# Horizon 2020


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

# Deliverable D2.3

## Report on the integration of algorithms on *flora robotica* and first tests

| | |
|---|---|
| **Date of preparation: 2018/03/31** | **Revision: 1 (r845)** |
| **Start date of project: 2015/04/01** | **Duration: 48 months** |
| **Project coordinator: UzL** | **Classification: public** |
| **Partners:** *lead:* **AMU** | *contribution:* **UzL, UNIGRAZ, CITA, ITU** |
| **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 D2.3 |
| Due date: | M36 |
| Delivery date: | 2018/03/31 |

| | |
|---|---|
| **Name:** | **Report on the integration of algorithms on *flora robotica* and first tests** |
| **Description:** | We report our development of algorithms and experimental approaches for bio-hybrid systems within *flora robotica*. We use different stimuli to influence plants in desired ways in terms of their motion, shape, and function and report the respective experiments. We apply the Vascular Morphogenesis Controller (VMC) to direct the growth of artificial structures. We use computer vision to construct a model of plant stem stiffening and motion dynamics by training an LSTM network. The LSTM network acts as a forward model predicting change in the plant, driving the evolution of neural network robot controllers. The evolved controllers augment the plants' natural light-finding and tissue-stiffening behaviors to avoid obstacles and grow desired shapes. As a benchmark task we choose obstacle avoidance. We also report the Integrated Growth Projection (IGP) to simulate the combined results of our models. The reported results are an important stepping stone of the project as they provide the basic methodology to develop bio-hybrids systems of natural plants and robots. |

| | |
|---|---|
| Partners owning: | **AMU** |
| Partners contributed: | **UzL, UNIGRAZ, CITA, ITU** |
| Made available to: | public |

# Contents

# 1 Introduction

The initial reporting period (M1-M18) was characterized by an amazing explorative journey. The main objective for this reporting period (M19-M36) was consolidation, focus on key concepts, and hard engineering work towards robust, reliable, and scalable solutions and hardware implementations. This is also reflected in this report as we have mastered the control of growth and motion of plants by light stimuli, we can grow steered artificial structures as scaffolds, and we have integrated concepts to project these combined approaches on architectural scales.

We report on algorithms for control of plant-robot bio-hybrids. Plant-robot hybrids consists of two different types of beings that require different means to steer and influence their growth. Especially the control of plants is challenging, because of their physiological complexity and intrinsic system complexity of living organisms. The plant's efforts to survive and reproduce, in turn, provide us with a base of processes that we can exploit and maybe even manipulate. Controllers for growth of plants and robots have been integrated on the levels of hardware (see D1.3) and software (in this deliverable). We use simulations of growth and motion that integrate different growth models to make predictions for our subsequent experimental verification. We have reported early assumptions for our controllers and have introduced our software approach in D2.2 (M24). Here we focus on recent developments of controllers, new methods for plant shaping, and the integration of our algorithms.

We report the essential implementation and the important engineering step of the Vascular Mophogenesis Controller (VMC) from simulation to an embodied system to showcase that we can grow and control tree-like and braided structures that may serve as scaffolds for our plants. We are able to form embodied tree structures in response to light conditions, that is a physical representation of the VMC and its effect. Moreover, RaspiNet is introduced, which embeds VMC in distributed hardware for bio-hybrid systems.

We have pushed the growth and motion control of plants by light to a reliable and robust approach. The controller enables us to guide a plant's growth to user defined targets and to grow a plant in desired patterns as showcased in an obstacle avoidance task. This obstacle avoidance task nicely showcases the challenge of controlling a plant's stiffening process by controlling its motion, anticipating future stiffening, and by integrating these effects into an appropriate holistic plant and tropism model.

While light serves as an attractor (exploiting phototropisms to steer a plant towards a target) we also require an additional stimulus that serves as a repellent (exploiting other tropisms to either steer a plant away from an area or to stop growth in an area). The use of vibrations motors was reported before in D1.3. Here, we explore additional options and focus on methods that will enable us to efficiently counteract attractor of blue light.

Having in mind a future potential utilization of nanotechnology for creating biohybrids, we also report progress in our molecular biology investigations.

The final chapter is dedicated to the Integrated Growth Projection (IGP), which is our approach to scale our methods to architectural dimensions. We combine the VMC and plant growth models, and allow for extensions by additional models. A number of different tasks are reported in this deliverable, including T2.2, T2.3, and T2.4.

# 2 Repellent

## 2.1 Introduction

Light was chosen as a main stimulus used to guide growth of plants. The robotic nodes for plant shaping in their current design have blue, red, and far-red LEDs. Light is our main plant-shaping tool to trigger plants' adaptive behaviors despite immanent and hardly mutable factors, such as gravity or genetic background. Light is also indispensable for a plant's metabolism, for photosynthesis, and for regulation of its development. Based the natural reaction of plants to different wavelengths of light we are able to interfere with different physiological processes using blue light. We have chosen to manipulate the phototropism of plants by blue light as an attractive stimulus, while using red light as 'feeding' light to allow the plant doing efficient photosynthesis. Mechanical stimulation, such as vibrations generated by vibration motors, have also been tested to influence growth (as reported in D1.3). However, vibrations have found to be of use mostly for woody plants and for long time experiments. There is still a need for another repelling stimulus that can complement our attractive stimulus (blue light). As a result of our analysis, we classify stimulations by three classes characterized by their advantages and disadvantages:

1. remote and with high reach stimulus: light, gravity, electromagnetic field, volatile/sprayed chemicals, air flow
2. precisely targeted, but direct contact required: phytohormones in cream, mechanical obstacles
3. transmitted by direct contact, but with high reach: vibrations made by vibration motors, also some phytohormones in cream
4. remote and potentially precisely targeted: laser light, concentrated LED beam

The range of remote stimuli has to be investigated more deeply to understand what areas and which plants/organs are influenced. Next, we show the good potential of auxins and their inhibitors to manipulate plant growth, our experiments to use liquids in pipes for reshaping, and the use of light intensities to influence plant growth rates.

## 2.2 Phytohormones for plants shaping

Direct application of phytohormones is our next approach following the utilization of volatile compounds (BVOCs) reported in D2.2. For BVOCs no shaping effect was observed, so we focus on basic growth regulators: phytohormones. In the future, BVOCs may be tested to help *flora robotica* plants in pathogen resistance by inducing defense priming, but that would require additional experiments. The influence of phytohormones on plant growth and plant development have been investigated since long, such as by their local application. Spraying of phytohormones is also possible, but due their broad effects on plants, we decided to apply them locally by manual daubing. Auxins are forming gradients within plant bodies, which are responsible of determining the plant's structure, shoots, and roots growth, and the plant polarity. For example, signaling triggered by light results in changes in auxin gradients which leads to the curving of a shoot (e.g., during phototropic reaction). Unilateral auxin application also leads to the elongation of cells on the side where the auxin was applied.

Different phytohormones (NAA, IAA, GA, kinetin) were diluted in lanolin to the final 1% concentration. Firstly we applied hormones to the upper region of dracaenas shoots, what was

Figure 1: Dracenas, that were growing on the bridge above braided structures, had been subjected to direct stimulation by different phytohormones dissolved in lanoline cream. The apical parts of shoots were treated manually once a week.

Figure 2: Curvature of a dracena's shoot which was induced by local application of NAA (auxin) in lanoline cream.

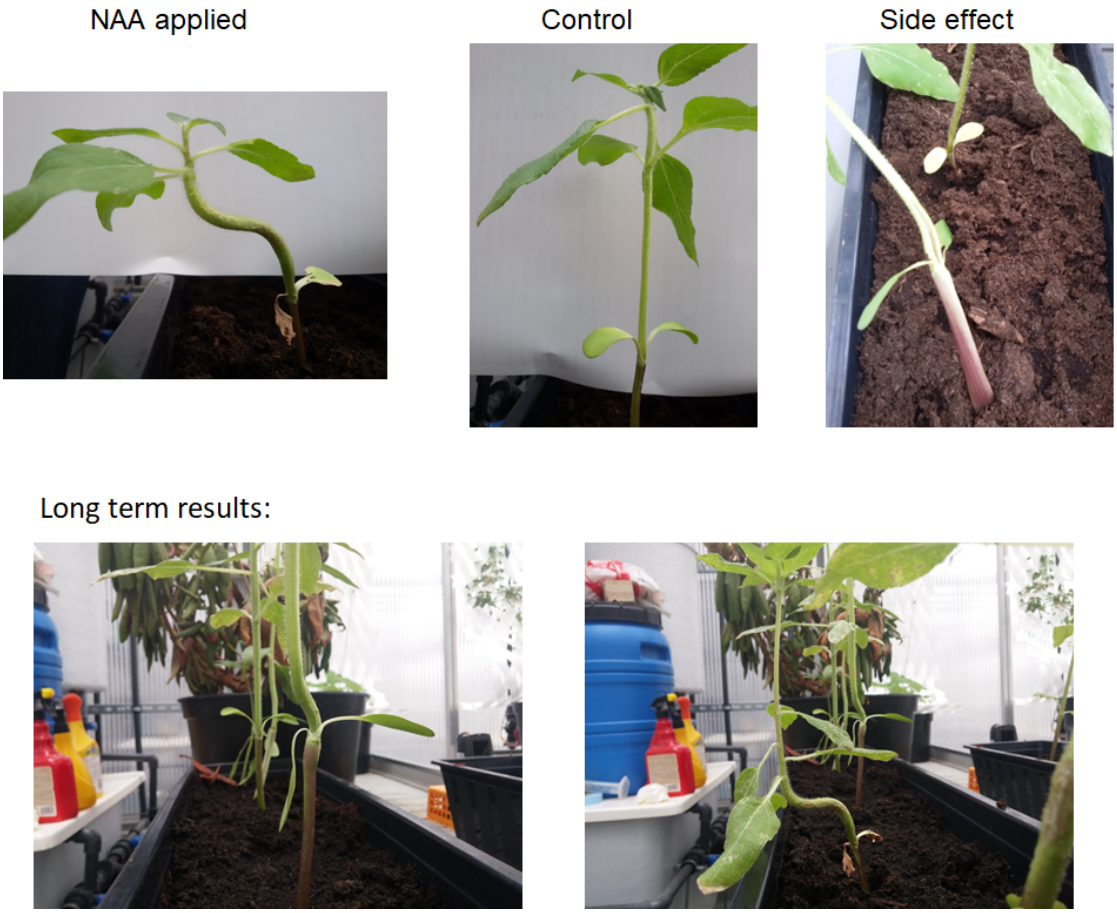



Figure 3: Effects of direct and targeted application of NAA to sunflowers. Bending of stems was observed. In a few cases a stem was deformed and plants even collapsed (as side effect).

repeated once every two weeks at the same place. The growth of Dracaenas (*Dracaena marginata*) was monitored for two months on a bridge above a braided structure (D1.3).

Dracaenas were the plants of choice due to their interesting development of shoots, in which growth depends on intercalary meristems (dracaenas are monocots), but later the shoots get woody and form stiff, stable shapes. Hence, we found dracaenas interesting models for *flora robotica*: moderately branched, potentially prone to shaping, but getting stiff at some time. Clear results were obtained for the application of NAA (1-Naphthaleneacetic acid) on one side of a dracaena green fragment of a branch (see Fig. 2.2). Later, the dracaenas continued their normal growth and curvatures remained, however, a bit straightened. Next, we set experiments with applications of 1% NAA in lanolin to other plant species. Diverse results were obtained for various species.

Localized applications of NAA on sunflowers turned out to be very effectively and temporarily changed their growth direction. The sunflowers strongly curved their shoots to the opposite side than where NAA was applied. This stimulus temporarily change the growth direction, however, sunflower shapes remained affected by NAA application (see Fig. 2.2). A cauliflower was the next species tested for its suitability for local NAA application. For cauliflowers we regularly

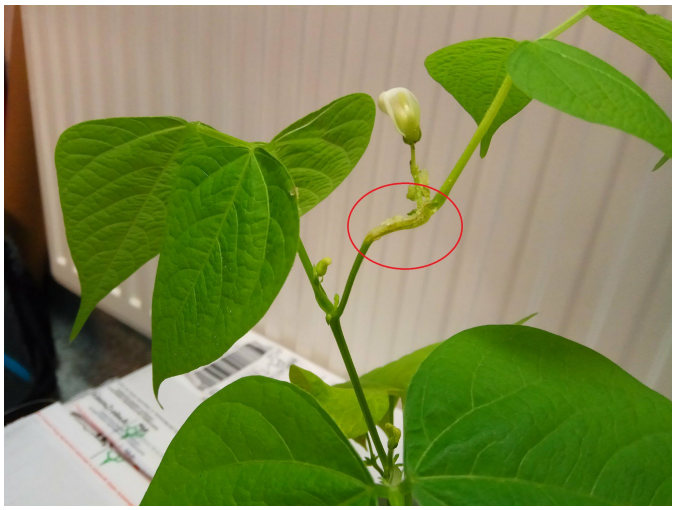


Figure 4: A bean's branch that was bent after application of NAA.

observed deformations of plant tissues at the place of the NAA application. Additionally, no change in stem curvatures was observed. For sunflowers also damaging side effects were observed for about 10% of the plants. Local NAA application was also tested for poplars and common beans. In the case of poplars no effect was observed, probably due to their impenetrable woody organs. Different regions of a main stem and lateral branches were tested, however, buds were omitted, because of their tiny sizes.

Shoots of common beans exhibit clear bending opposite to the side of NAA application. So this method is suitable to be implemented in our current *flora robotica* system, where a common bean is one of the main species of choice (see Fig. 2.2).

Due to its simple, straight structure and fast growth rate, the usage of sunflowers clearly shows potential of local phytohormone applications for introducing changes to growing plant shapes. In addition to the engineering challenges to robotize this procedure, the region subjected to phytohormone has to be specified in the case of different plants, also the hormone concentration and application schedule has to be developed to minimize undesired negative effects. Moreover, not every plant may be susceptible to this method – we failed to bend poplars in this way and induced tissue deformation in cauliflowers. Shaping with local NAA applications may significantly improve plant shaping effects implemented in *flora robotica*. When absorbed, auxins operate on a crucial level of a plant's reorganization, as it is the same mechanism as for phototropism. NAA local applications may be used as a help in shape control (e.g., our obstacle avoidance task), if the light control approach may turn out to be not efficient enough or even as a basic bean shaper combined with lights. The great advantage of using both stimuli, light and local NAA application, to shape (bean) plants is that they fall into two different methodological classes: light allows to remotely steer a plant, while phytohormones in cream applications require direct contact. A similar case is the application of light combined with vibration motors. Utilizing different approaches for *flora robotica* plant shaping probably will benefit from the potential of using bigger number of plants and/or species of more complex structures. Applications of phytohormones in cream or usage of vibration motors may be prominent stimulations in the next generation *flora robotica* system.

Next, we report our experiments with the inhibitor of polar auxin transport, TIBA (2,3,5-Triiodobenzoic acid), that is known for its antagonistic effect to auxin. 1% TIBA was applied


Figure 5: Beans treated with TIBA (2,3,5-Triiodobenzoic acid, the inhibitor of polar auxin transport) were characterized by smaller vertical growth rates, but developed bigger leaves (not measured directly; result of inhibition of apical dominance).

manually to tips of 7 DAG (Days After Germination) common beans. One week after application a significant difference in their growth rate was observed: while the control of only solvent-treated beans had grown an average of 1.55 cm, the TIBA treated plants have grown an average 0.2 cm (see Fig. 2.2). This result encourages us to start studying other species, such as Fallopia and Wisteria, based on sprayed TIBA applications. Applications of chemicals in spray are less targeted, but may be easier to be automatized in our robotic nodes.

## 2.3 Burdening tubes

In deliverable D2.1 the idea of bending a particular branch with a water-filled tube was discussed. Following this partially restricted approach (gravity vector), organs may be reshaped permanently or transiently. Time limits depend on particular plant properties and whether the weight is applied only for a restricted time. Using tubes that can be filled and drained multiple times is a first step that will make this method more applicable. Many possibilities would be available once these tubes could be moved actively. With movable tubes, a limited number of tubes would be sufficient to reshape any branch or shoot if required. For example, moving tubular robots have been developed recently by the Bertold Group from Harvard University ("crawling robot", Rafsanjani et al. [19]).

We applied catheters filled with colored water on the branches of different species: *Dracaena marginata, Crassula ovata*, and *Pinus sylvestris*. The volume of liquid injected into the catheter can be adapted as required in certain situations. The used device enables us to burden plant organs with a max. of 30 ml of liquid, which was distributed in tubes, mainly in a juxta-end "baloon". Few approaches in application of a filled tube to a branch of a *Crassula ovata* resulted in the whole plant deviating (see Fig. 2.3). Stiffness, weight balance, and connections between branches and stem are probable reasons that not one branch, but whole plant was bent. The Crassula plant was simultaneously monitored using phytosensor measurements of differential potentials (Fig. 2.3). One channel was inserted to the burdened branch and the second was inserted to a non-burdened branch. In the first few hours of the experiments potentials were increasing, although we did not find any strong correlations with the plant's bending. Bending was monitored by time-lapse imaging (Brinnio Camera, 5min. intervals).

For dracaena during the time of observation the applied weight has not induced any alignment with the gravity vector, probably due to too less weight applied (see Fig. 2.3). The influence of the applied weight could be revealed in longer experiments, as the growing branch is more prone to deformations than already grown section of the branch. For a pine's branch burdened with a tube quick bending was observed. However, this deformation was partially and rapidly reversed, once the weight had been removed (see Fig. 2.3). This method could also be applied


Figure 6: *Crassula ovata* burdened with liquid-filled catheter. The Crassula branch with applied extra weight bent, but other branches bent as well. No clear correlation with changes in the potential differences were observed.

as a repelling stimulus. Bending of a shoot or a branch that grows too much in an undesired direction, could probably also slow down the growth rate of the burdened organ. At least it could be prevented that a branch approaches an undesired point in space. The hypothesis of a branch's thickening at the cost of an increase in length was not yet verified experimentally, but it can be deduced from knowledge about applying mechanical load to plants and its consequences for the growth processes. This method is limited because we have to rely on gravity, although due to parallel utilization of phototropism and the plant's intrinsic self-organization capabilities a versatile plant shaping system can be developed.

## 2.4 Light intensity

Light is one of the most important environmental factors that enable plants to survive and to grow. Both, wavelengths of light and light intensity determine in what way the light influences plants. Light as provided in our experiments of the robotic nodes results in strong elongations of common bean stems, due to the low light intensity (not measured, evaluated by beans morphology). Generally, shortage of light induces plant growth upwards in order to overgrow a (supposedly) shaded space. We proposed that introducing high light intensities to our *flora robotica* system can have at least two advantages:

1. providing more photosynthetic light to establish more supportive conditions for the plant development and metabolism
2. providing a repellent stimulation, because the plants will slow down their vertical growth in comparison the light conditions as in our experiments with the robotic nodes. Instead of growing upwards, the plants develop bigger leaves etc.

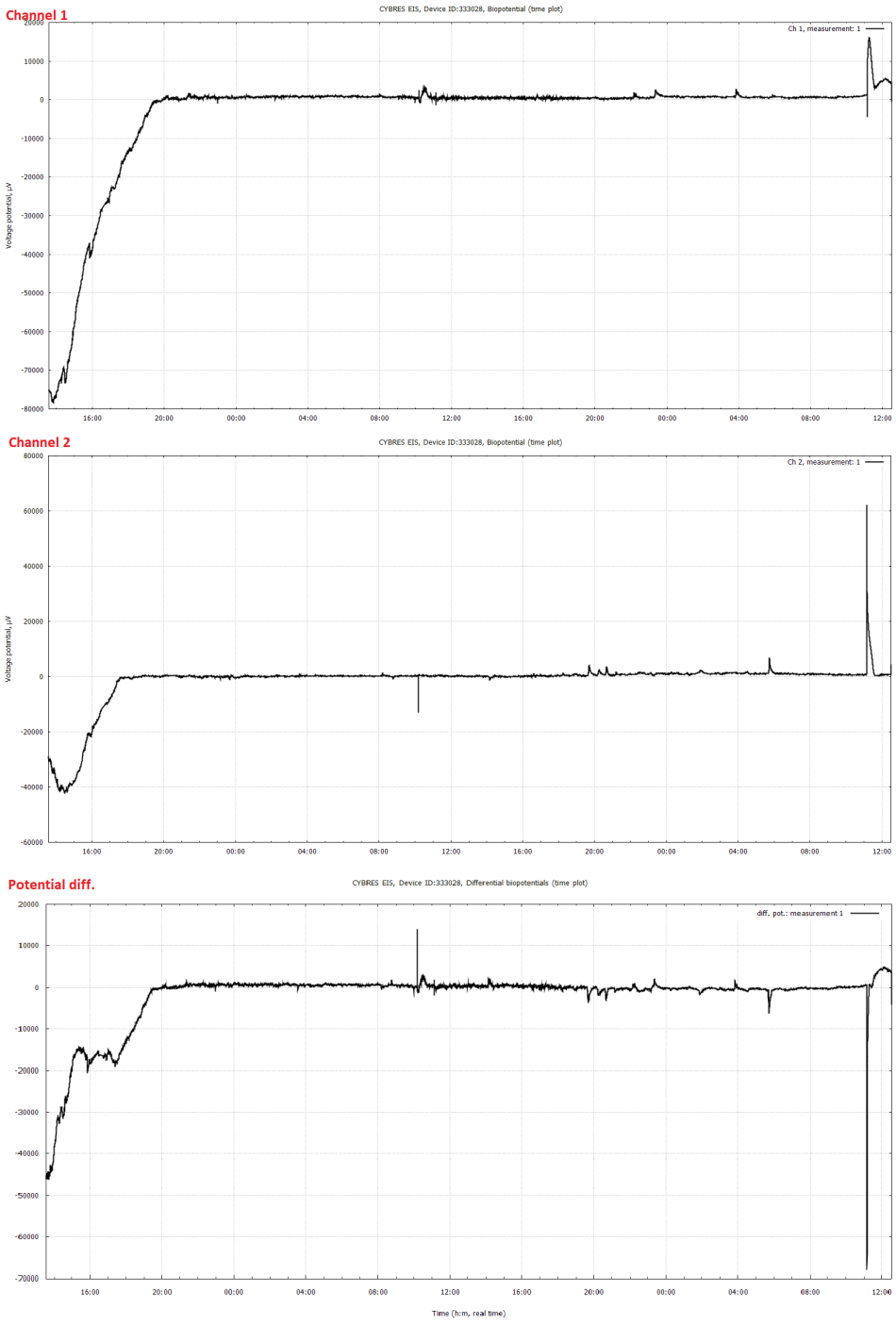



Figure 7: A phytosensor was used to monitor changes in the potential differences when bending branches with catheter's weight. The initial increases in voltage observed for both channels do not correspond to simultaneously observed bending. The bending was occurring continuously with the biggest motion of branches 24 hours after start.

Figure 8: Dracena has not responded to the applied weight during the time of observation.

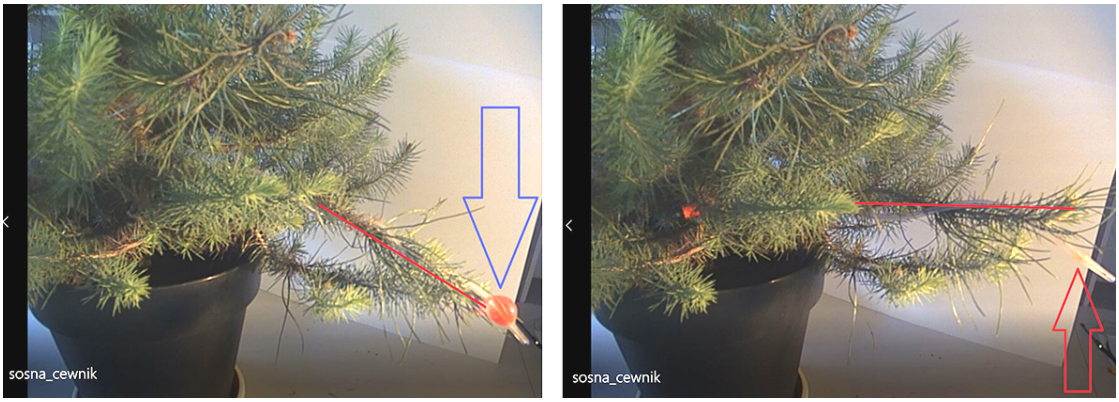



Figure 9: A pine's branch immediately bent after applying a weight. After removing the weight, the branch approached its initial position again, such that the effect was fully reversible.

Figure 10: Two plants on the left were growing in moderate light conditions: 70 $\mu Em^{-2}s^{-1}$. Two plants were growing on the right in low light conditions 20 $\mu Em^{-2}s^{-1}$.

Using two alternating phases might occur beneficial to our *flora robotica* setup. During the robotic node phase, plants are guided towards defined spatial targets. However, the plants' circadian clock may be perturbed with potentially harmful consequences. Hence, the duration of these phases has to be optimized. Moreover, too intensive light is destructive for plants and induces oxidative stress. Hence, also this case has to be avoided. Optionally, three alternate phases can be implemented with high light intensities, followed by low light, and a dark phase. The greater challenge that we see is to select plants or branches that are supposed to be illuminated by a robotic node's light or instead high light intensities. Two options are possible:

1. light beams could be concentrated (with carefully configured light intensities), similar to lasers, then targeted to points of interest
2. plants or organs could be prevented from illumination by screens, then both, action of high and low light could be effectively stopped for chosen plants/organs
3. interference and interplay between different nodes light (blue, red, far-red) together with strong white light possibly also can result in development of attractor and repellent combinations

Concentrations of utilized light beams have to be established. Investigations of scattered LED light could give information about which area and which plant is actuated by a certain light source. In order to show the effect of different light intensities on plant growth and the plant morphology we studied common beans in growth chambers with different light conditions (20 vs 70 $\mu Em^{-2}s^{-1}$). Temperature, humidity, and hours of illumination were the same. Plants that were growing in lower light intensity were taller, but developed smaller leaves (see Fig. 2.4).

