibitive amount of randomness at a certain timestep and scale, making it seem too unpredictable for architectural design. At such a state, a comprehensive description of the system would have to include a description of each individual in order to describe the behavior

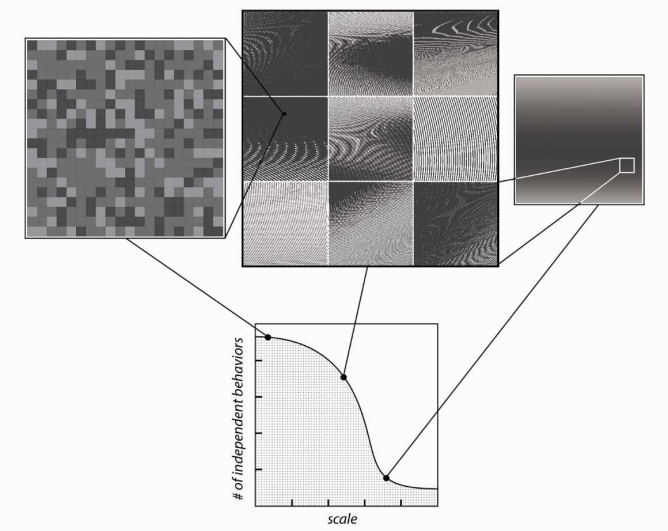


Figure 3: Author's illustration from [20] of the existing mathematical concept of the complexity profile [3], showing how one system can have different numbers of independent behaviors at different scales, depending on the system's scale of randomness, coherence or correlation.

of the system. But in that same system, there could be a coherent behavior at a different timestep and scale. At this state, a comprehensive description could be comprised of very little information, because it would need to describe only one behavior: that of the coherent group (see random, coherent and correlated behaviors [3]). The balance of complexity and simplicity in a designed system may likely depend on which scale is being described, and at which point in time. Therefore, the task of designing architecture that is emergent or distributed is not a task of designing behaviors with primarily unpredictable results. It is rather the task of designing a relevant phase space; it is the task of designing the scale at which coherent or correlated behavior occurs, without resorting to high-level control.

3.1.3. Case study: Controllers for solar responsive self-organizing growth

As described in [20], the case study selected is the task of designing an exterior plant-robot bio-hybrid system that responds to solar exposure. The design task of the response is to shade occupiable spaces from the sun by growing taller where needed, but to only grow where necessary, so as to leave as much occupiable space on the site as possible. However, the system should maintain some presence, even if the site receives no sunlight, so that the open area is divided into smaller occupiable spaces, and so that the system has a chance to react if the lighting conditions change at a later time. The controller should assume that the bio-hybrid system generally grows upward, to both align with plant dynamics and to be self-structuring. The bio-hybrid system should be able to be placed on any site and correctly interpret whether each element is in sun or shade.

A predetermined controller in this case (which would bring bio-hybrid growth to a certain height based on solar geometry derived from latitude and longitude) is not sufficient, because the controller would be unable to react to shade cast on the site by clouds, trees, or neighboring buildings (see Fig. 4). A centralized sensor-controller would also not be sufficient in this case because, though the sensor would be able to correctly interpret its local neighborhood's solar

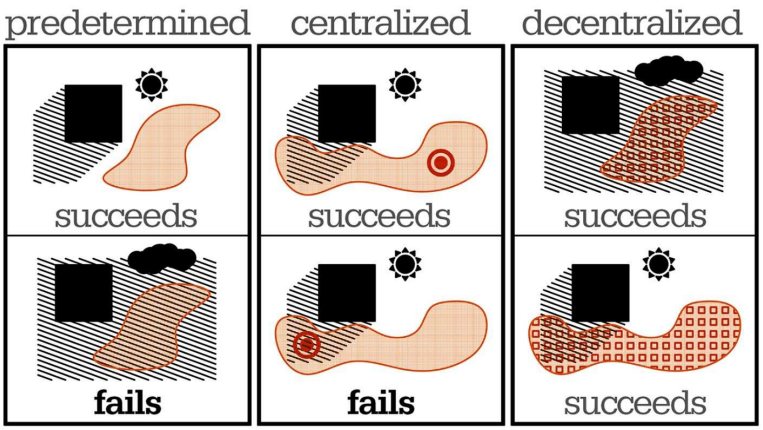


Figure 4: Comparison in this case study of predetermined and centralized (which use methods of averaging), and decentralized systems, showing the conditions under which they either fail or succeed to accurately respond to site conditions (figure from [20]). Orange indicates the actuators and scaffolds of the artificial system; red indicates sensors and controllers.

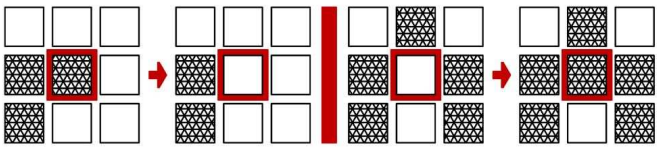


Figure 5: Author's illustration from [20] of two examples of transitions of the states of single cells in a Moore neighborhood majority rule CA [35], based on illustration strategies from [30]. The central cell in each block of 9 is the only cell transition depicted.

condition, it would assume that the solar condition is the same across the site, and would fail in conditions where neighboring buildings (for instance) cast shadow on half of the site (see Fig. 4). Decentralized sensor-controllers, by contrast, would not fail in any of the above mentioned scenarios (see Fig. 4, also see *D1.2 Evaluation of mechatronics prototype of the robotic symbiont including supporting software*, Sec. 3.6) and furthermore have the spatial consequence and design opportunity of increased heterogeneity. If the system has not only distributed control, but distributed communication, it has the added advantage of being able to create spatial clusters as it grows, to provide as much occupiable space on the site as possible. Therefore, with the design problem presented in the case study, it is appropriate to design a system that features distributed control and distributed communication.

3.1.4. Setup for the case study

In future work, we will use the formation space to assess *flora robotica* controllers for artificial growth (such as the VMC described in Sec. 3.2.4), but here we use a simpler case study, to more clearly describe the formation space.

As described in [20], we select cellular automata[10] (CA) as a typology to address this design

[10] http://mathworld.wolfram.com/CellularAutomaton.html

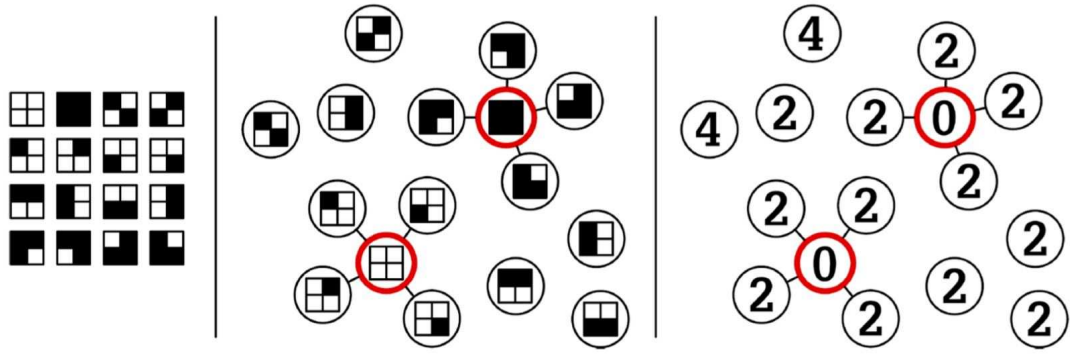


Figure 6: Formation space concept (as shown in [20]). Left, an enumeration of all possible configurations of a 2-dimensional binary state CA with 4 cells. Center, author's visualization, based on [30], of the phase space, attractors, attraction basins, and fixed points of the CA. Right, a visualization of isometrically mapping a formation space of the CA onto the phase space, according to the characteristic of boundary length.

problem, because it models decentralized communication in a way that is befitting of non-mobile architectural elements, it is discrete in both time and space, and the possible states of the system and the possible transitions it can undergo are finite [35, 36, 30]. We use a Moore neighborhood setup[11], in which, as each cell determines its state, it looks at the cells directly below, above, and beside it, and it also looks at the 4 cells diagonal to it. We use a standard majority rule [35], in which each cell decides its state by conforming to whichever state holds the majority in its neighborhood. The majority rule means that, if a cell is dark, 2 of its neighbors are dark, and 6 of its neighbors are light, the cell will become light in the next timestep (see Fig. 5). Likewise, if a cell is light, 5 of its neighbors are dark, and 3 of its neighbors are light, it will become dark in the next timestep (see Fig. 5). The majority rule was chosen because its dynamics are known to often lead to clusters, which are desirable in the given case study. In this setup, we use dimensions of 100 × 100 and periodic space, and in initial conditions use equal probability of randomized cell state distribution[12].

3.1.5. Using the formation space in the case study

As described in [20], the phase space of a CA is an enumeration of all possible configurations in that setup, according to the dimensions of the cellular automaton and the number of possible cell states. For instance, in the right of Fig. 6, we can see all the possible configurations in a 2-cell by 2-cell two-dimensional cellular automaton with binary cell states (two possible cell states). The number of possible configurations in this setup is few, so they are easy to imagine, visualize, and analyze. In the center of Fig. 6, we see a visualization of the phase space of this system (visualization method from [30]).

Each possible configuration of the system is subjected to the majority rule (in a Moore neighborhood in periodic space), and the configuration it goes to in the next timestep is recorded. Then, a graph is constructed showing each configuration transition as an edge connecting the two

[11] http://mathworld.wolfram.com/MooreNeighborhood.html

[12] All setups are completed in a combination of PyCXsimulator (http://pycx.sourceforge.net/), IronPython (http://ironpython.net/), and Grasshopper3d/Rhino3d (http://www.grasshopper3d.com/), with reference to the PyCX Project (http://pycx.sourceforge.net/).

configurations. Each component (cluster) in the graph reveals a basin of attraction in the system, and the hub of that component (center of the cluster) reveals the attractor (see [30]). The phase space visualization for this system shows that it has two equally large basins of attraction, going to the attractor configurations of fully dark or fully light. There are several other fixed point configurations that have no predecessors and do not change configuration when subjected to the rule. The only way for the system to achieve these states is to begin there, and, once there, it can proceed nowhere else (see [30]).

When using these systems in architectural design, however, we are unlikely to be concerned with the internal states of members of the system, which is what the phase space description is based upon. We are much more likely to be concerned with the material manifestation left behind as the system progresses. For this reason, we originate the concept of the mathematical formation space, as an extension of the phase space, intended specifically for the realm of architectural design.

The formation space is an isometric mapping onto the existing phase space (hence being considered an extension). It is a mapping of a particular characteristic of the configuration at a larger scale than the individual cell (or of the material deposition the configuration leaves), ideally at a scale at which some coherence or correlation of behavior is applicable to the design problem at hand. In the right of Fig. 6, we look at the particular characteristic of length of boundaries between opposing states throughout the system's entirety, isometrically mapped onto the phase space to form a formation space. In this mapping, we can see that the two large basins of attraction now lead to the same attractor, boundary length "0", and that the 7 states which used to be completely isolated can now be grouped together into boundary lengths "2" and "4". This method of visualization shown in Fig. 6 (for phase space and for formation space) is fine for this very small system. However, the number of possible configurations in a CA system is defined as the number of possible cell states, raised to the power of the number of cells [35], [30]. So, while this system shown in Fig. 6 contains only $2^4 = 16$ configurations, a system with only 16 cells would contain $2^{16} = 65,536$ configurations. It is easy to see how this visualization approach quickly becomes unmanageable.

Rather than visually interrogating every possible configuration in the phase space or formation space, we must find ways to approximate effectively and make reliable design decisions. The following lays out reasoning for the usefulness of the formation space (and related concepts), in architectural design, as motivation for further work to verify usefulness and find effective approximations. For this case study, we include 20 setups of the CA, and conduct 5 simulation runs for each setup. The 20 setups cover each possible combination if number of possible cell states $s \in \{2, 3, 4, 5\}$ and radius size of neighborhood $r \in \{1, 2, \ldots, 5\}$. Individual timesteps of single runs of two of the setups are shown in Fig. 7, and the 100th timestep of every run is shown in Fig. 8. In Fig. 7, we see that the upper setup settles on its steady states extremely quickly, while the lower setup has not yet reached its steady states even by the 100th timestep. In Fig. 8, we see that setups with smaller neighborhood radii reliably reach a steady state with many small clusters, with the initial possible states approximately equally represented. We also see that setups with larger neighborhood radii tend to end with the entire system in a homogeneous state, with only one of the starting cell states represented.

If we look at the resulting steady states from two runs of the same setup, such as those in Fig. 9, it is clear that it would be impossible to predict which state an individual cell will end up in, or which overall configuration the system will end up, if the starting conditions are randomized and unknown. In this way, considering the phase space of the system, its result seems very unpredictable, and perhaps not useful for architectural design. However, if we do not consider the cell states, system configuration, or phase space, instead considering the formation space and spatially-useful characteristics at the scale most relevant to the design problem at

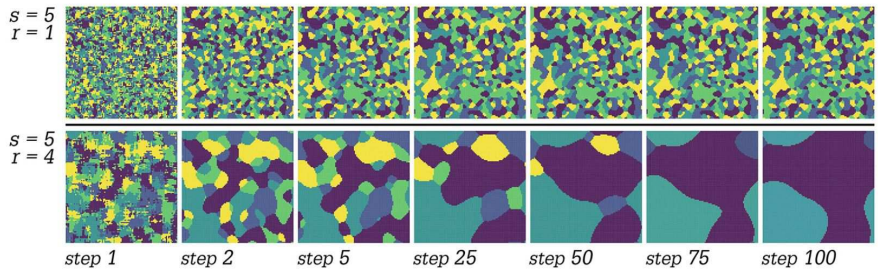


Figure 7: Progression of two setups. Top setup reaches its steady states very quickly, bottom setup does not yet reach its steady states at the 100th step (as shown in [20]).

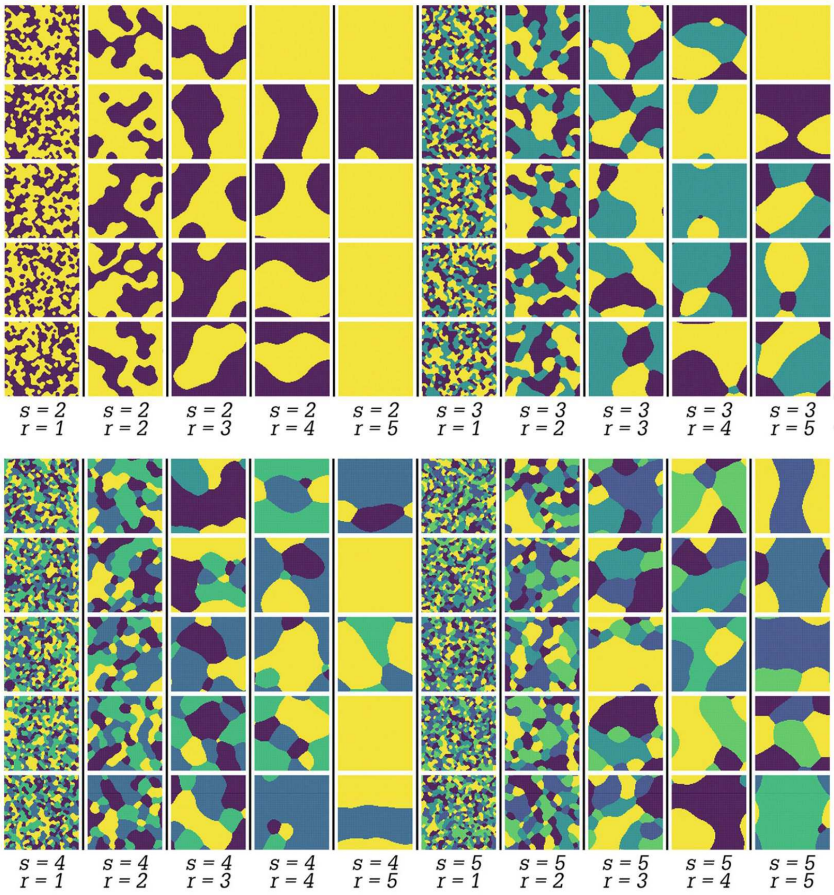


Figure 8: The 100th timestep of 20 different setups, with 5 separate simulation runs shown for each setup. Setups are indicated by number of states s and size of neighborhood radius r. This side-by-side comparison is meant as a method of visual analysis exploring dependencies in the system through changing variables (as shown in [20]).

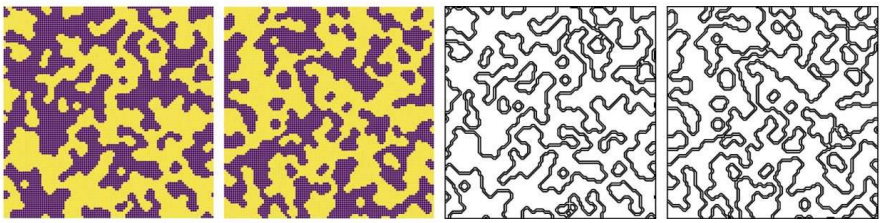


Figure 9: Visualizing the configuration of the steady states of two runs of the same setup, and visualizing the boundaries between binary states in those same two steady configurations (as shown in [20]).

hand, the system suddenly seems much more predictable. If we look only at the boundaries between states in these two simulation results, and consider the overall boundary length in the system, we can see that they are very similar. More precisely, they are of lengths 1,834 and 1,709; very similar for a system with 10,000 cells. In short, if we look at the phase space of this system it is difficult, or impossible, to predict precisely into which basin of attraction an unknown randomized initial condition will fall. But, if we look at the formation space, we can reliably project (perhaps, in some cases, even guarantee) an attractor that the characteristic will fall near to. Therefore, it is feasible that using this system, we could design low-level control rules to reliably produce an approximate desired density, number, and spatial scale of the occupiable areas present on the site.

The formation space is a representation that can work as a tool for designers, enabling them to select controllers that will predictably achieve high-level architectural objectives. This establishes impact to the architectural community by a novel approach architectural design approach of guaranteeing key attributes rather than specifying exhaustive descriptions of fine-scale details. If the controllers of the system are subject to evolution, the attractors of the formation space can be used to evaluate fitness.

3.1.6. Extension: evolving controllers in the case study

The future deliverable *D3.2 Architectural propositions* (in its section on propositions of use) will report on evolving controllers using the formation space. Here, we report a small probe study into evolving controllers in the simple case study above.

As described in [20], an example probe [32] study was conducted, using two formation spaces of high-level architectural characteristics to distributively evolve in simulation local controllers that respond to sensed ambient light conditions. The evolution of the controllers is rewarded for the low-level characteristic of sensed local shade, for the low-level characteristic of local non-growth (to encourage open area), and for maximizing the attractor value of the formation space for length of boundaries. The results of this probe study (see Fig. 10) show different growths on each site, suggesting that the controllers achieved site-specific evolution, in response to differences in shade provided by neighboring buildings. The first site, shaded only from the south, results in high density of boundary condition, tall growths, and open areas in the center, where they are shaded by growths. The second site, more shaded by buildings, results in shorter, fewer growths. The third site, already very shaded, results in low density of boundary conditions, and growths that slope downward and outward, thereby providing little additional shade to open areas. These preliminary outcomes show promise in fulfilling the high-level design requirements of maintaining open occupiable space and system growths as spatial dividers, and of shading

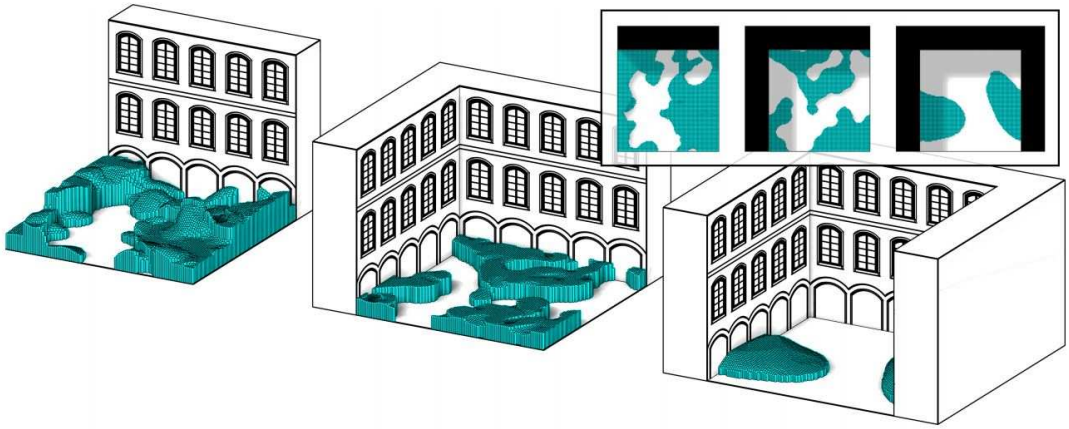


Figure 10: Results of probe study, evolving the CA majority rule controller in simulation on three distinct sites (as shown in [20]).

occupiable space that is not already shaded by neighboring buildings, suggesting that the probe into evolving controllers merits further investigation.

3.2 VMC for Growing Structures in Different Conditions

Vascular Morphogenesis Controller (VMC) is introduced in *D2.2 Report on the final algorithms and plant-affection of bio-hybrid organism* and is implemented here in a modular robotic controller described. As described by the algorithm, growth happens at the leaves of the VMC graph. When a leaf grows, it turns into an internal node with new leaves attached to it as its children. The new leaves then compete for the resource with all the other leaves in the VMC in order to get their chance to grow. In a physical system, for example, in a growable modular robot that is controlled by VMC, a leaf node is associated with every extension point of every module where new modules can potentially be added. When a particular leaf is supposed to grow, a new module is attached to the extension point associated with that leaf. The new module brings its own extension points and hence new leaves are added to the graph as the children of the previous leaf that now has become an internal node. The number of the new leaves is a feature of the modules. For example, in a homogeneous modular robot with two extension points, adding a new module to the structure means adding two more leaves (each associated to one of the extension points) to the graph as the children of a previous leaf.