In many setups highly intensive light could be provided by the sun without any energy costs. In turn, screens for shading would then be the only feasible option to avoid undesired high illumination of plants and organs. The interplay of different lights is, however, challenging, for example, because sunlight changes in its intensity across wavelengths during the day and seasons. Controlling and directing sunlight with a sophisticated mirror system seems too challenging in the case of our needs in *flora robotica*.

To summarize, we have reported different options to modulate plant growth. We have found options of repellent stimuli by appropriate plant experiments. The question of which repellent is optimal is challenging and arguably none of the reported methods can serve as an ideal repellent; probably because plants do neither reverse their growth nor do they not run away. The efficiency of blue light as attractor depends on other environmental factors and, first of all, on the complex developmental program that is realized by plants and their proprioception. For example, plants have to correlate their growth upwards and their branching with appropriate thickening of the main stem to avoid overloading and breaking. Methods that rely on supplementing soil with different nutrients were not discussed, due to the incompatibility with desired setups of *flora robotica*. The implementation of the chosen methodology will allow us to verify our proposals and to optimize the stimulations for the needs of our *flora robotica* system.

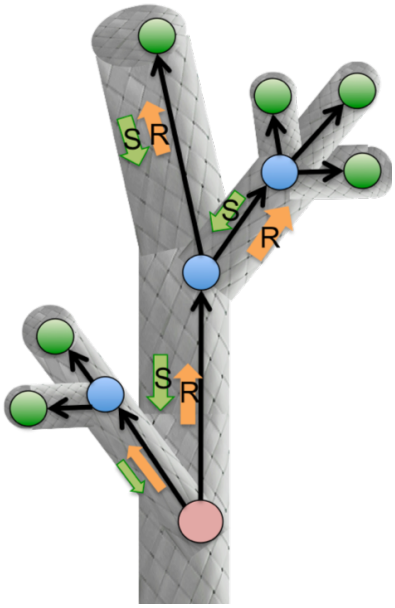



Figure 11: A schematic view on the VMC. Successin (S) is produced at the leaves, flows rootwards, and updates the thickness of all vessels accordingly. Resource (R) starts from the root and is distributed between the children of every node proportional to the thickness of their vessels.

# 3 Vascular Morphogenesis Controller (VMC)

The Vascular Morphogenesis Controller (VMC) is a distributed model of morphogenesis inspired by the competitions for common resources in plants. The model has been previously introduced in D2.2. In the following sections we present a close look into the dynamics of an improved version of the model and the effects of its parameters, then we describe how to set the parameters while satisfying different requirements. We present a software platform containing a standalone VMC implementation and a biologically plausible 3-dimensional simulation of trees in the simulation framework *Unity* that is controlled by the VMC. Next, we present a self-assembly study with an application of VMC in an embodied system (emulation of a scaffold in 2-d with mobile robots). Finally, we present a distributed hardware platform with embedded VMC for directing growth of braided structures. The integration of VMC with generalized braided structures and steered plant growth—implemented in simulation through the Integrated Growth Projectionb (IGP)—is presented at the end of this document (Sec. 6).

## 3.1 Looking closer into the dynamics

Here we describe an improved version of the VMC model [28], look into the details and how the different parameters influence the dynamic behavior of the system.

### 3.1.1. The VMC model

Fig. 11 shows a schematic representation of VMC. The figure shows the flow of Successin (S) produced at the leaves and propagating towards the root. The flow of Successin regulates the thickness of vessels (weights of the edges of the graph). A common Resource (R) starts at the root and is distributed between the children of each node proportional to their vessel thickness (V). Successin is produced at the leaves based on the local sensory inputs and constant parameters as defined by

$$S_{leaf} := \text{PRODUCTION}(\text{params}, \text{sensors}). \tag{1}$$

Successin flows towards the root passing through internal nodes. At an internal node $i$, the flow of Successin passing through may drop off based on the sensor inputs and constant parameters determined by a transfer function in the range of $[0, 1]$ defined by

$$S_{\text{non-leaf}} := \text{TRANSFER}(\text{params}, \text{sensors}) \sum_{b \in \text{children}} S_b. \tag{2}$$

The weight of each connection $(i, j)$ (thickness of the vessel) is adjusted based on Successin passing the vessel and the parameters determining the competition rate between the siblings are

$$\begin{aligned} V_{i,j} &:= V_{i,j} + \alpha(S_j^{\beta_i} - V_{i,j}), \\ \beta_i &= \text{COMPETITION}(\text{params}, \text{sensor}), \end{aligned} \tag{3}$$

where $V_{i,j}$ is the connection between node $i$ and its child node $j$. $S_j$ is the Successin of node $j$ flowing towards $i$.

In the current implementations used in the following sections, the above mentioned functions are defined by

$$\text{PRODUCTION}(\text{params}, \text{sensors}) = f(\omega_{\text{const}} + \sum_{s \in \text{sensors}} \omega_s I_s), \tag{4}$$

where $f(x)$ is

$$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{otherwise} \end{cases},$$

where $\omega_{const}$ is the constant production rate of Successin at a leaf, $\omega_s$ is the sensor dependent production rate which is the coefficient determining the dependency of Successin production on sensor input $s$.

The transfer function is defined by

$$\text{TRANSFER}(\text{params}, \text{sensor}) = g(\rho_{\text{const}} + \sum_{s \in \text{sensors}} \rho_s I_s), \tag{5}$$

where $\rho_{\text{const}}$ is a constant transfer rate, $\rho_s$ is a sensor-dependent transfer rate of Successin passing a node, and $g(x)$, in the current implementation, is

$$g(x) = \begin{cases} x, & \text{if } 0 \leqslant x \leqslant 1 \\ 0, & \text{if } x < 0 \\ 1, & \text{if } x > 1 \end{cases}.$$

The competition function is defined as

$$\text{COMPETITION}(\text{params}, \text{sensor}) = \beta_{\text{const}} + \sum_{s \in \text{sensors}} \beta_s I_s, \tag{6}$$

where $\beta_{const}$ and $\beta_s$ are a constant and a sensor-dependent competition rate of vessels respectively.

As it is described in the definitions above, PRODUCTION, TRANSFER, and COMPETITION are all functions of constant parameters combined with sensor inputs. However, by setting the according parameters to zero, the effect of the sensors can be removed if required depending on the application.

**Resource distribution over the structure.** The common resource starts at the root and is distributed over the structure according to the vessel thickness (weight of connections). A part of the resource reaching node $i$ ($R_i$) is partially consumed at that node and what remains is divided among its children. For a given child $j$ that is done proportional to the vessel thickness $V_{i,j}$ as defined by

$$R_j := (R_i - c)\frac{V_{i,j}}{\sum_{b\in\text{children}} V_{i,b}}, \tag{7}$$

where $c$ is the constant consumption rate of the resource at a node and 'children' is the set of children of node $i$. $c$ can be set to zero for a resource distribution only between leaves. The value of $c$ in relation to the amount of the resource at the root puts a constraint on the overall graph size. The common resource initiated at the root can be a constant value or a function of the environment and/or Successin that reaches the root from anywhere within the graph. In the current implementation, the $R_{\text{root}}$ is fixed to a constant value.

**Addition of nodes.** When the graph grows from a leaf, a number of new leaves appears as the children of the old leaf. The decision for growth happening at a leaf is based on the share of the common resource reaching the leaf and follows a growth strategy. Different strategies can be taken to direct the decision. For example, the leaf with the maximum resource value can be chosen as a candidate for growth, or the resource can be considered as the probability of growth. Here, we use a threshold $th_{add}$ to determine whether a leaf grows. If a leaf's resource is above this threshold, the leaf is a candidate for growth in the next time step. At each time step, one of the candidates are randomly picked to implement the growth.

**Deletion of nodes.** Leaves can be removed from the graph based on the resource value reaching the nodes and following a chosen deletion strategy. Here, we define a threshold $th_{del}$ to decide on the deletion of a node's children. In the currently implemented strategy, we remove the children of a node if all children are leaves and the amount of the resource at the node is below the threshold. For all nodes with that property the children are removed in each time step.

### 3.1.2. A Discussion on the Effect of Parameters

In order to get an initial understanding of each parameter's impact on the growth behavior of the structures, we use a simplified setup with a simplified 1-dimensional structure consisting of a root and its two children (growing along a line to the left and the right) where all the leaves in the setup can get only one child. Fig. 12 gives an example structure. We assume that the left and the right branch contain a leaf each and the number of nodes between the leaves and the root are $n$ and $m$ respectively. The sensor-dependent transfer and competition rates are set to zero ($\rho_s = \beta_s = 0$), and the Successin produced at the left leaf and the right leaf are correspondingly $S_{\text{leafL}}$ and $S_{\text{leafR}}$.

**Intrinsic tendency towards shorter paths** In a structure as in Fig. 12, the amount of Successin reaching the root from the left and right branch converges to $S_L = S_{\text{leafL}}\rho^n$ and $S_R = S_{\text{leafR}} \times \rho^m$. If the total resource at the root is $R$, the consumption rate of resource at the internal nodes is $c = 1$, and with the competition rate $\beta$, the edge thicknesses for each branch of the root converge to $V_L = S_L^\beta$ and $V_R = S_R^\beta$ with a speed of $\alpha$ as the adaptation rate. The

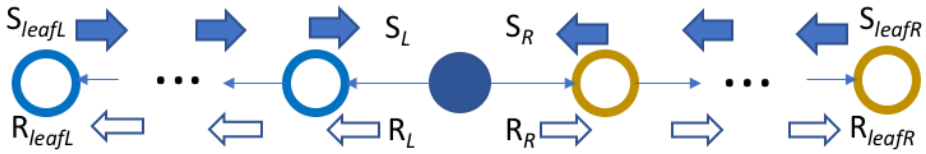



Figure 12: A simplified 1-dimensional VMC setup with a root having two children while all other nodes have at most one child.

amount of the resource reaching each leaf converges to

$$R_L = R\frac{S_{\text{leafL}}^{\beta}\rho^{n\beta}}{S_{\text{leafL}}^{\beta}\rho^{n\beta} + S_{\text{leafR}}^{\beta}\rho^{m\beta}} - n, \tag{8}$$

$$R_R = R\frac{S_{\text{leafR}}^{\beta}\rho^{m\beta}}{S_{\text{leafL}}^{\beta}\rho^{n\beta} + S_{\text{leafR}}^{\beta}\rho^{m\beta}} - n. \tag{9}$$

Therefore, whenever $m = n$, the equations can be simplified to

$$R_L = R\frac{S_{\text{leafL}}^{\beta}}{S_{\text{leafL}}^{\beta} + S_{\text{leafR}}^{\beta}} - n, \tag{10}$$

$$R_R = R\frac{S_{\text{leafR}}^{\beta}}{S_{\text{leafL}}^{\beta} + S_{\text{leafR}}^{\beta}} - n, \tag{11}$$

and the difference between the amount of resources for each leaf depends on the amount of Successin that the leaf produces. In turn, in an environment that gives identical sensor values for all leaves causes identical production of Successin at the leaves. Then the leaf of the shorter branch would get more of the resource. The resulting desired property of VMC to prefer shorter paths has been demonstrated in a maze scenario by Zahadat et al. [29].

**Regulation of growth of specific paths by using a sensor-dependent transfer rate.** In the previous example, the transfer rate $\rho$ was a constant value not depending on sensors. However, the transfer rate can be a combination of a constant parameter and several sensor values. For example, while a light sensor can be used to affect the production rate of Successin at the leaves, a stress sensor (e.g., associated with the joints of a physical structure) with a negative effect on the transfer rate can be employed at internal nodes. Thus, in a two-branched structure of Fig. 12 with $m = n$, a high physical stress at the left branch would decrease $\rho$ for that branch and therefore we have $S_L < S_R$. That consequently leads to $R_L < R_R$ and eventually results in a preference for growth at the right branch.

### 3.1.3. Numerical Investigation of Parameter Effects

To provide a general understanding of the behaviors of VMC in constant and dynamic environments, a set of experiments are performed in simulation. We limit our study to a setup with two children for every node. The simulation starts with a root node with two children each of which at one side of the root. None of the nodes of this initial structure are allowed to be removed during the experiment. The two children of the root make the two main branches of the structure and their morphological properties are observed and compared in the experiments.

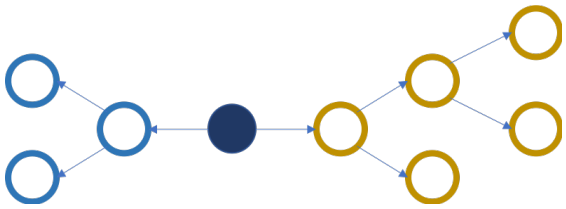


Figure 13: A schematic example structure with three nodes at the left main branch and five nodes at the right main branch.

In the simulations, the radius of a node is set to nine pixels while the length of connections between the nodes is three times as long as the node radius (21 pixels from node center to node center). When growth is triggered at a leaf, two new leaves are added. The orientation of the connections between the new leaf and the old leaves is defined as

$$\theta = \pm\pi/20 + X, \quad X \sim \mathcal{U}(-\pi/20, \pi/20)\,.$$

Here, nodes cannot perceive other nodes (they have no physical effect on each other). We chose this option in order to limit the effects of certain implementations of physical simulations on the generated shapes and to keep the focus mainly on the behavior driven by the VMC.

To define a stop condition for an experiment, the maximum size of the structure during the development is recorded and the experiment continues until 50 time-steps after the recorded maximum size. Reaching this stop condition in a limited time period is guaranteed by setting the consumption rate at the nodes ($c$) to a positive value.

In the following, a collection of experiments using this simulation implementation is presented demonstrating the effect of different parameters on the structure's morphology driven by VMC. Table 1 represents the set of parameters that we varied in the following. Table 2 represents the parameters that are fixed in this study (unless explicitly stated otherwise in particular experiments).

| parameter name | description |
|---|---|
| $\alpha$ | adaptation rate of vessels |
| $\beta_{\text{const}}$ | competition rate of sibling vessels, constant rate |
| $\beta_s$ | competition rate of sibling vessels, sensor-dependent |
| $\rho_{\text{const}}$ | transfer rate of Successin at the internal nodes, constant rate |
| $\rho_s$ | transfer rate of Successin at the internal nodes, sensor-dependent |
| $\omega_{const}$ | production rate of Successin at the leaves, constant rate |
| $\omega_s$ | production rate of Successin at the leaves, sensor-dependent |

Table 1: investigated parameters

| parameter name | value | description |
|---|---|---|
| $R_{\text{root}}$ | 20 | common resource initiated at the root |
| $c$ | 1 | consumption rate of the common resource at each node |
| $th_{\text{add}}$ | 3 | threshold of resource at a node for adding new leaves |
| $th_{\text{del}}$ | 1 | threshold of resource at a node for deleting its leaves |

Table 2: constant parameters

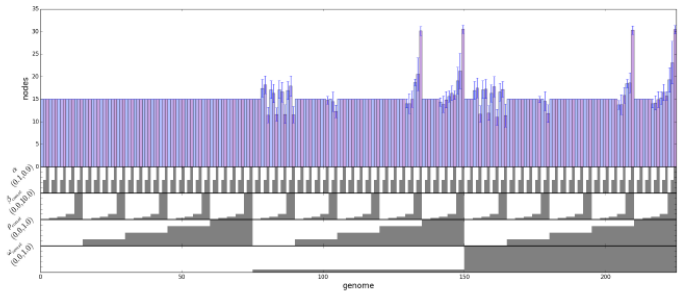
| configuration | environment | parameters | values |
|---|---|---|---|
| EMP1 | No environmental effects (EMPty environment) | $\alpha$ | $\{0.1, 0.5, 0.9\}$ |
| | | $\beta_{\text{const}}$ | $\{0, 0.5, 1, 2, 10\}$ |
| | | $\rho_{const}$ | $\{0, 0.25, 0.5, 0.75, 1\}$ |
| | | $\omega_{const}$ | $\{0, 0.1, 1\}$ |
| EMP2 | No environmental effects (EMPty environment) | $\alpha$ | constant (0.1) |
| | | $\beta_{\text{const}}$ | $\{1, 3, 5, 7, 9\}$ |
| | | $\rho_{\text{const}}$ | $[0.1, 1.0]$ in steps of 0.1 |
| | | $\omega_{\text{const}}$ | constant (0.1) |
| SW | Half-bright environment, SWitching the brightness when growth is completed | $\alpha$ | $\{0.1, 0.5\}$ |
| | | $\beta_{\text{const}}$ | $\{1, 2, 3\}$ |
| | | $\rho_{\text{const}}$ | $\{0.25, 0.5, 0.75\}$ |
| | | $\omega_{\text{const}}$ | $\{0, 0.1, 1\}$ |
| | | $\omega_s$ | $\{-0.1, 0.1\}$ |

Table 3: experimental configurations

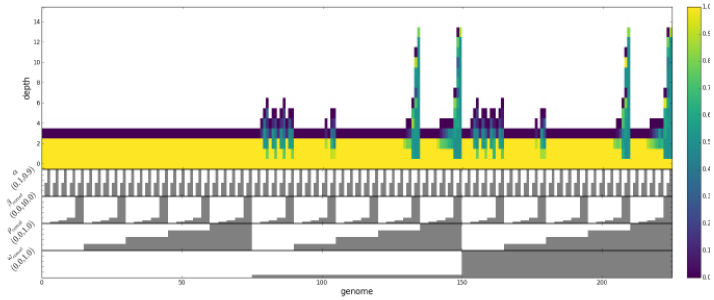
#### 3.1.4. Investigated Behavioral Aspects

We investigate a set of static and dynamic morphological aspects in different experimental setups with different parameter values. The investigated behavioral aspects are:

- $N_{\textbf{TOTAL}}$**:** total number of nodes of the structure at the end of the run.
- $N_{\textbf{LEFT}}, N_{\textbf{RIGHT}}$**:** number of nodes of the two main branches (the two children of the root) counted at the end of the run
- $N_i$**:** The number of nodes in a particular depth $i$.
- $D$**:** number of nodes that are deleted during the course of the experiment
- **Asymmetry:** asymmetry of the structure in respect to the two main branches; computed by
$$\text{asymmetry} = (\max(N_{\text{LEFT}}, N_{\text{RIGHT}}) - min(N_{\text{LEFT}}, N_{\text{RIGHT}}))/N_{\text{TOTAL}}$$
- $rA$**:** asymmetry of the structure in respect to the two main branches; computed by
$$rA = (N_{\text{RIGHT}} - N_{\text{LEFT}})/N_{\text{TOTAL}}$$
- **Adaptation:** adaptation level is defined as a measure of change in the asymmetry of the structure in response to a change in the environment,
$$\text{Adaptation} = 0.5|rA_1 - rA_2|,$$
where $rA_1$ and $rA_2$ are respectively the $rA$ of the structure at the end of the first phase (before the environmental change) and the second phase (after the environmental change)




(a) Total number of nodes ($N_{TOTAL}$). The bars represent the value of $N_{TOTAL}$ averaged over all the repetitions and the whiskers represent their standard deviation. The colors is for a clearer representation.




(b) *Fullness* of depths

Figure 14: EMP1 setup (no environmental effects)

- ▷ **dynamicity:** dynamicity of the structure is measured by the number of nodes that are deleted from the structure over time divided by the size of the structure at the end of the run,

$$\text{dynamicity} = D/N_{\text{TOTAL}}$$

- ▷ **fullness$_i$:** The fraction of nodes in a certain depth $i$ that are not leaves; computed by

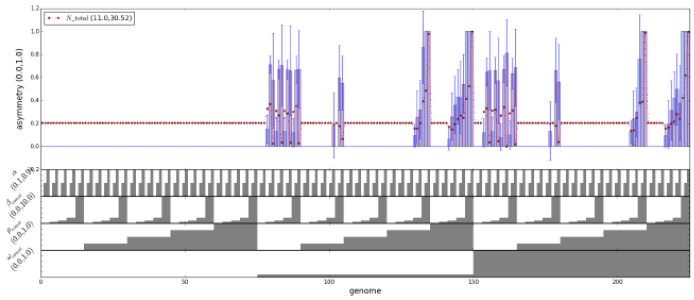
$$\text{fullness}_i = N_{i+1}/(2N_i)$$

### 3.1.5. No Environmental Effects

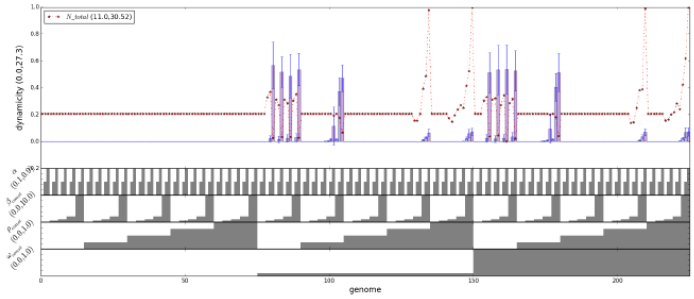
In the first sets of experiments (for configurations EMP1 and EMP2 see Table 3), we excluded any effect of the environment by setting all the sensor-dependent parameters to zero.

**Coarse overall parameter sweep.** While the sensor-dependent variables are set to zero, a coarse parameter sweep is performed on the other variable parameters ($\omega_{\text{const}}, \rho_{\text{const}}, \beta_{\text{const}}, \alpha$). The configuration of EMP1 (see Table 3) is used for the parameter settings. Every parameter setting is used in 25 independent runs. Fig. 14a shows the $N_{\text{TOTAL}}$ for all the investigated settings. The bars represent the value of $N_{\text{TOTAL}}$ averaged over all the repetitions and the whiskers represent their standard deviation. Fig. 14b shows the fullness$_i$ for every existing depth in the investigated settings. The demonstrated values are the mean values of all repetitions of the respective setting.

As seen in Fig. 14a, in this empty environment with $\omega_{\text{const}} = 0$, the number of nodes is constant (15 nodes). This is not surprising considering the eq. 4 that describes the production




(a) *Asymmetry*




(b) *Dynamicity*

Figure 15: EMP1 setup (no Environmental effects)

of Successin, as in this case the amount of Successin is always zero for all the nodes due to $\omega_{\text{const}} = 0$. This leads to equal $V$ for all the children of every node ($V_i = 0$) which results in an equal distribution of the common resource between all the children of nodes and a symmetrical shape for the resulting VMC graph.

When $\omega_{\text{const}}$ is more than zero, Successin is produced in positive amounts at the leaves flowing back towards the root. Even though the production of Successin is exactly the same in all leaves, new nodes are not added all at the exact same time. By adding a new node, the amount of Successin flowing in different branches differ. Thus, new nodes lead to different $V$s for their parents. Depending on the values of the other parameters, this effect can be temporary or it can get permanent via the positive feedback caused by the further growth at the corresponding branches.

Fig. 14b shows the *fullness* of the structure at each existing depth of the structure in every setting. Also the maximum depth for each setting is shown. The maximum depth is three for all setups with $N_{\text{TOTAL}} = 15$ (see Fig. 14a) and all the structures are full in every depth except for the last (depth of three) where the nodes have no children. In other setups, even if the number of total nodes is lower, the maximum depth is larger than three meaning that the structure is asymmetric.

Fig. 15 shows the *asymmetry*, and *dynamicity* of the structures in the different settings. Fig. 15a confirms the asymmetry of the setups with $N_{\text{TOTAL}} \neq 15$. Such setups also have depths of more than three.

By considering $\rho$ in the settings with $\omega_{\text{const}} > 0$ in both Figs. 14 and 15, three different groups of setups can be identified: setups with $\rho < 0.5$, setups with $\rho = 0.5$, and setups with $\rho > 0.5$. While all the setups with $\rho = 0.5$ lead to symmetric structures without any *dynamicity* during the growth process, the other setups can be asymmetric depending on other parameters.

Figs. 15b and 15c show a high amount of *dynamcity* in asymmetric settings with $\rho < 0.5$. The reason is found by looking at the extreme case of $\rho = 0$. In this case, no Successin passes from one level to the next. Therefore, the only effect of Successin in these settings is to regulate the thickness of edges before leaves. After the first growth step (that occurs at either of the main branches), the edge behind the grown node starts to decay towards zero (with the rate of $\alpha$) and therefore it gets a smaller share of the resource. But on the other hand, the resource that was already delivered to that node continues being distributed among the node's children. All leaves with $R > th_{\text{add}}$ are potentially capable of growth and depending on the value of resource at the root, the newborn leaves of the grown main branch can be capable of growth. This fact also applies to the new leaves as they emerge. Since the number of leaves at the side of the grown branch is larger than at the other side which only has one leaf (the initial leaf), the grown branch has a higher chance to keep growing (as long as the share of the resource at the leaves are still higher than $th_{\text{add}}$). However, after a few steps the resource is shared too much and no leaves of the larger side get enough resource anymore. Thus, they stop growing while the other side starts the growth with a large amount of resource which again goes through the same process of distribution and growth. At the same time, the previously larger side looses its nodes due to its small amount of resource until there are no nodes left at that branch except the initial leaf of that side. The leaf then attracts the resource again and the fluctuating asymmetry between the two main branches continues.

The speed of the decay at the edges, which are not right before the leaves, depends on $\alpha$. With higher $\alpha$, the decay rate and therefore the switch between the directions of asymmetries occurs faster leading to higher dynamicity.

In the settings with high amounts of $\rho$ ($\rho > 0.5$), the value of $\beta$ has a high contribution to the dynamics and the shape of the structure. Recall that $\beta$ contributes to the competition rate between siblings meaning that higher values of $\beta$ increase the competition and therefore reinforce the initial asymmetries caused by other aspects of the system.

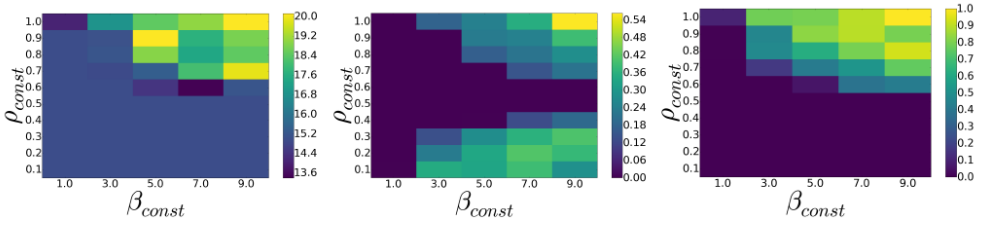
At $\rho = 0.5$, the structures are symmetric. The reason is that the number of immediate children (#children) for every node in the current implementation is two, meaning that the growth of a node adds two children to the node and thus two times of the amount of Successin. This is then multiplied by the transfer rate $\rho = 0.5 = 1/\#\text{children}$ which does not change the Successin passing the node (an example with different number of children is shown in the next section confirming the inverse relation between the number of children and the critical value for $\rho$).

With a small $\alpha = 0.1$, the *dynamicity* is always small and in most cases its even zero if the structure is asymmetric. High values of both competition rate $\beta$ and adaptation rate $\alpha$ promote *dynamicity* although the behavior highly depends on the value of the transfer rate $\rho$ (and the existence of production rate $\omega$).

**Combined effect of competition and transfer rates.** In order to have a closer look at the combined effects of the transfer rate $\rho$ and the competition rate $\beta$, the experiments are repeated for a parameter sweep of higher resolution for $\rho$ and $\beta$ (EMP2 configuration in Table 3). The other parameters are chosen such that *dynamicity* and asymmetry are possible $\omega = 0.1$ with a slow adaptation rate $\alpha = 0.1$.

Fig. 16 shows the mean values of all the repetitions for $N_{\text{TOTAL}}$, *dynamicity*, and asymmetry in the different settings. The change in the behavior around $\rho = 0.5$ (as discussed in the previous section) can be seen. In a large region of the parameter space, the final number of nodes ($N_{\text{TOTAL}}$) is 15 and the structures are symmetric. This holds for all the settings with $\rho \leq 0.5$. We also find that the *dynamicity* increases when the competition rate $\beta$ increases.

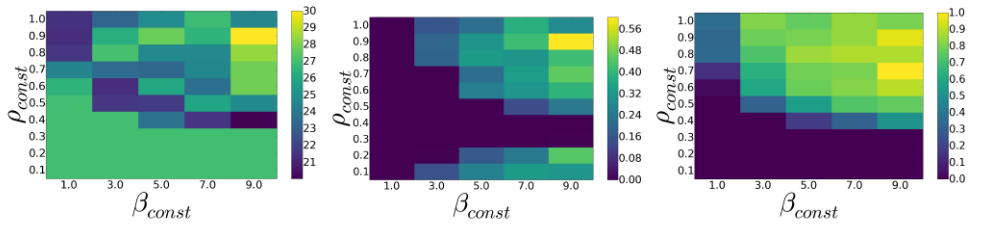
In an additional experiment we study structures with three children (instead of two). In order




(a) $N_{TOTAL}$ (b) *Dynamicity* (c) *Asymmetry*

Figure 16: $\beta$ vs $\rho$ for EMP2 for structure with 2 children and $R = 20$

to avoid too small structures, we used a higher value of the resource at the root ($R_{\text{root}} = 30$). The results are shown in Fig. 17. The inverse relation between the critical value of $\rho$ and the number of children ($\rho_{\text{critical}} = 1/\#\text{children}$) can be seen for values of $\rho = 0.3$.

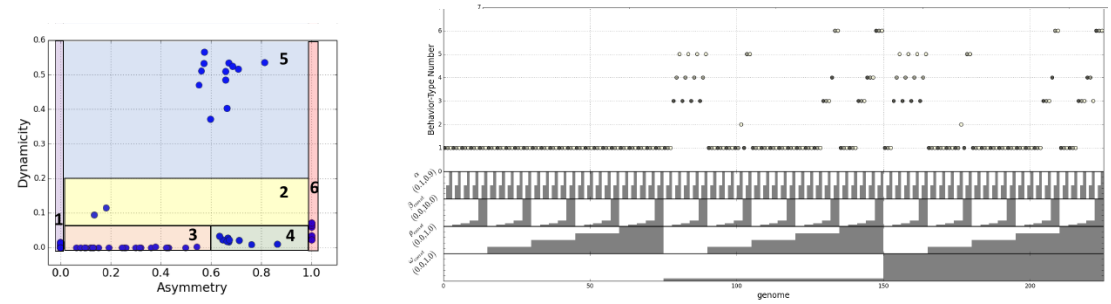



(a) $N_{TOTAL}$ (b) *Dynamicity* (c) *Asymmetry*

Figure 17: $\beta$ vs $\rho$ for EMP2 for structure with 3 children and $R = 30$

**Different classes of behaviors.** Here we classify different behavioral types in groups with similarity according to certain aspects. Two behavioral aspects of *asymmetry* and *dynamcity* are considered together in order to detect groups of settings that can be classified into same behavioral types. Fig. 18a shows the different settings (each setting represented by a dot) and how we have defined the six different classes of behaviors. The classes are defined based on the clusters of the dots in the figure. The classes 1 and 6 respectively represent the behaviors with full symmetry and full asymmetry. Classes 3 and 4 represent behaviors with low *dynamicity* spanned over different degrees of asymmetry where the behaviors in class 4 are more asymmetric than class 3. Class 2 represents behaviors with relatively low *dynamicity*. The settings in class 5 demonstrate high levels of *dynamicity*.

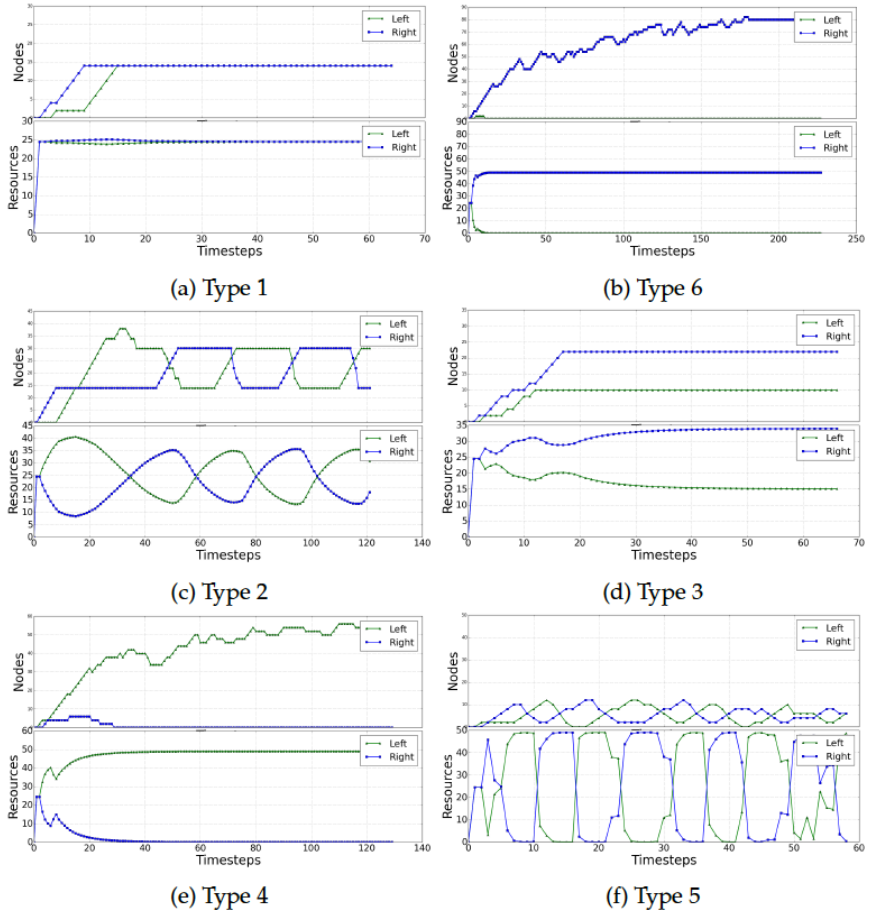
Fig. 19 shows example behaviors of each class in EMP1. The number of nodes and the total resource at each of the two main branches of the structure are shown. Fig. 18b shows which settings fall into which behavioral class for experiment EMP1. Most settings lead to behaviors of class 1 (complete symmetry). For $\omega > 0, \rho > 0.5$ ($\rho > 1/\#\text{children}$), increase in the value of $\beta$ (competition rate) leads to behaviors of class 3, then 4, and eventually 6 which are all of comparatively low *dynamicity* in this configuration. In the region of $\rho = 0$ with $\omega > 0$, the behavior type only depends on the adaptation rate $\alpha$.




(a) Classifying behaviors into different types

(b) Behavioral types of each setting. The different shades of gray are for visualization purposes.

Figure 18: Different behavioral classes defined based on the asymmetry and dynamicity for the experiment EPM1 (no environmental effect)




(a) Type 1 (b) Type 6

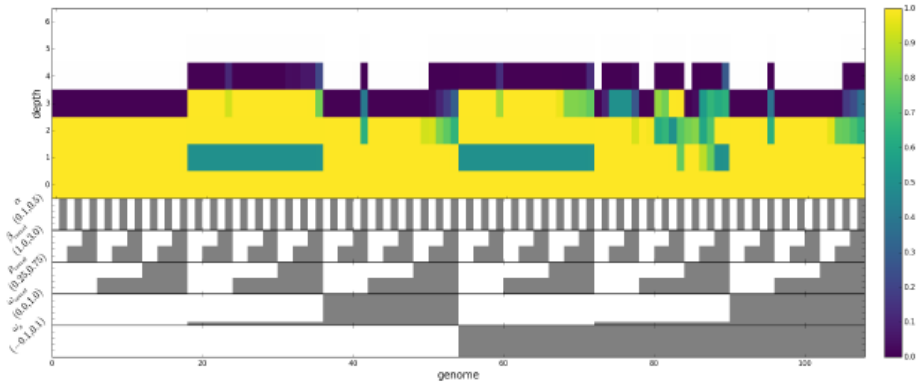
(c) Type 2 (d) Type 3

(e) Type 4 (f) Type 5

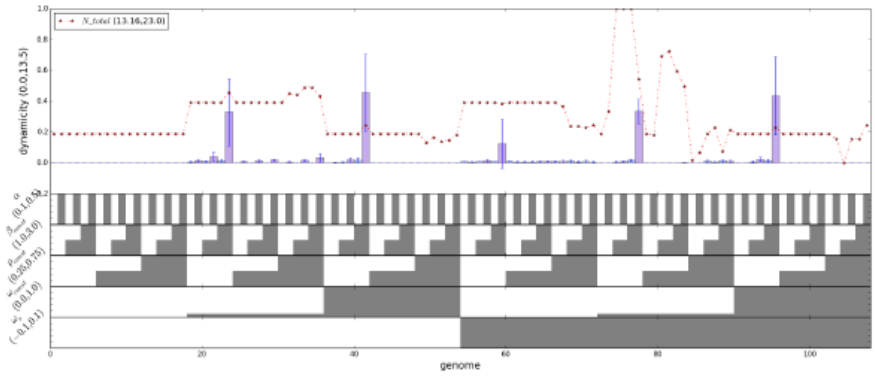
Figure 19: Dynamics of the number of nodes and resources at each of the two main branches (left/right) over runtime for an example setting from each behavior class in experiment EMP1 (in these examples $R = 50$ in order to have a higher resolution of dynamics in the behavior).

### 3.1.6. Environmental Switch

In another set of experiments (configurations SW, Table 3), we test the behavior of the structures in different settings in an environment with two different regions ("half-bright"): the right side of the environment is bright ($I_s = 1$) and the left side is dark ($I_S = 0$). One of the main branches are placed in the bright side of the environment, the other in the dark side. The structure grows until 50 time steps after its maximum size is reaches. Then the brightness switches to the other side and the growth continues until 50 time steps after the maximum size of it in the bright side is reached. In this experiment, two different values are tested for the sensor-dependent production rate of Successin, $\omega_s$: a positive one $+0.1$ promoting Successin production in light, and a negative production rate $-0.1$ promoting Successin production in shadow. According to the eq. 4, the value is used as a coefficient to the sensor value and the result is summed up with the constant production rate determining the amount of Successin for the leaf.




(a) *Fullness* of depths.

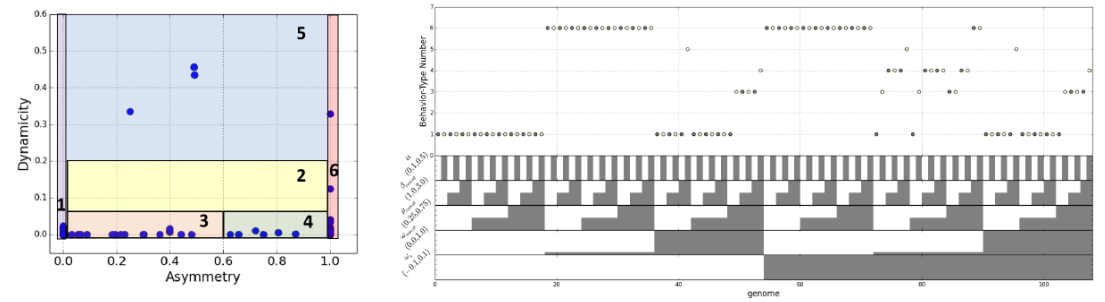



(b) *Dynamicity*

Figure 20: *Fullness* and *dynamcity* for each setting in the experiment SW ("half-bright" environment). The values are the mean values of 25 independent runs of the experiment for every setting.

Fig. 20 shows the *fullness* and *dynamicity* of the structure in the different settings at the end of the first phase of the experiment (before the switch). According to eq. 4, with $\omega_{const} = 0, \omega_s = -0.1$, there is no environmental effect and no production of Successin at the leafs. Symmetric

structures with a maximum of depth three are found for such settings. With the high constant value of $\omega_{\text{const}} = 1$ and the small value of $\omega_s = \pm 0.1$, there is not much effect of the input $I_s$ in the behavior of the structure. As expected, the combination of high values of $\beta$ and $\alpha$ with low value of $\rho = 0.25$, leads to high *dynamicity*. With the values of $\omega_{\text{const}} = 0.01, \omega_s = -0.1$, $\omega_{const} = 0, \omega_s = 0.1$, and in the short range of $\omega_{\text{const}} = 0.01, \omega_s = 0.1, \rho = 0.75, \beta = 3$, Fig. 20a shows $fullness = 0.5$ at depth one followed by $fullness = 1$ at depth two meaning that one of the initial nodes of the structure is a leaf while the other node is full (both of its children are internal nodes).




(a) Classifying behaviors into different types

(b) Behavioral types of each setting

Figure 21: Different behavioral classes defined based on the asymmetry and dynamicity of the settings for the experiment SW (half-bright environment)

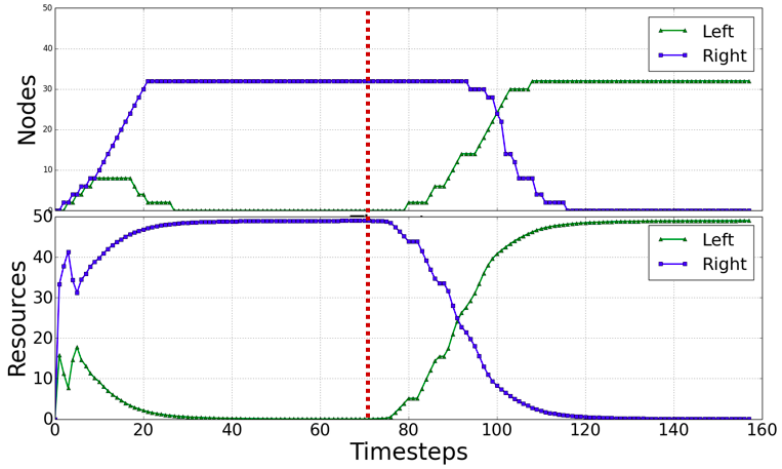



Figure 22: Example behavior of a random setting from class 6 in experiment SW. The vertical red line represents the switching event.

Fig. 21 shows the behavior types of the settings in experiment SW at the end of the first phase before the switching event occurs (the same classification of behaviors as EMP1 are used here). Fig. 22 demonstrates an example run of a random setting of behavior class 6 in the SW experiment.

Fig. 23 shows the *Adaptation* levels for the SW experiment and the corresponding *Asymmetry* of the structures before the switch. The high value of $w_{const}$ (comparable to $\omega_s$) turns adaptivity

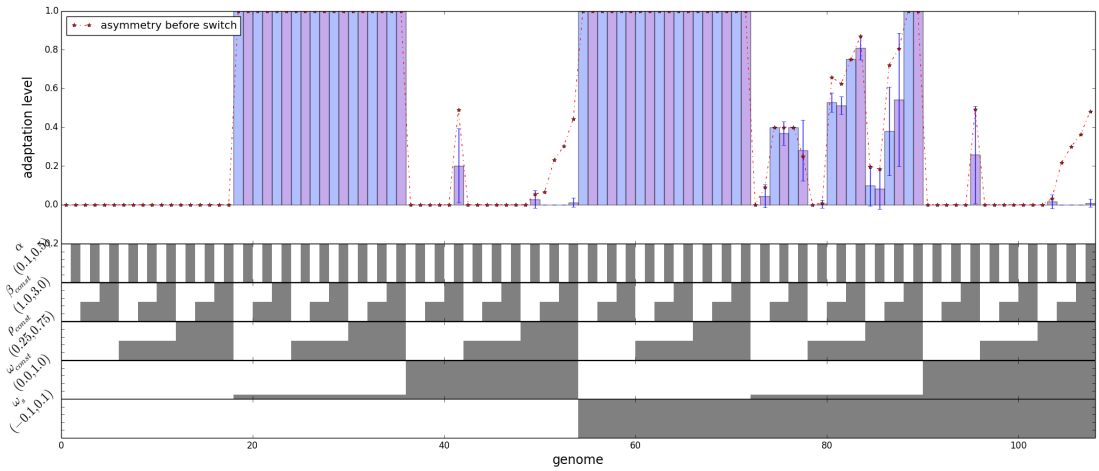



Figure 23: Adaptation level for every setting in experiment SW

off and results in structures that do not adapt after the switch. Considering the input value of $I_s = 1.0$ at the bright side, and $I_s = 0$ at the dark side of the environment, the combinations of $\omega_{\text{const}} = 0.1, \omega_s = -0.1$, $\omega_{\text{const}} = 0, \omega_s = 0.1$, and $\omega_{\text{const}} = 0.1, \omega_s = 0.1$ lead to Successin production rates of $(0, 0.1), (0.1, 0), (0.2, 0.1)$ respectively for the nodes at the bright side and the dark side. Thus, the Successin rate at the leaves is lower at the bright side for the first combination, and higher for the second and third combinations. This leads to structures that, if asymmetric, grow against the light and towards light respectively. However, in the third set of combinations, both sides get a positive amount of Successin production and consequently the settings tend to produce less asymmetric structures in most cases. The adaptivity is higher in the settings with higher values of competition rate $\beta$.

### 3.1.7. Summary

We carried out a set of analysis in order to make a better understanding of the behaviors of the parameters in the VMC. Although there are some details and special cases where the parameter space should be searched more carefully for specific behaviors, we can conclude the following overall effects for the investigated parameters:

**Production rate $\omega$:** determines the production of Successin at every leaves. It conveys information about the status of the structure (more leaves can produce more Successin), as well as information about the environment ($\omega_s$). The values of $\omega_{\text{const}} = 0$ lead to no dynamics and full symmetry. The values of $\omega_{const} > 0$ lead to asymmetry. A positive/negative value of $\omega_s$ motivates/demotivates the growth towards the high environmental inputs. The combination of $\omega_{\text{const}}$ and $\omega_s$ determines the sensitivity of the structure to the differences in the different regions of the environment.

**Transfer rate $\rho$:** determines the rate of information about the structure passing to the root. $\rho = 1$#children leads to no dynamics. $\rho < 1$#children tends towards fluctuation between the main branches - it is a result of the current growth strategy where a random node from the pool of nodes with $R > th_{\text{add}}$ is chosen to grow. The fluctuations are constrained by $R_{\text{root}}$. $\rho > 1$#children *dynamicity* is low while it is more than zero with increasing $\beta$. With higher values of $\beta$, asymmetry increases considerably

**Competition rate $\beta$:** Competition rate $\beta$ motivates sensitivity of the structure to differences

in the different parts of the structure. Large $\beta$ tends to create higher dynamics as well as higher adaptability of the structures in response to changes in the environment.

**Adaptation rate $\alpha$:** Small $\alpha$ tends towards low dynamics even in asymmetric structure. Where both $\alpha$ and $\beta$ are large, the structures tends towards high dynamics.

The above analysis and conclusions can be used as a recipe and preliminary knowledge for limiting the search space for specific behaviors and even enable one to manually set parameters in particular cases.

## 3.2 VMC guiding the growth of simulated trees

The VMC draws a separation between the physical level and the information (logical) level of the growing system and does not make many assumptions about the physical system in order to keep the generality. As a result, it can be implemented in various physical systems (either simulated or real) without a need to change the implementation of the controller. In a physical system with distributed processing units, a wise decision is to benefit from the distributed nature of the controller by embodiment of the controller in the distributed physical units. In a simulated environment, the desired implementation depends on the simulation of the physical system. In order to develop a system that facilitates the application of VMC in different simulation environments without a need of reimplementation, we developed a modular software system that separates the logical and physical sides as standalone software and allows the user to run each part on different computer in a network (see Fig. 24).

Over the last years, a high number of tree simulations were published. Among them are models of functional tree growth [27], reflecting the competition for light [7], as also the competition for water [13]. The L-system [18] represents a class of dynamic plant growth models, which are based on genetically driven branching patterns. In contrast to this model where environmental influences were built into the equations, Hart et al. [9] presented a simulation of tree growth and response where the tree growth is split into genetic influences and environmental causes. Another more recent model by Pirk et al. [17], using a graph-based description of the tree's structure, is able to simulate natural development and aging of the tree's morphogenesis.

Here, we use the VMC for guiding the growth of trees that we simulate on the Unity simulation platform [23]. The tree implementation follows a number of mechanisms in biological trees. In the future, the simulation will be further developed in different environmental conditions and with co-existence of different trees and possibly in Virtual Reality environments.

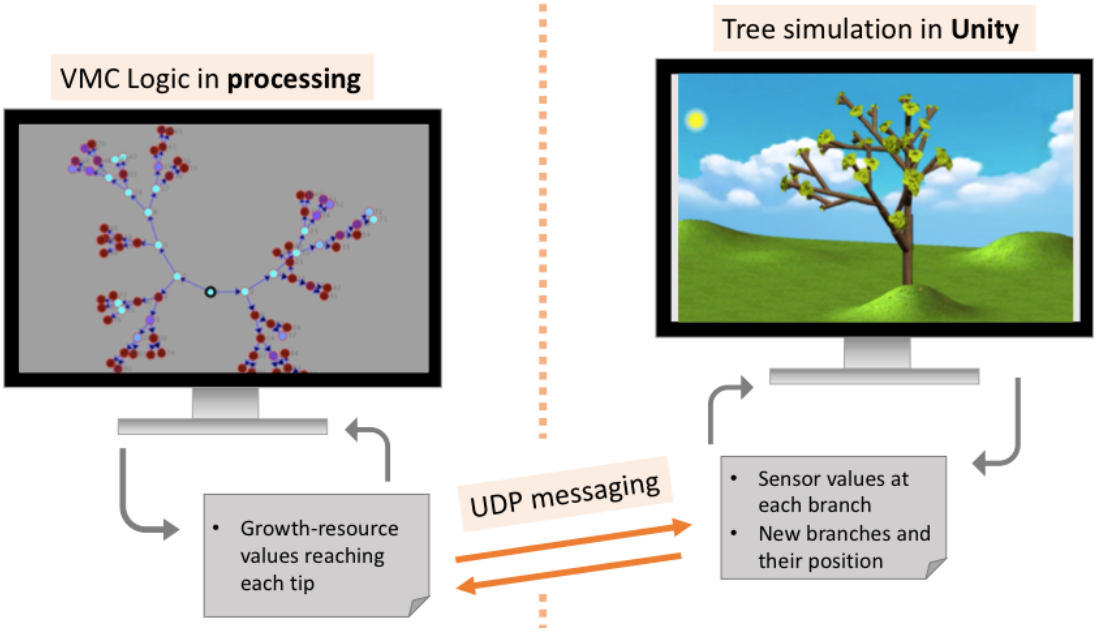



Figure 24: Standalone implementation of VMC logic communicating with a simulation environment.