Here we are interested to use VMC to drive the morphology of a growing structure in presence of some physical effects such as gravity and elasticity. For that, we have built a simple modular robot where extending its morphology is possible during run time. Unlike a biological organism, our modular robot cannot grow autonomously and the growth process is sequential (one module at a time). The growth is achieved by the help of a human operator who follows the suggestions of the distributed controller running on the modules of the structure.

3.2.1. Design of the Modular Robot

The designed modular robot is built based on a small mobile robot called Thymio [23] that is available off-the-shelf (Fig. 11(a)). We have connected several Thymios to each other to build the single modules of our modular robot (Fig. 12). The Thymios are connected in a way that their

(a) Thymio robot in normal use

(b) A growing structure

Figure 11: A Thymio robot and an example of a growing structure made out of non-branching modules

local communication is still possible. The built modules are to some extent flexible and elastic due to the usage of rubber bands and zip-ties for keeping together the rigid parts of the modules which are basically the Thymios (please see the attached video for an impression of the physical modules[13]). Two types of modules are built: non-branching and branching modules. Fig. 12 represents a module of each type both in physical and in simulated versions. The elasticity of the modules and their bent shapes makes the structure building more challenging and closer to real world. Fig. 11(b) shows an example structure built out of non-branching modules growing against gravity. In a branching module, the two upper Thymios can communicate with the bottom robot (and vice versa) via Infra-Red (IR) sensors. Each Thymio robot contains an accelerometer that is used for sensing the orientation of the module against the vector of gravity.

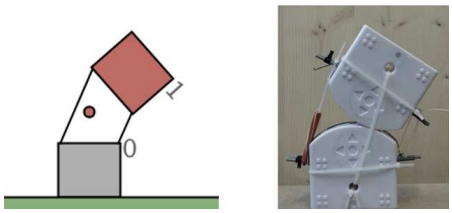
In this work, we have mainly used the branching modules in simulation. A simplified physics-based 2-dimensional simulation is designed based on the physical robot and considers gravity and elasticity of the modules due to the use of rubber bands. It allows the modules to bend to some extent and the structure is prone to collapse when the center of gravity is sufficiently far from the base. The local communication between the modules is constrained in the simulation in order to emulate the constraints in the physical robot. The simulation is developed in the Processing framework[14] with Box2D physics engine[15].

3.2.2. Implementation of VMC in the Robot

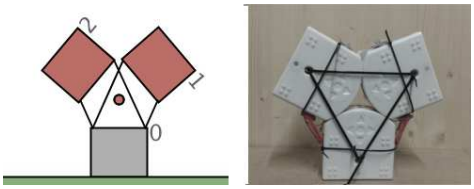
The growth of the robot starts from a base module that is a non-branching module fixed to the ground making a tilted flexible starting point for the growing structure. All the other modules of the robot are branching modules. Growth in this structure means attaching a new branching module to a potential extension point. Every branching module has two extension points (two Thymios on the upper part of the module) and a point of attachment (Thymio on the bottom).

As the base is non-branching, the VMC of this structure initially contains a root and its single child (a leaf). The root constantly produces a fixed resource value of 1 unit that is distributed between the leaves via the connections. At every time step, the modules that contain the leaves show to human operator, an indication of their share of resource. The human attaches a new

[13] https://youtu.be/tAj_uHk9oNg
[14] https://processing.org
[15] http://box2d.org/



(a) A simulated and a physical non-branching module



(b) A simulated and a physical branching module

Figure 12: Single modules

module to the leaf that indicates the highest resource. In simulation, the process of adding a new module is done automatically and regularly at every 150 time steps.

The accelerometers at the leaves are used to perceive the vector of gravity relative to the nodes. The magnitude of this vector is used as the local sensor input at the leaves and is available for influencing the production of Successin. In addition to the vector of gravity, the modules can perceive the light intensity (IR in the real robots). There is no sensors used to perceive the presence of obstacles (e.g., other modules or the ground) by the modules. Hence, the VMC leaves are not aware of the presence of other leaves or the ground even if they are facing them. If the robot intends to grow at a leaf and fails due to the presence of obstacles, the failing leaf goes to a non-growable mode and will not produce Successin anymore for the rest of the experiment.

3.2.3. Evolving Parameters

VMC is implemented here to grow a modular robot in different setups and objectives. To evolve the VMC parameters for each setup, a genetic algorithm is used. The population size of the genetic algorithm in all the experiments is 20 and the populations are evolved for 24 generations. The experiments are repeated for 20 independent runs in every setup. Elitism of one genome and a crossover rate of 20% is implemented. All the genomes (except the elite) are mutated with a *stepsize* $\sim \mathcal{N}(0, 0.2)$. A genome is a set of VMC parameters. The length of the genome depends on the number of the sensor types used for the VMC in every particular setup. By using the light sensor in both leaves and the internal nodes, and using the accelerometer of the modules (for sensing the angle of gravity) at the leaves, the genome would look like: $(\omega_{\text{const}}, \omega_{\text{acc}}, \omega_{\text{light}}, \rho_{\text{const}}, \rho_{\text{light}}, c, \alpha, \beta)$ (see *D2.2 Report on the final algorithms and plant-affection of bio-hybrid organism* for the description of the parameters). The genomes are randomly initialized between $[-1, 1)$ for the sensor-related parameters and between $[0, 1)$ for the others.

For evaluating the fitness, each genome is used to parameterize VMC to grow a structure in a given setup. At the end of a run, the structure is evaluated based on the given objective function according to the setup. Since the setups are noisy due to the physics and the additional

randomly imposed force to the structures in some of the experiments, each genome is evaluated in three independent runs and the fitness is the minimum of the three evaluations.

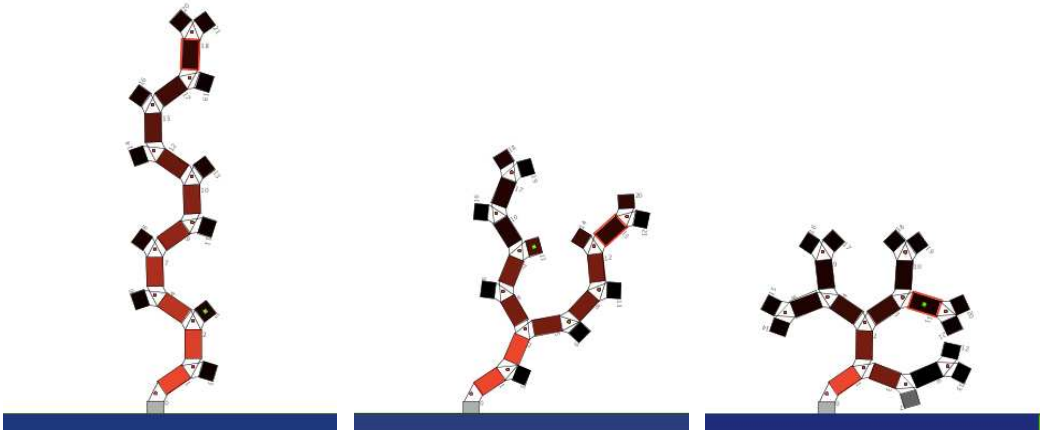
3.2.4. Experiments: Growing in Different Conditions

In the first set of experiments, VMC is used to grow a robot with 10 modules as tall as possible against gravity. An impulse force is imposed to the base module randomly from the left or the right side of the module. Considering the flexibility of the modules and the effect of gravity, the structures might collapse during the growth ending up to (even long) structures which are fallen onto the ground and therefore are not considered tall against gravity, thus not meeting the objective of the experiment. We evolved the system in two different setups: a low impulse force and a high impulse force. The strength of impulse in the latter setup is three times as large as in the former setup and thus the structures in the latter setup are more prone to collapse. In this experiment, the accelerometer sensors at the leaves are used. The fitness is computed as the height of the highest module of the robot at the end of the experiment. Fig. 13(a) and 13(b) show the structures developed by the best genomes evolved in each setup (see a video here: `https://youtu.be/xBEIwFixtJs`). As shown in the figure, the morphologies are different in each setup. To investigate the difference, we developed two robots each in one of the two environments and positioned them both in the harsher environment (high impulse setup) for a while without any further growth. The result was the collapse of the robot developed in the calmer setup, while the robot from the harsher setup stayed upright until the end of the experiment, showing a higher stability. Fig. 15(a) and 15(b) show the fitness trajectory of the best genomes accumulated from all 20 evolutionary runs.

In the second experiment, we evolved VMC for growing a bushy robot. For that, the fitness is computed as the number of modules with no children (i.e., both extension points of the module are free). Fig. 13(c) shows the resulted morphology in this setup. The gray leaf in the figure indicates that the robot has tried to grow at that leaf but it has failed due to the collision with ground. Finding the VMC parameters for this setup seems to be easy for evolution (many possible solutions) such that the solution is found already in the first couple of generations (not shown).

In order to get a better understanding of the effect of each of the VMC parameters in the final morphology of the robot, the parameter sets with the highest fitness in the three above setups are collected from the several runs and represented in Fig. 16. As it can be seen in the figure, the range of β (addition rate) is quite similar in all the setups (statistical Mann-Whitney U-test shows no significant difference). The value range of parameter α (adjustment factor of V), and parameter c (decay rate of V) are narrow for all the setups. The highest variation between different setups can be seen in the values of ω_s and ω_{const} that deal with production of S at the leaves (respectively, regarding and regardless of the sensor inputs), and also ρ which is related to the transfer rate of S from a node to its parent. The ω_{const} for a bushy morphology is narrow and close to zero, and ω_s for the same morphology is always negative leading to no production of flow at the leaves. The range of the parameter ω_{const} is broad for both of the tall morphologies. The value of the parameter ρ is very close to 1 for the tall morphology in the low impulse environment, meaning that (almost) all of the S is transferred from each node to its parent which implicitly increases the effect of the S generated at the leaves.

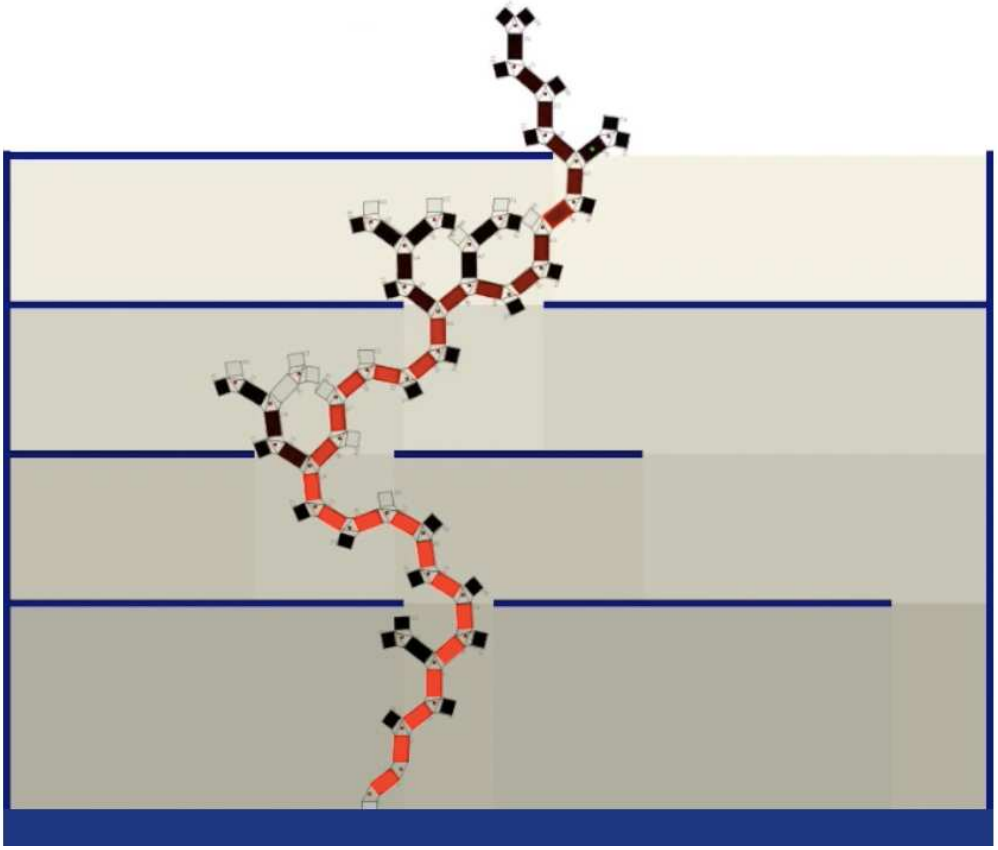
Based on the evolved parameters in the previous experiments and our gained understanding of the effects of different parameters, we manually parameterized the VMC for a special setup. In this setup, in addition to the accelerometer and light sensors at the leaves, the light sensors are also used at the junctions. The aim here is to use the same set of parameters in environments with different light conditions and the robot is supposed to grow bushy in high intensity of



(a) growing tall in a calm environment

(b) growing tall in a harsh environment

(c) growing bushy



(d) finding the brightest layer in a layered environment

Figure 13: Growing the structure in different setups

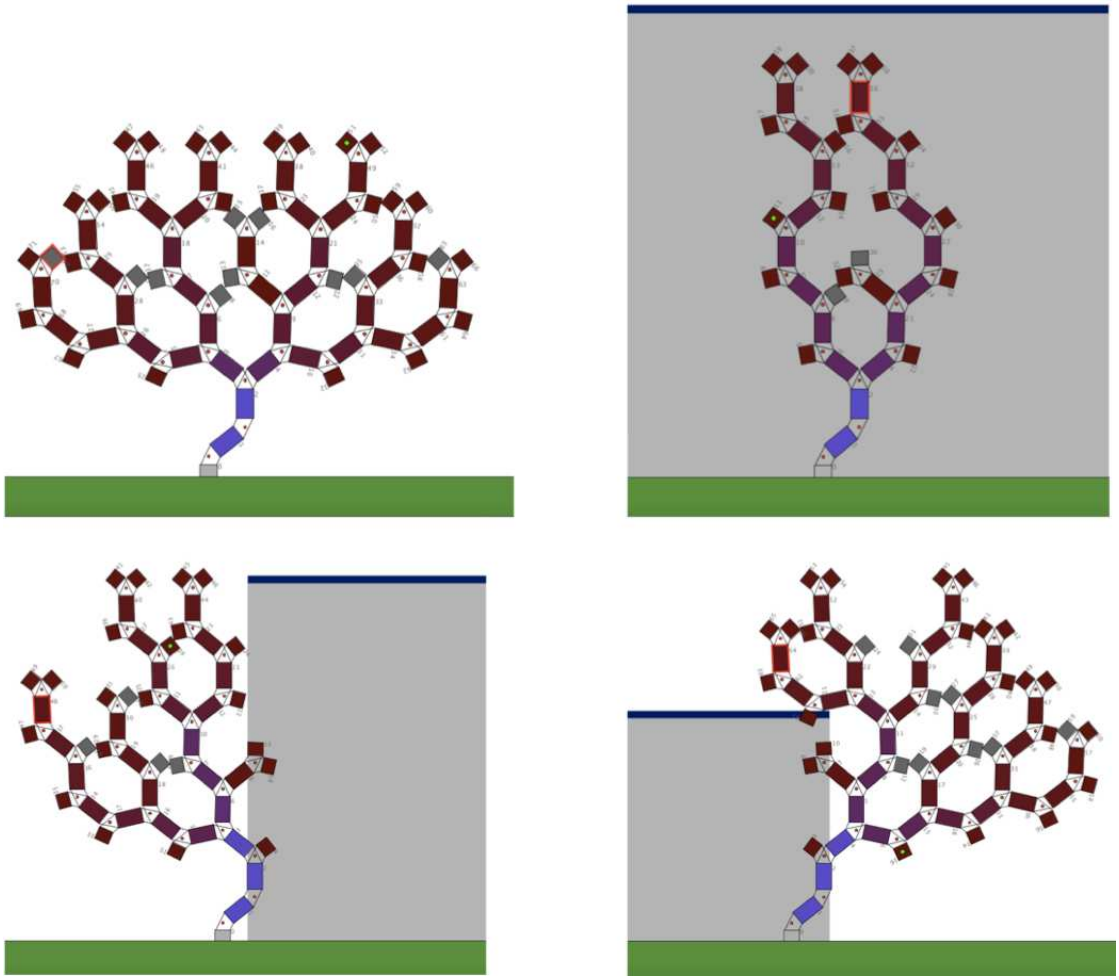
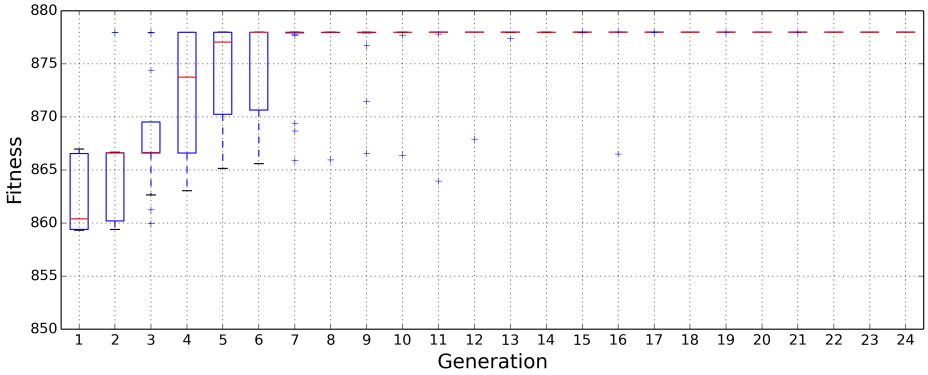
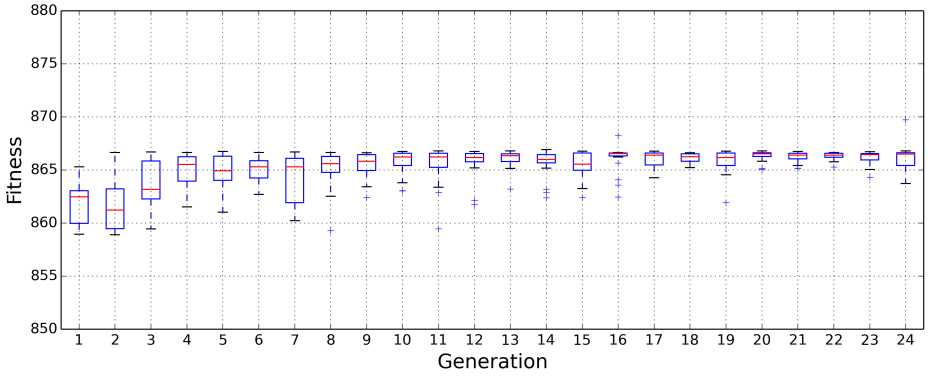


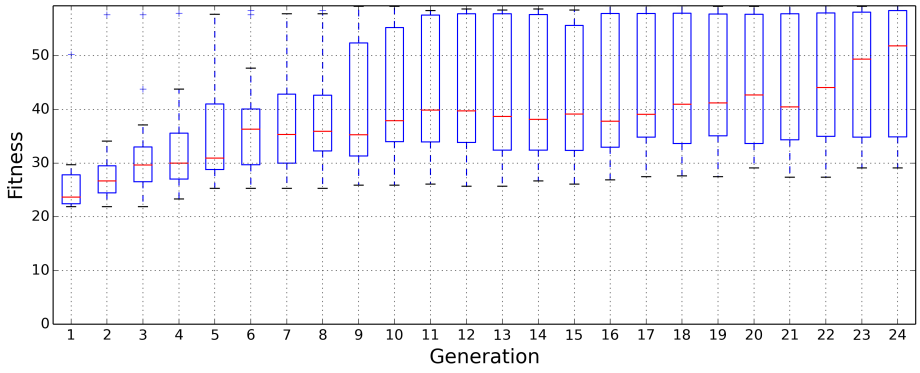
Figure 14: identical parameter set grows the structure differently in different light setups



(a) Growing tall against gravity in a calm environment



(b) Growing tall against gravity in a harsh environment



(c) Growing to get more light

Figure 15: Fitness trajectory of the best genomes over generations collected from all the independent runs.

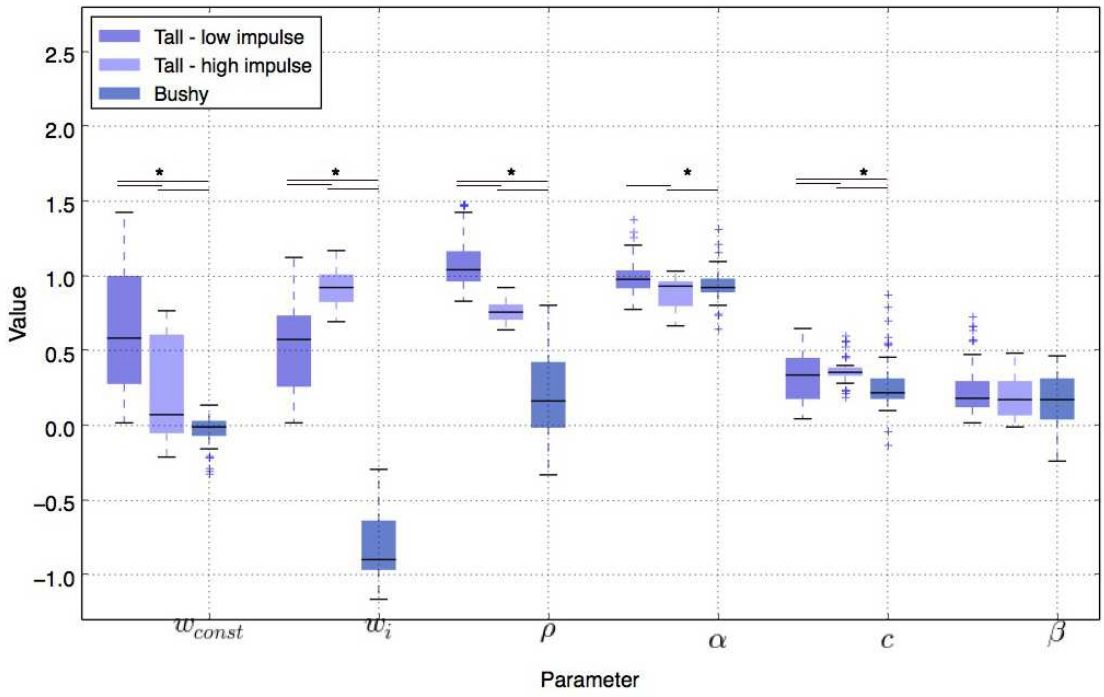


Figure 16: Comparison of parameter values of controllers evolved for growing tall in low and high impulse setups, and growing bushy. The parameters are collected from the best controllers of the runs. The horizontal lines with the asterisks indicate statistically significant differences (Mann-Whitney u-test, $p < 0.05$)

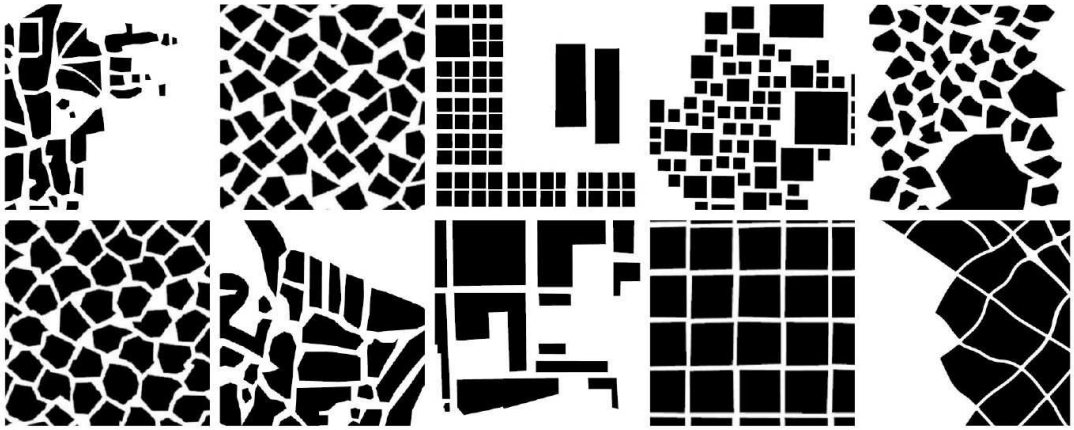


Figure 17: 2-d morphologies of various example groups of aggregated artifacts.

light, and to grow tall in low intensity of light (shadow). For that, we set a ρ_{light} to a negative value such that the high intensity of light reduces the transfer rate, thus a more bushy structures emerge in light. On the other hand, we set the ω_{light} higher than ω_{acc} such that the S production is more sensitive to light than to the angle of gravity. Thus, the parts of the structure that are placed in light are more likely to grow than the parts in shadow. The parameters are chosen as follows: $(\omega_{\text{const}} = 0.01, \omega_{\text{acc}} = 0.6, \omega_{\text{light}} = 1, \rho_{\text{const}} = 0.9, \rho_{\text{light}} = -0.5, \alpha = 1, c = 0.25, \beta = 0.2)$. Fig. 14 shows the result of growing a robot controlled by the same parameters in light, shadow, and half-shadow environments.

In the final experiment, we evolved VMC parameters with the aim of growing a robot in a layered environment such that the structure gets the highest levels of light. The environment consists of several obstacles in different layers. The highest layer has the maximum intensity of light (say 1 unit). With every layer down, and below every obstacle, the light intensity decreases by 0.1 units. The intensity of light is constant all over each block and the nodes are only capable of sensing the local light (not direction). Fig. 13(d) shows the environment and the robot developed by the best evolved VMC in this environment (see a video here: `https://youtu.be/dSkjohr5Nqs`). The leaves use both the accelerometer and the light sensors. In order to evaluate the fitness, the robot is developed up to 80 modules and the fitness is computed as the total sum of the light intensity sensed by all the nodes. Fig. 15(c) shows the fitness trajectory of the best genomes accumulated from all the runs.

3.3 Modeling aggregate properties of low-level morphological change in artifacts

Architectural characteristics that contribute to spatial potential often depend both on local relationships between neighboring artifacts and global relationships between a single artifact and the entire construction of which it is a member. As such, a useful representation of these properties must account for their multiscalar nature. This representation of aggregate morphological properties in artifacts encodes some features of spatial potential, such that they can be used in fitness functions when evolving controllers.

This representation of properties depends on a characterization of homogeneity and heterogeneity in artifact morphology, capturing dependencies and important macroscale information from aggregate low-level descriptions of shape. Aspects of spatial potential such as local and

global density, heterogeneity of morphology, boundary conditions, clustering, and residual occupiable space can be encoded using such a representation (see Fig. 17 for illustrations of such morphological properties). The details and usage examples of this representation will be described in *D3.2 Architectural propositions* in the context of specific architectural propositions of use. The representation will assist in evaluating not only the spatial potential of *flora robotica* artifacts on their own, but their spatial potential in context, when surrounded by existing artifacts in urban or social settings.

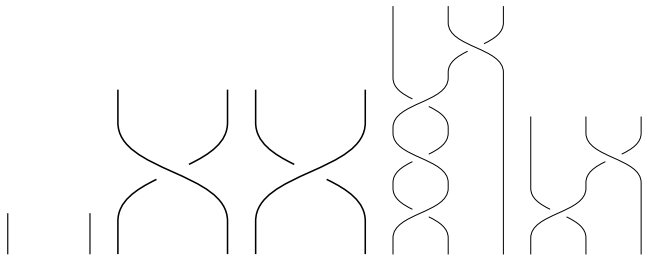


Figure 18: Braids for two braiding robots.

4 Representations of braid as organizational logic

As braid is the organizational logic not only for mechanical scaffolds but for embedded robotic elements, representations of braid are important for integrating models of robotic control into the design of architectural propositions.

4.1 Modeling self-organized construction of braid

In the *1st Periodic Technical Report* and in our paper [21], we investigated early studies into self-organized construction of braided structures, and the modeling of those structures. As reported in [21], in the context of braiding, we have investigated possibilities of the so-called 'Braid Theory' following Artin [2, 6]. We investigate this theory as a mathematical formalism for the organizational properties of the braid (as opposed to mechanical properties). It has been used, for example, to achieve a guaranteed mixing in multi-robot systems [15]. The standard theory is about planar braids and uses the mathematical construct of a group. Different braids or braid operations σ_1, σ_2, σ_3 are distinguished (see Figs. 18a, b, c for the simplest example of two strings) and help to define braids resulting from sequences of such operations. For example, we would get $\sigma_2 \cdot \sigma_3 = \sigma_1$. We say σ_3 is the inverse of $\sigma_2 = \sigma_3^{-1}$ and executing one after the other reverts the braid to its original configuration. Each element σ_i has an inverse element $\sigma_i^{-1} = \sigma_j$, we have σ_1 as the identity element, and associativity holds $((\sigma_1 + \sigma_2) + \sigma_3 = \sigma_1 + (\sigma_2 + \sigma_3))$. With the help of this theory we can take sequences of braiding operations $u = \sigma_i \cdot \sigma_j \cdot \sigma_k \ldots$ as input, process them, and output the resulting braid v by removing reversions from the sequence. This way we implement a map $f(u) = v$. For example, with three strings and for input sequence $\sigma_5 \cdot \sigma_3 \cdot \sigma_2 \cdot \sigma_1 \cdot \sigma_3$ we get $\sigma_5 \sigma_3$ (see Figs. 18d, e). For our work, we need a small change of the standard theory because we are braiding in rings, that is, the left most and right most strings are neighbors and can also switch position with each other. This application of the braid theory also nicely formalizes a concept, that we follow in *flora robotica*. When agents generate persistent structures, we call that 'embodied memory'. The trajectories of the robots are represented as input sequences u and the map $f(u) = v$ gives the memorized structure that only represents the robots' trajectories partially.

Low-level representation of braided structures. To work with the operations of braid theory, we define a graph notation for the low-level representation of braided structures. In this notation, each intersection of strips is a vertex in the graph. The vertices are given a unique identifier at the time of generation. The unique identifier is numeric, but does not carry

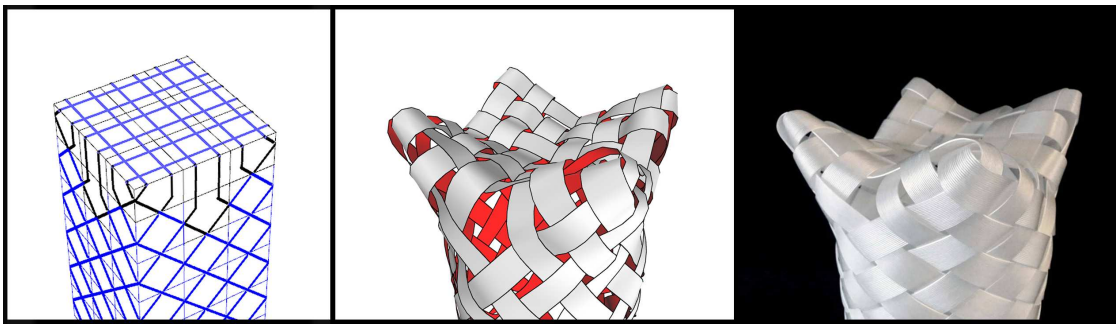


Figure 19: Side by side comparison of the real and modeled foot feature.

information about the vertex location. It is rather defined according to the method of generation being used in that instance. Each edge (representing a segment of a braid strip) is then described according to the identifiers of the two vertices it connects. The strip intersection relationship of under/over is indicated by positive/negative on the identifiers. For instance, if a strip connects vertices 7 and 22, then an edge $\{7, 22\}$ indicates that the strip is in the 'over' position at the intersection having identifier '7,' while an edge $\{-7, 22\}$ indicates that the strip is in the 'under' position at the intersection having identifier '7.'

4.2 Physical braid features that impact geometric modeling

During preliminary investigations into braid as a construction logic, for future deliverable *D3.2 Architectural propositions*, many test objects were hand braided. Through these studies, a great variety of possible shapes was found, much greater than we had previously seen in the context of industrially manufactured braid products. This variety impacted the requirements we pursued for geometric modeling of braided structures.

Biaxial braids are those whose filaments cross each other in only two opposite directions. Triaxial braids add another set of filaments in a third direction, usually parallel or perpendicular to the direction the braid growth. Sheets are braids that resemble surfaces, which result in finite filament elements trimmed at the boundaries of such surface. Solid braids those that resemble solid geometries and can have infinite length filament elements, thus describe a growing direction, Bifurcations are braid splits, where at any point in the braiding process an n-filament braid set subdivides into two or more braid subsets which continue the braiding process individually. Inversions are particular to solid braids with non-uniform cross section filament elements or strips, where the main direction of the surface of a set of filaments is inverted at any point of the process of braiding, resulting in a change from concave to convex bending in the cross section of the strips.

Features such as bifurcations and inversions are necessary to fabricate geometries topologically more complex than a planar shape. As it came clear after developing the first, voxel-based pattern generation/shape generation workflow (described in section 4.3.2) modeling of such features inherently introduces vertices with variable valence. Such vertices make it complex to solve the braiding pattern locally, which becomes a tedious task in the macro scale of the model.

From the pattern generation point of view, the most challenging is certainly to retain the control of the overall arrangement of the strips, that is to be aware of where each of them start and end, and how does it exactly traverse the underlying geometry, also in relation to the other strips. This task required a pattern generation system which is as generic and as robust as

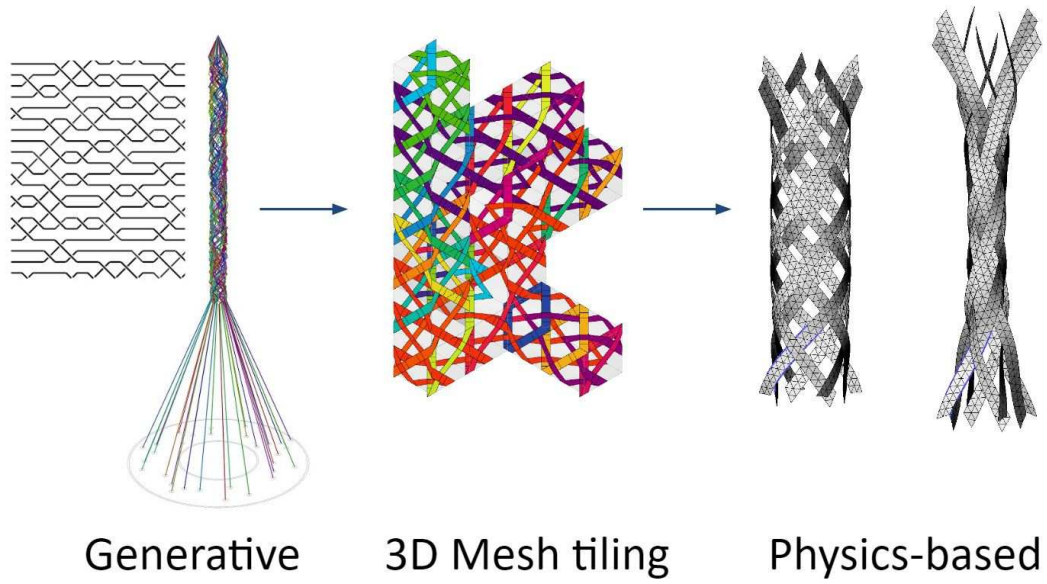


Figure 20: In order to develop a generalized modeling approach that can be the basis for integrated design workflows, the modeling process is approached as three distinct steps, from generative of organizational pattern, to 3-d mesh tiling of the pattern, to physics-based relaxation into a realistic geometric representation of a braided artifact.

possible, coupled with a mesh generator which can create a acyclic topology used as a guide for the pattern generation system.

Bifurcations A bifurcation if modeled incorrectly can cause some strips to turn back the direction of braiding, rendering the model useless for machine fabrication. There were two solutions to that problem developed, one for the voxel based system (described in section 4.3.2), the other one for the tile based pattern generator as seen in section 4.3.3. Both of those solutions take the constant number of strips in any cross-section as the target, which is the indicator of the local braid topology correctness.

Non-industrial braid features While industrial braiding machines are limited with the constraints of the directionality and constant number of strips per branch, there are some interesting features possible to be braid by hand. One of those is the "foot" as seen in the Fig. 19, which requires to introduce 2 types of patterns: the regular braiding pattern and the so-called weaving pattern as seen on the top of this model. It implies the pattern generation system which isn't simply based on one type of tile (which was already seen in the literature before [29]), but has to introduce more tile-specific patterns. This model served as an inspiration to create the second representation approach, the tile-based system.

4.3 Geometric modeling of braided artifacts

Realistic geometric modeling of braided artifacts is an important representation task in *flora robotica*. For users to engage in interactive evolution with the growth of their individual *flora robotica*, as part of the future *D3.3 Internet-connected social garden*, the user needs access to

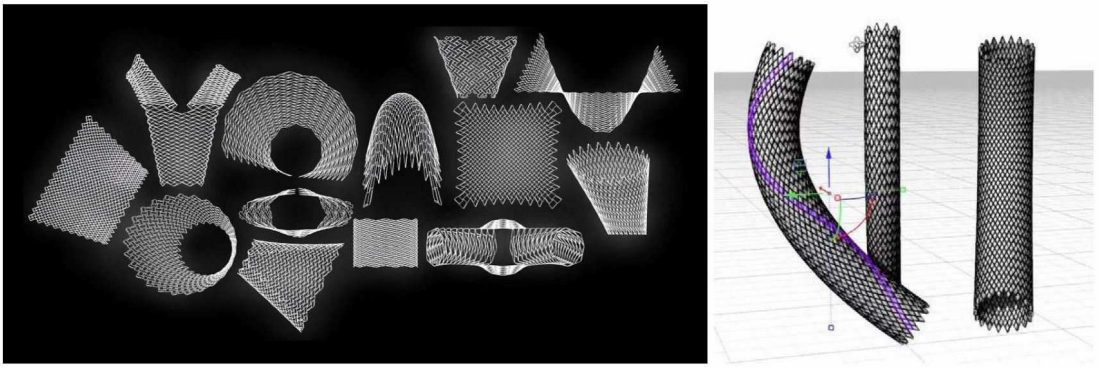


Figure 21: Left, quadrilateral polygon mesh 3D models of many braid types. Right, an example of tubular braid models skinning simple graph representations, with one continuous 3D mesh strip highlighted.

3D models of the system's expected long-term growth that have enough realism to be easily understandable by a naïve viewer. These models should also be detailed enough to be subjected to analyses according to high-level architectural objectives (such as global density, heterogeneity of morphology, etc). Furthermore, the representation of braided structures should link generative methods (such as the VMC controller, Sec. 3.2.4) to fabrication techniques (such as braiding robots or manual braiding). In order to achieve this link, the modeling of braided artifacts is partitioned into three primary steps. A generative process is used to define the organizational pattern of the braided artifact, which is interpreted into a 3-d mesh geometric model, which is then relaxed with a physics-based approach into a geometric model that realistically represents the physical braid (see Fig. 20).

This section focuses on the geometric modeling of braided artifacts, which contain the fluidly combined functions of braided mechanical scaffolds and braided robots (see *D1.2 Evaluation of mechatronics prototype of the robotic symbiont including supporting software*).

First we tried a purpose-specific approach skinning macroscale graph edges with stiff cylindrical meshes that approximated a braid pattern. After extensive exercises in the manual fabrication of braids (see future deliverable *D3.2 Architectural propositions*), we abandoned that approach for the advantages of a generalized approach. The generalized approach is able to build braid meshes around an input macroscale graph, like the purpose-specific approach, but is also capable of handling a vast array of other inputs. Furthermore, the generalized approach is able to encompass the wide range of geometric complexity that we discovered was possible in braided structures, and is able to geometrically model complex examples in a sufficiently realistic way, by incorporating physics-based relaxation. This allows for more sophisticated analyses than the simplistic purpose-specific approach. Beyond that, the generalized approach is able to solve for continuity of filaments and other fabrication parameters, making the representation useful for the process of construction, rather than merely for visualization. On the whole, the generalized approach in a substantial improvement in sophistication and usefulness of our geometric modeling of braided artifacts.

4.3.1. Preliminary study of a purpose-specific approach

In the preliminary directed pipeline, the input is a graph representation with variables assigned to each edge of the graph. The variables are braid diameter along the x-axis, braid diameter

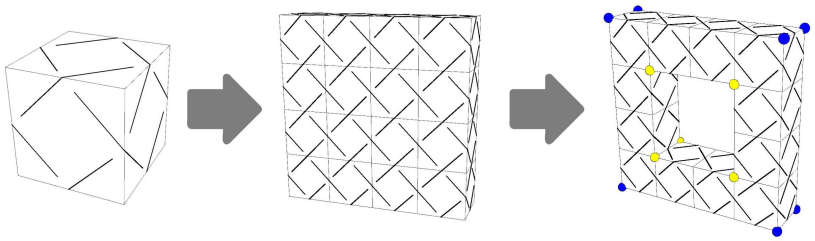


Figure 22: Image showing the voxel-based workflow. Given a voxel with a simple tile pattern, it is possible to stack it retaining the braiding pattern correctes (in terms of under-over order of the strips.) The pattern gives a solution to all of the possible local vertex valence configurations, like 3- and 5-connected vertices shown in color.

along the y-axis, number of filaments, width of filaments, and width of gaps between filaments. These variables, and the xyz-coordinates of the graph nodes, are used to define a global master surface with properties such as branching and openness. The master surface is subdivided, according to the variables, into a polygon mesh with diagrid quad faces. Each quad face is divided into four braid filament faces, which are offset on both sides, forming continuous braid strips.Simple deformation of the braids is modeled geometrically by manually manipulating the xyz-coordinates of a node in the graph (see video[16] of deforming braids). Edge lengths in the graph are constant; when a node's coordinates are manually changed, the coordinates of the other nodes in that component are automatically updated.

4.3.2. Overview of the generalized approach

The methods of representation for the generalized geometric modeling approach are defined to encompass the great variety and potential complexity found in *flora robotica* braided artifacts.

Background to the approach. Assuming no higher-level qualities of the braid pattern are the interest, a simple representation of the braiding pattern can be achieved by discretizing the target geometry as mesh geometry using any method. With a single tile it is possible to create a braid/weaving pattern on any topology, yet there is little to nothing control on the global properties of the strips. This approach was exhaustively explored in the literature and is suitable for non-machineable fabrication processes [29].

There are two widely different expectations towards the pattern generation system which are in the scope of the project. The first one is to create a flexible and easy to use system, so that creation of the patterns is made as easy as possible. Such patterns should represent a variety of combinations and features which were analyzed in the previous sections of this deliverable. On the other hand, in the context of robotically fabricated braids, the system should be predictable and robust, with a structured organization of the pattern, so that it can serve as a basis for generation of the robot instructions. Both tasks are approached with two systems: the voxel-based system and the tiling system, which proved the superiority of the latter. The tile approach is further equipped with graph based mesh generation method which yields topologies possible to fabricate with unidirectional braiding robots.