### 3.2.1. Separation of VMC logics and simulation of physical structures

The implemented platform consists of three standalone software parts. Each of the three parts can be exchanged by a different version or implementation independent of each other. The complete system consists of a VMC-graph simulation (VMC logics and visualization), a tree simulation (physical-simulation of trees), a messaging software, and two files for communicating data between the different sides: a Resource-file, and a Sensor-file. The VMC-graph simulation and the physical-simulation implement the logical side and the physical side of the growing system and can be run on different computers. The resource file is written by the logical side and copied over to the other computer by the messaging software to be used by the physical simulation. The sensor file is written by the physical simulation and is copied over to the other computer by the messaging software to be used by the VMC-graph simulation. The messaging software keeps the files up-to-date at each end of the connection.

**Resource and sensor files.** Both files are CSV tables. Every row in the resource file consists of the graph ID, node ID, and resource value. Every node in the sensor file consists of the command ID, graph ID, node ID, parent's node ID, and the list of all the sensor values. The resource file is updated by the VMC-graph simulation and the sensor file is updated by the Physical-simulation.

The messaging software is coded in Python language and uses User Datagram Protocol (UDP) for sending the files over the network.

The software consists of two parallel processes, one for sending data and the other one for receiving data from the paired computers. The sending software checks regularly for new data in the CSV tables and once new data is found, the software sends the data to the paired computer.

**VMC-graph simulation** We have implemented a VMC-graph simulation in a processing framework that takes care of the logics of the VMC controller including connections between the logical nodes of the graph, transferring the flows and the update of the vessels. The nodes in the VMC-graph simulation are virtually run in parallel and communicate only with their neighboring nodes in order to keep the concept of the distributed controllers. In every time step of the experimental runs, the leaf nodes write down their resource value (R) in a CVS file along with their ID number and the ID number of the tree that they belong to it, and therefore the system works with several trees running in parallel. The receiving part of the messaging software listens for new messages and updates the data in the corresponding CSV files as soon as they arrive.

### 3.2.2. Visualization of a tree in Unity

In order to test the VMC controller in a 3-dimensional dynamic environment, we implemented a completely controllable tree simulation within the game engine Unity [23]. The implementation uses a tree mesh generation asset from Wasabimole [26] that was downloaded from the asset store in Unity. The asset generates procedural trees using a C#/.Net script and offers a variety of parameters to achieve different shapes. A branch of a tree in general consists of a set of segments with a large radius at the base segment and small radius at the tip segment. The number of segments for every branch is fixed. Every segment has a length, a radius, and an angle in relation to the previous segment. The length and radius of each segment can increase over time according to the environmental modalities and genetically encoded parameters. The angles between the segments can vary depending on environmental conditions, such as light and gravity.

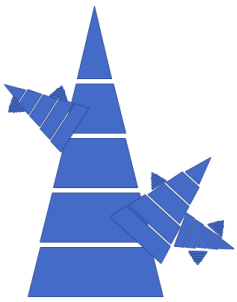


Figure 25: A schematic tree with five segments at each branch.

**Script method.** The mesh is created with a recursive function in the script that is called once in the beginning to build a ring of vertices at the branch's base (see Fig. 25). This function is recursively entered until the top of the trunk is reached. In order to add branches each exit of recursion is followed with a conditioned entrance to start a new branch recursion. In this way, the script starts at each ring of vertices at new branch either randomly, driven by environmental features, or due to other parametrized conditions.

The algorithm from Wasabimole produces random trees where the shape of the tree changes with every recomputing step. There is no control or identification system implemented, which would allow specific changes to be made or to reproduce the same shape. In order to allow a dynamic tree growth, we extend the existing algorithm with an identification system that allows to control the shape of the branch in any specific way as the user or the controlling environment directs it.

**Branch identification.** Each branch in a tree can be identified by its rooting point in the vertex ring of the parent branch and by it's parents identification. In this way each branch has an identifying trail of root segments. The only ambivalence in this system is caused by to twin branches growing on the same vertex ring and on the same branch. This ambivalence can be resolved by introducing branching angles for the identification trail. For this first model we kept the algorithm without branching angles, which is due the excluded cases of twin branches.

In our current script version, new branches can potentially grow at specific segments of a branch. The potential branches are implemented as ordinary branches that are in a "bud" state. A branch in bud state does not grow in size or get new branches. The bud-branches are made in analogy to buds in real plants which are undeveloped (embryonic) branches (see Fig. 26).

The segments that contain these bud branches are determined based on their position in the branch. There is a fixed number of segments in between every two segments that contain bud branches. In order to allow for collisions of the tree with environment or other trees, we include a mesh collider that adapts immediately when there is a change of shape or size. As soon as any other virtual object touches the tree, a collision is registered. Using the mesh colliders also prevents the interference of colliding objects.

**Leaves.** On the top of each branch there is a set of leafs. These leaf sets serve for visual representations of healthy branch tips. The leaves cast shadows on the tree branches below, which influences the resources and growing pattern similar to real plants in nature.

Figure 26: An example bud in a natural plant and in a schematic tree.

**Growth.** In each time step, the length and width of each branch segment grows. The growth of the branch diameter depends on the branch resource and follows a linear equation with a constant growth parameter and a linear resource dependency. The growth of branch lengths is a global constant.

**Implementation in the script.** To organize the tree simulation, several object classes were implemented. Each tree object is represented by a tree class. The tree objects are organized in an array which store all the tree in the current simulation. Each tree object has a collection of members. These members are built up by parameters, such as the trees random seed, branch twisting, segment length, and diameter. These are real-time accessible and can be changed at runtime in the Unity editor. Each tree object has a list of branch objects constructed from a branch class. Each branch object has members, such as the array of branching points, parent branch ID, branch length, and diameter. In order to organize the developed and evolution of the tree, the branch object also has members, such as the current resource and sensor data (light and relative gravity direction). The branch class contains methods for creating new branching points.

**Rendering a tree.** To render a tree, the software traverses the branches recursively starting from the main branch. In every branch, the segments are visited (without being rendered) starting from the base segment until the tip segment. Rendering occurs after visiting the tip segment and during the unwinding of the recursion loop. During that, once a segment with a connection to another branch is visited, the new branch is traversed and rendered. Then the unwinding and rendering continues in the main branch towards its base.

**Phototropism.** The shape of a branch is influenced by the direction of light during the growth of the branch segments in order to simulate phototropism in biological plants. Phototropism is the attraction of the branch tips towards the direction of light (see Fig. 27).

Here, phototropism is implemented by measuring the light intensity on each branch tip. The light intensity is calculated by drawing an artificial "light cylinder" from the sun position to the branch tip position. This control script counts the objects that collided with the light cylinder that is in between the branch tip and the sun position. The default value of light is then decreased by a function that is proportional to the number of objects that collided with the light cylinder. The local amplitude of light at each tip influences the resources flowing to that branch. In each

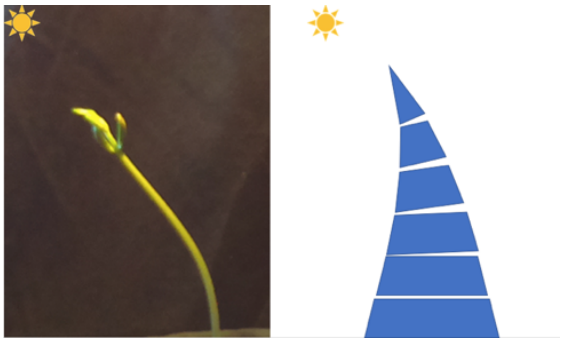


Figure 27: An example of phototropism in a natural plant and in a schematic tree.

time step the light cylinder is iterated through all the branch tips of each tree. Apart from that the branch bends also towards the light direction. This effect is achieved by adapting the angles between each branching segment. The resulting angle between each branch segment is a sum of light direction and a noise component.

**Association of VMC nodes to the tree.** Control of the branching (whether a bud should be activated and become a branch or not) is controlled by VMC which is running separately as described before at section 3.2.1. A VMC node of a VMC graph is associated to every bud and to every branching segment of a tree (see Fig. 28). The sensor values of the buds and the branches are communicated to their associated nodes. The Successin is produced, the vessels are updated, and the resource from the root is distributed between the buds and branches. Once the resource at a bud is above a predefined threshold, the bud is activated and grows into a new branch.

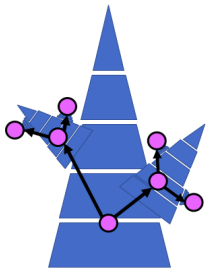


Figure 28: A schematic association of VMC nodes to the simulated tree.

**Example trees grown in Unity.** Fig. 29 shows an example of a tree simulated in Unity and its associated VMC graph. The parameters are set such that the growth of the tree is influenced by light and gravity with preference of light with a high effect and growing against gravity with a lower effect comparing to light. Parts of the tree cast shadows on other parts. The tree in Fig. 29 has spread its branches while growing towards the light. The VMC graph shows the nodes where each internal node holds four children representing four initial buds of each branch. The internal nodes represent the branches and the leaves represents the buds that are not yet activated (so not yet turned into branches).

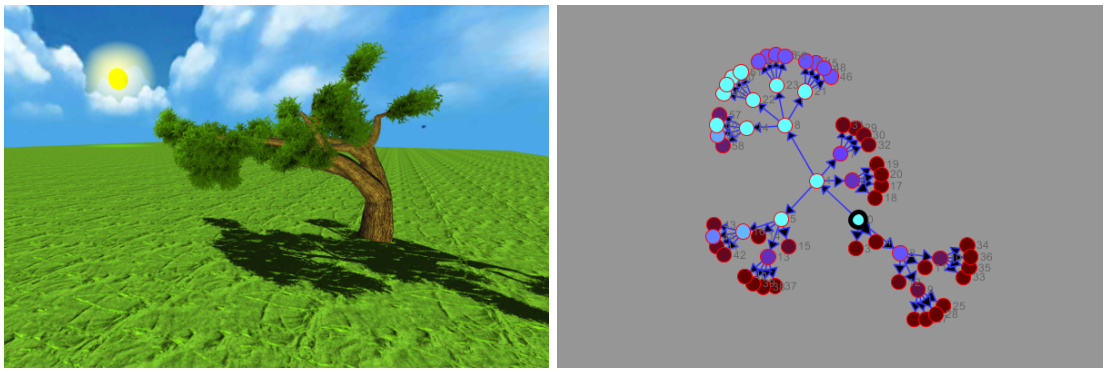


Figure 29: A tree simulated in Unity (left) and its associated VMC controlling the growth of its branches (right). The VMC node with a thick outline represents the root of the graph. The colors represent the resource value at the nodes. The brighter the color is, the higher is the amount of the resource.

Fig. 30 shows an example of four trees with the same parameterization growing in parallel in the same environment. The light source (not shown) is fixed and positioned on top of the environment. The difference in shape of the trees is due to a different relative position towards the light source, some randomness in branch bending in order to increase the variety of possible shapes developed by an identical parameter set, and the shadows cast by the trees on the trees.

Figure 30: A set of trees with identical parameters influenced by the light and their shadows.

## 3.3 Embodied system based on VMC

An essential engineering challenge in *flora robotica* is to 'grow' artificial structures as scaffolds in sync with natural plants. Ultimately we want to solve that with growing braided structures (growing here means autonomously braided). In a preliminary study we investigate here how an embodied system can be controlled using VMC. In order to have a high degree of flexibility and to simplify the setting, we use the Kilobots as a mere emulation of modular braided structures.

Self-assembly has many applications, such as the autonomous construction of architectural artifacts (e.g., by the means of re-braiding of filaments in braided scaffolds). Using methods of swarm robotics [8], self-assembly has been shown before [21] where Kilobots [20] were used to form predefined shapes. Remaining challenges are, how to self-assemble robots adaptively to their environment and how to make the self-assembly robust to damages enabling self-repair and regrowth adaptively reacting to changes in the environment.

The underlying vision is to grow architectural artifacts and braided scaffolds autonomously and adaptively to the plant growth and other environmental features. Here, we built on our previous work reported in D2.2, which focused on an adaptive seed selection mechanism as a collective decision-making about where to start the self-assembly process. We use an adapted version of our VMC algorithm to create a tree network in a swarm of Kilobots, in relation to light conditions in the environment. The VMC algorithm, once implemented on the robots, gives us adaptivity and self-repair almost for free. The future work will be focused on robotic nodes in the project that guide the growth of 3D scaffolds adaptive to environmental features and natural plants.

### 3.3.1. Implementation on Kilobots

Fig. 31 gives a schematic example tree of physical robotic agents and the virtual underlying VMC tree of the current implementation. Every agent in the tree contains an internal node with a leaf and the possibility of adopting the node of two other agents as its children (except the root that can adopt three children). Each agent maintains the weight of the edge ($V$) between its node and the parent agent, computes its Successin ($S$) to be transfered to the parent agent, calculates the total sum of the edges of its children ($V_{CH}$) to be communicated to them, and computes its own Resource value ($R$) based on the parent's Resource value ($R_p$) and the total edges received from the parent agent ($V_{CH,p}$). The only sensor values ($I_s$) used here are the light sensor of the Kilobots.

We perform physical experiments with 70 Kilobots. While we keep a root robot fixed in the middle of the arena, all other robots move randomly. The robots' controller is iterating an infinite loop interrupted by messaging (see Alg. 1).

### 3.3.2. Experiment setup

To verify the algorithm, we designed an experiment with a dynamic environment. The light distribution is similar to the "half-bright" environment introduced in Sec. 3.1.6, is changed over a certain time period in the middle of the experiment, and we monitor its effect on the tree structures. The robots are expected to stay adaptive (re-) build tree structures in the bright area.

An experiment lasts ten minutes (600 s). After 200 s into the experiment, we switch the dark and bright sides by projecting a video that moves the bright area to the other side in a slow transition (200 s $\leq t \leq$ 400 s). For the final 200 s the light conditions then remain static. Furthermore, we verify the swarm against a local change in the environment, by putting a dark narrow bar to the bright side to see if the tree can repair itself.

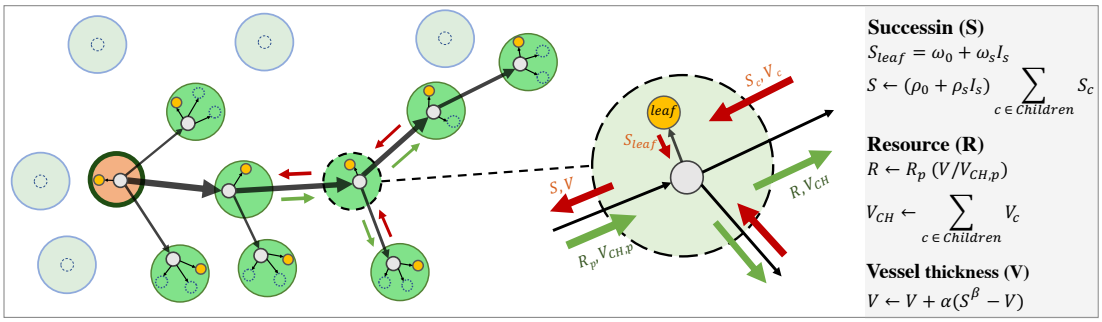



Figure 31: A schematic tree structure of robots and their virtual underlying VMC tree (left), and the equations of dynamics in each agent (right). A seed robot contains the root of the VMC tree. The yellow small circles represent the leaves of the tree and the gray small circles represent the internal nodes. The dotted small circles represent the possible adoptable children of the robot.

**Algorithm 1:** Pseudocode of our robot controller

```
while true do
    measure light;
    send message;
    if in walking state then
        move randomly;
        if message received then
            if from tree and confirmed then
                stop;
                join tree;
            else if from walking or saturated agent then
                turn;
            end
        end
    else
        // in tree
        calculate parameters;
        if dry then
            leave tree;
            reset;
        end
        if message received then
            if from potential child and not saturated then
                add child to children list;
            else if from child or parent then
                update parameters;
            end
        end
    end
end
```

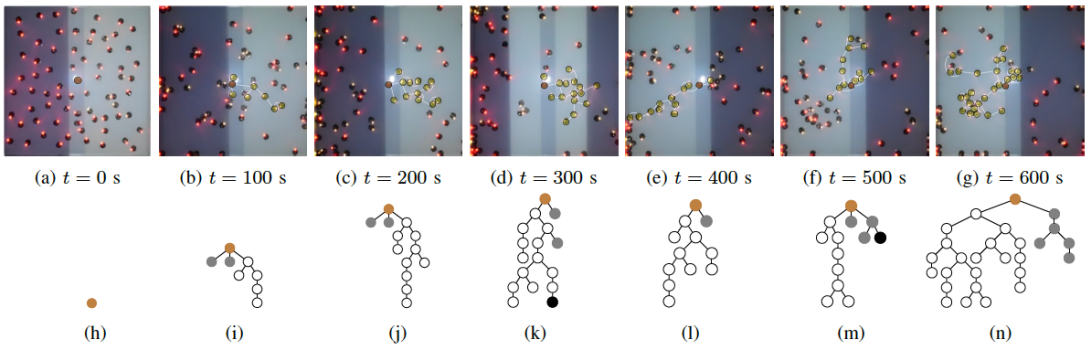



Figure 32: Photos of robot configurations from above over time and schematic drawings of formed trees. The tree structures are highlighted on the images and given as graphs beneath the corresponding pictures. The brown node is the root and the nodes shown in gray, black, and white refer to the robots in the gray, dark, and bright areas respectively. Having more white than black nodes indicate that the robots properly adapt to the dynamic light conditions, hence build, decompose, and rebuild the tree to grow mainly in the bright area.

### 3.3.3. Results

We change the color-space of the video frames to gray scale and apply the Gaussian blur filter. Filtering out the background, we have the images for analyzing the swarm. The difference between a frame and its successive indicates the motions in the experiment. We use these differences to reveal the motion pattern among the agents. By monitoring these motions throughout the experiment, we can see when and to which extent the robots moved. Fig. 33 and Fig. 34 demonstrate this motion for one and all experiments. The gray area separating bright and dark sides is excluded from the pictures.

The median values of the last 30 frames changes for one (Fig. 33) and eight experiments (Fig. 34) are shown besides the upper and lower quartiles. Hence, Fig. 33 plots the raw, scattered points (every fifth value).

The movement among the agents in the bright side decreases over time (represented by red color) as the robots join the tree. The free space around the robots for random movement decreases as well since they cluster around the corners or around the wall, but not as much as the ones forming the tree structure. In a time window of 200 seconds, depicted with a green bar in both Figures (Fig. 33 and Fig. 34) the light slowly shifts towards the other side, making the formerly bright side darker, and the dark side gradually brighter. On $t = 200$ the tree starts to dis-assemble and the robots leave in random motion and diffuse around the arena. Therefore, the movement captured on both sides (in general) increases starting from the 200th second. From the middle of the experiments, the agents at the formerly dark side, now in the bright side start to form a tree and the tree grows which can be inferred from the fall of the blue line.

Layering the frames of all experiment offers a prospective on where the robots spent most of their times (see Fig. 35). Big clusters were forming around the corners and walls and thick dark boundaries of the heatmap demonstrate this effect. Kilobots can not negotiate the obstacles. Some of the proposals in the literature are using beacons [24], a sophisticated arena [2], or a certain light distribution around the walls that could make the robots aware of the walls. We contented to the random motion which to some extent reduces the formation of big clusters around the corners and walls but does not prevent it. Hence, the heatmap reveals that often

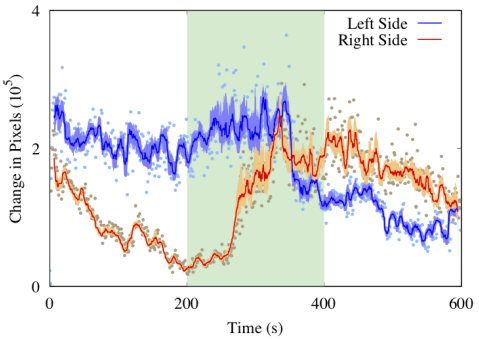



Figure 33: Detected motion (in pixels) during a single experiment, every fifth value is plotted. The red line shows the median motion on the right side and the blue line shows the median motion from the left side of the arena. The area between 200 s to 400 s is the light shifting interval.

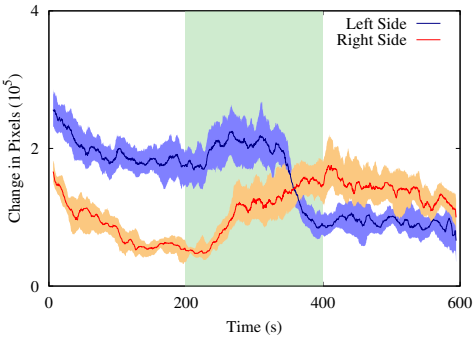



Figure 34: Median pixel changes of all experiments over time with upper and lower quartiles.

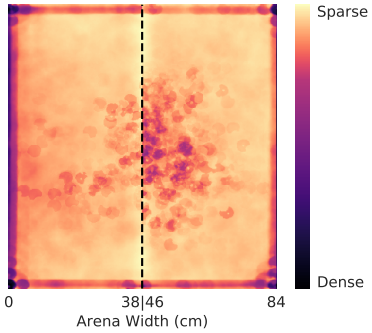



Figure 35: Heatmap of the robot distribution in the arena, robots spend a lot of time at the seed and at the boundaries (a common issue with Kilobots

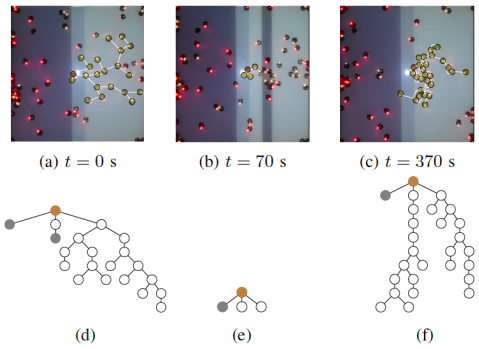



Figure 36: A dark line was projected on the tree structure ($t = 0$ s). The robots under the dark line left the tree and as a result, all of their successors dis-assembled (after 70 s). By removing the dark bar, the tree repaired itself and rebuilt the structure in 300 s.

the side which was bright first contains larger tree structures as seen by higher density in the right half of Fig. 35 (also noticeable with the deeper valley in the left half of Fig. 34). Once the light switches that many robots at the formerly dark side (now bright) are at the boundaries so a large tree mainly will not emerge (sparser in the left half of Fig. 35). Images of higher resolution and videos of all experiments are available online[1]. Fig. 32 consists of pictures taken every 100 s from one of our experiments. The tree starts to grow in the right side until the light distribution started to change. The tree started to decompose where it was getting darker and rebuilt the structure in the brighter area continuously and grew to the other side. The

[1] https://doi.org/10.5281/zenodo.1186516

corresponding schematics under the pictures help to see the dynamic tree structure. The white nodes represent parts of the tree in the bright zone and they outnumbered the other nodes in the graph which indicates that despite the dynamic light conditions, the robots continuously built the structure and adapted themselves.

Fig. 36 demonstrates the result of our verification for robots capability in reacting to sudden local damage to the structure. The swarm was able to self-repair the assembly against the sudden local damage and rebuilt the structure.

### 3.3.4. Summary

This emulation of growing braided scaffolds using VMC to control the aggregation process of mobile robots shows the potential of this approach. First, we have successfully implemented VMC on simple and limited hardware. Second, we have shown that this VMC-based system is immediately able to grow tree-like structures that are adaptive to the environment. The system is robust to changing light conditions and even to cutting parts of the structure. This study prepares us well for the future work on autonomously growing (modular) braided structures.

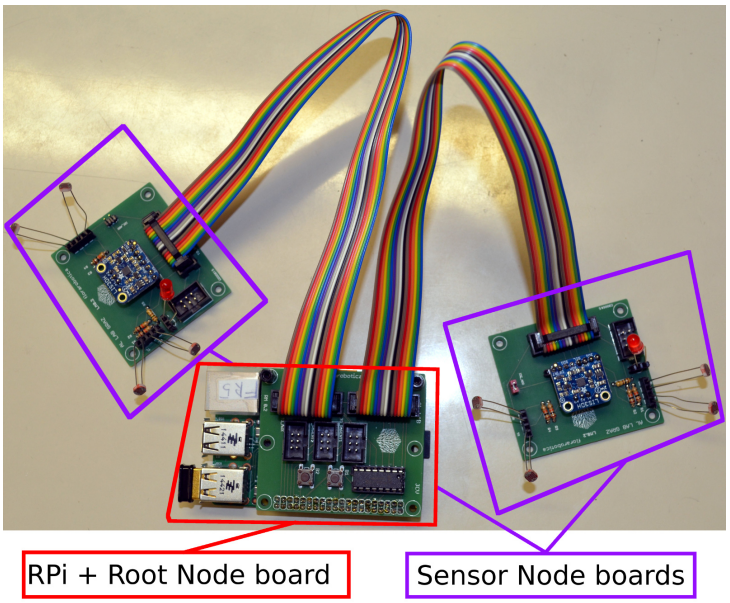



Figure 37: A single RPN-module. Marked in red is the Raspberry Pi (here a model 2 with WiFi dongle attached) with the Root-Node board stacked to the GPIO-header (GPIO: general-purpose input/output). In purple, the two Sensor-Node boards (SNs, the 'leaves' of the module) are highlighted. Note that on the right SN, the jumper bridge to select the second $I^2C$ address is in place.