[16] https://vimeo.com/169727229

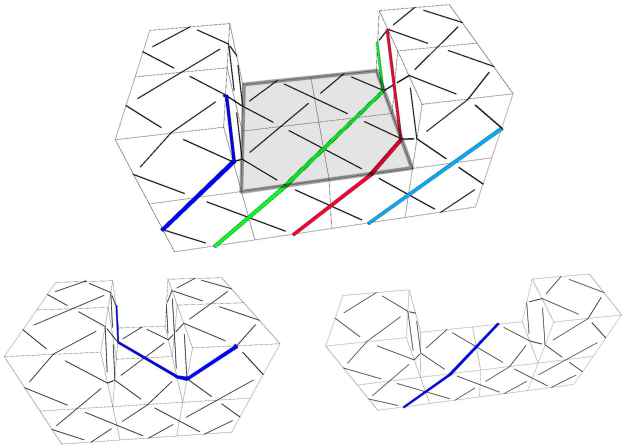


Figure 23: The top bifurcation preserves the direction of the strips, while both of the bottom cases break this logic.

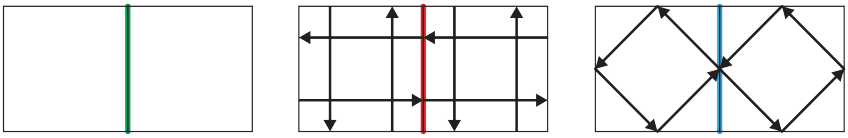


Figure 24: The three possible interfaces between tiles.

The initial approach Initially the pattern generation was based on a single pattern applied to each of the voxel sides, which thanks to its properties and symmetries was able to yield a pattern for any voxel configuration. The problem arose when a greater degree of control over the macroscale pattern properties was required. It was necessary to introduce additional rules in this system to constrain the braid to the fabrication requirements, which is best seen in the bifurcation case.

Whenever the bifurcation occurred, the introduced gap between the voxels had to have a square shape, giving each edge of the square the same amount of strips passing through (Fig. 23). This rule yields a bifurcation solution which is possible to braid in one direction (which is the braiding robot requirement).

With one type of braiding pattern and only one type of possible bifurcation, this method fails to provide a simple solution when the global shape's bifurcations don't match the described pattern. Nevertheless, this is a concise workflow which surely can produce a wide range of complex geometries.

4.3.3. Pattern generation through a tile dictionary

The pattern generation system[17] takes inspiration from the methods described in the existing literature. The primary advantages achieved by the developed method are simplicity and generalizability. It is tailored to work with orientable manifold mesh geometry, but the same method can be easily generalized to braid in 3d (volumetrically). The complexity of the resulting patterns cannot be simply classified as braid, plain weave or triaxial weaving, yet the method is able to reproduce each one of them. It is based on Mercat's method [27] but it is extended in a way that supports many additional types of braiding and weaving, also mixed together in various configurations. The greatest advantage of the described method is the implicit nature of the under-over relationships, which results from the set of the tile generation rules described in this subsection. In general this method uses a predetermined set of tiles, which, in their underlying logic, combine different approaches seen in the literature. Just as the voxel-approach this system requires some additional rules to guarantee the macroscale pattern is possible to be fabricated with braiding robots. Those rules and the solution for their preservation are further described in section 4.4.1.

The first step in the pattern generation workflow is to define how many types of interfaces are possible between the tiles (i.e., determining the possible transition cases). In the developed system, there are three such cases are: no connection between the faces (Empty), connection with two strips separated (Plain), and connection with two strips passing through the middle point (Braid, as in Mercat's method). The interfaces are hereafter interchangeably referred to as colors. (Fig. 24)

Once the number of colors is defined, it is necessary to define the number of types of polygons. (Though the dictionary and implementation encompass both triangles and quads, for clarity, the description will refer only to quads). Given the number of polygon sides to be colored and the number of possible colors, a finite set of color combinations becomes evident. The precise number of combinations can be calculated using Polya's Counting Theorem [28].

For three colors and four-sided polygons, the number of possible color combinations after reducing the rotational symmetries is equal to 24. The implementation uses tuples to identify a particular color combination. By sorting the tuples, the user gets a convenient overview of the complete set, useful for editing the predefined tiles. An example tile set can be seen in the Fig. 25.

[17] See images at https://zenodo.org/record/437462#.WN6fUk1MR9A and source code at https://github.com/florarobotica/polymesh-braid.

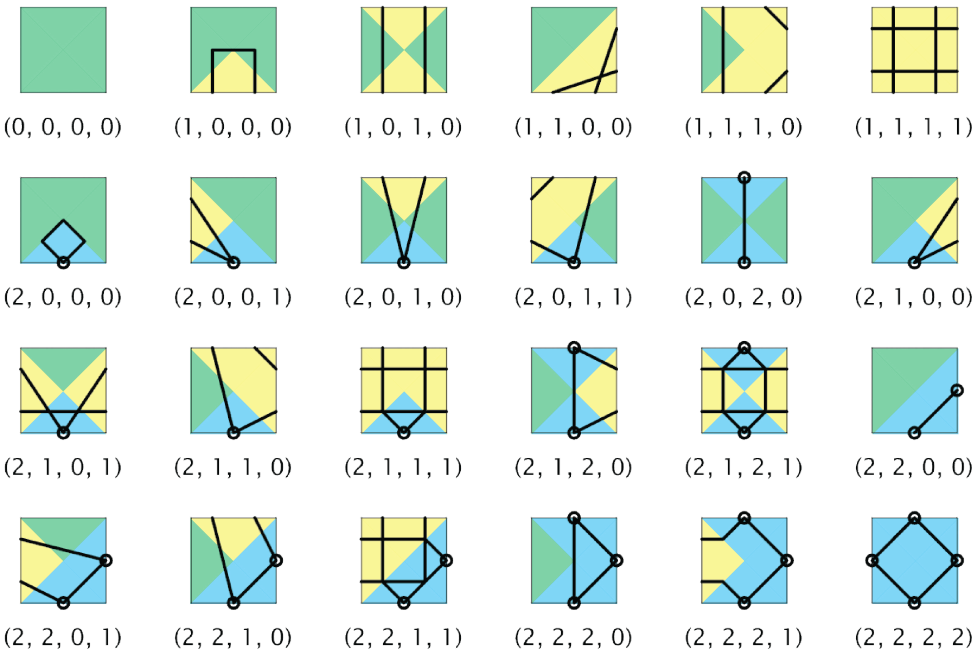


Figure 25: A complete tile dictionary for the 4 sided, 3 colored braiding pattern.

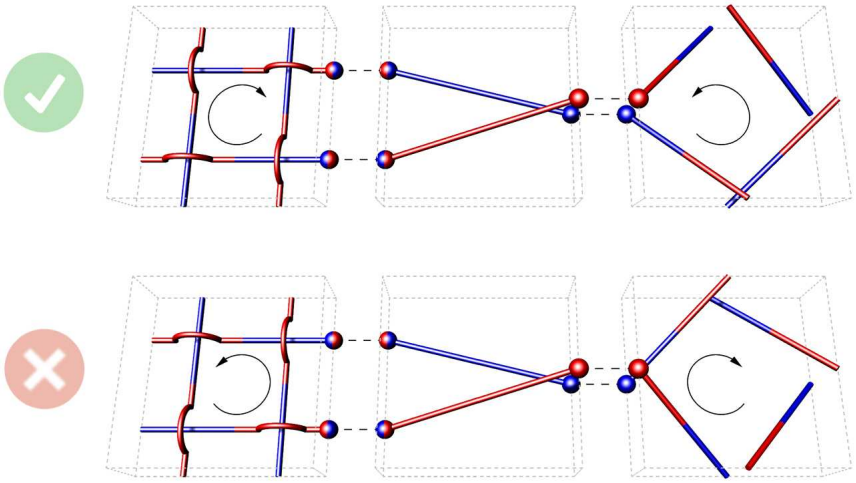


Figure 26: Notation and elemental rules for defining tiles.

This dictionary is therefore a Wang tile set [33], with all the possible combinations of colors. However, from the point of view of the user, it is more intuitive to consider the colors of individual edges than particular tiles. It is important to note that contrary to Wang tiles, the ones presented here can be rotated, such that the number of combinations required for a complete set is reduced by rotational symmetry.

Tile topology rules While the edge colors describe the overall pattern, the tiles have to be designed with the interfaces in mind, so that the resulting pattern has a desired under-over order (in the implementation it's a plain weaving order). There is a potentially infinite amount of weaving patterns with a particular color combination, so the rules specified here have to apply only to the strips which are taking part in the tile-to-tile interface (contrary to cycles contained wholly within a tile).

As mentioned before, there are three interface types in the implementation, from which two have to be resolved - the Braid and the Weave interface (Empty transition is skipped because it is an obvious case.) The blue color hereafter represents the under state and the over strips are marked with red.

In case of the Braid interface the coloring is straightforward, the under point is marked with blue color as seen in the Fig. 25 and the over point is red. It is important to design tiles in a way following the colors and their meaning. Not crossing over with a red strip means that the blue part wont consequently go under.

The interface points notation has to be modified in case of the Plain interface. Each point has 2 colors indicating the order of the strip, which means that the strip changes its order after passing through it. Note the left and right points are marked differently, therefore mirroring the tile will produce a wrong interface as seen in the bottom case in the Fig. 25. Once all the tiles are solved, it is possible to tile any orientable surface with any combination of colors. A small sample of colorings and resulting patterns is shown in the Fig. 27.

Bifurcation solution using the tiling system (mesoscale feature) The tiling system has no information about the global topology of the underlying geometry, and that's why it is necessary to develop a mesoscale bifurcation solution which meets certain local criteria. Those are: each branch has to have it's own direction, and the thickest branch has to lead inwards, while the two thinner ones lead outwards (or the exact opposite); thickness of the largest branch has to be equal to the sum of thicknesses of the other branches; lastly there can be only one inward branch, but as many outward as needed (if that condition is not met, the solution for such case is ambiguous.) The method for generating such geometries was developed and is described in the 4.3.5 section.

4.3.4. Strip geometry construction and discretization

To be able to simulate the physical characteristics of the braiding pattern, it is necessary to translate the tile notation into geometry. This section describes the construction of such geometry at the tile-level.

While it is possible to define a continuous parameter space for geometry construction as shown in [1], the described method uses a point grid to ease the tile geometry construction process, making the task of strip designing less error-prone. Each strip is declared as a series of grid-based coordinates. The grid with the underlying strips is mapped onto a B-spline surface constructed from the 4 corner points of a particular mesh quad. There is a small chance of self-intersection, particularly when the target face is non-planar. The under-over order is expressed by offsetting strip points by an interpolation of faces' vertex normals.

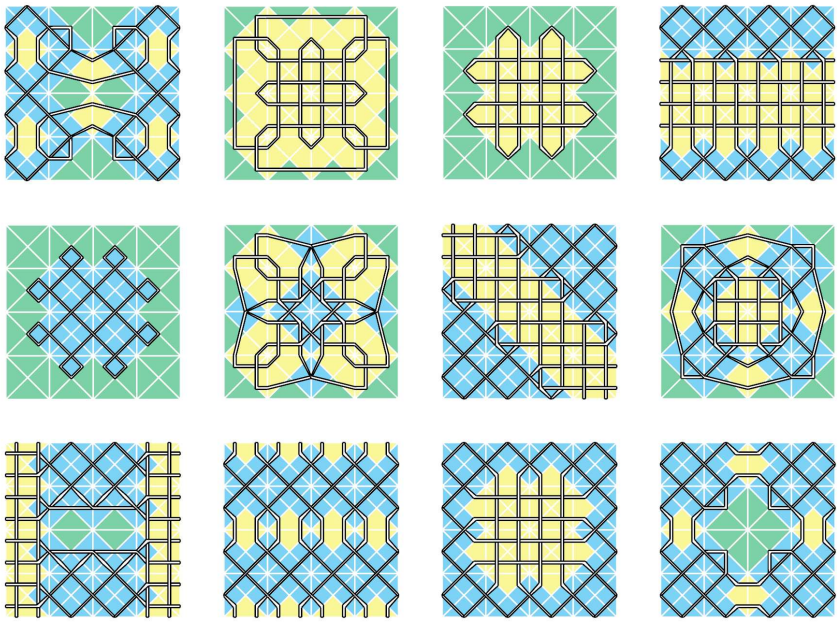


Figure 27: A set of examples showing the possibilities given by the tile-based system.

The grid points are also used in the process of mesh drawing. As shown in the Fig. 28, the point P1 is the point from the predefined strip. The angle of the strip turn determines the choice of the P2 and the P3. The resulting variance of the mesh width is later unified by the geometry solver.

The last step is to divide each of the strip segments into triangles, having in mind the final length is dependent on the number of divisions as seen in Fig. 29. While it's not the most intuitive way to set the final length for each strip, it provides an uniform and simple triangulation method.

4.3.5. Preliminary geometry relaxation at the macroscale

While the simulation method described in Sec. 4.3.6 is able to precisely simulate the physics of the braid at the microscale, with bigger braids it can take up to few minutes to converge (benchmarks in section 4.3.7). Therefore a method to quickly come to a roughly precise shape was developed. It uses the same projection based geometry solver, which is given the mesh geometry from the section 4.4.1 to process. The constraints supplied to the solver are:

1. Maintain minimal distance between the mesh vertices (realized via sphere collision)
2. Treat each mesh edge as a spring (realized via springs following the Hooke's law)
3. Start with all vertices anchored at their initial positions and during the relaxation decrease the weight of this constraint.

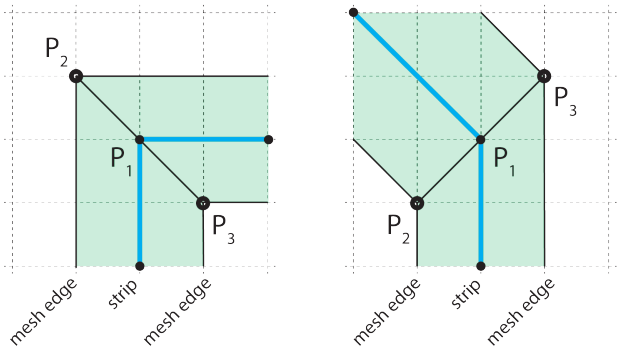


Figure 28: The strips initially vary in width, which is a result of the grid based construction. This will be later corrected in the relaxation routine.

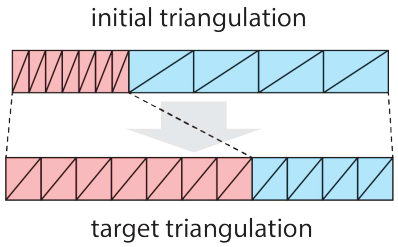


Figure 29: The geometry solver has an objective to equalize all the mesh edges to the same length. The discretization of the initial geometry is one of the input parameters for the solver and determines the final length of each of the strip segments (red and blue).

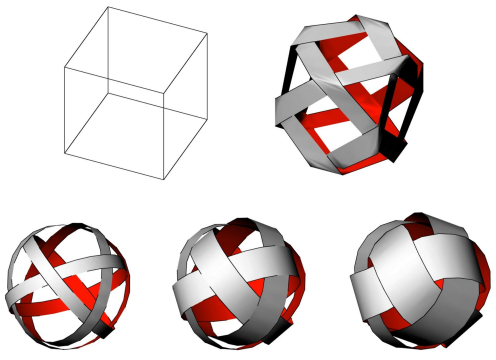


Figure 30: From left to right: base mesh model – initial triangulation – geometry solver. The geometry solver equalizes strips gradually to prevent from overshooting of filaments.

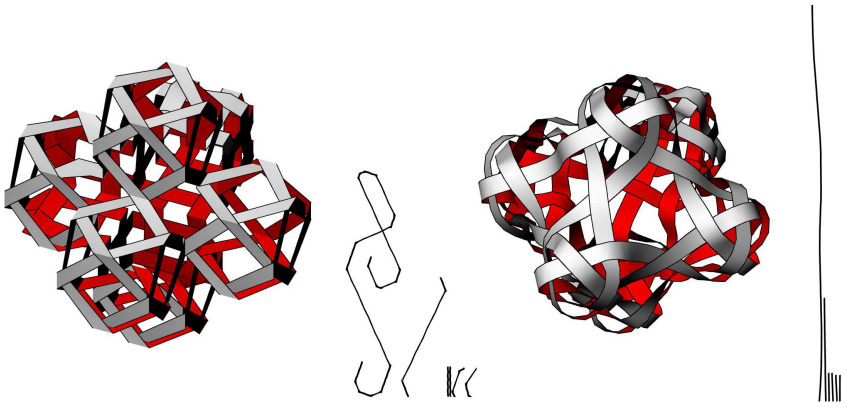


Figure 31: Left: mesh original geometry Right: mesh after geometry solver

4.3.6. Geometry solving and simulation at the microscale

Here we conduct physics-based geometry solving and simulation at the microscale[18]. In order to evaluate physical properties of digital model such as strip-strip interaction and self-weight, strips cannot be simplified to a grid-shell like model, because of lack of the torsion induced by the flat strip geometry. As a result mesh topology is constructed from triangle meshes with varying density to properly model the physical properties. The constraint-based geometry solver Kangaroo2 plays the critical role for the simulation and fabrication purpose. It both enables to predict the anticipated physical appearance, but it also forces the generated mesh strips into straight shape thanks to the custom coded constraints. By unrolling the mesh strip geometry it is possible to evaluate how close digital geometry is to the physical prototypes - the straighter the unrolled strip, the closer it is to the straight band used in the fabrication process.

Detection of collision is a great impediment in case of 2-dimensional manifolds. Because of the discrete simulation time-step and geometry with no thickness, rapid check of the side of collision can be rendered impossible (depending on the approach). The under-constrained nature of fabric simulations makes this problem even harder, as a variety of types of collisions occurs in such simulations: self-collisions, low and high speed collisions, etc. [7].

The spatial grid method [31] is chosen for collision detection as it is appropriate for range searches in a dynamic environment of the simulation and it is not necessary to rebuild spatial grid for each frame. The initial mesh model as established in section 3.2 defines the relationships between the particles and utilizes the naked edges of the mesh for collisions. The middle points of those edges are stored in the hash table and used to quickly find nearby points, therefore the necessity to perform time-consuming line-line collisions is greatly reduced.

The 3D bucket size is set to half of the target length of the naked edge. While in the initial geometry there are edges of various lengths, the solver aims to equalize them to the target size and therefore the collision detection gets more robust over the time of the simulation (Note the collisions are getting more frequent over time as well.)

Overshooting may be caused when the triangles of the mesh are undesirably stretched or compressed by large percentages. A rule of thumb in computational mechanics is that triangle edge should not change length by more than 10% in a single time step [10] to prevent it. This can be enforced by either adaptively decreasing the time step or decreasing the strain rate (the latter

[18] See images and videos at https://zenodo.org/record/438631#.WN6gsE1MR9A and source code at https://github.com/florarobotica/Kangaroo2-Braid-Simulation.

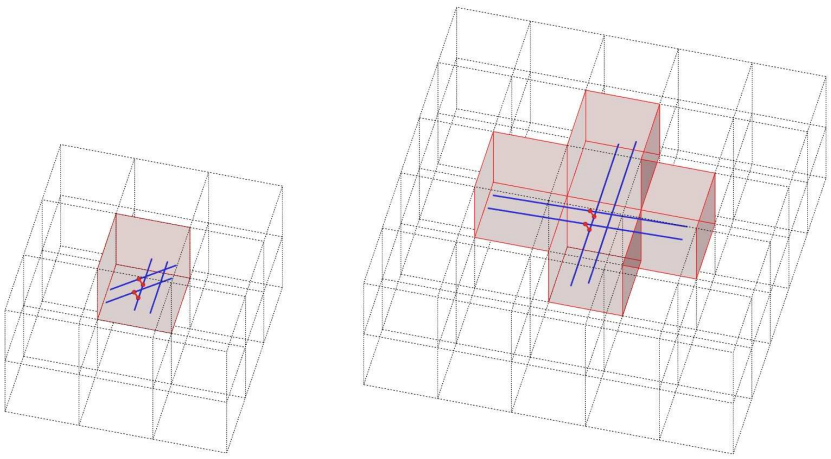


Figure 32: The spatial-grid consists of small bins (cubes) and each cube may contain a pair of lines to detect a collision. If the bin size is too small it is possible to include neighbor bins for collision detection.

is used in the implementation). In usual cases, the projection solver would change the length of each edge to the target value as quickly as possible and in few iterations the under-over order would be lost. A dynamic constraint is introduced as a solution for this overshooting problem. It starts with the initial edge lengths and incrementally adds up to reach the target. This results in longer calculation time but helps to achieve extremely tight braids with strips composed of well shaped isosceles triangles.

4.3.7. Results of the generalized approach

The geometry obtained with the presented method is closely following the manufactured prototypes, as seen in the Figs. 19 and 36. As the problem of cloth and fabric simulation is by definition an underconstrained problem, it is unreasonable to expect exactly the same results. For the purpose of design exploration and macro-scale behavior prediction, the presented approach seems to be accurate enough. The side-by-side comparisons reveal the same kind of artifacts emerging in the real-world pieces and their virtual models. Furthermore the strips which after unrolling remain straight, satisfy the fabrication requirements

The simulation method was tested on a series of different models and topologies, with varying complexity and triangle count. When it comes to performance, thanks to the projection-based geometry solver a simulation with a triangle count between 1,000-10,000 converges after 15-120 seconds depending on the case (Fig. 37 b,c,d,g). Smaller forms like example (e) with few hundred triangles, reach their final shape in a couple of seconds. The amount of time required to reach equilibrium is also dependent on the global shape, and with simple forms like the example (a) (around 40,000 triangles) it took 120 second to converge as well.

Modeling predetermined design targets. Mesh tiling helps to represent hand made braid models, when predetermining mesh edge colors to change braid directionality. In order to model a “feet” typology (see Fig. 38), strips have to change their orientation on the top and the bottom parts. Furthermore, it is necessary to control the subdivision of mesh strips because it has a direct relation between geometry solver and wanted geometry. For example, “foot” was modeled in such

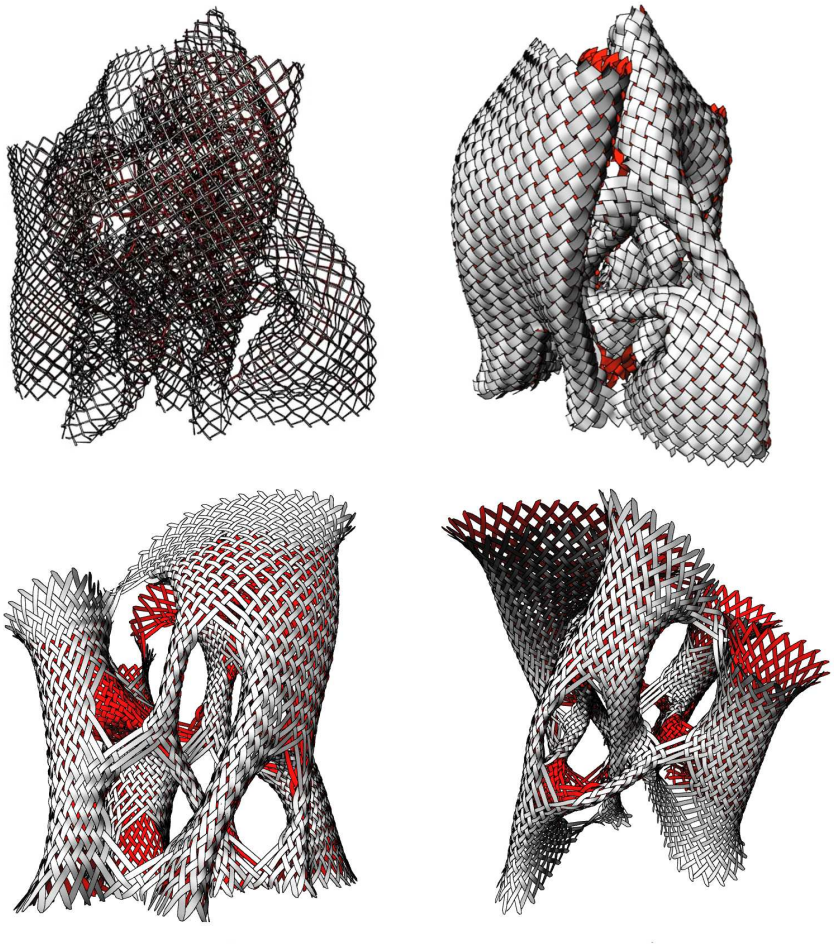


Figure 33: Top left: initial mesh typology. Top right: after geometry solver. Bottom: initial mesh typology after geometry solver with lower value for mesh edge length.

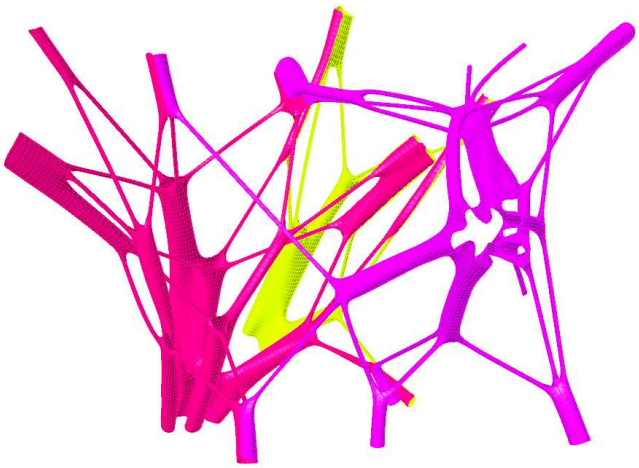


Figure 34: A mesh and a properly directed global graph make braided structures possible to braid in one direction (with a braiding robot for instance).

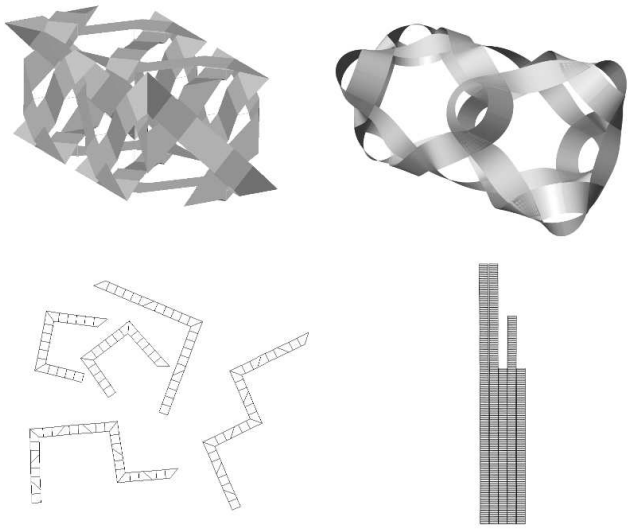


Figure 35: Left: Initial mesh and unrolled geometry. Right: Strips after geometry solver.

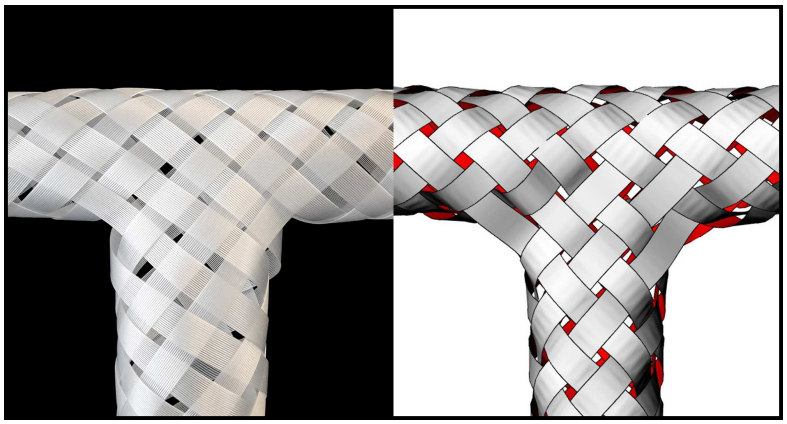


Figure 36: A side by side comparison of the real and simulated geometry of the T-junction.

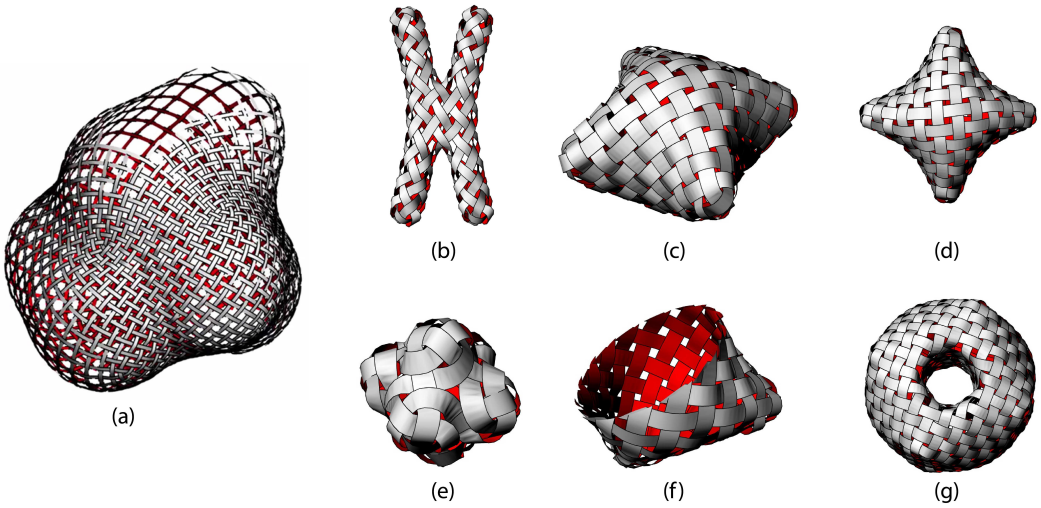


Figure 37: Various simulation results.

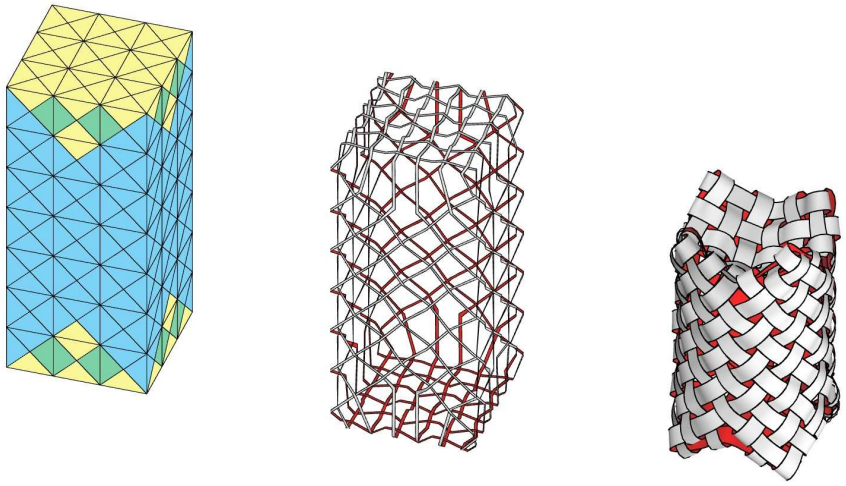


Figure 38: Modeling "foot" geometry of braid.

a way that green tiles had very low resolution meshes. In this way strips were shortened and resulted in four pointed corners. Calibration between mesh tiles orientation and mesh resolution has a direct relation to geometry solver and has to be adjusted each time to have one to one physical appearance of physical artifact.

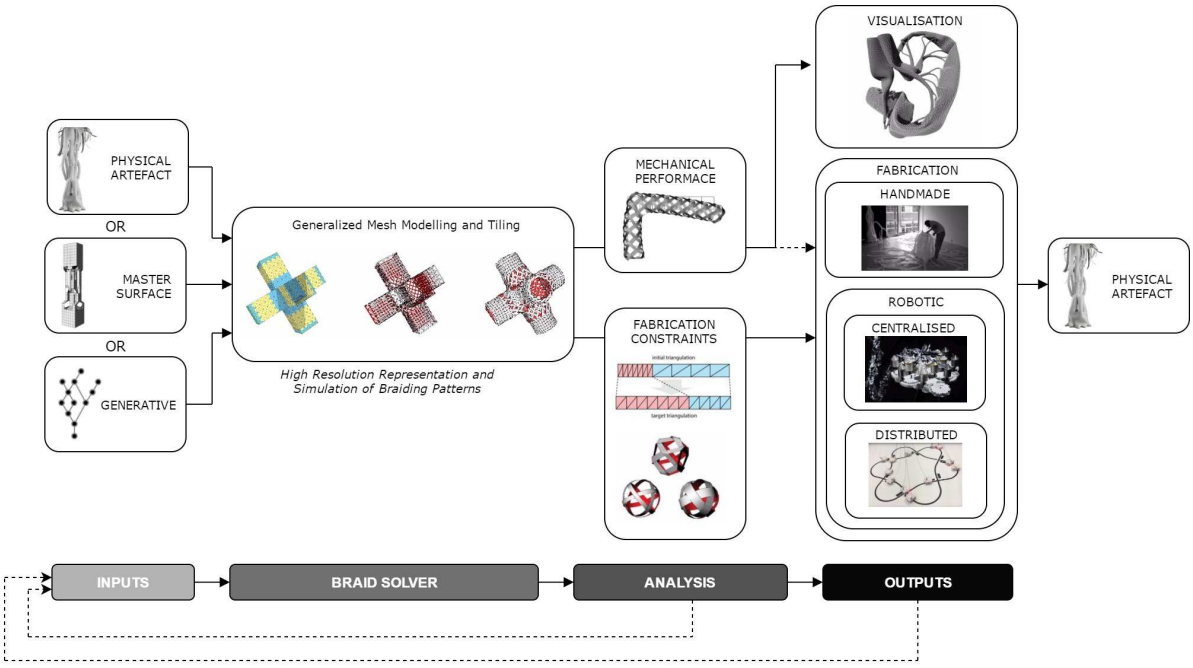


Figure 39: Overview of the workflows supported by the generalized modeling approach for braided artifacts.

4.4 Integrating the generalized mesh modeling approach with design workflows

The generalized modeling approach can be based on many inputs, including a target physical braided artifact, a target master surface, or a generative input such as the VMC. The generalized mesh model can be interpreted in several ways, such that it can be used to generate hand braiding instructions, instructions for a centralized braiding machine, or controllers for self-organizing mobile braiders. See Fig. 39 for an overview of possible workflows. Workflows using these various inputs and outputs are developed as part of future deliverable *D3.2 Architectural propositions*. An extension to the generalized modeling approach is needed to support these workflows, in order to reliably model braided structures that can be manufactured with the centralized braiding machine described in *D1.2 Evaluation of mechatronics prototype of the robotic symbiont including supporting software*. This extension is detailed below.

4.4.1. Fabrication-oriented macroscale design for tubular braids

To guarantee that a braiding pattern can be fabricated with a braiding robot, the underlying mesh has to have all the faces marked with a direction pointing in the fabrication direction. A macroscaled directed graph approach was developed to generate meshes following this constraint. The graph serves as a scaffold for mesh geometry generation step described in subsection 4.4.1, and on it's own has to comply to few constrains given by the fabrication method and mesh generation routine. Those requirements are:

1. The graph has to be acyclic.

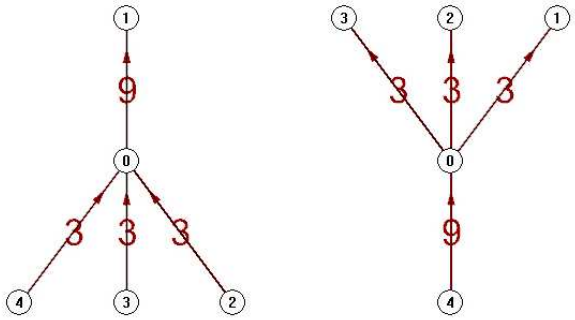


Figure 40: Two possible bifurcation cases. Single-in, multiple-out and the other way around.

2. The "start points" (sources) for the braiding fabrication procedure are defined as the vertices without any incoming edges.
3. The "end points" (targets) are the points which don't have any outgoing edges.
4. Bifurcation can happen only to one branch at a given vertex. This is represented as one incoming edge with multiple outgoing edges and vice versa.
5. All possible walks from the "start points" to the "end points" have to have the same length.

In the first step the user defines any undirected graph which is then fed to the solver. The solver performs a breadth-first search ordering on vertices and assigns values to each of them, indicating the distance to the topologically closest source (which is to be manually defined by the user). In the next step the solver traverses the graph vertex-by-vertex, solving the local neighborhood of each. The vertex can only have an edge pointing to another vertex with greater distance to the source value. This means some of the edges will have to be skipped in the final, directed-graph. An example showing the result of this procedure is shown in the Fig. 41.

Deriving the edge values Given the acyclic directed graph, the next step before obtaining the mesh is to assign values to each edge, which will indicate the number of strips running through. It is derived using a simple approach:

1. Find all the possible paths from each source to each output vertex.
2. For each path:
3. For each edge in path:
4. Check if the edge was visited already. If not then mark the whole path as necessary.
5. If the path is necessary:
6. For each edge in path:
7. Add 1 to the edge strip count.

This routine is guaranteeing that each edge will be visited once, meaning no edge in the directed graph is skipped. Because of the source-to-target continuity of each path, it is guaranteed that each bifurcation will meet the strip value conditions as previously described in Sec. 4.2.

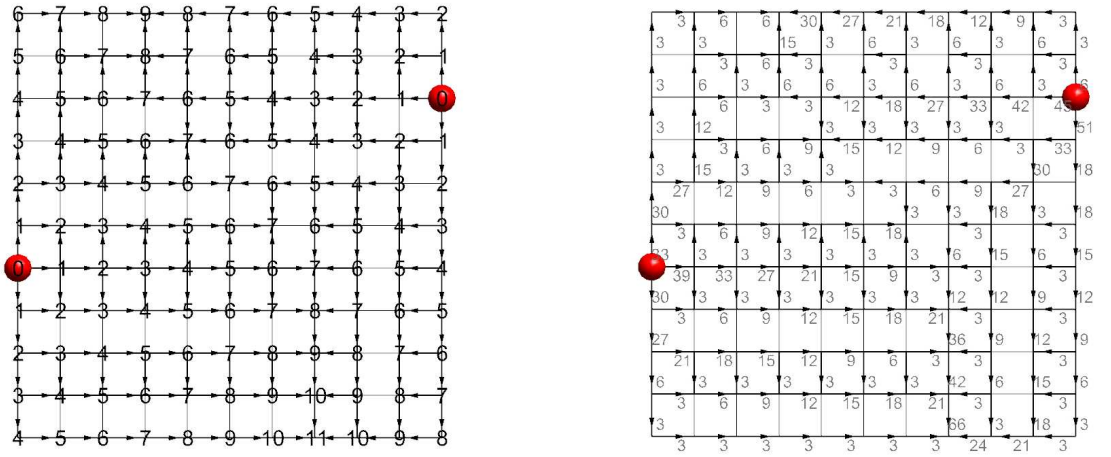


Figure 41: Left: graph with solved edge-directionality. Adding any of the missing edges would break the graph constraints, therefore rendering the resulting braid impossible to fabricate with a braiding robot. Values indicate the distance to the closest source (red). Right: graph with assigned edge values.

Deriving the edge lengths An another fabrication constraint has to be satisfied at this stage. Given the source-to-target paths from the previous step, it is necessary to set each edge length so that any walk in the directed graph will result with a path of equal length. The method is as following (illustrated with the Fig. 42):

1. Perform breadth-first search ordering on the graph vertices
2. For each vertex in the BFS order:
3. a. Get all the topological distance values V of the vertices leading to the current vertex.
4. b. Find the vertex with the greatest value V
5. c. Assign value V+1 to the current vertex.
6. For each edge in the graph:
7. a. Set the edge length as the difference between it's head and tail vertex values.

Deriving the macroscale geometry from the graph Once the directed graph has the edge strip counts assigned, the mesh topology is possible to derive. There are two main routines in that process: the first one meshing the nodes, the second one is meshing the edges.

The node meshing has few subroutines for robustness, which are applied depending on the number of adjacent nodes. While a node connected with 2 others is a trivial case, a node connected to 3 and more nodes requires a separate procedure.

Each edge meeting at the node is queries for it's value and the one with the greatest value is marked as the "master". On each edge there is circle perpendicular to the edge direction constructed and trimmed in radius so that it doesn't intersect with any of the other circles in

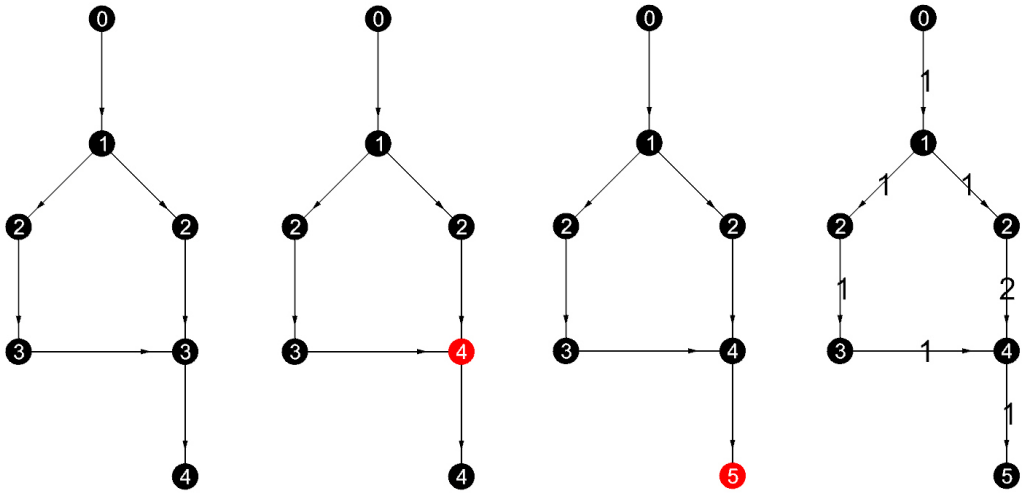


Figure 42: Deriving the edge lengths.

the node. The "master circle" assigned to the "master" edge is then splitted into arcs associated with each of the smaller circles. Finally, for each "master circle arc" and "small circle" pair, a mesh is created by dividing each of the curves into segments and lofting each segment pair afterwards. The number of segments is indicated by the edge value of the small circle.

The points on the circles created in the node generation procedure are stored in a list and queries by the strut meshing method. For each graph edge it collects the points from both ends of the edge and through a brute force search finds the orientation of the points yielding the shortest sum of loft lengths between the points.

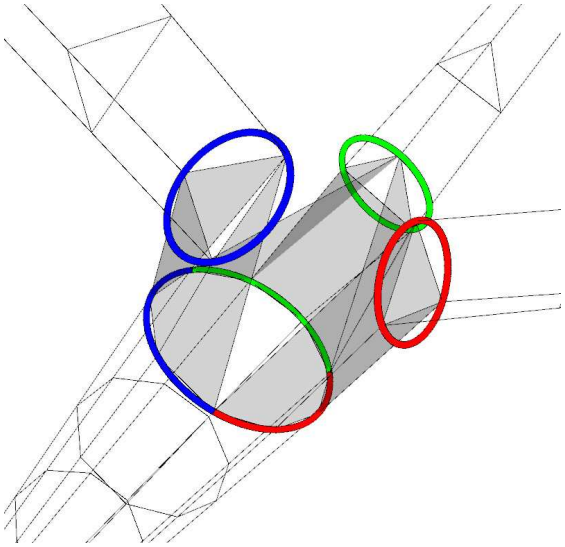


Figure 43: A node geometry (in gray) with "master circle" split into 3 parts for each of the struts.