## 3.4 Growing physical structures: VMC embedded in distributed hardware

To leverage the major benefit of a distributed algorithm, the computations need to be distributed. This way the system scales well. To this end, modules based on Raspberry Pi (RPi) that can each connect to four other modules were designed and assembled. A fifth interface is reserved to connect to different devices, for example, the plant-guiding robot that was developed (see D1.2). We call the system *RasPi-Net* (or *RPN* in short). In Fig. 37 a full RPN-module with its submodules is shown. One is the *Root-Node* (RN) add-on board (see Fig. 38) that is stacked onto the RPi's header, the other two are the *Sensor-Node* (SN) boards (see Fig. 39). The structural basis is a Y-shaped braided module that can be deformed in various ways (see Fig. 40). Similar to the modules that build up a plant shoot (*phytomers* [3]), an RPN-module has an orientation. The RPi with the RN board is placed at the bottom. It houses plugs to connect three parent RPN-modules, two buttons, an ADC to use eight analog sensors and relays all the lines from the sensor-node boards to the RPi-header (or the ADC). If unconnected, it represents a *root*-node of the VMC-graph. The sensor-node boards (if unconnected) represent the leaves of the VMC-graph. Each of them has a single plug to which a child RPN-module can be attached (by connecting the plug to the parent plug on the RPi add-on board of the child module). In addition, each sensor-node is equipped with a triple-axis accelerometer[2], four photoresistors[3], and an LED to signal a leaf's amount of resource, which correlates positively with its 'willingness' to grow, that is, attach to another RPN-module.

[2] Adafruit LIS3DH: https://www.adafruit.com/product/2809

[3] Sparkfun CdS photoconductive cells SEN-09088: https://dlnmh9ip6v2uc.cloudfront.net/datasheets/Sensors/LightImaging/SEN-09088.pdf

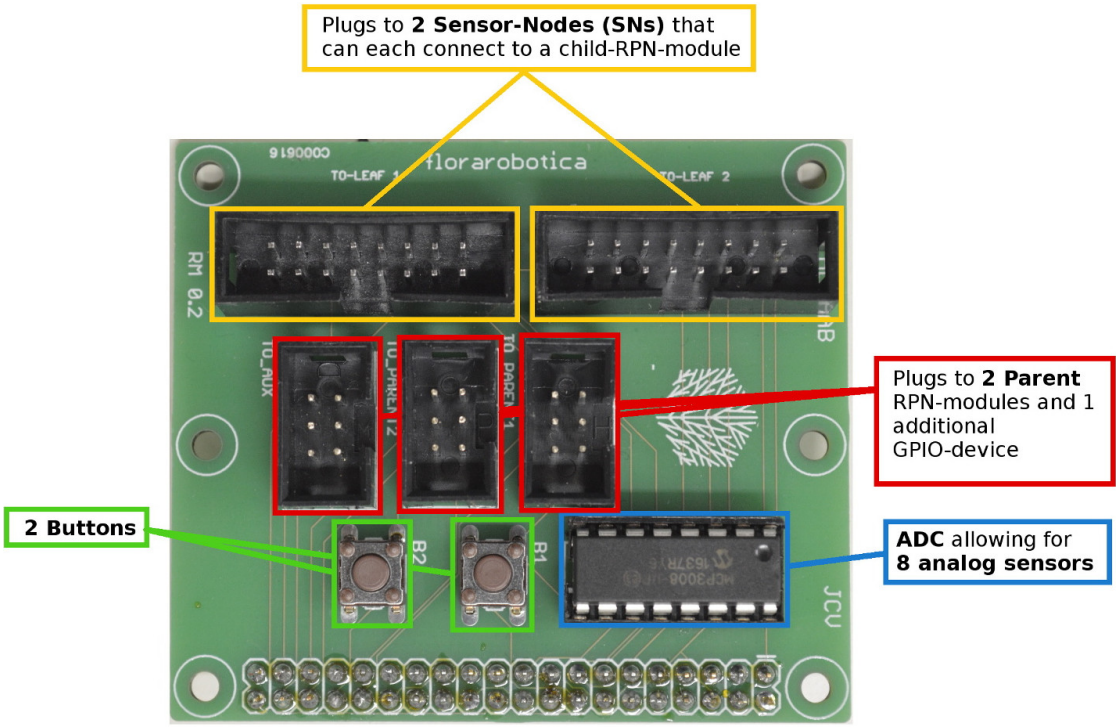



Figure 38: Assembled Root-Node board.

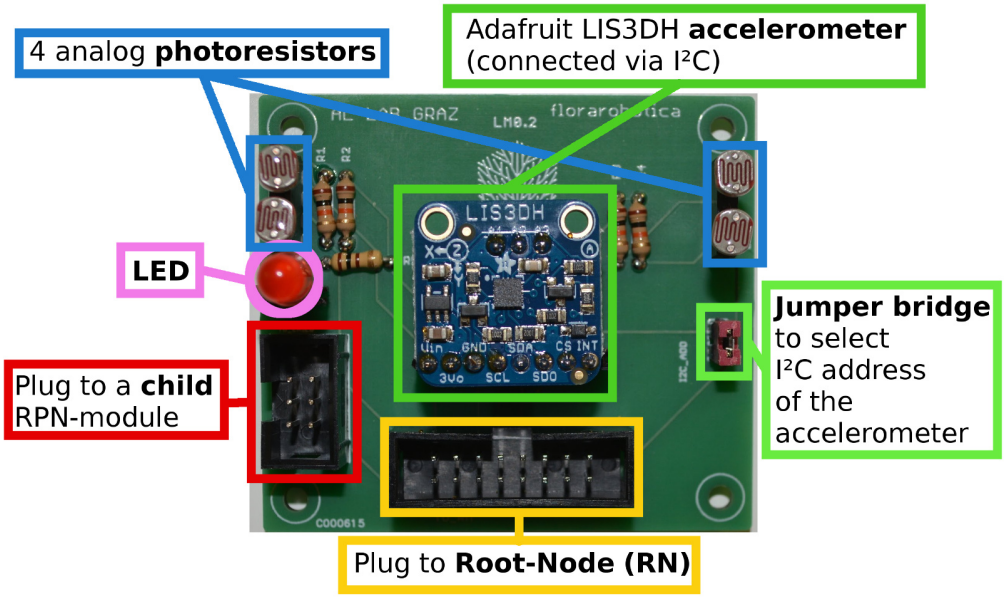



Figure 39: Assembled Sensor-Node board.

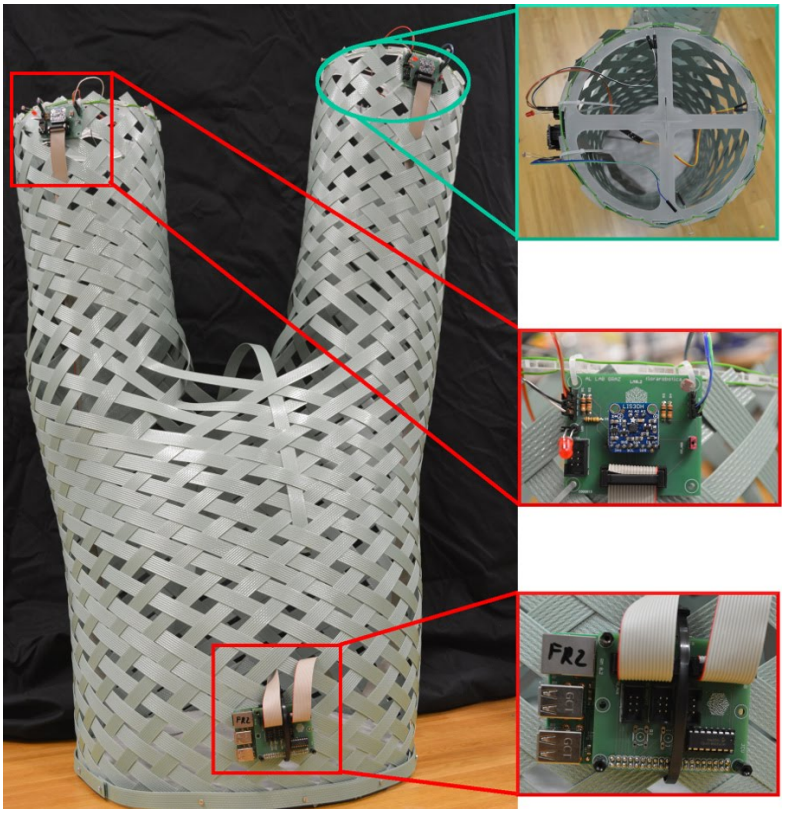



Figure 40: Y-shaped braided module with RPN-module attached.

### 3.4.1. Local communication

To allow neighboring RPis to exchange values, we use GPIO pins with the *gpiozero*[4] library for Python[5]. Three wires connect two Raspberry Pis: one for receiving, one for transmitting, and ground. This requires two GPIO pins per connection on each RPi. A transmitter pin on one RPi is connected to a receiver pin on the other RPi and vice-a-versa. This way, RPis can safely be plugged together and unplugged during runtime, with changes being noticed on all involved RPis.

The sender pin is configured as a *gpiozero.PWMOutputDevice*[6] that performs pulse-width modulation. We set the initial duty-cycle to 0.2 (at a frequency of 100 Hz), such that a receiver is guaranteed to detect a connected neighbor at any time, even when the VMC-node sends the value 0.0. The duty cycle thus varies in the interval $[0.2, 1.0]$, transmitting values that are scaled to the range $[0.0, 0.8]$ and added to the basic duty cycle.

The receiving pin is set up as a *gpiozero.SmoothedInputDevice*[7] that is running a queue in the background that continuously polls (and stores) the pin's value. A single value can either be 0 or 1, depending where in the PWM duty cycle we measure. To deduce the sender's value reliably (accuracy: $\pm 0.01$), we use a large queue length of 5000, that is, the last 5000 values are averaged every time we read the receiver pin. The receiver's threshold is set to 0.1, above which value the device is considered *active*, indicating that a connection to another Raspberry Pi is established and alive. Since the sender's basic PWM duty cycle is 0.2, this is a robust approach and reciprocal detection is guaranteed.

### 3.4.2. VMC implementation

Following an OOP-approach, for each of the five interfaces to neighboring RPN-modules, we implement a class *Neighbor* that includes all the methods and attributes necessary for local communication between two RPN-modules. For interfaces to children, we derive a subclass *Child*, that contains the necessary interfaces to all the sensors attached to the leaf nodes. The Child class also contains the methods to either generate (if a sensor node is a leaf) or transfer Successin to a parent module (if a sensor node is connected to a child module and thus acts as an intermediate node, as well as the methods to adjust vessel thickness and for receiving resource from a parent. The *Parent* class, which is also derived from the Neighbor class, is simpler, as it does not contain any additional sensors. It has a method to receive resources from (up to three) connected parent RPN-modules, or, if unconnected, generate it. It also has a method for distributing the Successin coming from the leaves or children to connected parent RPN-modules. A parent that delivers more resource, also gets a greater share of the Successin.

The parameters of the VMC and the configuration of experiments are stored in two config-files, an immutable one containing the VMC-constants (i.e., the 'genome' of the syntheticophyte[8]) and another one that stores the current state of a module and all its interfaces. The latter one allows Pis to resume from where they stopped after a system-crash or manual restart or even the replacement of a module.

[4] https://github.com/RPi-Distro/python-gpiozero. At the time of writing, the currently released version (1.4.0) contains a crucial bug in the gpiozero.SmoothedInputDevice, where the median of queued values is taken instead of the arithmetic mean (this makes sense for jittery sensors, but not in our case). We have fixed this for ourselves and raised the issue on GitHub. It has now been fixed by exposing the choice to the user, but one has to clone the most recent version from GitHub and manually install it. For compatibility reasons, this is the version we now use in this work.

[5] https://www.python.org (version 2.7.14)

[6] https://gpiozero.readthedocs.io/en/stable/api_output.html#pwmoutputdevice

[7] https://gpiozero.readthedocs.io/en/stable/api_input.html#smoothedinputdevice

[8] coinage for 'artificial plant'

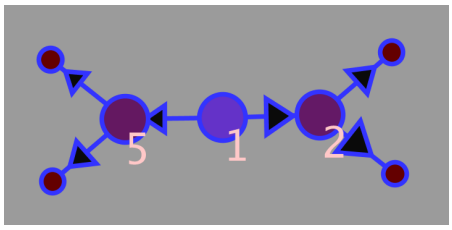



Figure 41: Visualization of the VMC-graph of a system of three RPN-modules.

The main script is run as a background process managed by *systemd*[9]. Upon inception, it first parses the two configuration files and initializes the *Parent*- and *Child*-interfaces with those values. It then enters an infinite loop, during each iteration of which first the VMC-algorithm is worked through. The chronology of events:

1. Receive resource from all connected parents or else generate resource.
2. Receive Successin from all children (leaves or child-modules).
3. Adjust vessel thickness according to the amount of Successin received from each child.
4. Distribute the resources (received in step 1) to the children according to their vessel thickness.
5. Distribute the Successin to the parents according to the amount of resource received from each (in step 1).

After this is done, all the sensors are each read for themselves. This is only done for sending them to the visualization (see below) and for debugging reasons. Finally, all information from a single iteration is stored as a row in a dedicated CSV-file, the mutable configuration file is updated and the information is published to the WiFi (also see below).

To provoke asynchronicity between all the RPN-modules, the main-script then waits a random period of time (within an interval that can be set in the immutable configuration file) before the next iteration begins.

### 3.4.3. Wireless communication and visualization

Each Raspberry Pi continually publishes its state to the local WiFi network using ZeroMQ[10] *publisher* sockets. A PC in this network runs a *subscriber* socket (the publisher's counterpart) that listens for a given interval (twice the maximal interval that can pass between two iterations of the main loop on the RPN-modules themselves, this information can be parsed from the configuration files which are also available on the PC), receives the information from all RPN-modules and aggregates it in an array (if more than one message is received from an RPN-module, only the latest message is processed). Since each RPN-module has a unique hostname, but does not know to which other RPN-modules it is connected, we currently keep a manually maintained CSV file in the repository that contains each module's parent ID[11]. The respective parent IDs are also stored in the array, which is then not only stored locally on the PC with a time stamp,

[9] *systemd* is a system and service manager for Linux operating systems

[10] `http://zeromq.org` (asynchronous messaging library)

[11] In the future, this will be autonomously handled by the RPN-modules themselves, either by adding a third line of communication between them or by switching to a different communication protocol.

but is read in by another script written in *Processing*[12] that visualizes the whole VMC-graph at any given time. In the case that this script runs on yet another machine, we have code in place that can send the CSV via UDP sockets.

[12] https://processing.org/

# 4 Evolving robotic control of plant stimuli

In the *flora robotica* system we want to control the direction of plant growth with a distributed autonomous system. We have developed our dedicated hardware for this purpose, the so-called robotic nodes. In order to efficiently steer plant growth, we need a sufficient understanding of the whole plant and in particular the plant's tropisms (here, predominantly phototropism). We get this understanding by mostly data-driven models and as described in the following by a high-tech machine learning approach using the most recent methodology (deep learning, data augmentation, etc.). We train an LSTM network, that is a network composed of so-called Long short-term memory (LSTM) units; basically a recurrent neural network (RNN). LSTM networks are particularly useful to predict time series. Based on these holistic plant growth and motion models using LSTM networks, we evolve robot controllers for our robotic nodes using evolutionary computation and regular Artificial Neural Networks (ANN). The work-flow is complex:

- ▷ preliminary experiments with real plants to gather data,
- ▷ digest data by image processing to obtain appropriate time series of plant growth and motion,
- ▷ feed the data to our machine learning technique and train the LSTM network,
- ▷ use the successfully trained LSTM network to simulate plant behavior and evolve robot controllers using methods of evolutionary robotics [16],
- ▷ and test the evolved controller in a real plant experiment to check for the reality gap.

We have mastered this complex work-flow and report successful experiments of controlling a plant's motion, which is then integrated by the plant into a stiffened stem of desired shape, that is the plant's embodied memory of our robot's actions.

## 4.1 Model setup

### 4.1.1. Dataset experiments

The plant model is derived from dataset experiments with real plants, in a bio-hybrid setup. These include six repetitions with a simplistic, non-reactive controller, following [25], and three repetitions with a closed-loop adaptive controller, following [11]. The open-loop controller switches between two light sources in regular time periods of six hours for a total duration of 83.3 hours. The closed-loop controller switches light sources according to the plant's tip location, following the approach of [11]. In each experiment the plant is photographed every five minutes. The plants show equal influence by growth and motion, with substantial motion in the horizontal direction. They also show variance between the behaviors of individual plants. Observing variance in plant experiments is a well-known phenomenon in plant science, which requires high numbers of repetitions. However, in the context of this research, where the focus is on evolutionary computation and robotics, such high overheads for experiments are infeasible. Instead we test our approach based on an engineering perspective by testing whether the model, that results from these experiments, helps us to successfully control a plant. We also test if the evolved controllers are able to perform properly with such dynamic and unexpected plant behavior.

#### 4.1.2. Stem motion tracking

We describe our computer vision method for stem motion tracking. The 10-point description of stem geometry forms the basis of training data for our LSTM-based *Stem stiffening and motion model*. We process images from the dataset experiments described above, to record a 10-point $xy$ description of the full stem at each timestep, representing its phototropic motion and stiffening dynamics. The images are sampled[13] at 1/8 resolution and processed according to the following method, both for the dataset experiments described above, and for the reality gap experiments detailed in Sec. 4.3.2. Before processing the dataset experiment images, a set of images is compiled showing the setup without a plant. The setup images include all states of the controller and any slight variations in lighting conditions. The set of images is sampled[13], isolating the green RGB channel value at each pixel position $(i, j)$ and remapping it onto the domain $[0, 1]$, forming sequence $\Lambda$ containing a matrix $M$ of green values for each image. To represent the interval of possible green channel values present in the setup, matrices $L$ and $H$ are constructed by

$$L_{i,j} = \min_{M \in \Lambda}(M_{i,j}), \quad H_{i,j} = \max_{M \in \Lambda}(M_{i,j}). \tag{12}$$

After constructing the setup matrices, dataset experiment images of plants are processed. The green channel value is isolated for each pixel $(i, j)$, remapped to the domain $[0, 1]$, and saved into the matrix $R$. Pixels within a window are identified as containing plant material if: $(R_{i,j} < L_{i,j} - \theta_1) \vee (R_{i,j} > H_{i,j} + \theta_1)$, for threshold $\theta_1 = 0.2$. Each identified plant pixel is extracted to set $P$, and their $(x_p, y_p)$ coordinate positions are used to identify two possible locations of the plant's tip. In order to locate the growth tip $g = (x^g, y^g)$, plant pixels are compared to the globally defined anchor $a = (x^a, y^a)$, representing the position where the plant stem emerges from the soil. Two possible $xy$ growth tip positions are identified (corner point $c$ and high point $h$) and one selected as $g$ based on the prior timestep, such that

$$c = \underset{(x_p, y_p) \in P}{\arg\max} |x^a - x_p| + |y^a - y_p|, \tag{13}$$

$$h = \underset{(x_p, y_p) \in P}{\arg\max} |y^a - y_p|, \tag{14}$$

$$g_n = \begin{cases} h, & \text{if } d(g_{n-1}, h) < d(g_{n-1}, c) \\ c, & \text{otherwise} \end{cases}. \tag{15}$$

The remaining intermediate points $((x^{S_2}, y^{S_2}), \cdots, (x^{S_9}, y^{S_9}))$ describing the stem are preliminarily identified from set $P$, and then smoothed while preserving the representation of stiffening dynamics. For these eight points, the $y^i$ are distributed evenly between the tip and anchor, as

$$y^{S_i} = \frac{i}{9}|y^a - y^g| + y^a, \tag{16}$$

and $x^{S_i}$ are set as the averaged $x_p$ for pixels in set $P$ that have $y_p$ within threshold $\theta_2$ of the respective $y^{S_i}$, such that

$$x^{S_i} = \overline{x_p} : \forall x_p \in P : |y_p - y^{S_i}| < \theta_2, \tag{17}$$

where $\theta_2 = 30$ pixels. In this way, a 10-point description $S$ of the full stem $S = (a, (x^{S_2}, y^{S_2}), \cdots, (x^{S_9}, y^{S_9}), g)$ is defined. Due to minor variations in images caused by shadows and light reflections on the stem, this 10-point detection contains some errors. We

[13]The sampling was duplicated in two separate programming platforms. One is standard Python, utilizing the OpenCV library for computer vision. The other is Iron-Python and native libraries of the VPL Grasshopper pertaining to computer vision.

address these errors using a simple algorithm based on Smoothing via Iterative Averaging (SIA) [14], which preserves the key topological features of the curve being smoothed. For each point in $S_n$, our algorithm utilizes the equation

$$(x^{S_i}, y^{S_i}) = \left(\frac{1}{2}(x^{i-1} + x^{i+1}), \frac{1}{2}(y^{i-1} + y^{i+1})\right) \tag{18}$$

iteratively, according to the following steps: 1) for $i = 2, 4, 6, 8$, apply equation 18, 2) for $i = 3, 5, 7, 9$, apply equation 18, 3) for $i = 2, 4, 6, 8$, apply equation 18. In this way, the intermediate stem points are smoothed with the SIA-based process, while the tip and anchor remain unchanged. The newly smoothed sequences $S$ are converted to cm and scaled to match physical setup dimensions. The anchors are then unified to standardize the data, by translating $(x^{S_i}, y^{S_i})$ points in $S_n$ according to $((x^{S_i}, y^{S_i}) + (a - \overline{a} : \forall a \in S_n))$. The resulting data is reformatted to sequence $\Psi$ of 18-dimensional vectors $\psi_j = (x_j^{S_2}, y_j^{S_2}, \cdots, x_j^{S_9}, y_j^{S_9}, x_j^g, y_j^g)$, without the now redundant anchor values. These vectors are the basis for regression data for our LSTM-based *Stem stiffening and motion model*, described below.

#### 4.1.3. LSTM trained as stem stiffening and motion model

Building a holistic model of plant stem dynamics is a complex task [4] that would benefit from deep learning. However there is a lack of existing data, and the substantial overhead associated with plant experiments makes it infeasible to obtain large amounts of new data (many plants can be grown in parallel, but controlled light conditions, monitoring, and tracking are costly). Therefore, having obtained a small amount of data from real plant experiments – described above – we develop a method to artificially expand that data, avoiding overfitting when training the LSTM.

**Preparation of stem data for regression.** After manually removing data in $xy$-areas that are too sparsely populated to provide reliable data (mostly in zones far from the origin, where only one plant of nine may have reached by coincidence), we process the experiment motion tracking data in two ways to expand the set. Firstly we add noise, to reduce the tendency of overfitting, and secondly we add a generic model, such that the typical plant behavior is dominantly represented in the data for the LSTM.

In order to add normal distribution noise, in addition to the experiment data in sequence $\Psi$, we define noisy data in sequences $(\Psi_1, \cdots, \Psi_n)$, where $\psi_{n_j} = ((x_j^{S_2})^{\Psi_n}, (y_j^{S_2})^{\Psi_n}, \cdots, (x_j^{S_9})^{\Psi_n}, (y_j^{S_9})^{\Psi_n}, (x_j^g)^{\Psi_n}, (y_j^g)^{\Psi_n})$. The noise values applied to each growth tip $(x_j^g, y_j^g)$ in $\Psi$ are computed according to the mean $\mu$ and standard deviation $\sigma$ of a finite quantity $(\theta_3)$ of the closest growth tips $(x_i^g, y_i^g)$ that have the same light condition $b$. To calculate this, for each growth tip $(x_j^g, y_j^g)$, all other growth tips in $\Psi$ sharing the same light condition $b$ are placed into sequence $\mathbf{dist}^j$, and are then sorted according to their Euclidean distance from the respective tip at $m = j$. The $\theta_3$ closest tips for each respective tip are defined as $W_\Psi$, such that

$$w_j = (x_i^g, y_i^g) \in \Psi \quad | \quad n \leq \theta_3 \in \mathbf{dist}_n^j, \tag{19}$$

where $\theta_3 = 100$. The mean $\mu$ for noise is calculated as

$$\mu(x_j^g) = \frac{1}{|W_\Psi|} \sum_{j=1}^{|W_\Psi|} w_j x_i^g, \tag{20}$$

with a symmetrical equation for $\mu(y_j^g)$, and standard deviation $\sigma$ as

$$\sigma^2(x_j^g) = \frac{1}{|W_\Psi|} \sum_{j=1}^{|W_\Psi|} w_j \left(x_i^g - \mu(x_j^g)^2\right), \tag{21}$$

with a symmetrical equation for $\sigma^2(y_j^g)$. The noisy data in each new sequence $\Psi_n$ is calculated by first defining the noisy growth tips and then defining the noisy intermediate points in relation to the tip output. The new noisy tips $((x_j^g)^{\Psi_n}, (y_j^g)^{\Psi_n})$ in $\Psi_n$ are calculated using normal distribution noise, according to the existing tips $(x_j^g, y_j^g)$, and $\mu$ and $\sigma$ values scaled by factor $\omega$ such that

$$(x_j^g)^{\Psi_n} = x_j^g + \mathcal{N}\left(\mu(x_j^g), \sigma(x_j^g)\omega\right), \tag{22}$$

with a symmetrical equation for $(y_j^g)^{\Psi_n}$, where scaling factor $\omega = 0.1$. The noisy intermediate points $((x_j^{S_i})^{\Psi_n}, (y_j^{S_i})^{\Psi_n})$ in $\Psi_n$, are calculated according to the noisy tips $((x_j^g)^{\Psi_n}, (y_j^g)^{\Psi_n})$, and are scaled by the values $\omega_2(x^{S_i})$, $\omega_2(y^{S_i})$. The noise values are generated through an artificial mean $\mu_2$ and standard deviation $\sigma_2$, defined according to the calculated standard deviation and the generated change in position of the noisy growth tips, such that

$$\mu_2\left((x_j^{S_i})^{\Psi_n}\right) = x_j^{S_i} + \left((x_j^g)^{\Psi_n} - x_j^g\right)\omega_2(x^{S_i}), \tag{23}$$

$$\sigma_2\left((x_j^{S_i})^{\Psi_n}\right) = \left(\sigma(x_j^g)\omega\right)\omega_2(x^{S_i}), \tag{24}$$

with symmetrical equations for $\mu_2\left((y_j^{S_i})^{\Psi_n}\right)$ and $\sigma_2\left((y_j^{S_i})^{\Psi_n}\right)$, where scaling factors $\omega_2(x^{S_i})$, $\omega_2(y^{S_i})$ are defined according to the extents in $\Psi$ of the respective intermediate point $(x^{S_i}, y^{S_i})$ in comparison to the extents of growth tip $(x^g, y^g)$. These are defined as

$$\omega_2(x^{S_i}) = |\min_{x_i^{S_i} \in \Psi}(x_i^{S_i}) - \max_{x_i^{S_i} \in \Psi}(x_i^{S_i})| \div |\min_{x_i^g \in \Psi}(x_i^g) - \max_{x_i^g \in \Psi}(x_i^g)|, \tag{25}$$

with a symmetrical equation for $\omega_2(y^{S_i})$. In new noisy data $\Psi_n$, the intermediate points $((x_j^{S_i})^{\Psi_n}, (y_j^{S_i})^{\Psi_n})$ are defined using normal distribution noise, according to $\mu_2$ and $\sigma_2$ and scaled by $\omega$, such that

$$(x_j^{S_i})^{\Psi_n} = x_j^{S_i} + \mathcal{N}\left(\mu_2((x_j^{S_i})^{\Psi_n}), \sigma_2((x_j^{S_i})^{\Psi_n})\omega\right), \tag{26}$$

with a symmetrical equation for $(y_j^{S_i})^{\Psi_n}$. In the methods implementation described in this deliverable, we conduct three runs of equations 11-15 and their respective symmetries, generating three unique sequences of noisy data: $\Psi_1, \Psi_2, \Psi_3$.

In order to add a generic plant model, we manually select experiment data associated with the natural plants' smoothest and least noisy movements (identified by observation), and then follow a data-driven approach. We reinforce these generic movements as dominant by adding additional translations of them, distributed over small $xy$ distances. In addition to experiment data $\Psi$ and noisy data $\Psi_n$, we define new generic model data in sequence $\Psi_\Phi$, with each vector $\psi_{\Phi_j}$ structured as those in $\Psi$, defined as:

$$(x_j^{\Psi_\Phi}, y_j^{\Psi_\Phi}) \in \psi_{\Phi_j} = \left((x_j \in \psi_j) \pm 10^\lambda, (y_j \in \psi_j) \pm 10^\lambda\right), \tag{27}$$

where $\lambda = (-3, \cdots, -6)$, generating 64 new $xy$ translations in $\Psi_\Phi$.

The data sequences of $\Psi, \Psi_n$, and $\Psi_\Phi$ are combined to form $\mathbf{\Psi}^*$. Vectors in $\mathbf{\Psi}^*$ are then mirrored across the $x$-axis, as we assume the targeted plant behavior to lack left-right bias. This also doubles the quantity of data. Then $\mathbf{\Psi}^*$ is reformatted according to timestep, such that each new vector $\psi_j^*$ is composed of 18 dimensions representing the current $xy$ stem position, 18 dimensions representing the next stem position, and one dimension representing the Boolean light condition. Vectors are removed if they 1) contain duplicate stems at the current and next positions, or 2) if the $xy$ change is greater than $\times 20$ the average $xy$ in $\mathbf{\Psi}^*$.

**Training procedure.** In order to obtain a holistic plant model, we train the LSTM using Keras [6], a high-level wrapper of TensorFlow [1]. The data in $\mathbf{\Psi}^*$ is formatted as described above, in vectors containing nine 2D stem points at a given time step and at the subsequent timestep, together with current light conditions (left/right light on). The LSTM network has 19 input units (current nine points and light condition), 50 LSTM memory blocks, and 18 output units (next nine points). We shuffle the vectors $\psi_j^*$ and split them into training (70%), validation (20%), and testing set (10%). We train the LSTM network with the training set in sequence of batches of size $N = 30$ for 200 epochs at a steady learning rate of 0.001 using Adam optimizer [1]. The training loss $L_t$ is the mean absolute error (MAE), defined as

$$L_t = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{18}\sum_{j=2}^{10} \mid x_p^j - x_t^j \mid + \mid y_p^j - y_t^j \mid, \tag{28}$$

where $x_t^j$ and $y_t^j$ are the true $xy$ coordinate values of the stem point $j$, and $x_p^j$ and $y_p^j$ are the corresponding predicted coordinates. The validation dataset is used to track the training progress through validation loss $L_v$ (calculated similarly to $L_t$ but not in batches). An early stopping callback is implemented to prevent overfitting by tracking $L_v$ and stop the training process with patience of ten epochs (i.e., if the $L_v$ stops improving for ten epochs). As seen in Fig. 42, the training process stops at the 27th epoch when $L_v$ stopped improving for ten epochs at $L_t = 1.56\times10^{-3}$ and $L_v = 1.55\times10^{-3}$. $1.55\times10^{-3}$ Then, we calculate the MAE for each of the three datasets when used as input to the network. The error values for the training, validation and testing datasets are $1.55 \times 10^{-3}$, $1.55 \times 10^{-3}$, and $1.44 \times 10^{-3}$ respectively. On average, the error is $\approx 1$ mm at each coordinate value, showing that the resulting model represents plant behavior closely[14].

## 4.2 Controller setup

Our controller is an ANN operating two light sources. The input to the ANN at each time step is a set of 10-points (20 real numbers), the current coordinates of the target (2 real numbers), and coordinates of the obstacle (4 real numbers) representing the current plant position and shape as described above. We have two setups: in silico (simulation) and in vivo ('wet' setup with plant and hardware). In silico, the 10 points are directly generated using the *stem stiffening and motion model*. In vivo, a camera and computer vision detects the actual plant and forms the corresponding 10 points. The output of the ANN is the control triggering light sources for stimuli.

### 4.2.1. Task: Obstacle avoidance by shaping the plant

The controller has to shape the plant appropriately by navigating it around a virtual obstacle to then reach a target area. The plant should not touch the obstacle with any part of its body

[14]Find a video at: https://vimeo.com/254632635

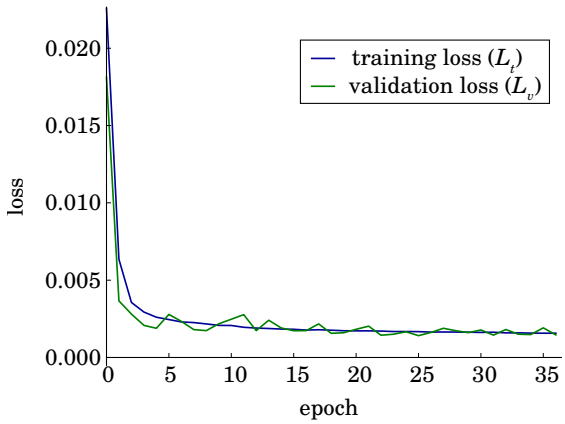



Figure 42: LSTM-based model training.

throughout the experiment. Since the obstacle is virtual, it neither casts a shadow, nor does it give other physical cues (e.g., a mechanical barrier) that would allow the plant to avoid or grow around it by itself. We perform the obstacle avoidance task in two different experiment settings. In the first experiment (*left target* experiment), a fixed target is located at 5.12 cm to the left of the plant anchor and 17.9 cm above it. We evaluate the controller in four different scenarios where a rectangular obstacle ($7 \times 3$ cm$^2$) is centered at four different locations. In the first scenario the obstacle is centered ≈8.24 cm left of the plant anchor point and at a height of 8.8 cm. In the second scenario, the obstacle is pushed 2.67 cm closer to the plant, making the task more challenging. In the third scenario, the obstacle is pushed again 2.67 cm, making it impossible for the plant to reach the target. Finally, the obstacle is pushed again, but 5.33 cm this time clearing the area for the plant towards the target. In the second experiment (*middle target* experiment), a fixed target is located above the plant anchor at a height of 17.9 cm. Here, we have only two scenarios. In the first scenario, the obstacle is centered at ≈3 cm right of the plant anchor and at a height of 8.8 cm. In the second scenario, the obstacle is centered at ≈3 cm left of the plant anchor and at a height of 8.8 cm. Hence, the controller requires different strategies to control the plant for different target/obstacle configurations, which makes the task more challenging. In addition, the plant stiffens only over time, requiring the plant tip to be guided in wide deviations from the plant's ending configuration.

### 4.2.2. Evolutionary approach

We use MultiNEAT [5], a portable library that provides necessary Python bindings to the original implementation of NEAT, to evolve ANN controllers. We use the NEAT parameters set from similar work [25, 11] which have shown successful results. We follow a step-wise simulation approach, where the 10-point plant stem description $S_t = (x^a, y^a, x_t^{S_2}, y_t^{S_2}, \cdots, x_t^{S_9}, y_t^{S_9}, x^g, y^g)$, the target position $\mathbf{x}_i^*$, and the coordinates of an obstacle $\mathbf{x}_i^o$ are input to the ANN at each time step $t$. The output of the network $C_t$, regulates the light settings. If $C_t \leq 0.5$, the left light source is triggered, otherwise, the right light source is triggered. The current plant location and light setting $(\mathbf{x}, C)_t$ impact the plant's behavior during that time step. The *LSTM stem stiffening and motion model* is used to predict the next ten-point plant stem description $S_{t+1}$ accordingly. In case of experiments in reality, an image of the plant is captured and processed to determine $S_{t+1}$ (see Section 4.1.2). The simulation is stopped when the tip's $y_t^g$ value is equivalent to approximately 21 cm in height or once the plant touches an obstacle. Beans

require approximately 72 hours to grow that high, hence, performing the reality experiments in relatively short period of time.

This also allows the plants enough growth to exploit stem stiffening and avoid hitting the obstacles.

At the end of the simulation (at $t = f$) the performance of the ANN controller is evaluated using a behavioral fitness function $F$ (according to the classification in [15]). Where the motion of the plant is rewarded by measuring the distance traveled by its tip $g$ towards the target along both $x$ and $y$ axes as

$x_r = |x^*| - |x^* - x_f^g|$, $y_r = |y^*| - |y^* - y_f^g|$,

where $(x^*, y^*)$ is the position of the target. The fitness $F$ is then calculated by

$$F = \frac{x_r + y_r}{|x^*| + |y^*|}, \tag{29}$$

where $|x^*| + |y^*|$ is the theoretical best fitness value the controller can achieve. According to the experiment settings, if the controller is evaluated at different scenarios, then it's fitness value is the average from all evaluations.

## 4.3 Results

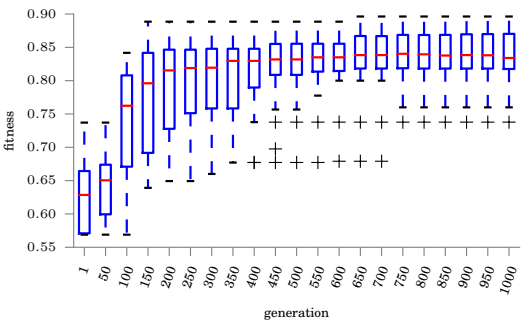
Based on the *stem stiffening and motion model* (see Sec. 4.1.3), we evolve the robotic controllers and evaluate their performance in simulation. Next, we transfer the fittest controllers to reality and investigate the extent of the reality gap.

### 4.3.1. Evolving controllers in simulation

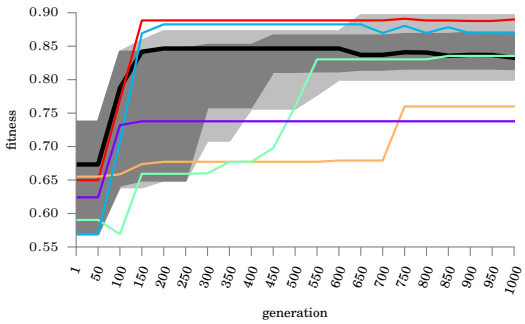
First, we report the results of the *left target* experiment. The boxplots in Fig. 43(a) and function boxplots in Fig. 43(b) show the performance of 20 independent evolutionary runs, 1000 generations each. The best fitness per generation for all the evolutionary runs is considered. Notice the steady increase in the median until the convergence is reached around the 500th generation. In this experiment the controller is evaluated according to four different scenarios (see Sec. 4.2.1). Therefore, according to the behavior of one of the best controllers (fitness of 82.5%), the controller is able to determine whether it needs to exploit the stem stiffening or not, in order to avoid hitting the obstacle. In case there is a possibility to hit an obstacle (e.g., second scenario), the controller steers the plant away into the opposite direction of the obstacle, long enough to obtain sufficient stiffness at the lower parts of the stem, see Fig. 43(d), then steers the plant back towards the target area, see Fig. 43(e). In case the obstacle is not blocking the way (e.g., forth scenario), the controller leads the plant directly towards the target area (i.e., no stiffening is necessary). The behavior in all scenarios can be seen in the video[15].

Second, we report the results of the *middle target* experiment. Here, we also have 20 evolutionary runs, 1000 generations each as shown in Fig. 44(a) and 44(b). In contrast to the previous experiment, the convergence is reached earlier, around the 350th generation. This indicates that the task here is relatively easier. The expected behavior of the evolved controller, is to steer the plant away, obtain some stiffness, then steer the plant back, similar to the behavior in *left target* experiment. However, the controller here (fitness of 97.3%) steers the plant to the opposite side of the obstacle, see Fig. 44(c). Then switches between the two light sources, until the plant obtains enough height and stiffness without hitting the obstacle, see Fig. 44(d). Finally it steers the plant tip towards the target, see Fig. 44(e).

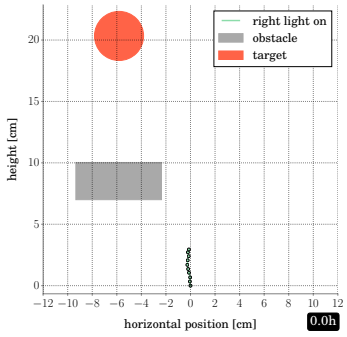
[15] Find a video at: https://vimeo.com/254632635



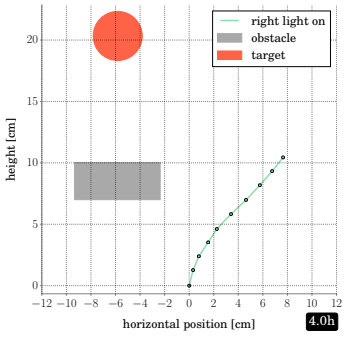

(a) *left target* exp., boxplot of best fitness per gen.



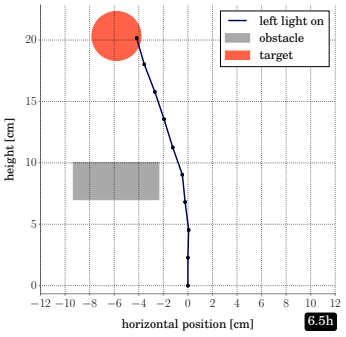

(b) *left target* exp., functional boxplot, best fit. per gen.




(c) Initial stem geometry.

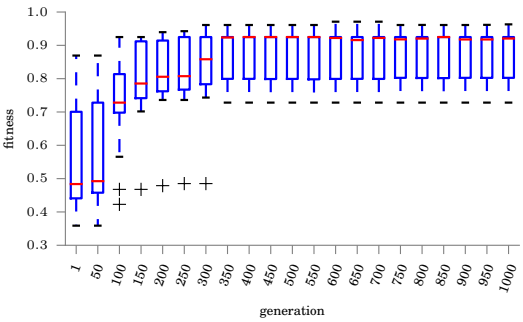



(d) Stem geometry at 4.0 simulated hours.

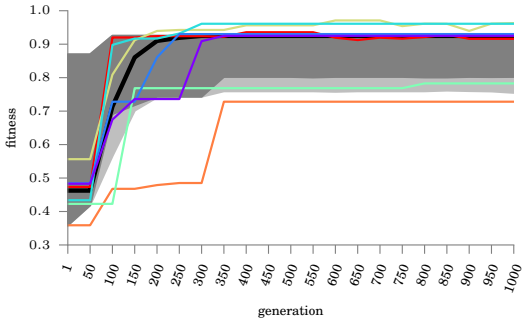



(e) Stem geometry at 6.5 simulated hours.

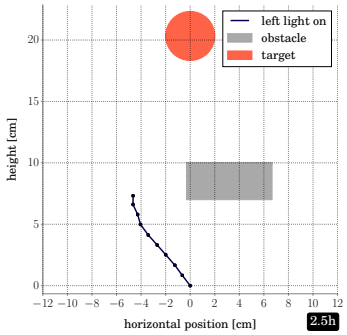
Figure 43: Performance of the evolutionary process in the *left target* experiment over generations for 20 evolutionary runs.



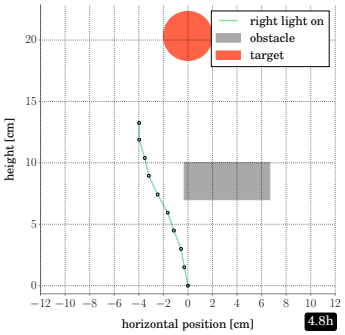

(a) *middle target* exp., boxplot, best fit. per gen.



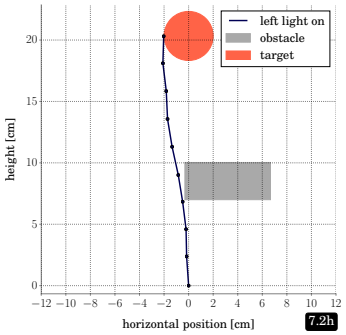

(b) *middle target* exp., functional boxplot, best fit. per gen.




(c) Stem geometry at 2.5 simulated hours.

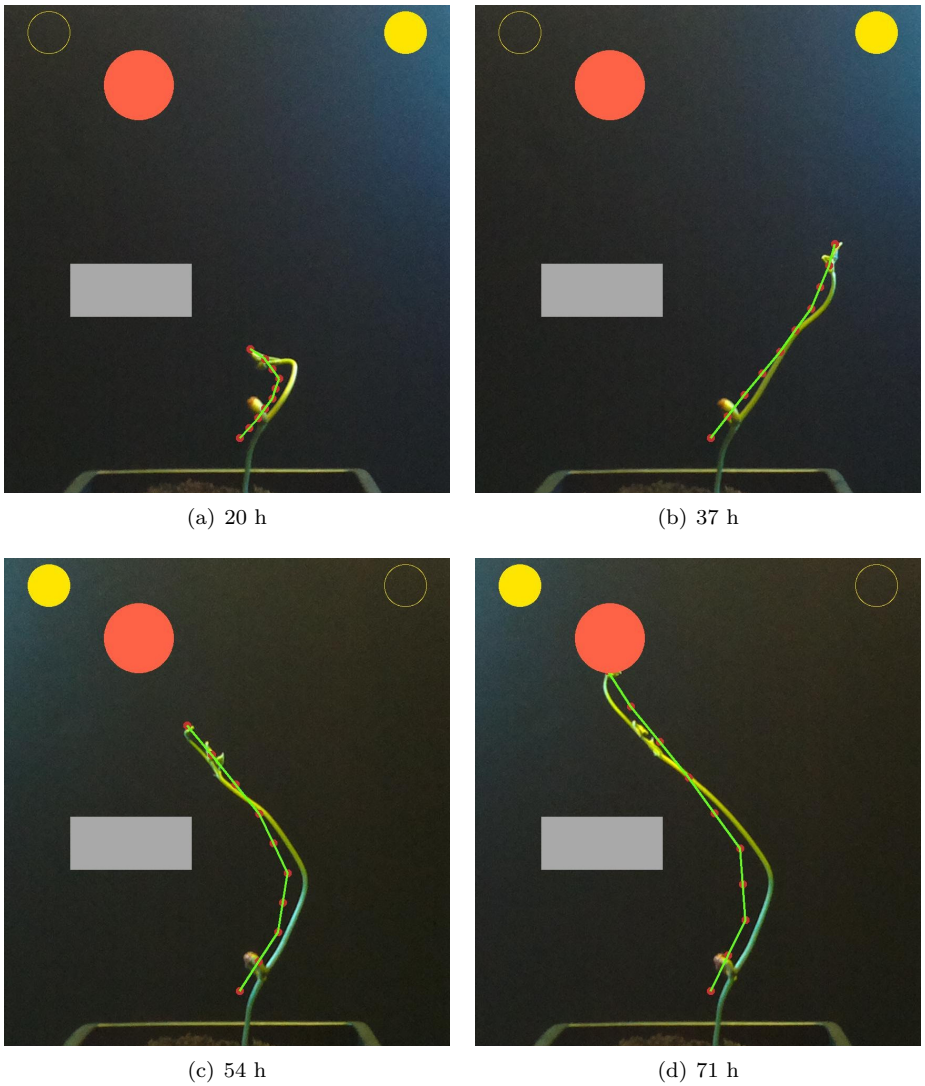



(d) Stem geometry at 4.8 simulated hours.




(e) Stem geometry at 7.2 simulated hours.

Figure 44: Performance of the evolutionary process in the *middle target* experiment over generations for 20 evolutionary runs.



(a) 20 h (b) 37 h

(c) 54 h (d) 71 h

Figure 45: Sequence of images showing the course of the reality-gap experiment. The yellow circles on top indicate which light is on (filled: on, empty: off). The larger filled red circle is the target area and the gray rectangle is the obstacle the plant is not allowed to touch.

### 4.3.2. Performance of controllers in reality

To test the controller's performance in the physical world, we examine whether it can guide a natural bean plant around a virtual obstacle (without colliding) and reach the target area. This addresses the reality-gap problem [12], which states that controllers evolved in simulation do not always transfer to a real setup, because the simulation is limited in principle. To test the reality gap, we used the bio-hybrid setup. The camera and computer vision detect the stem, feeding into the ANN evolved in simulation, which then control the light stimuli provided to the real bean plant. The controller we selected was evolved according to *left target* experiment and the rectangular obstacle is centered ≈6.3 cm left of the plant anchor point and at a height of 12.5 cm

from the soil. The bio-hybrid setup completed the task using the real plant, although the plant's circumnutation behavior brought it in close proximity of the obstacle at times (see Fig. 45 and video[16]). During the experiment, the controller initially kept the right light on to guide the plant away from the obstacle. It did so for about 37 h, until the plant was 7.4 cm to the right of the plant origin and 15.4 cm above the soil, see Fig. 45(b).

It switched to the left light, quickly bringing the plant tip to the opposite side, while the stem retained some stiffness. After less than 2 h, the plant tip is roughly in the center, with a pronounced curve in the stem, see Fig. 45(c), as the tissue at the base has already stiffened. There is then a phase of quick alteration between light, as the controller guides the plant tip close to the obstacle's edge, thereby leaving it near to the target after clearing the obstacle. The left light is then triggered for another 37 h, successfully guiding the plant to the target, while the curvature of the stiffened stem allows it to entirely avoid the obstacle. The evolved controller together with the real plant achieves 100% fitness, see Fig. 45(d). Though repetitions are required to confirm the result, given the successful outcome of this experiment, we consider it feasible that the controller transfers to reality.

## 4.4 Conclusion

We have successfully managed the above mentioned complex work-flow and have reported our experiments of controlling a plant's motion, its stiffening process of the stem into a desired shape, and we have shown that we successfully bridge the reality gap. Especially the modeling challenge is difficult and requires complex state-of-the-art techniques (deep learning). Despite the complexity of the current approach, we are curious to keep pushing the boundaries. In future work, we may try to shape plants into more complex patterns, to control several plants at a time (e.g., plant interaction), and to go to much bigger scales.