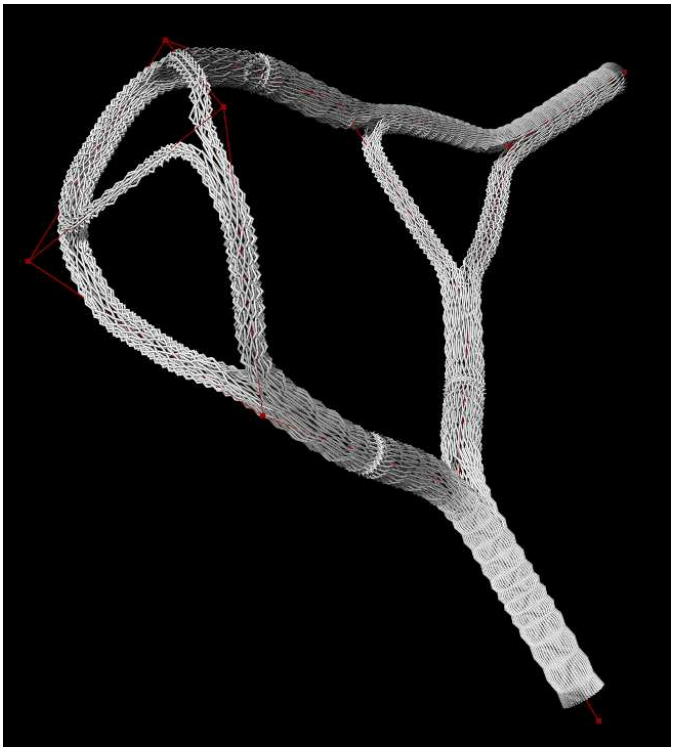


Figure 44: Mesh generated with the described method. The strip pattern is naively meshed by thickening the tile polylines.

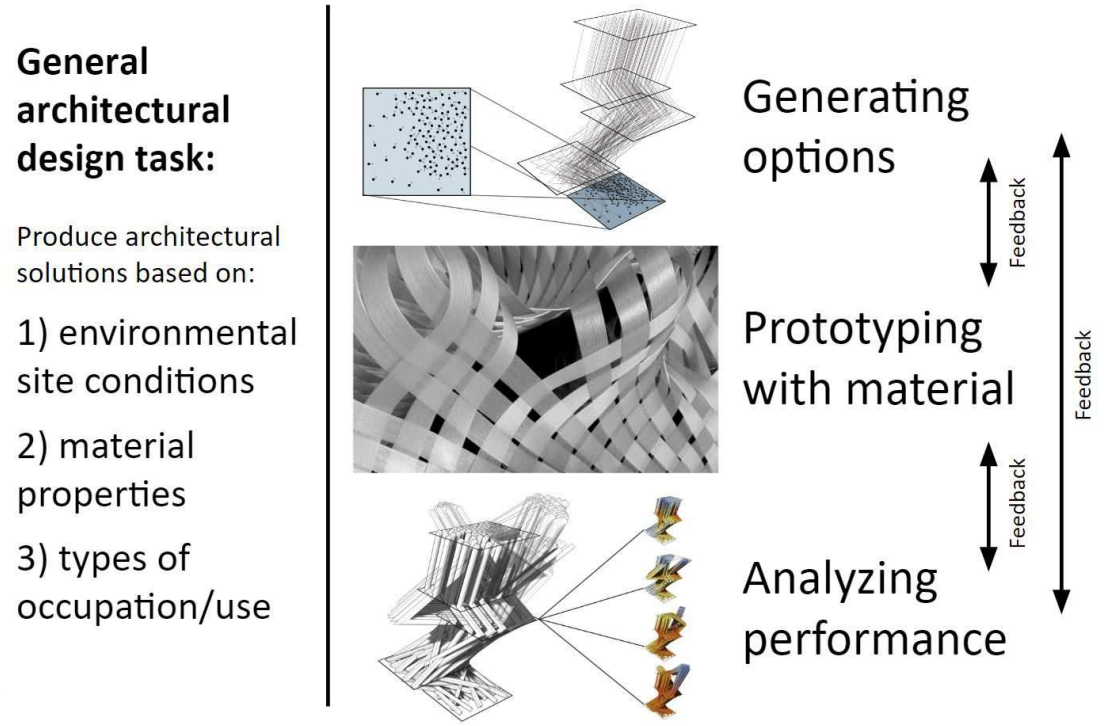


Figure 45: The general task of architectural design.

5 Evaluation of modeled artifact propositions

The representations introduced above serve as the basis on which propositions of use will be evaluated in*D3.2 Architectural propositions*. Realistic geometric modeling of braid (Sec. 4.3) will be crucial for evaluations of structural and environmental performance and spatial potential. Artifact states generated by robotic controllers such as the VMC (Sec. 3.2.4) will be modeled in full as braided artifacts, and then evaluated. In the context of architectural design, characteristics and objectives that are encodable are crucial to assessment, alongside more abstract considerations that require human judgment through visual analysis.

Encodable objectives. Evaluation of a controller's performance according to encodable high-level objectives will be supported by the formation space (Sec. 3.1) representation of that controller, which can be used to assess the predictability of achieving the objectives. Evaluations of spatial potential will be supported by modeling aggregate morphological properties of artifacts (Sec 3.3), for encodable spatial characteristics such as local and global density, heterogeneity of morphology, boundary conditions, clustering, and residual occupiable space. Some of these encodable objectives require extensions to the representations described above. These extensions are addressed below.

Human judgment. In the *flora robotica* context, individual users in the Internet-connected social garden will have personal preferences in aesthetics and other considerations. These personal preferences will be incorporated through interactive evolution and inputs from physical

interaction, in future deliverable *D3.3 Internet-connected social garden*. The interactive evolution portion of the social garden relies on visualizations of projected growth that are visually realistic enough to be understandable by an untrained user. Realistic geometric modeling of braids (Sec. 4.3) is a crucial part of achieving this visual legibility. Beyond personal preference (and before getting to the user-interaction stage in the project), systematic considerations of architectural spatial potential that are not easily encoded can be evaluated by a trained architect. This evaluation relies on visual analyses of propositions, modeled at fine-scale detail, and is conducted by comparing and contrasting the modeled proposition with known existing examples (according to e.g., relevant scale; use case; programmatic typology; tectonic organization and style; relationship to context; type and quantity of users). This type of visual assessment will help us to evaluate the propositions in *D3.2 Architectural propositions*, as we fine-tune the objectives for steering artificial and biological growth. In architectural design, the importance of various design criteria is subject to human judgment, making it difficult to represent the task in a fitness function when using artificial evolution. Assessing encoded performance objectives (e.g., environmental, structural) in a way that allows for human judgment of the trained architect is addressed below.

5.1 Evaluating structural and environmental performance and spatial potential

Here we describe extensions to the existing representations that will enable evaluation of structural and environmental performance and spatial potential in the artifact propositions presented in the future deliverable *D3.2 Architectural propositions*.

Structural performance Multiple investigations are being conducted for assessing the structural performance of geometrically modeled braided artifacts. In the macroscale and microscale relaxation of braid modeling (Sec. 4.3.5 and 4.3.6), the physical geometry of of a braid organization is approximated according to strip width and weave density. This method assumes that the structural stiffness of the strips is sufficient to keep the physical braid in this approximate shape. However, as discussed in the construction logics portion of the *1st Periodic Technical Report* and *D3.2 Architectural propositions*, we investigate varying stiffnesses of filaments, as well as actuation by deformation. Depending on the relationship of strip stiffness, self-weight, and overall geometry, the strips in the braid may incrementally settle over time, resulting in changes in geometry (see Fig. 46). Designing material stiffness to prevent or predict this effect is an important task. Furthermore, we seek to purposefully deform the braids as a method of robotic actuation, making an understanding of the braids' stiffness properties essential. So far, investigations to this end include 1) the setup for structural testing of tubular braids (described here) and 2) the setup for structural testing of braids stiffened with interwoven plants (see *1st Periodic Technical Report* for preliminary test with plants, and *D2.2 Report on the final algorithms and plant-affection of bio-hybrid organism* for structural testing with plants). The setup for structural testing of tubular braids is intended to lead to a precise model of mechanical deformation. A double-layer braid (with polymer strips and fiberglass rods) is mounted with fixed connections perpendicularly to a plate. To deform the braid, a central actuation string is affixed to a Mecmesin universal computer-controlled force tester, which pulls the string away from the braid as the grip-head moves upward. Red tracking dots are placed on the side of the physical braid, so that their positions can be recorded into a data set (in addition to the load applied). With an extensive set of tests in this setup, data can be gathered to lead to a precise model of braid deformation according to load. In the setup test (see Fig. 47), the braid is fully erect in its starting position, with the grip-head in the lowest position (Fig. 47, position 1). The braid is bent nearly perpendicular to the starting position in its fully deformed position, with

Figure 46: Large ≈ 2 m braids with strips that have incrementally settled over time, causing the braid to deform into a bent position, though it was originally erect when fabricated. Left, deformed braids after settling, approximately six months after fabrication. Right, fully erect self-structured braids immediately after fabrication.

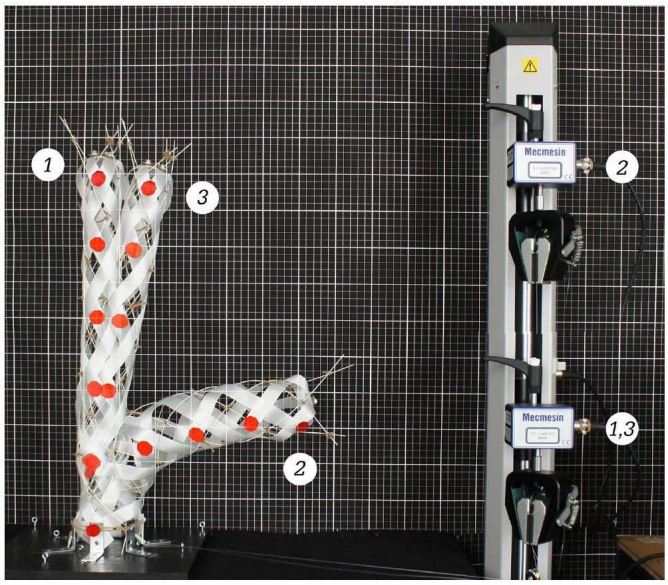


Figure 47: Double-layer braid (polymer strips and fiberglass rods) with red tracking dots. Braid is deformed with an actuation string, attached to a computer-controlled force tester. Braid position 1 and 3 both correspond to force tester position 1,3. Braid position 2 corresponds to force tester position 2.

the grip-head in the highest position (Fig. 47, position 2). When the grip-head is returned to the lowest position (equal to the starting position), the braid does not fully return to its original starting position, instead remaining slightly deformed (Fig. 47, position 3). This is an example of a deformation scenario that would be included in a precise mechanical model of the braid. Further work on investigating the material stiffness and structural performance of braid is in the scope of future deliverable *D3.2 Architectural propositions*.

Environmental performance The formation space representation needs to be extended to test more aspects of environmental performance (beyond shading occupied areas), and needs to be tested in combination with the integrated projections (Sec. 2) so that plant growth might be considered. Furthermore, the usefulness of the formation space needs to be tested for long-term growth and evaluation. The various datasets collected to record artificial and biological growth (described below in Sec. 6) should be able to be used to improve the robotic controllers over time. This extension can be used to ensure the success of the other representations.

Spatial potential As discussed above, the evaluation of spatial potential is supported by the visually realistic geometric modeling of braided artifacts. The geometric modeling above is quite good for this purpose. The extensions that are needed for accurate structural analysis will also be useful for visual assessment of spatial potential, specifically in circumstances where the artifact's geometry changes significantly (e.g., actuation by deformation). Also, the representation of aggregate properties (Sec. 3.3) needs to be extended for aspects of spatial potential that are deemed useful for the propositions of use in *D3.2 Architectural propositions*.

5.2 Visualizing multi-objective design spaces for incorporating designer judgment

The task of assessing performance according to architectural objectives with subjective priorities is supported by the designer's visual analysis and understanding of conflicting and contrasting objectives. The approach below follows our paper [19].

In multi-objective optimization, solution sets with more than three dimensions cannot readily be understood visually, and the problem of visual representation becomes increasingly difficult with increased dimensionality. However, in using the representation method of gradient interspersion, described below, combining the six example gradients remains highly legible. In this method, areas of density (where there is the most overlap of objectives) are simple to visually identify.

The method of gradient interspersion can be applied both to a multi-dimensional parameter space and to the surface topology of 3D models of individual solutions. The method of mapping gradient interspersions onto surface topologies can be applied to our 3D models of braided artifacts.

5.2.1. Background of early phase architectural design using evolutionary algorithms

As described in [19]: as architectural projects have become more complex and begun to engage more numerous complex systems, the computational task of assessing a design's performance according to many disparate objectives has become more prominent. Computational design teams frequently search for performative solutions with optimization algorithms or metaheuristic approaches such as evolutionary algorithms. The prerequisites for utilizing these methods, while well-suited to engineering, are less perfect for the full scope of architectural design tasks. They require comprehensive numeric formalization of objectives and their relative priorities [14].

Predefined fitness functions may be a poor companion to exploring design nuance, discovering commonalities between encodable objectives and those requiring human judgement, and leveraging designer understanding of emergent relationships between various real-world systems. Therefore, rather than seek a more perfect method of automated optimization, we propose visualizing multi-objective design spaces (as a complementary tool to automated search methods) for situations when a designer does not already know the correct fitness function (or even the final task), and needs to explore the commonalities between objectives in order to design appropriate fitness functions. Our approach intends to enhance understanding of objectives and fitness in the process of determining priorities in architectural design. This requires that improvements of visualizations, as tools from which designers can make reliable inference, become a priority.

It is well established that solution sets with more than three dimensions cannot readily be understood visually, and that the problem of visual representation becomes increasingly difficult with increased dimensionality [12]. Therefore, we intentionally construct a simple case study with more than three dimensions in the objectives and in the parameter set.

5.2.2. Approach for the problem of multidimensional visualization

As described in [19]: as pointed out by Coello et al [12], it is readily accepted that legible visualization of optimization results is typically limited to three dimensions. It is with this problem that the key underlying algorithm of our approach deals. One way to visualize three dimensions is with xyz plots. We assume here that plots will be used for the parameter space, leaving color as the visualization medium for objectives. Gradients are commonly used in optimization visualizations, to visualize a single set of values (Fig. 48). Our underlying algorithm therefore deals with visualizing more than one set of values, by combining multiple gradients in a single parameter space. In the examples of in this section, the combination is first shown with full gradients (distributed across the value set). It is then also shown with a threshold at the median, creating a binary caricaturization.

First, we look at the problem of gradient combination with two gradients (representing value sets) (Fig. 49(a)). Both the method of averaging RGB and of assigning each value set to an individual RGB channel are visually clear.

Assigning value sets to RGB channels gives visually clear results not just with two sets, but with three (Fig. 49(b)). It is an approach often used, and is for example used by the Grasshopper add-on Octopus. However, its limit is reached at three value sets, when the R, G, and B channels are all occupied.

Next, we look at the strategy of averaging the RGB values of multiple gradients representing value sets. Averaging two gradients (Fig. 49(a)) was visually clear. Averaging three gradients (Fig. 49(b)) is still mostly visually clear, but as we continue to add more gradients, the legibility declines significantly. Averaging six gradients becomes quite illegible when using binary colorization, and completely illegible when using full gradients (Fig. 50).

By contrast, when we use our algorithm for gradient interspersion (see below) combining six gradients remains quite legible using full gradients, and highly legible using binary colorizations (Fig. 51).

When using our strategy of combining many gradients representing value sets for objectives, on one parameter space, one can visually identify areas of density. These areas of density occur when there is the most overlap of gradients, and therefore the most commonality between many objectives on specific individual solutions in the parameter space (Fig. 52). In other words, the simple task of visually identifying density allows the viewer to intuitively identify optimal solutions.

These results acquired through the user's visual judgement and inference can be confirmed

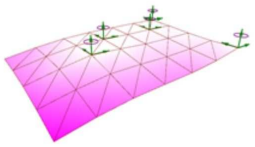
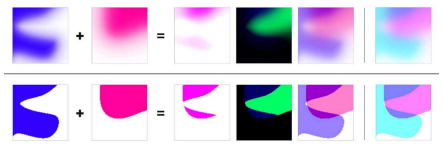
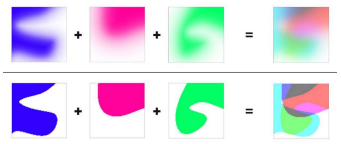


Figure 48: KARA MBA3D. An example of one value set visualized with gradient.



(a) From left to right, first gradient, second gradient, adding RGB, subtracting RGB, averaging RGB, assigning value sets to individual RGB channels.



(b) Assigning three value sets to the R, G, and B channels.

Figure 49: Combining RGB channels.

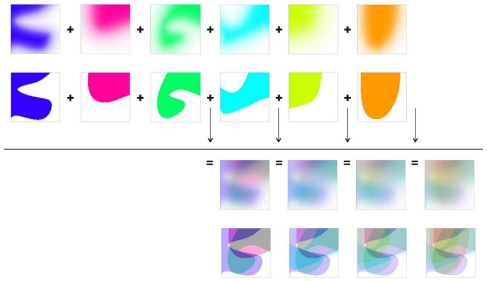


Figure 50: Averaging the RGB values of up to six gradients.

algorithmically, by identifying the isoline curves for each gradient and calculating the intersection areas of those curves (Fig. 53).

Setup of case study and baseline for comparison The case study scenario we construct to detail the implications of this strategy involves a six-dimensional parameter set. The parameters inform the coordinates of a grid of control points defining a NURBS surface, representing a gridshell. The points are distributed into six groups (Fig. 54), and the z-coordinate of each point group becomes a parameter. Because these are NURBS control points, each parameter's individual impact on surface topology can only be determined relative to the rest of the parameters. The z-value of the defined surface, at a given xy position, depends not only on the z-value of that respective control point, but on the other control points in the set.

Each parameter's variables are continuous and periodic, as is typical in an optimization setup. Specifically, each of the six groups of control points can have a z-value of 0, 2, 4, 6, 8, or 10. However, the possible parameter-based NURBS surfaces are time-dynamic morphologies, not static objects, undergoing constant geometric change by continuously updating to surrounding environmental conditions. In this case study, the geometric scaling of the z-dimension of the

Figure 51: Left to right, averaging RGB, our approach of interspersing full gradients, our approach of interspersing binary colorizations.

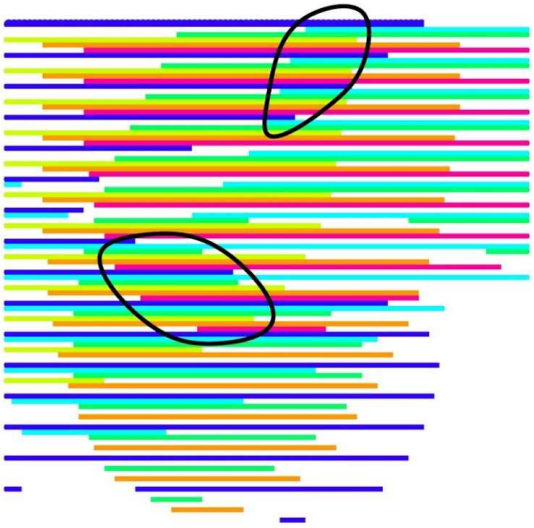


Figure 52: Areas of density in the interspersed visualization, indicating commonalities between objectives, and therefore optimal solutions.

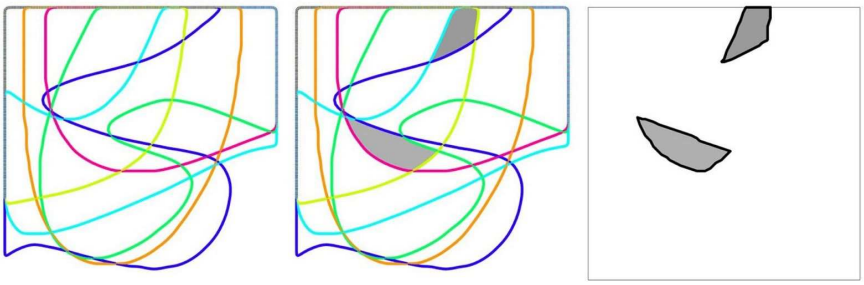


Figure 53: Left to right, isolines, isolines with intersections, intersections.

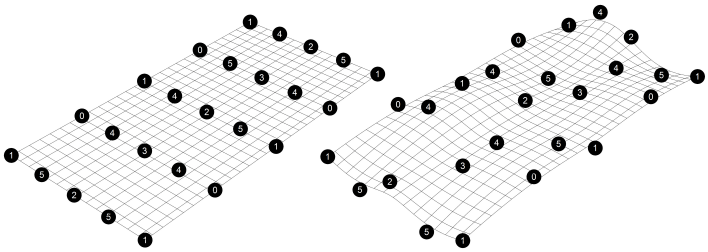


Figure 54: The six groups of NURBS control points (numbered 0-5) in two positions, with the NURBS surfaces defined.