[16] Find a video at: https://vimeo.com/254632635

# 5 Molecular biology

## 5.1 Long distance signaling

In D2.2 we have reported investigations of long distance signaling in plants as a part of an elucidating processes that may prove to be useful in the development of new bio-hybrid communication channels between robots and plants. Using *Arabidopsis thaliana* Jas9:VENUS overexpression lines enables us to observe in *in vivo* conditions an increase of jasmonic acid (JA) in response to biotic and abiotic stress. VENUS is a yellow fluorescent protein that forms the common polypeptide with Jas9, a motif derived from the JAZ protein. At AMU we prepared several lines that exhibit Jas9:VENUS overexpression in different knockout backgrounds. Until March 2018 several double homozygotes lines were obtained:

1. Jas9:VENUS x *rbohd*
2. Jas9:VENUS x *mca2*
3. Jas9:VENUS x *glr3.6*
4. Jas9:VENUS x *tpc1*
5. Jas9:VENUS x *glr3.3*
6. mJas9:VENUS x *rbohd*
7. mJas9:VENUS x *glr3.6*
8. mJas9:VENUS x *tpc1*

These lines are now analyzed concerning their difference in spreading of regions with increased level of JA within the roots. The knocked-out genes are involved in rapid long distance signaling and mechano-sensing. We induce jasmonic acid biosynthesis (activation of precursor) in different ways: by applying $10\mu M$ isoxaben, by cutting a leaf, by changing aspects in relation to gravity, and root growth through narrow aperture. Induced stress resulted in the increase of active jasmonic acid, which can be detected by a decrease of the Jas9:VENUS level (see Fig. 5.1). We focus mostly on spreading of intercellular signaling, which induces an increase in JA. Images from lines: Jas9:VENUS and line 2), 3) were analyzed and 4D images were captured. However, the data is not completely analyzed yet and conclusions could not be formed yet. Lines containing mJas9:VENUS are used as a control, due to mJas9:VENUS there is no degradation in the presence of JA. Additionally, we prepare constructs for transient expressions, in both protoplasts and *Nicotiana benthamiana* epidermis, genes associated with long distance signaling and potential mechanoreceptors. Coexpression of two genes makes it possible to investigate protein colocalization and direct interactions (with FLIM-FRET). The transient expression systems will also be used to analyze spatial relations between key signaling genes (e.g., RBOHD) with a cytoskeleton (actin and microtubules). The cytoskeleton is crucial, for example, in forming cell walls, that is, in establishing mechanical properties of plant cells and the whole plant, and also in intracellular trafficking of organelles and vesicles. Subcellular localization of proteins of interest in a certain relation to the cytoskeleton could allow us to formulate new hypotheses about intercellular signaling in plant cells.

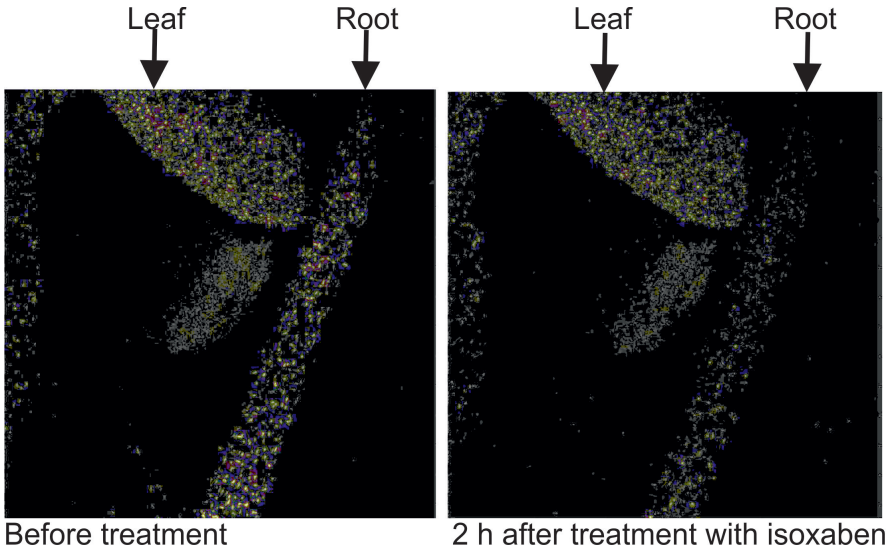



Figure 46: Isoxaben inhibit cellulose biosynthesis, leads to mechanical stress, and increases the active jasmonic acid level. We are collecting data from the original Jas9 line and Jas9 crossed with different knock-out lines to investigate long distance signaling. Application of isoxaben resulted in increase of jasmonic acid level and decrease of VENUS fluorescence signal in roots.

## 5.2 Interactome of plants gamma-secretase

Protein complex of gamma-secretase has an activity of intramembrane protease and is responsible for the production of beta-amyloid in human neurons. Beta-amyloid is well known to be responsible for forming plaques in the human brain, which is closely related to the pathogenesis of the Alzheimer disease. Hence, gamma-secretase has been investigated intensively in mammals. A lot of molecular data was obtained about complex composition, assembly, activity, and functions. A presenilin (there are two presenilins genes in most of Eucaryotic genomes) is a catalytic subunit of the complex and it was found as a independent player being responsible for intracellular calcium flux in gamma-secretase independently. Fully functional gamma-secretase and presenilins were shown to be indispensable, for example, for long term potentiation (LTP) by modulating the neurotransmitter release. LTP is a process that is crucial in formation of memories and learning. During recent years, we have started research on plant gamma-secretase and preliminary results have been published [22]. Localization of *A. thaliana* homologs of gamma-secretase subunits was shown and also direct interactions between proteins. Published results suggest abundance of gamma-secretase in plant cells and potential involvement in autophagy and protein trafficking processes.

Gamma-secretase subunits belong to an interesting set of proteins that posses many neuron-specific functions in mammals, but that are present also in plants, which have no neural system. Plants generally sense environmental and internal stimuli, integrate them, and respond, in different ways from animals. However, some mechanisms and molecules are similar when comparing animals and plants. Elucidating *in planta* functions of homologs of neural proteins may lead to a better understanding of how plants regulate their response to stimulations and how much molecular mechanisms of plants are similar to the better known mechanisms in animals. One such example is are Toll-like receptors in non-adaptive immunity and recognition of elicitors.

We have focused on interactome of plant presenilins and other gamma-secretase subunits.

| **Gene identifier** | **Description** | **Detergent used** | **FLIM-FRET verification** |
|---|---|---|---|
| AT2G29900 | Presenilin 2 | Digitonin, C12E8, Triton X-100 | - |
| AT4G12650 | EMP7/TMN12 | Triton X-100 | - |
| AT5G10840 | EMP1/TMN8 | Triton X-100 | No |
| AT2G46170 | RTNLB5 | C12E8 | - |
| AT4G23630 | RTNLB1 | C12E8 | Yes |
| AT4G27000 | RBP45C | digitonin | No |
| AT2G33410 | RNA-binding protein | digitonin | - |
| AT3G02420 | dihydroflavonol 4-reductase/flavanone protein | digitonin | No |

Table 4: Proteins identified by MS after TAP-tagging purification of presenilin 2 from root cell cultures. Transmembrane domain proteins are underlined. In the case that an FRET-FLIM verification was done, the confirmation of the interaction is "Yes", while "NaN" refers to no FRET-FLIM was done to verify the interaction, and "No" refers to the finding of a denied interaction.

TAP-tagging purification and co-immunoprecipitation were used for initial identification of presenilin 2 and PEN-2 protein partners. Recently we have verified interactions that were identified by TAP-tagging (Fig. 5.2). Confocal microscopy and FRET-FLIM analysis enabled us to confirm some of the previously identified interactions. Many of the TAP-tagging proposed interactions were denied. Despite this, initial information of gamma-secretase subunit interactions were elucidated. Reticulon family protein, RTNLB1, was identified in both TAP-tagging and FRET-FLIM as presenilin 2 interacting protein. RTNLB1 is involved in forming tubular regions in endoplasmic reticulum (ER) and is indispensable for trafficking of FLS2, which is a key receptor of bacterial infection in plants (see Fig. 5.2). Moreover, we showed that presenilins interact with two single transmembrane domain (TMD) proteins: TIRP and PHYL1.1. TIRP is a representative of TIR-X protein family, which suggests the involvement in Effector Triggered Immunity (ETI) – the second layer of plant cells recognition of pathogen activity. PHYL1.1 belongs to phytolongings, plants R-SNARE proteins, that run secretory pathways and vesicle targeting, fusion processes. Both, PHYL1.1 and TIRP could be substrates of gamma-secretase, due to single TMD, but more experiments are required to prove this. Moreover, we discovered that RTNLB1 also interacts with TIRP, hence the interaction networks that are uncovered here, are probably more dense and complex. Cellular and physiological processes, in which plants gamma-secretase is involved, are not elucidated yet, but our findings suggest an association with the regulation of endomembrane systems and the immune response. Understanding molecular processes that make a plant exhibiting reasonable responses to changing environmental cues, may allow us to develop better plant-robot communication. We are in the process of publishing our results concerning gamma-secretase interacting proteins.

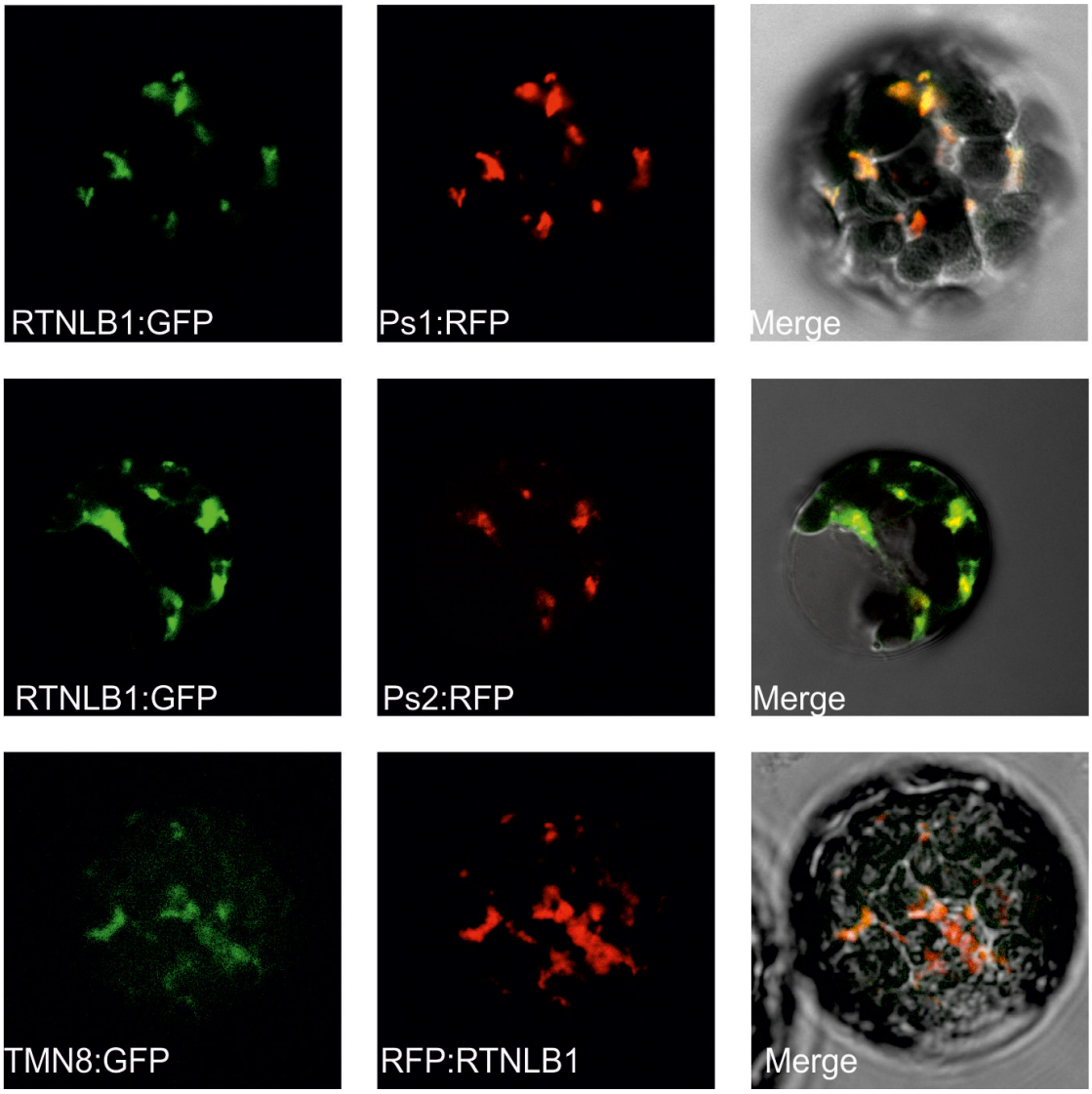



Figure 47: Coexpression of RTNLB1 with presenilins and TMN8. Colocalization was observed and FRET-FLIM confirmed presenilins-RTNLB1 interactions. TMN8, another TAP-tagging identified protein, was shown to colocalize with RTNLB1 and also partially presenilins (data not provided here), but no interaction was confirmed with FRET-FLIM.

# 6 Integration: the Integrated Growth Projection (IGP)

This section introduces the integration of *flora robotica*'s models of artificial growth (e.g., the VMC, see Sec. 3) with models of steered natural plant growth (e.g., evolved control of light stimuli, see Sec. 4.1.1). The approach described in this section is currently at its intermediary stage of development, so is introduced here with a high-level description. (Details of the next stages of development will be presented in future deliverables D2.4 and D3.3). The method may be adapted to also incorporate future models (e.g., of plant repellents, Sec. 2), and can be linked to the planned central database of the Social Garden (future deliverable D3.3).

## 6.1 Introduction to IGP

The *Integrated Growth Projection* (IGP) takes the models and controllers of *flora robotica* as inputs, and unifies them into a single growth simulation of both the artificial and natural components. This setup is used to project the growth of a whole *flora robotica* system at a given instance. Its output projections are intended to be useful in a range of workflows, including:

1. to manually refine the parameters of the incoming controllers,
2. to assess the performance of simulated grown artifacts, for evolution of the controllers,
3. to provide certain types of projection-based performance feedback for controllers running in reality,
4. to facilitate simulated *Growth Careers* for architectural envisioning and design processes (see D3.2),
5. to allow users of the Social Garden to explore the possible growth futures of their own *flora robotica* instance, and
6. to allow those users to impact that future, through the tool of Interactive Evolution.

Here we introduce the IGP integration of three primary models of *flora robotica*:

- ▷ VMC in 3D (finalized, except for minor updates),
- ▷ braid 3D mesh modeling (finalized; see D3.1 and D3.2), and
- ▷ plant growth and motion model for control by light stimuli (currently represented in the simulation setup by a placeholder, as a version of this model for *climbing plants* is still under development)

The IGP may be extended to incorporate additional *flora robotica* models. Furthermore, some of the currently incorporated models will undergo future developments (e.g., the plant growth and motion model). Therefore, the IGP requires interoperability with the incoming models, such that the IGP can be updated seamlessly. As part of the *Internet-connected Social Garden* (future deliverable D3.3), consortium partner ITU is developing a planned central database to host *flora robotica* models and data streams. In D3.3, the IGP is planned to be adapted to take its model inputs directly from this database.

Example outputs of the IGP—for Workflow 4 listed above—are presented in deliverable *D3.2 Architectural propositions*. The features of the IGP that enable it to be used effectively with Interactive Evolution for the future Social Garden (Workflow 6 listed above), will be reported in future deliverable D3.3. Detailed developments and results of the IGP for any of the above Workflows 1-3 or 5-6 will be reported in future D2.4 and D3.3.

## 6.2 Integrating the FR artificial growth models (VMC, braid)

For the IGP,[17] the VMC is implemented into 3D in a flexible and open-ended way. The interpretation of that 3D VMC into any physical structure—including braid—is also decoupled from the behavior of the VMC itself, to dramatically expand the range of possible artifacts able to be generated or grown using the VMC.

### 6.2.1. Open-ended 3D growth with VMC

In a two-dimensional growth plane, the usability of the VMC has been demonstrated above and previously, for tasks such as growing tall in a harsh environment and navigating a maze. Above, the VMC has also been used in 3D to grow tree-like structures in Unity (see Sec. 3.2). Here, we extend the VMC to encompass more open-ended growth in three-dimensions. In this 3D VMC, each decision to grow results in three new branches from the respective leaf (i.e., edge vertex). Three branches provides the minimum information needed to sense an advantageous direction in a 3D environment. Each new set of branches is oriented around the axis of its parent branch. The relationship of each individual new branch to the parent is user-defined, according to 1) the distance of the new leaf from its parent leaf, and 2) the new branch's angle of inclination from its parent branch.

Figure 48: Examples of user-defined branch sets (below), with example resulting structures (above), when used in a non-deterministic 3D VMC setup. The example branch sets are (left) symmetrical with wide proportions, (center) symmetrical with narrow proportions, and (right) uneven. Uneven branch sets tend to create morphologies that have directional bias, forming curved primary axes.

Not only do new branch sets vary in width-to-height ratio, but they can be radially symmetrical or can be uneven. The user-defined shape of new branch sets can dramatically impact the overall morphology of a 3D structure grown by VMC, as seen in Figure 48.

[17] Parts of this subsection also appear in an *in-print* conference paper [10].

### 6.2.2. From control logic to physical structures and building components

When interpreting the VMC into a material structure, the studies shown above focus on direct relationships between the physical and the encoding, where each edge and vertex of the VMC graph has a one-to-one relationship with a physical element in the structure (see Sec. 3). This interpretation style creates material structures that are tree-shapes, shown in the previous VMC studies with built structures from Thymio robots and for tubular braid shown above.

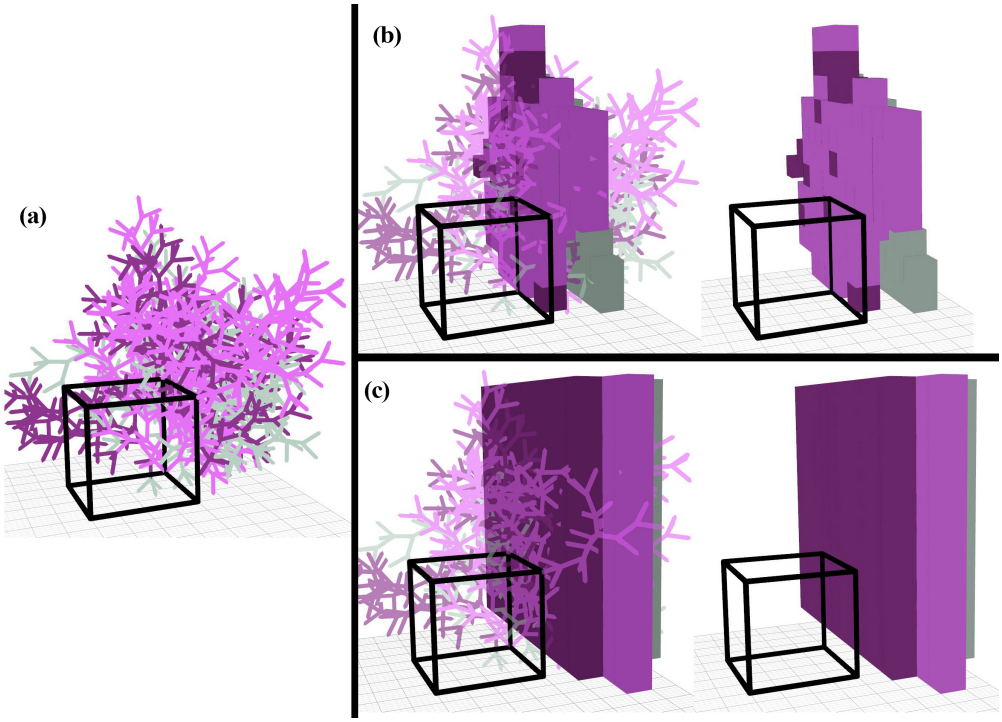



Figure 49: VMC and material aggregation control, as grown in response to an obstacle in the environment. (a) VMC structure without a second layer of control for material aggregation, (b,c) the same VMC output, with its respective structure when incorporating simple rule-based control for material aggregation, with either (b) control for placing bricks along a barrier line, or (c) control for placing a solid wall.

In the *Integrated Growth Projection*, we extend to flexibility in physical structure by decoupling the control of material aggregation from the control logic that discovers environmentally advantageous locations for growth. A second layer of control for material aggregation allows for the distribution of heterogeneous tasks, and therefore the creation of hybrid control. Such hybrid control can provide flexibility to a designer or user of the system.

When being hybridized with a self-organizing controller (the VMC), control for material aggregation can itself also be self-organizing, or can instead be simply rule-based. Here we illustrate with rule-based control for placement of basic building components: bricks or walls (see Figure 49). In this scenario the VMC can be used to sense and avoid the environmental obstacle, thereby solving the specified task reliably through features of its behavior, while the second layer of control - for material aggregation - allows to have flexibility in the structure built and building components used, rather than being restricted to tree-shaped structures.

Further examples of the types of structures able to be grown will be reported in future D2.4 and D3.3.

### 6.2.3. Integrating VMC with braid

Because *flora robotica*'s chosen method of construction is braid, here we discuss the integration of the VMC with our braid 3D mesh modeling (see D3.1). The braid modeling approach takes a 3D quad mesh and a list of edge colorings as its inputs. Furthermore, there are certain fabrication and mechanical constraints on the scope of quad meshes able to represent braids that are both possible in reality and possible to be fabricated by the braiding machine (see D1.3). To go from a VMC graph output to a quad mesh that represents a real braid that can be machined, we primarily utilize two categories of approach:

1. using the above technique to locally or globally aggregate tiles, voxels, or other compatible modules (or to globally aggregate more amorphous material that can then be approximated into tiles or voxels), see Fig. 50; or

2. using a separate technique that interprets the VMC graph into a quad mesh that undergoes multiscalar physics-based relaxation in order to approximate tubular braids with bifurcations, see Fig. 51.

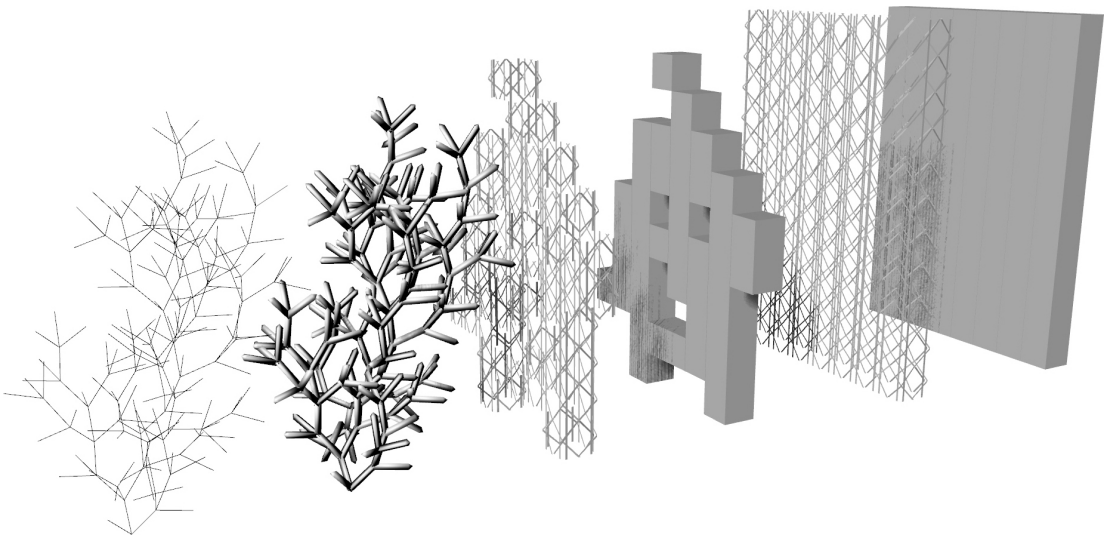


Figure 50: Interpreting the VMC into braided structures through the approach of aggregating voxels compatible with the *flora robotica* braid 3D mesh modeling. (Left) A VMC output graph, and a 3D mesh model of that graph interpreted one-to-one as a tree-like structure. (Center) that graph interpreted into a 3D mesh model of a braided structure resembling a brick-based partition, through the aggregation of voxels and a matching tile colorings list. (Right) that graph interpreted into a 3D mesh model of a braided structure for a basic wall, through the definition of a wall mass that can easily be approximated into mesh tiles and a matching tile colorings list.

In the first category of approach, material can simply be aggregated in the full area that the 3D VMC graph output occupies, for example by casting the nodes of the graph as a 3D point cloud and running a controller that aggregates blocks anywhere in the environment that is within

a given distance from that point cloud. However, control that is more specific can also govern the aggregation step, enabling the output to solve a much wider variety of construction tasks, where the range of shape features appropriate to the task may be very narrow. For instance, a very common and basic element found in construction is a standard partition or wall, where a partial or full barrier is erected between two differentiated spaces, but should normally use only as much material thickness as is structurally and programmatically necessary. In order to build this kind of artifact using the VMC as input, a controller can be used to detect certain shape features in the topology of the VMC graph output, and aggregate material in response to those shape features. Although for most intended workflows this step needs to be completed with decentralized control and to send feedback to the VMC, for the purposes of this high-level introductory description, we use global rule-based control to provide a simple and illustrative example.

In this example (see Fig. 50), the topology of the VMC graph (left) is analyzed simply to determine the opposing axis' widest point at any height of the graph along a given directional axis, and a partition of bricks (center) or a solid wall (right) is aggregated within the zone of the VMC graph, along that opposing axis. In this way, a consistently non-leaning and constant-thickness partial-cover partition or full-cover wall is built in response to environmental conditions, through the control provided by the VMC.

For this approach, where there is no global physics-based relaxation occurring to the quad mesh before it is input to the braid 3D mesh modeling, the real braided structures able to be represented with close approximation are limited to those in which the filaments or their organization display a certain degree of structural stiffness. This is not particularly limiting, as there are quite a number of approaches to establishing structural stiffness in braid, laid out in *D3.2 Architectural propositions*. The benefit of adopting this constraint of structural stiffness is that this approach of integrating the VMC with braid is able to be implemented without any global analysis or control, and therefore can be run using the real decentralized hardware used in *flora robotica*, detailed both above, and in D1.1-3 and D2.1-2.

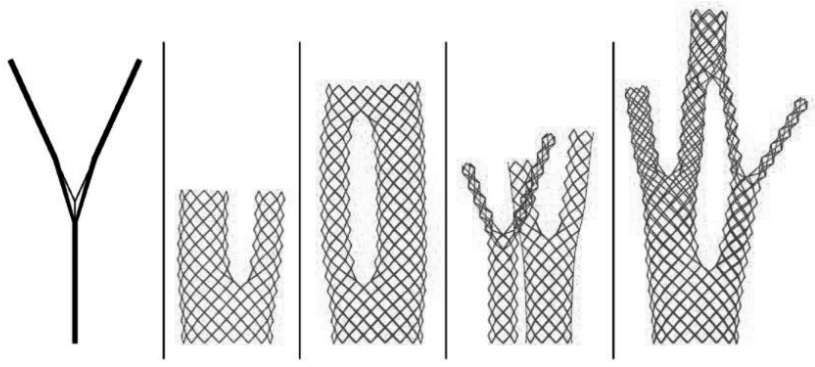


Figure 51: (Left) A simple four-node graph that might be output from a 2D implementation of the VMC. (Right) An example set of four possible tubular braid conditions that, having been first output from the multiscalar physics-based relaxation of the braid 3D mesh modeling, can then be used as a mid-level voxel-based dictionary for integrating the VMC with braid.

By contrast, the second category of approach is not able—at least directly—to be implemented in a decentralized way due to its partial dependency on global physics-based mesh relaxation (see Sec 4.3.1 in *D3.2 Architectural propositions*). It will, however, reliably produce simulated results that very precisely represent real tubular braids with bifurcations, even when constructed with

flexible materials and organizations. Its results maintain this high degree of precision for detailed features of the real braids (see Sec 3.3.3 in *D3.2 Architectural propositions*).

Indirectly, a set of tubular braid outputs of this approach can be set up as a mid-level voxel-based dictionary in lieu of the standard low-level tiles and voxels described above. This allows the VMC output graph to be interpreted not necessarily one-to-one, but into a range of possible tubes and tubular bifurcations (see Fig. 51), which can be dependent on environmental conditions or control parameters. This keeps some of the precision of the second approach, and brings in some of the flexibility of the first approach. This indirect usage is very productive for generating and envisioning (i.e., Workflow 4 listed above), for large-sized braided structures that both perform an architectural role and in principle represent a real braid that can be machined (see D3.2).

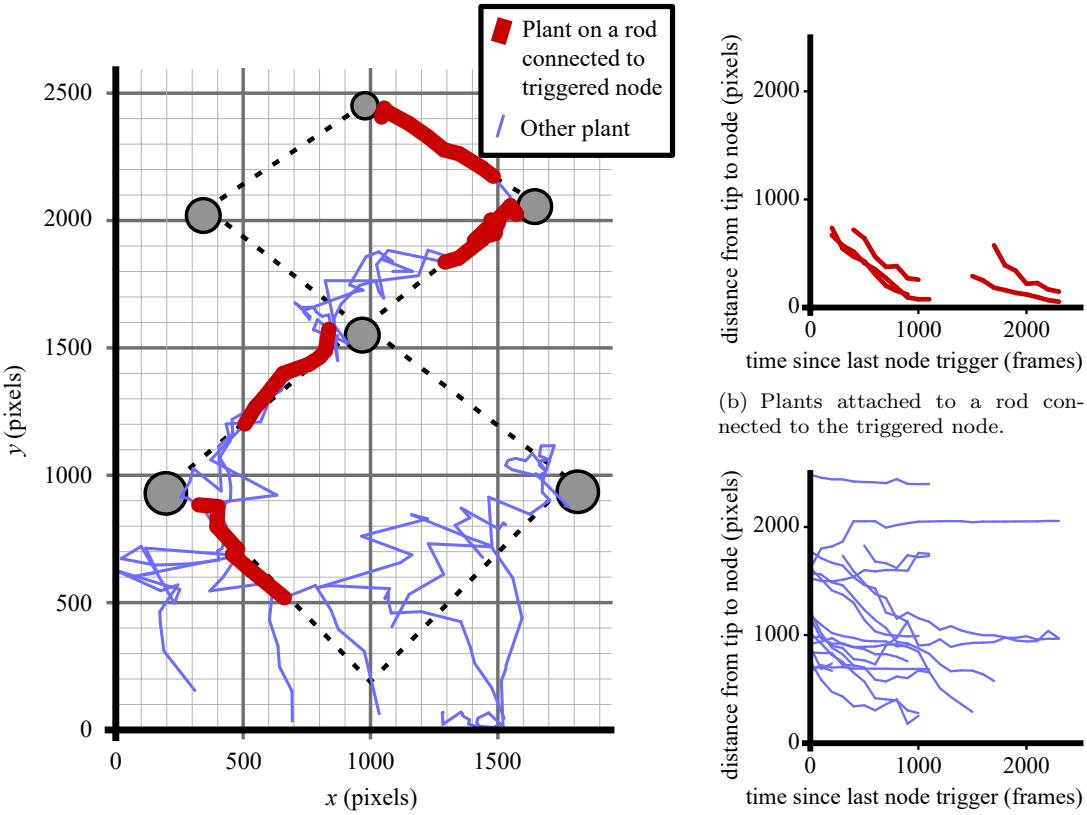
## 6.3 Integrating the FR natural growth models (steered plants)

In this high-level description of the IGP, we limit to projecting artifacts built with climbing plants, specifically common beans, used in building our models for plants steered by light stimuli. In order to project the growth patterns of climbing plants on braided structures, in response to light stimuli, the models used will ideally encompass:

- ▷ the growth and motion of unattached plant tips
- ▷ the behavior of tips' attachment onto scaffolds
- ▷ the growth of a tip that has attached
- ▷ the behavior of new branching events

A unified model of these growth behaviors in climbing plants in under development, so the current IGP setup makes use of a placeholder based on the models and data already developed. The growth and motion of unattached bean plants has been developed so far for tip control (see D2.2) and for stem control (see Sec. 4) for light stimuli from 2 sources. Though it is not verified that these models also apply directly to conditions where the plant has previously been attached to a scaffold, they are currently useful to approximate tip behavior in those conditions. The other three behaviors, regarding scaffold attachment and branching, are studied and successfully steered in prior work that uses robotic nodes to provide stimuli (see D2.2 and D1.2). Here (see Fig. 52) we present a new data-driven analysis of those prior experiment results, which gives us a placeholder for growth models of the two behaviors related to scaffold attachment, useful for placeholder projections that roughly approximate expected growth. Though the data used here does contain some information about branching behavior as well, it is not currently used in the placeholder for the IGP.

For the data-driven analysis of experiment results concerning attachment behaviors, plots of results in Fig. 52 show the progression of plant tip movements over the seven week course of an experiment presented in D1.2. The *leading* plant tip successfully attaches to each of the four correct rods (i.e., those connected to the concurrent *guiding* node). The fastest attachment is very near to the rod's beginning, and slowest is near to the middle of the rod. Other than the leading plant, one other plant also manages to attach to one of the correct rods (i.e. connected to a triggered node). In Fig. 52(a), we see that the plant tips take a much shorter path to the next attachment when the target rod is on the nearby side of the node. When the plant has to surpass the node to find the target rod, it searches in a large area. Even though it passes by an easily reached rod—and the plant's *twining* behavior is a strong influence—the plant chooses to continue following the attractive stimulus until it reaches the target rod, even if this requires a *branching* event. In Fig. 52(b), we see that when plants are attached to a rod connected to




(a) Path of plant tips, relative to the image frame.

(b) Plants attached to a rod connected to the triggered node.

(c) Other plants.

Figure 52: Movement of plant tips visible in experiment photographs, in a pre-defined pattern experiment using a Z-shaped pattern. Tip locations recorded manually, at 200 minute time intervals (i.e., at every 100th frame). (a) shows the $xy$ movement path of visible plant tips in the image frame, colored according to whether the plant has attached to a scaffold rod that is connected to the currently triggered node (i.e., *guiding* node). (b) and (c) show the changes in plant tips' distance from the currently triggered node, throughout the experiment, relative to the time since the last node triggering event. For plants attached to a rod connected to the currently triggered node (b), a plant tip on average moves closer to the triggered node by 64.9 pixels (approximately 3.2 cm), per 100 frames (i.e., 200 minutes). For other plants (c), a tip on average moves 28.1 pixels (approximately 1.4 cm) closer.

the *guiding* node, their tips approach the node consistently and quickly, at an average speed of $\approx$ 1.0 cm per hour. The stimulus provided by the nodes therefore successfully completes the intended task. In Fig. 52(c), we see that the attractive stimulus additionally influences other plants favorably, as even without helpful attachment they on average approach the *guiding* node at $\approx$ 0.4 cm per hour.

We currently use two approaches in the IGP setup to utilizing these model placeholders. Once the models of growth behaviors to be used are finalized, the approaches to implementation will likely need to be refined, or even dramatically modified. However, the placeholders are adequate and useful for the architectural envisioning that was conducted (Workflow 4 listed above, with results presented in *D3.2 Architectural propositions*), as well as for preliminary steps in the ongoing work for Social Garden growth futures and the use of Interactive Evolution (Workflows 5 and 6 listed above). The first approach is to combine motion vectors from both the data of unattached plant tips and the above data of attached growth, and to use means and standard deviations of this data to approximate fully stochastic plant growth at each timestep, where the locations of light source stimuli are either not present or are unknown (see Fig. 53). The second is to follow essentially the same approach, but to use it to approximate growth in response to known light source locations (see Figs. 54 and 55). Because we are still using placeholders, we currently only use this approach with light source at randomized locations, which are either all triggered stochastically at each growth timestep (see Fig. 54), or where some zones are triggered stochastically and some zones are triggered continuously (see Fig. 55). When some zones are triggered continuously, currently the growth is approximated as fully stochastic in that zone, as our current data described above does not encompass conditions with more than one nearby triggered light source. As it simply uses means and standard deviations, the current placeholder setup is in principle easily receptive to updates in our available growth data.

The IGP setup is generalized to encompass growth not only on braided structures, but also any other scaffold type with shape features that provide attachment points for climbing plants, and are able to be represented by 3D polygon meshes (not necessarily quad; see Fig. 53).

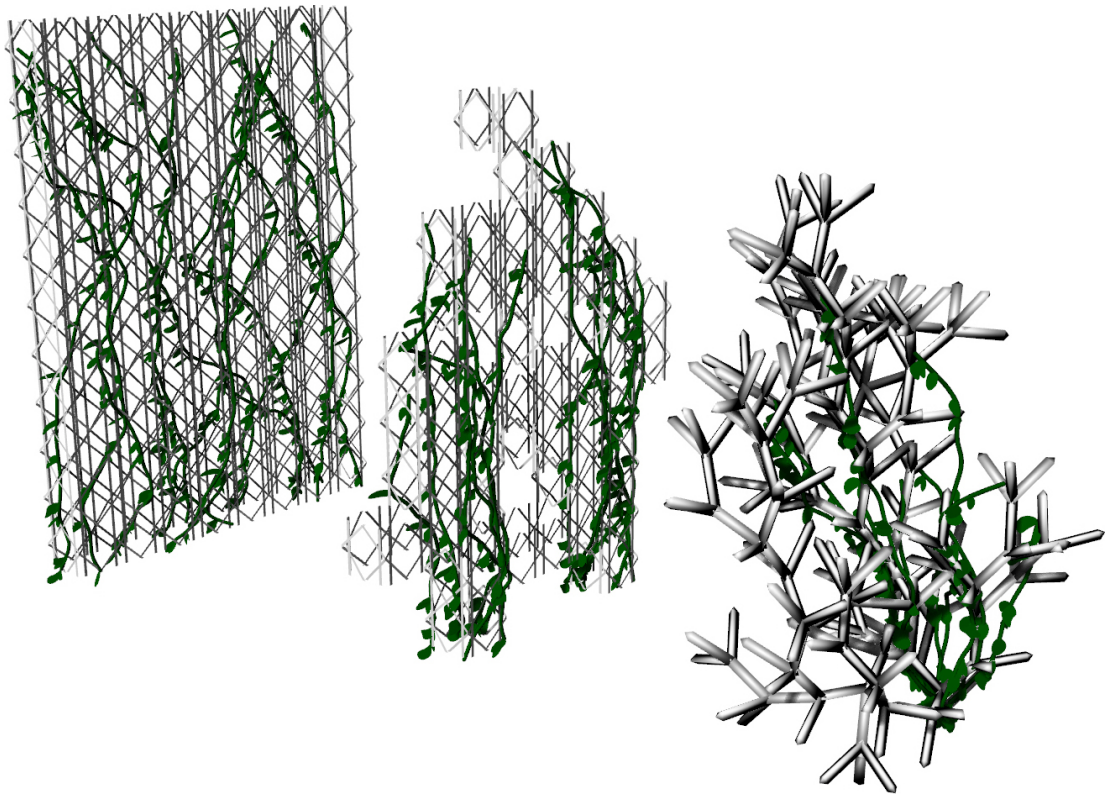


Figure 53: Three example IGP outputs, approximating fully stochastic plant growth with unknown light source locations, on three scaffolds: (right) one-to-one 3D mesh interpretation of a VMC output graph, (center) brick-based braided structure forming a partial cover partition, and (left) braided structure forming a full-cover wall. In these cases where the plant growth is not steered, the growth spreads fairly evenly over the scaffold when it is continuous (left). Such evenly spread growth is sometimes desired for the architectural artifacts (see *D3.2 Architectural propositions*). Without steering, this is not achieved in the IGP on structures with discontinuous shape features (center, right), especially when there are only limited points where the structure is anchored to potential plant root points (center). Likewise, without steering, growth concentrated in specific areas is not achievable in the IGP on continuous structures (left).

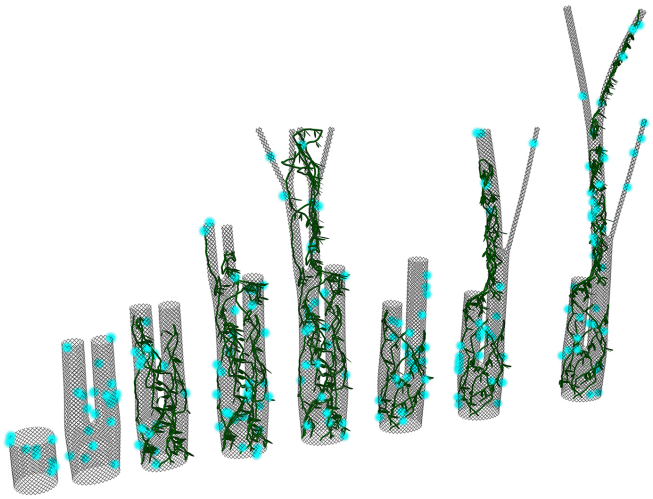


Figure 54: Two example growth careers from IGP, with all nodes triggered stochastically. (Left) Growth career #1, steps 1-5 shown in the first five braids; (right) growth career #2, steps 1-3 shown in the last three braids. Though light sources are all triggered stochastically, in some cases growth still consolidates in certain zones (tops of 5th, 7th, 8th braids), likely due to the size and proximity of the scaffold branches.

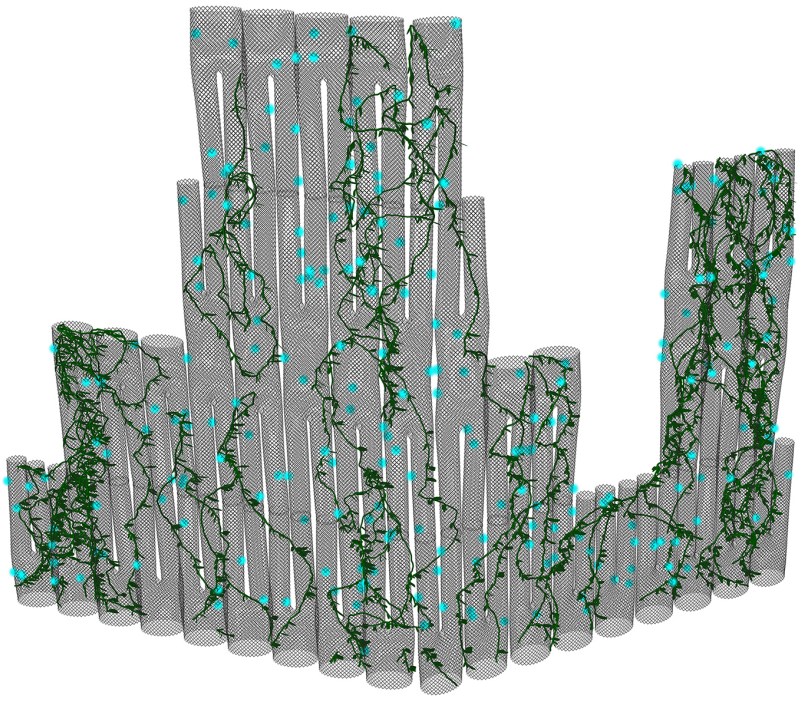


Figure 55: Large braided structure with nodes in the center zone triggered stochastically at each timestep, and zones in the two narrow end zones triggered continuously. Growth tends to consolidate in the zones where light sources are triggered continuously.

# 7 Conclusions

We have reported our recent results and developments that belong to WP2 "Becoming". This work is mostly concerned about growth control of both artificial and natural structures. The ongoing work on the VMC has resulted in the development of a controller that establishes plant-supporting growth of artificial structures and scaffolds. Our evolved controllers for the robotic nodes help us to steer the motion and growth of plants. The potential of simultaneously growing plants and braided structures, that are controlled by VMC and the evolved controllers, is modeled by Integrated Growth Projection (IGP). The robotic nodes and the combined RaspiNet/VMC approach can be easily integrated to grow our robot-plant symbiont. Using the VMC, we can generate plant-supporting artificial growth that discovers and explores environmentally advantageous locations for natural growth. Both, the robotic nodes and the RaspiNet/VMC approach can be used to augment in a truly bio-hybrid approach the adaptive behaviors of natural plants as well as their decision-making capabilities.

Our new findings of options for growth repellers open up new options of how to control plant growth. This will qualitatively change our approach and will be a significant extension of our capabilities for plants shaping within *flora robotica*. Using phytosensors to establish plant-robot communication is another element that is going to alter the 'being' of *flora robotica* into a system that keeps plants at its center and that is robust (cf. recent progress reported in D1.3). In our planned benchmark task we will show the integration and feasibility of our approach by growing desired architectural artifacts.

## References


[1] Martín Abadi et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[2] Anthony Antoun, Gabriele Valentini, Etienne Hocquard, Bernát Wiandt, Vito Trianni, and Marco Dorigo. Kilogrid: A modular virtualization environment for the kilobot robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*, pages 3809–3814. IEEE, 2016.

[3] Daniel Barthélémy and Yves Caraglio. Plant architecture: A dynamic, multilevel and comprehensive approach to plant form, structure and ontogeny. *Annals of Botany*, 99(3):375–407, 2007. doi: 10.1093/aob/mcl260. URL https://doi.org/10.1093/aob/mcl260.

[4] Renaud Bastien, Stphane Douady, and Bruno Moulia. A unified model of shoot tropism in plants: Photo-, gravi- and propio-ception. *PLOS Computational Biology*, 11(2), 2015. URL https://doi.org/10.1371/journal.pcbi.1004037.

[5] Peter Chervenski et al. MultiNEAT, 2018. URL http://www.multineat.com/.

[6] François Chollet et al. Keras. 2015. URL https://keras.io/.

[7] Paul-Henry Cournede, Amelie Mathieu, Franois Houllier, Daniel Barthelemy, and Philippe de Reffye. Computing competition for light in the greenlab model of plant growth: A contribution to the study of the effects of density on resource acquisition and architectural development. *Annals of Botany*, 101(8):1207–1219, 2008. doi: 10.1093/aob/mcm272. URL https://doi.org/10.1093/aob/mcm272.

[8] Heiko Hamann. *Swarm Robotics: A Formal Approach*. Springer, 2018.

[9] John C. Hart, Brent Baker, and Jeyprakash Michaelraj. Structural simulation of tree growth and response. *The Visual Computer*, 19(2):151–163, May 2003. ISSN 1432-2315. doi: 10.1007/s00371-002-0189-4. URL https://link.springer.com/article/10.1007/s00371-002-0189-4.

[10] Mary Katherine Heinrich, Payam Zahadat, John Harding, et al. Using interactive evolution to design behaviors for non-deterministic self-organized construction. In *Proc. of SimAUD, Symposium on Simulation for Architecture and Urban Design*, 2018. in print.

[11] Daniel Nicolas Hofstadler, Mostafa Wahby, Mary Katherine Heinrich, Heiko Hamann, Payam Zahadat, Phil Ayres, and Thomas Schmickl. Evolved control of natural plants: Crossing the reality gap for user-defined steering of growth and motion. *ACM Trans. Auton. Adapt. Syst. (TAAS)*, 12(3):15:1–15:24, 2017. URL https://doi.org/10.1145/3124643.

[12] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Trans. on Evo. Comp.*, 17(1):122–145, 2013.

[13] Vincent Le Chevalier, Marc Jaeger, Xing Mei, and Paul-Henry Cournède. Simulation and visualisation of functional landscapes: Effects of the water resource competition between plants. *Journal of Computer Science and Technology*, 22(6):835–845, Nov 2007. ISSN 1860-4749. doi: 10.1007/s11390-007-9105-8. URL http://dx.doi.org/10.1007/s11390-007-9105-8.

[14] Mohsen Mansouryar et al. Smoothing via iterative averaging (SIA) a basic technique for line smoothing. *Int. J. of Computer and Electrical Eng.*, 4(3), 2012.

[15] Andrew L. Nelson, Gregory J. Barlow, and Lefteris Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57:345–370, 2009.

[16] Stefano Nolfi and Dario Floreano. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT Press, 2000.

[17] Sören Pirk, Till Niese, Oliver Deussen, and Boris Neubert. Capturing and animating the morphogenesis of polygonal tree models. *ACM Trans. Graph.*, 31(6):169:1–169:10, November 2012. ISSN 0730-0301. doi: 10.1145/2366145.2366188. URL http://dx.doi.org/10.1145/2366145.2366188.

[18] Przemyslaw Prusinkiewicz. A look at the visual modeling of plants using L-systems. In Ralf Hofestädt, Thomas Lengauer, Markus Löffler, and Dietmar Schomburg, editors, *Bioinformatics*, pages 11–29, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[19] A. Rafsanjani, Y. Zhang, B. Liu, S. M. Rubinstein, and K. Bertoldi. Kirigami skins make a simple soft actuator crawl. *Science Robotics*, 3(15):eaar7555, Feb. 2018. doi: 10.1126/scirobotics.aar7555. URL http://dx.doi.org/10.1126/scirobotics.aar7555.

[20] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *IEEE International Conference on Robotics and Automation (ICRA 2012)*, pages 3293–3298. IEEE, 2012.

[21] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014. doi: 10.1126/science.1254295. URL http://dx.doi.org/10.1126/science.1254295.

[22] Michalina Smolarkiewicz, Tomasz Skrzypczak, Micha Michalak, Krzysztof Leniewicz, J. Ross Walker, Gwyneth Ingram, and Przemysaw Wojtaszek. Gamma-secretase subunits associate in intracellular membrane compartments in arabidopsis thaliana. *Journal of Experimental Botany*, 12(65):0153027, Apr 2014. doi: 10.1093/jxb/eru147. URL http://dx.doi.org/10.1093/jxb/eru147.

[23] Unity. Unity simulation platform website, 2018. URL https://unity3d.com/.

[24] Gabriele Valentini, Heiko Hamann, and Marco Dorigo. Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In R. Bordini, E. Elkind, G. Weiss, and P. Yolum, editors, *14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015)*, pages 1305–1314. IFAAMAS, 2015. URL http://dl.acm.org/citation.cfm?id=2773319.

[25] Mostafa Wahby, Daniel Nicolas Hofstadler, Mary Katherine Heinrich, Payam Zahadat, and Heiko Hamann. An evolutionary robotics approach to the control of plant growth and motion. In *Self-Adaptive and Self-Organizing Systems (SASO), 2016 IEEE 10th Int. Conf. on*, pages 21–30. IEEE, 2016. URL https://doi.org/10.1109/SASO.2016.8.

[26] Wasabimole. Wasabimole asset package website, 2018. URL https://assetstore.unity.com/packages/tools/modeling/procedural-tree-32907.

[27] Hong-Ping Yan, Meng Zhen Kang, Philippe De Reffye, and Michael Dingkuhn. A dynamic, architectural plant model simulating resourcedependent growth. *Annals of Botany*, 93(5): 591–602, 2004. doi: 10.1093/aob/mch078. URL http://dx.doi.org/10.1093/aob/mch078.

[28] Payam Zahadat, Daniel Nicolas Hofstadler, and Thomas Schmickl. Vascular Morphogenesis Controller: A generative model for developing morphology of artificial structures. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 163–170, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4920-8. doi: 10.1145/3071178.3071247. URL https://doi.acm.org/10.1145/3071178.3071247.

[29] Payam Zahadat, Daniel Nicolas Hofstadler, and Thomas Schmickl. Development of morphology based on resource distribution: Finding the shortest path in a maze by Vascular Morphogenesis Controller. In *14th European Conference on Artificial Life (ECAL-2017)*, volume 14, pages 428–429, 2017. URL https://doi.org/10.7551/ecal_a_071.