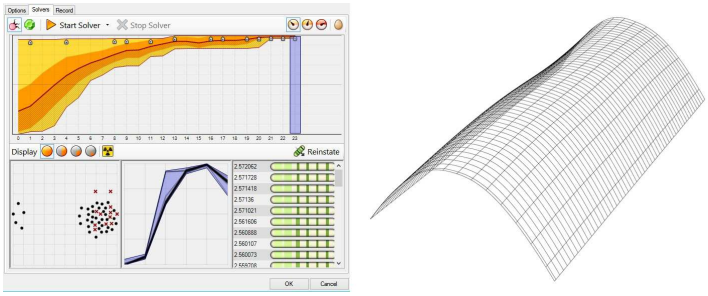


Figure 55: Left to right, Galapagos interface in this case study, Galapagos case study output topology.

defined surface is tied to the real-time solar conditions of the site. For instance, if parameters A, B, and C were equal to 2, and parameters D, E, and F were equal to 8, and these parameters did not change, the resultant NURBS surface would still have a different morphology in morning solar conditions than in it would in afternoon. The following four objectives are applied to the case study: acoustic scattering (black), rain water drainage (blue), wind sheltering (purple), and solar exposure (red). These objectives are evaluated according to respective geometric abstractions. They are chosen because they do not inherently have geometric commonality. From now, they will be referred by their assigned hues. We first run the scenario through the Evolutionary Solver in Grasshopper's Galapagos in Rhino3D. The native output of any single run of Galapagos is a single design solution that has been evolved according to a single fitness function. When optimizing for multiple objectives in Galapagos, the multiple fitness functions must be numerically combined. Weighted combination into a single objective is arguably the most common strategy of optimizing for multiple objectives [14]. Our approach could be used in combination with this tool, by exploring and designing fitness functions that can then be used in Galapagos, or another solver. For purposes of comparison, the objectives are given equal weights to be run in Galapagos, and the resulting solution is a topology similar to a barrel vault (Fig. 55).

We then investigate the same scenario with our approach, with the purpose of developing effective visual semantics to facilitate a user's deductive evaluation of dynamic design solutions. The possibility for such an interface is achieved by precalculating a comprehensive database of performance evaluations and then continually referencing that database as the user interacts with the interface. The precalculation of the reference database in this case study, in this early non-optimized prototype, took less than two hours of runtime. This suggests that an optimized version could be similar to the runtime of a Galapagos optimization (for this case

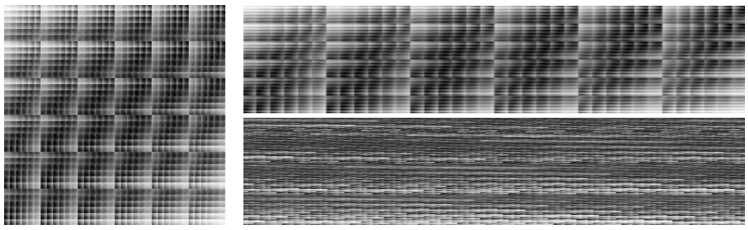


Figure 56: Left to right, squarified treemap, possible hierarchical orders

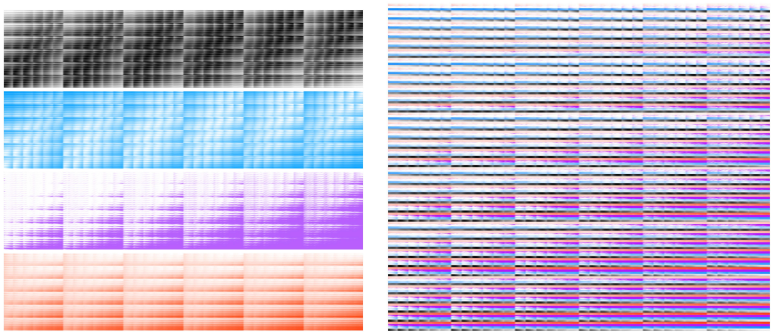


Figure 57: Left to right, extended squarified treemap for side-by-side comparison, and interspersion of that extended treemap.

study specifically, which contains 46,656 solutions; see [19] for scaling of the approach). This aspires to embodied interaction to "help support the process of improvised, situated action by making the immediate circumstances of the work more visible" [17], applied to visualization of multidimensionality. In addition, this approach allows representation and engagement with both the full scope of possible solutions, and the full scope of environmentally-dependent timesteps that can occur in each solution, in stark contrast to the typical architectural design optimization strategy of summarizing.

Squarified treemapping and interspersion in the case study The six-dimensional parameter space is visualized as a nested set model, through the visualization strategy of squarified treemapping [8]. The evaluation criteria, visualized as an alpha-channel gradient on the objective's assigned hue, is overlaid onto the structure of the squarified treemap (Fig. 56, left). Through the interface, the user can browse hierarchical orders for parameters, allowing the user to choose the hierarchy that makes trends in the evaluation criteria as visually legible as possible in that bespoke scenario (Fig. 56, right).

The four-dimensional objective space can be visualized in the interface both as an extension of the squarified treemap enabling side-by-side comparison (Fig. 57, left), and as an interspersed gradient map (Fig. 57, right), loosely inspired by genotype microarray visualization strategies [24]. This enables the user to make evaluations not just about the performance of design solutions, but also about the appropriateness of the user-specified objectives to the design problem at hand, a fundamental issue of multi-objective optimization [14]. For instance, a user can easily ascertain from the side-by-side comparison that the black objective and the blue are in direct competition with each other, and can never both be fully optimized in a single design solution. The user might, in that instance, manage one of these competing objectives with a different strategy in

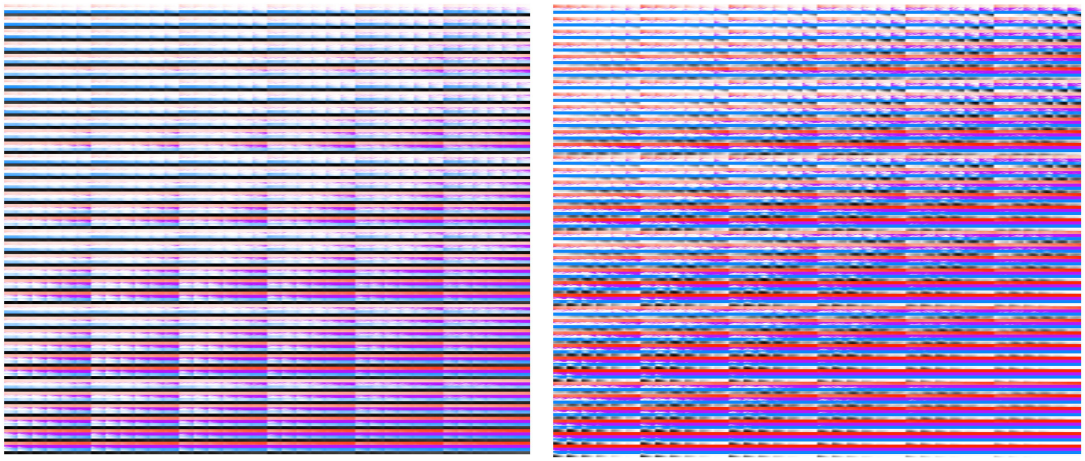


Figure 58: Left to right, treemap with thresholds lenient for the black objective, treemap with thresholds that prioritize black objective.

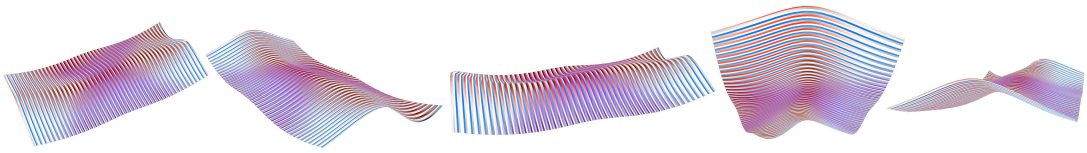


Figure 59: A single topologically mapped evaluation, from multiple viewing directions.

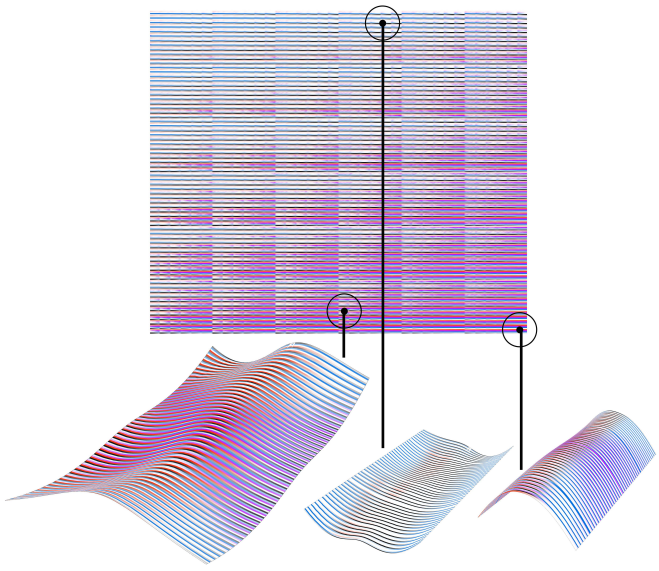


Figure 60: Solutions' treemap locations and topologically mapped evaluations.

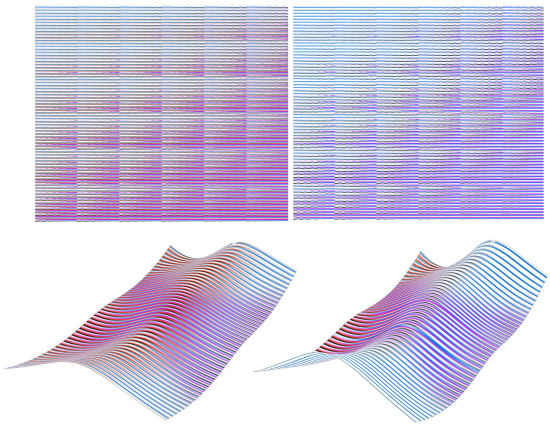


Figure 61: Evaluation of time-based deformation in a scenario with dynamic architecture components. Top, on the squarified treemap; bottom, topologically mapped.

order to achieve them both. Also, a user could ascertain that the distributions of the purple objective and the red are similar, and could accordingly make more informed decisions about the design. Both of these insights, though almost trivial when viewing a visualization of the entire evaluation set, could not be derived from the native output of the Galapagos evolutionary solver in any design problem that has many diverse local optima.

Dynamic fitness thresholds. In this interface, the user has the option to change the weights of the objectives with real-time updates to the squarified treemaps and interspersed treemaps, through the manipulation of evaluation thresholds (Fig. 58). This allows the user to interrogate their design of objective fitness, thresholds, and weights, a task that can be very problematic in typical evolutionary solver setups. For instance, the treemap in Fig. 57, right (equal weights for all objectives), shows that under the current thresholds, no solution can be fit for all objectives, and the most advantageous solutions are those in the bottom right (where there is low black, but high blue, purple, and red). After viewing this, a user could decide to make a trade-off [14] to be more lenient on the black objective and change the threshold to be more inclusive (Fig. 58, left), making solutions in the bottom right corner fit for all objectives. Conversely, a user may look at the Fig. 58 (left) treemap and realize that the initial objective weights is a poor reflection of the design problem at hand, and needs to be modified substantially. If the user were to increase the importance of the black objective, and decrease the importance of all others (Fig. 58, right), those in the center of the map would surpass those in the bottom right and become the most fit.

If a user wished to modify the weights and thresholds using evolutionary optimization in Galapagos or a similar tool, the solver would have to be re-run for each change in objective weighting, and it would be difficult and counterintuitive to begin to understand the trends present in the dependencies of weights, fitness, and solutions.

Mapping local evaluations onto solution topologies. The interface allows the user to select a single solution from the treemap (either by defining the parameter variables or by defining the position on the treemap), and to simultaneously view the local evaluations of that solution's geometric topology, according to each objective (Fig. 60).

This visualization, like the treemap, allows the user to toggle between side-by-side comparison and gradient interspersion. As solution browsing has real-time feedback both on the treemap and

the topology, the user can internalize an intuitive understanding of which topological features spur which fitness results. This could empower the user to make a non-optimal selection from the solution set, with the knowledge that a certain objective could be achieved by making other changes to discrete problematic spots.

The user can also run time scenarios, with assessment interfaces similar to that of simulation (see Fig. 61). The user can simultaneously view the treemap and an individual solution topology as they undergo their time-based changes and continuous re-evaluation.

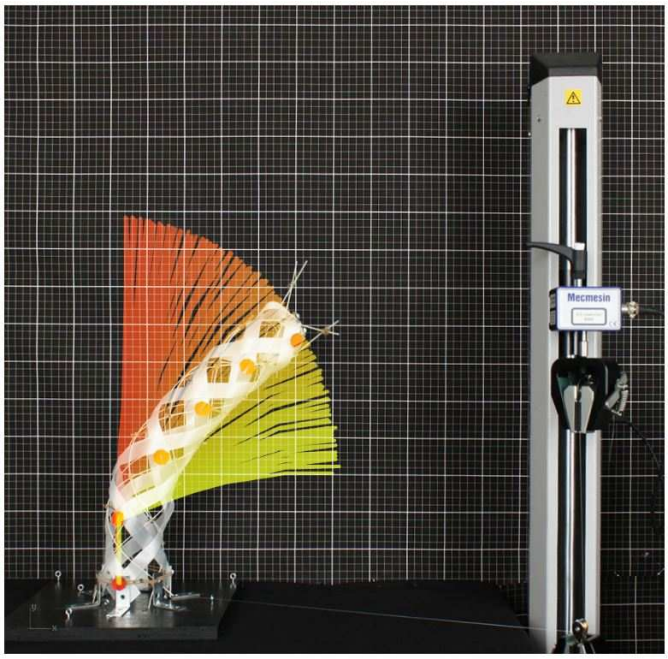


Figure 62: Positions of tracking dots on the side of a braid are used to record a polyline representation of the braid's shape.

6 Detecting and recording histories of growth for long-term evaluation

As the *flora robotica* system grows, the progression of the artifacts must be evaluated and compared to expected performance. For this purpose, the various methods described below are useful for recording histories of artificial growth, biological growth, and the environmental conditions impacting growth. Further methods later developed in other aspects of the project will also be useful for this purpose. The data sets described here, and any others later developed, with be tested for usefulness in evaluations of propositions of use in *D3.2 Architectural propositions*.

Below, there are both distributed and centralized methods for recording histories of growth. Eventually, only the distributed sources will be used to impact controllers. However, the centralized methods are very useful for archiving the results of our experiments, and comparing them to our expected results.

6.1 Artificial growth

Markers placed on mechanical scaffolds can be used to easily track their deformation via image sampling. This is a centralized method. For distributed sensing of deformation, we can sense the deformation indirectly by sensing the changes those deformations cause in their environment.

6.1.1. Mechanical scaffold detection through image processing

For setting up a method of stress tests to lead to a precise model of mechanical deformation, a double-layer braid made with polymer strips and fiberglass rods was tested with a load test machine. In order to account for the braid's deformation during the load test, a point tracking

process through image sampling was used (see Fig. 62). For this, a pre-process of low resolution image sequence contrasting red markers placed in the braided structure was performed. Then, a process of image sampling converts pixels into a colored mesh. After that, a script searches for points with a specific value, red in this case, within a certain tolerance, clusters them by proximity, and averages them into one point per cluster, which is the digital representation matching the physical marker. Points are then sorted and ordered to match the topology of the braid skeleton by searching from the first xy dot then finding and iteratively connecting the next closest dot to form an linear abstraction of the braid at its bending points.

6.1.2. Photoresistors for detecting scaffold morphology change

Aiming for distributed decision-making, embedded sensors are included in braid filaments. The preliminary embedded-sensor filament is described in *D1.2 Evaluation of mechatronics prototype of the robotic symbiont including supporting software*. It contains eight analog light sensors connected to a Raspberry Pi 3 (model B) at one end. Having such a setup located discretely within a braid filament can provide us with information regarding braid morphology and deformation, as described in our paper (add citation). For instance, as a braid arm bends, it can self-shade new portions, and expose others to light. This would be reflected in the state of the light sensors embedded across the braid artifact's surface. With an appropriate interpretation method for these changes in sensed conditions, the histories recorded could distributedly provide information about artificial growth and morphology.

6.2 Natural growth

Plant tips and stems can be tracked via image sampling, and 3d point cloud descriptions of entire plants and plant groups can be obtained via laser scanning. These are both centralized methods. The data from image sampling is interpreted into a useful description and model of plant growth, and can therefore be used for simulating growth, evolving robotic controllers, algorithmic comparison to expected results, and other tasks. The point cloud data from laser scanning is primarily meant for recording geometric descriptions that are detailed enough for visually analyzing the spatial potential of grown artifacts. If we in the future aim to use it for models and simulations of growth, a method of interpretation will need to be developed. For distributed tracking of plants, we choose IR-proximity sensing for the moment. This sensing gives far less detail than the centralized methods, so will primarily be used as inputs for robotic controllers, and its histories will be referred to in that context during evaluation of artifacts. The centralized methods will be needed for finer detailed evaluation of the bio-hybrid architectural artifacts.

6.2.1. Plant detection through image processing

For the aim of evolving controllers to steer plant growth and motion, experiment photos of the plants are sampled to obtain a model of the plant tip and of the full plant stem (as described in *D2.2 Report on the final algorithms and plant-affection of bio-hybrid organism*). Rather than simply recording the presence of plants, the sampling is interpreted into a smoothed multi-point description (see Fig. 63), allowing us access to analysis attributes like curve radius (or apex point of curve, inflection point, contraflexure point, etc) when evaluating histories of growth and deriving fitness functions for evolution. This method will also be useful for tracking results of grown plants in future experiments, to compare to the expected results. The histories of growth recorded in this way can be used to develop, evaluate and improve the representation of plant growth in integrated projections. Because the sampling method is generalized to any background

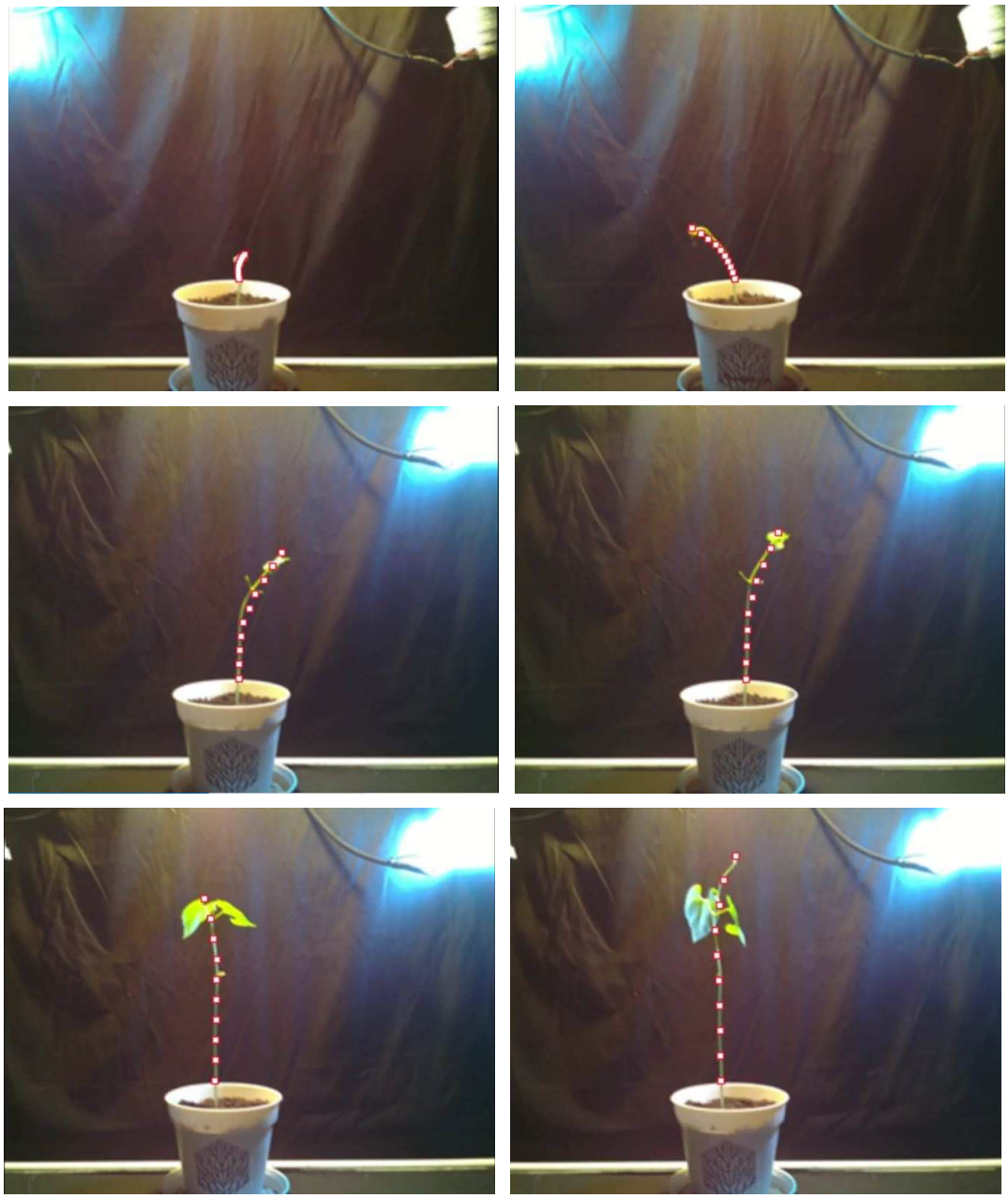


Figure 63: Example time steps of plant stem tracking.

by comparing to a baseline of that background, we are not limited to using this method only in the experiment setup tested so far. It could be extended to other experiment setups.

6.2.2. Sensor types for distributed plant detection

As described in *D2.2 Report on the final algorithms and plant-affection of bio-hybrid organism*, three types of sensors were tested for distributed sensing of plant proximity: capacitive touch sensing, ultrasonic sensors, and IR-proximity sensors.

In an early phase test of capacitive touch sensors, the plants were correctly detected only about 50% of the time. As this rate of success is similar to random guessing, we did not pursue this method further. The ultrasonic sensors worked most successfully when facing a flat object perpendicularly, maximizing reflections. As plant tips and stems are often small and oddly shaped, this condition is not ideal. The IR-proximity sensors achieved high accuracy when the direction of the approaching plant was predetermined. Therefore, the IR-proximity sensor is a good match for the experiment conditions of the Plant Binary Decision Wall (PBDW, see *D1.2* and *D2.2*). The IR-proximity sensors can be used to detect changes in reflections. While they are useful for detecting a newly approaching plant, they may not be as useful for detecting quantity of approaching plants, or for detecting a new plant approaching from the same direction as an existing plant. If we use the IR-proximity sensors for more general detection of plants (rather than from only one specific scaffold direction), it will be important to develop a way to interpret the data of many embedded sensors with overlapping fields of view.

6.2.3. Plant growth recorded through laser scanning: hardware and setup

High-resolution 3D models are needed for analysis according to high-level architectural objectives. For the purpose of comparing (and verifying) actual growth with projected growth in the final instance of a *flora robotica* system, a method is developed for obtaining a high-resolution record of plant, robot, or scaffold growth stages. In testing the method, two plant setups were recorded by laser scanning.

Two independent sites were selected to monitor plant foliation and growth during spring and early summer 2016.

1. Site 1: Bonsai. A small scale isolated bonsai tree was chosen as this has a short growth period and could be recorded in a controlled environment (see Fig. 64).

2. Site 2: Poplar. A larger scale group of trees were also recorded in order to test the setup in a larger scale real environment (see Fig. 64).

The sites were recorded at regular time-intervals using a FARO Focus3D laser scanner[19] (see Fig. 65 to produce 3D point-cloud records of growth. The scanner is a terrestrial scanner, which produces a spherical scan (see Fig. 66) capturing all objects within the line of site with an accuracy of ±2mm.

Each site is composed of several scans from the terrestrial scanner. These scans are registered together by using automated procedures in the Faro Scene software (a proprietary software from the producer of the 3D terrestrial scanner)[20] to produce a full point-cloud for the entire site, and the site is cleaned and cropped in the same software environment before being exported to an E57 file format[21]. By compositing successive point-clouds, a full 3D geometric record of growth has been produced for each site. In order to produce these records of growth over

[19] http://www.faro.com/en-us/products/3d-surveying/faro-focus3d/overview
[20] http://www.faro.com/en-us/products/faro-software/scene/overview
[21] http://www.libe57.org/

Figure 64: Left, the bonsai site with one isolated bonsai tree and target spheres for registration. Right, the poplar site with 8 poplar trees.

Figure 65: Faro Focus3D terrestrial scanner.

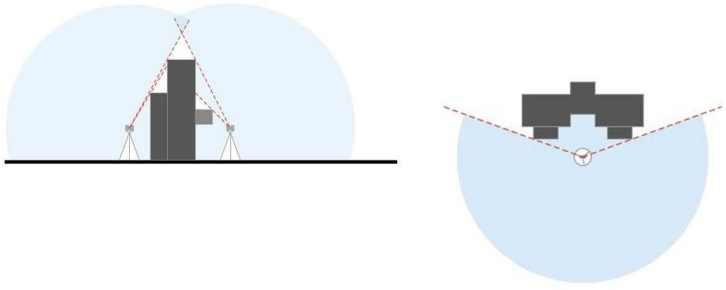


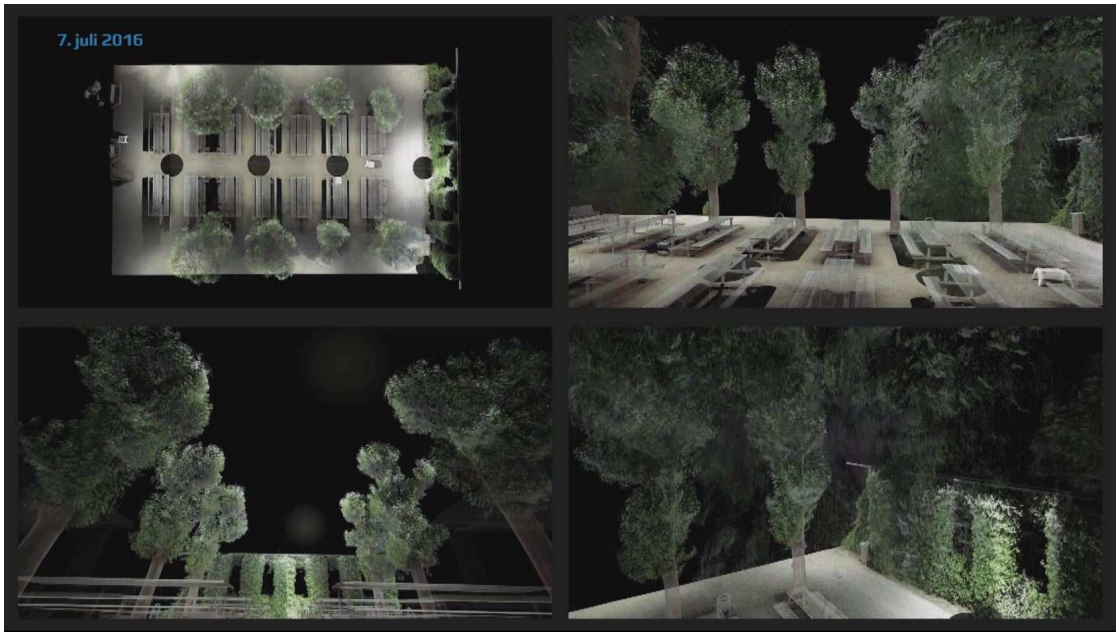
Figure 66: Diagram showing 3D scanning concept with almost spherical terrestrial scanning. (Diagram by LE34)

(a) The bonsai point-cloud at time-step 01.

(b) The bonsai point-cloud at time-step 18.

Figure 67: Two frames from the bonsai site video displaying the growth state at time-step 01 and 18.

(a) The poplar point-cloud at time-step 01.



(b) The bonsai point-cloud at time-step 25.

Figure 68: Two frames from the poplar site video displaying the growth state at time-step 01 and 25.

time Rhino/Grasshopper3D and the point-cloud editing plug-in Volvox[22] for Grasshopper3D has been used. With this setup it proved possible to construct a "4D" registration of growth for each site. The datasets could then be rendered to produce visualizations of recorded growth from any chosen location within the digital model (as shown in the two frames of Fig. 68, with each frame split into 4 views). And with a later post-production in Adobe Premiere Pro[23] videos have been produced for each site, to visualize their respective growth periods. Beyond visualization, the dataset provides a 3D model of growth time-steps for two species at two different scales (Bonsai and Poplar).

6.2.4. Laser scanning: single plant

The bonsai site consists of one isolated bonsai tree. For this site each successive point-cloud consists of 6 scans around the bonsai tree with a total of approximately 6 million measured points. Small spheres situated next to the bonsai tree was used as targets for later registration of the 6 scans in Faro Scene (see Fig. 64). Over a period of 11 days (04-05-2016 - 15-05-2016) a total of 18 point-cloud has been produced. The growth recorded can be seen in the video[24] produced from these point-clouds (or see Fig. 67).

6.2.5. Laser scanning: group of plants

The poplar site consists of 8 Poplar trees and each successive point-cloud consists of 10 scans registered together using natural feature registration (top view and iterative closest point) in Faro Scene. A point-cloud of the site per time step consists of approximately 130 million measured points. The site has been recorded over a time period of 3 months (11-04-2016 - 11-07-2016) with a total of 25 point-clouds. The growth recorded can be seen in the video[25] (or see Fig. 68).

6.3 Environmental conditions affecting growth

Environmental conditions affecting biological growth are sensed with the phytosensors described in *D1.2 Evaluation of mechatronics prototype of the robotic symbiont including supporting software*, the Global Environment Monitor described in *D1.2*, and other sensors. The data from these sensors is useful for the task of evaluating the biological growth and comparing to expected results. Environmental conditions that are sensed for the purpose of steering artificial growth through robotic controllers is important information for evaluating the results of artificial growth and morphology changes.

Photoresistors for guiding scaffold morphology change As described in *D1.2*, three photoresistors are used as inputs to promote actuation of a Glass Fiber Reinforced Polymer (GFRP) rod braided structure tower robot. Their ability to detect light variation in the environment

[22] http://www.food4rhino.com/app/volvox
[23] http://www.adobe.com/products/premiere.html
[24] Video of Bonsai point cloud: https://youtu.be/SocePcSHVEI
[25] Video of point clouds for group of Poplar trees: https://youtu.be/p009IM-UB5M

help the embodied evolution process to analyze the environmental light conditions and actuate the motors that the robot uses to point its tip towards the prominent light source or avoid it. In a setup with a group of such robots and changing environmental conditions, it will be useful to use the history of sensed data to evaluate the actions of the various robot controllers, and the performance of the architectural artifacts shaped by those actions.

6.4 Human occupancy in response to growth

Patterns of human occupancy have crucial impact on architectural use cases, typologies, organization, and other issues of spatial potential. Therefore, sensed human occupancy is useful both as an input to robotic controller evolution and as a data set from which to evaluate the success of architectural artifacts in terms of spatial potential. Much of this investigation falls under the social garden task in *D3.3 Internet-connected social garden*. Below is a description of the first probe study toward this aim, an installation completed at the November 2016 *flora robotica* review (see Fig. 69).

6.4.1. Probe tests into sensor types for occupancy detection

Ultrasonic sensors are equipped with an ultrasonic speaker, which emits a sound in a frequency of roughly 40 kHz (above audible sound), and a receptor capable of detecting this frequency. Distance measuring works analog to a sonar radar, where a sound pulse is emitted and distance is measured by the time sound takes to bounce from any physical surface back to the sensor. Ultrasonic sensors of the type HC-SR04 connected to a Raspberry Pi 3 controller where used to detect human presence. This sensor provides 20 to 4000mm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm, and a detecting angle within a range of 300mm.

6.4.2. Ultrasonic sensors in the Social Garden prototype

Ultrasonic sensors were tested in short to long range situations, multiple heights and orientations and calibrated to the required environment so that they would be able to detect the presence of human occupants within multiple obstacles. The information provided by simultaneous readings of multiple strategically place ultrasonic sensors will be able to produce an accurate understanding of the physical variation of an environment, including human motion within a determined space. Data gather from the sensors can also be remapped to generate a reliable measurement of the proximity of objects in conventional units at a fast rate. In the social garden prototype, Ultrasonic sensors collected data which manifested the amount of local presence of a human inhabitant, as well as an overall estimation of occupancy within the whole installation.

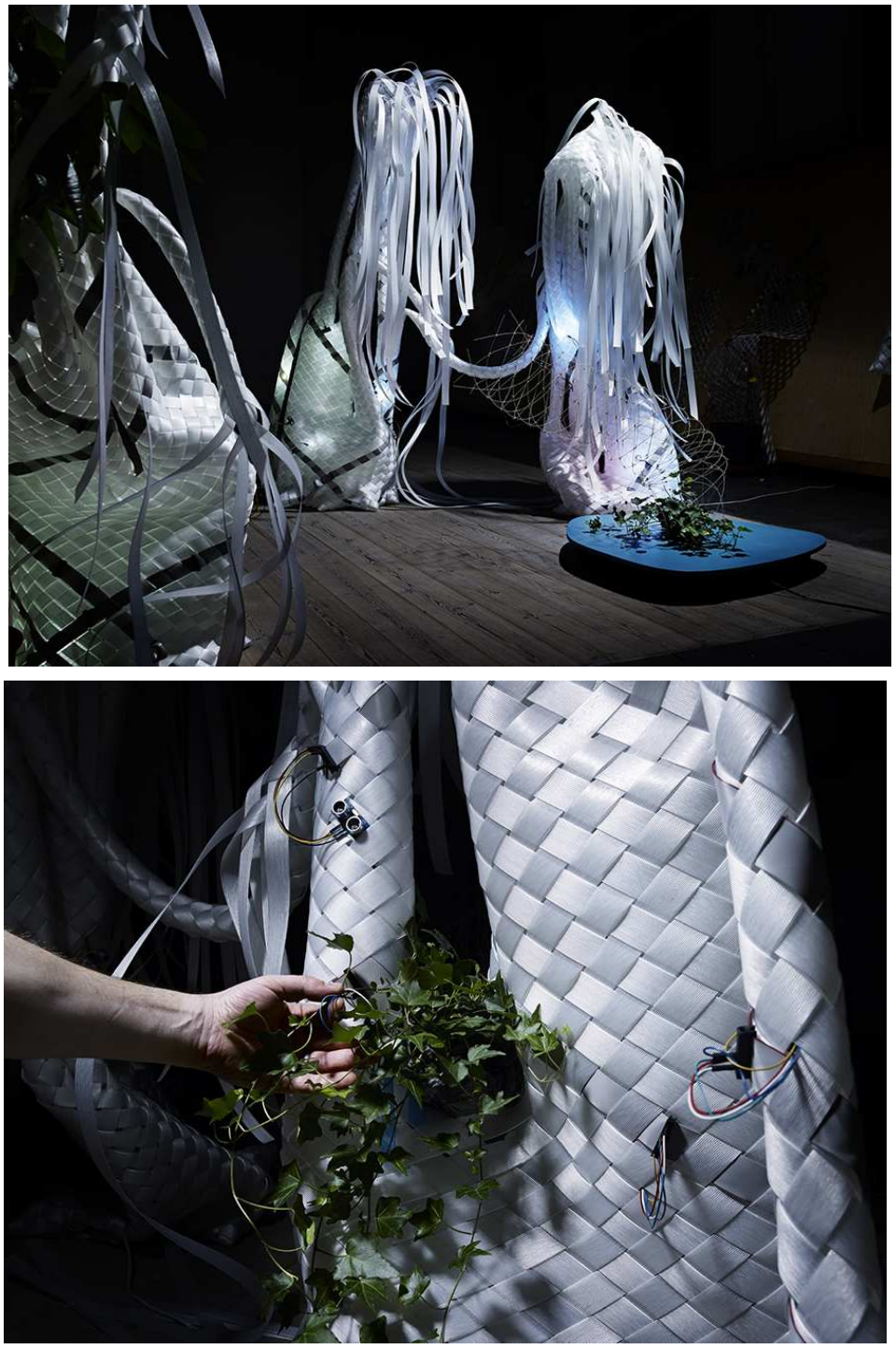


Figure 69: Social Garden probe installation at the November 2016 *flora robotica* review, incorporating ultrasonic sensors for tracking human occupancy.

7 Conclusion

In this report, we have presented various representational methods that meet needs of the *flora robotica* project, namely:

1. generation of steered artifact states from robotic controllers and biological growth;
2. realistic geometric modeling of braided artifacts to integrate with generation, visualization, and fabrication; and
3. short-term and long-term evaluation of natural and artificial growth in bio-hybrid artifacts in the context of environment, occupancy, and architectural propositions of use.

A key contribution to the architecture scientific discipline is our approach to designing for self-organizing robotic controllers. In typical architectural design, the shapes of artifacts must be entirely predetermined at a very fine-scale level of detail. The approach described here, of steering self-organizing artifacts to reach certain objectives without specifying every fine-scale detail of the final shape, is a significant departure.

Steered self-organizing controllers, as a method of generating bio-hybrid artifact states, can be used as an input to our generalized geometric modeling approach for braided structures. This allows the modeling of architectural propositions to be not only a representation of mechanical scaffolds, but to be integrated with robotic control, as braid is the overall organizational logic for both scaffold and embedded robotic elements. The link between geometric modeling and generation of instructions for robotic fabrication of braid (such as for the braid machine reported in *D1.2*) needs to be further developed in the context of construction logics and propositions of use (in the future *D3.2*). The integration of modeled propositions with models of biological growth needs to be developed, linking models such as evolved control of plant motion (reported in *D2.2*) to braided artifacts through the development of the Integrated Projection.

Proper methods of evaluation for such bio-hybrid architectural artifacts need to be developed in the context of propositions of use (in the future *D3.2*). The representations presented here are useful for evaluations of structural and environmental performance and spatial potential, and the long-term viability of such representations is supported by various methods for recording histories of natural and artificial growth. Extensions to the representations will be developed to further serve the tasks of evaluation, both in terms of fitness functions for evolution and visual analysis for the incorporation of human judgement.

Methods for realistic geometric modeling of braid and visual analysis for evaluation will also be further developed in the context of the Internet-connected social garden (in the future *D3.3*).

References

[1] Ergun Akleman, Jianer Chen, Qing Xing, and Jonathan L. Gross. Cyclic plain-weaving on polygonal mesh surfaces with graph rotation systems. *ACM Transactions on Graphics*, 28 (3):1, 2009. ISSN 07300301. doi: 10.1145/1531326.1531384.

[2] E. Artin. Theory of braids. *Annals of Mathematics*, 48(1):101–126, 1947. doi: 10.2307/1969218.

[3] Yaneer Bar-Yam. *Dynamics of complex systems*, volume 213. Addison-Wesley Reading, MA, 1997.

[4] Yaneer Bar-Yam and Hiroki Sayama. Formalizing the gene centered view of evolution. In *Unifying Themes in Complex Systems*, pages 215–222. Springer, 2006.

[5] Philip Beesley and Subtle Technologies Symposium. *Responsive architectures: subtle technologies 2006;[proceedings of a conference held in Toronto, Ont., June 1st 2006]*. Riverside Architectural Press, 2006.

[6] Joan S. Birman and Tara E. Brendle. Braids: A survey. *arXiv preprint arXiv:math/0409205*, 2005.

[7] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Transactions on Graphics (ToG)*, 21(3):594–603, 2002.

[8] Mark Bruls, Kees Huizing, and Jarke J Van Wijk. Squarified treemaps. In *Data Visualization 2000*, pages 33–42. Springer, 2000.

[9] Anatoliy G. Butkovskiy. *Phase portraits of control dynamical systems*, volume 63. Springer Science & Business Media, 2012.

[10] EJ Caramana, DE Burton, MJ Shashkov, and PP Whalen. The construction of compatible hydrodynamics algorithms utilizing conservation of total energy. *Journal of Computational Physics*, 146(1):227–262, 1998.

[11] Matthew TK Chan, Rob Gorbet, Philip Beesley, and Dana Kulič. Curiosity-based learning algorithm for distributed interactive sculptural systems. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3435–3441. IEEE, 2015.

[12] Carlos A Coello Coello, Gary B Lamont, David A Van Veldhuizen, et al. *Evolutionary algorithms for solving multi-objective problems*, volume 5. Springer, 2007.

[13] Peter Cook and Colin Fournier. *A friendly alien: ein Kunsthaus für Graz*. Hatje Cantz Pub, 2004.

[14] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. Wiley, 2005.

[15] Yancy Diaz-Mercado and Magnus Egerstedt. Multi-robot mixing using braids. In *52nd IEEE Conference on Decision and Control*, pages 2001–2005, Dec 2013. doi: 10.1109/CDC.2013.6760175.

[16] Ulf Dieckmann and Richard Law. *The geometry of ecological interactions: simplifying spatial complexity*. Cambridge University Press, 2000.

[17] Paul Dourish. *Where the action is: the foundations of embodied interaction*. MIT press, 2004.

[18] John Frazer. *An evolutionary architecture*. Architectural Association Publications, 1995.

[19] Mary Katherine Heinrich and Phil Ayres. For time-continuous optimisation. *Living Systems and Micro-utopias: Towards Continuous Designing*, 2016.

[20] Mary Katherine Heinrich and Phil Ayres. Using the phase space to design complexity. *Complexity & Simplicity*, 2016.

[21] Mary Katherine Heinrich, Mostafa Wahby, Mohammad Divband Soorati, Daniel Nicolas Hofstadler, Payam Zahadat, Phil Ayres, Kasper Støy, and Heiko Hamann. Self-organized construction with continuous building material: Higher flexibility based on braided structures. In *Foundations and Applications of Self* Systems, IEEE International Workshops on*, pages 154–159. IEEE, 2016.

[22] Mike Hurley. Attractors: persistence, and density of their basins. *Transactions of the American Mathematical Society*, 269(1):247–271, 1982.

[23] Thymio II. Thymio robot - website, 2017. https://www.thymio.org/.

[24] Robert Kincaid, Amir Ben-Dor, and Zohar Yakhini. Exploratory visualization of array-based comparative genomic hybridization. *Information Visualization*, 4(3):176–190, 2005.

[25] Branko Kolarevic and Ali Malkawi. *Peformative Architecture*. Routledge, 2005.

[26] Manuel Kretzer and Ludger Hovestadt. *ALIVE: Advancements in adaptive architecture*, volume 8. Birkhäuser, 2014.

[27] Christian Mercat. Les entrelacs des enluminure celtes. *Dossier Pour La Science*, 15, 2001.

[28] George Polya and Ronald C. Read. *Combinatorial enumeration of groups, graphs, and chemical compounds*. Springer-Verlag, 1987. ISBN 0387964134.

[29] Xing Qing, Esquivel Gabriel, Collier Ryan, Tomaso Michael, and Akleman Ergun. Weaving Methods in Architectural Design. In *ACADIA 2011 Regional: Parametricism*, pages 59–66, 2011.

[30] Hiroki Sayama. *Introduction to the modeling and analysis of complex systems*. Open SUNY Textbooks, 2015.

[31] Matthias Teschner, Bruno Heidelberger, Matthias Müller, Danat Pomerantes, and Markus H Gross. Optimized spatial hashing for collision detection of deformable objects. In *Vmv*, volume 3, pages 47–54, 2003.

[32] M Thomsen and M Tamke. Narratives of making: thinking practice led research in architecture. In *Proceedings of the Conference Communicating (by) Design*, pages 2009–17, 2009.

[33] Hao Wang. Proving thorems by pattern recognition II. *Bell Systems Technical Journal*, 40: 1–41, 1961.

[34] Justin Werfel and Yaneer Bar-Yam. The evolution of reproductive restraint through social communication. *Proceedings of the National Academy of Sciences of the United States of America*, 101(30):11019–11024, 2004.

[35] Stephen Wolfram. Statistical mechanics of cellular automata. *Reviews of modern physics*, 55(3):601, 1983.

[36] Stephen Wolfram. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, 10(1-2):1–35, 1984.